Die approbierte Originalversion dieser Diplom-/Masterarbeit ist in der Hauptbibliothek der Technischen Universität Wien aufgestellt und zugänglich.
http://www.ub.tuwien.ac.at

The approved original version of this diploma or master thesis is available at the main library of the Vienna University of Technology.
http://www.ub.tuwien.ac.at/eng

FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Kosteneffizientes Ressourcenmanagement in verteilten Cloud Datenzentren

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Informatik

eingereicht von

Andreas Egger
Matrikelnummer 0626885

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Univ. Prof. Dr. Ivona Brandić
Mitwirkung: Dražen Lučanin, MSc.

Wien, 21.04.2016

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT
FÜR !NFORMATIK

Faculty of Informatics

Cost aware resource management in distributed cloud data centers

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

by

Andreas Egger
Registration Number 0626885

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Univ. Prof. Dr. Ivona Brandić
Assistance: Dražen Lučanin, MSc.

Vienna, 21.04.2016

(Signature of Author)

(Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Andreas Egger
Ospelgasse 35/3, 1200 Wien

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum)

(Unterschrift Verfasser)

Acknowledgements

I would like to express my sincere gratitude to my supervisor Ivona Brandic for supporting me throughout the whole process. She gave me invaluable hints and suggestions on how to improve my work and concentrate on relevant topics.

I would also like to thank Drazen Lucanin who has been extremely helpful in providing advice and support as well as assist in issues regarding implementation. Our discussions helped me see the bigger picture when I got too caught up in details.

A special thanks goes out to Daniel Schall, my supervisor at work who helped me get a broader perspective and supported me with regard to implementation. Without his review the practical part would have been far less profound and advanced.

Last but not least I want to thank my family for their continuous support and love, which helped me persevere and get through hard times. Without them this work would not have been possible.

Abstract

Cloud computing services experienced increasing popularity over the recent years which caused the power demand of data centers to increase significantly. As energy costs represent a significant part of the total cost of data centers energy cost reductions in cloud computing environments play an important role for cloud providers to remain competitive and provide affordable cloud services. This thesis introduces a cloud framework to evaluate energy cost reductions in multi electricity market environments. It is shown that substantial cost savings are possible using intelligent resource scheduling algorithms with integration of energy price forecasts.

This thesis consists of two parts. In the first part different forecasting models are evaluated to assess the performance of the models for energy price time series of different energy markets. A forecast simulation framework is introduced capable of dynamically managing energy price data from different power markets and generating forecasts based on that data. The framework incorporates methods for automatic model generation and evaluation as a basis for extended forecast simulations. A large scale forecast evaluation is conducted to gain insights into model accuracy across a wide range of energy price data. Different training periods, forecast horizons and energy price datasets lead to a comprehensive evaluation of the forecasting models.

The second part is dedicated to large scale cloud simulations comprising different cloud schedulers and scenarios. An existing cloud simulation framework has been extended to perform simulations across a wide range of energy price data where different scheduling mechanisms have been introduced. A utility function has been defined for sophisticated evaluation of different criteria to determine the most suitable VMs for migration at any point in time. Criteria involve probability of SLA penalties and maximum cost benefits to appropriately handle the tradeoff between minimizing costs and providing an adequate level of quality of service.

Results show that significant cost savings are possible with schedulers incorporating different criteria and energy price forecasts which reveal promising results. Based on the defined cloud settings it illustrates the applicability of the proposed approach to real world scenarios of geo-distributed data centers connected to energy markets.

Kurzfassung

Cloud Dienste haben in den letzten Jahren zunehmend an Popularität gewonnen wodurch die Energiekosten von Datenzentren signifikant stiegen. Da die Energiekosten einen bedeutenden Anteil an den Gesamtkosten eines Datenzentrums betragen ist es für Cloud Betreiber von großer Bedeutung diese auf ein Minimum zu reduzieren um wettbewerbsfähig zu bleiben und leistbare Cloud Dienste anbieten zu können. Diese Arbeit präsentiert ein Cloud Framework zur Evaluierung von Energiekosteneinsparungen mit Anbindung an unterschiedliche Elektrizitätsmärkte. Es wird gezeigt dass signifikante Kosteneinsparungen mit Hilfe von intelligenten Verteilungsmechanismen unter Berücksichtigung von Energiepreisvorhersagen möglich sind.

Die Arbeit besteht aus zwei Teilen. Im ersten Teil werden unterschiedliche Vorhersagemodelle untersucht um die Qualität der Modelle bezüglich Energiepreisdaten unterschiedlicher Energiemärkte zu bestimmen. Ein Simulationsframework für Vorhersagemodelle wurde erstellt das die dynamische Evaluierung von Energiepreisdaten ermöglicht und in der Lage ist Vorhersagemodelle basierend auf diesen Daten zu erstellen. Das Framework stellt Methoden zur automatischen Modellgenerierung bereit das als Basis für weitere Simulationen dient. Zudem wird eine weitreichende Auswertung von Vorhersagen durchgeführt um Erkenntnisse über die Genauigkeit der Vorhersagen für eine größere Menge an Energiepreisdaten zu erhalten. Verschiedene Trainingsperioden, Vorhersagezeiträume und Auswahl an Energiedaten ergeben eine umfassende Analyse der Vorhersagemodelle.

Im zweiten Teil werden Simulationen über größere Zeiträume hinweg durchgeführt mit unterschiedlichen Verteilungsalgorithmen und Szenarien. Ein bestehendes Simulationsframework wurde erweitert um Simulationen über größere Datenmengen von Energiepreisen durchzuführen wobei verschiedene Verteilungsmechanismen vorgestellt werden. Eine Nutzenfunktion wurde definiert die basierend auf unterschiedlichen Kriterien eine Entscheidungshilfe bereitstellt um die best geeignetsten virtuellen Maschinen für die Migration zu einem bestimmten Zeitpunkt zu ermitteln. Beispiele von Kriterien sind die Wahrscheinlichkeit von SLA Strafen und die maximal möglichen Kostenreduktionen die entsprechend gewichtet werden um einen Kompromiss zwischen der Kostenreduktion und den dadurch entstehenden SLA Strafen zu erreichen.

Resultate zeigen dass erhebliche Kosteneinsparungen möglich sind unter Verwendung von Verteilungsmechanismen die unterschiedliche Kriterien und Energiepreisvorhersagen berücksichtigen. Basierend auf der simulierten Cloud Umgebung kann die Anwendbarkeit der vorgestellten Methode auf reale Umgebungen bestehend aus geographisch verteilten Datenzentren mit Anbindung an Energiemärkte gezeigt werden.

Contents

List of Figures x

List of Tables xv

Listings xvii

1 Introduction 1

1.1 Motivation . . . 1
1.2 Problem Statement . . . 3
1.3 Main contributions . . . 4
1.4 Methodological Approach . . . 5
1.5 Structure of the work . . . 6

2 State of the Art 7

2.1 Electricity price forecasting . . . 7
2.2 Cost optimization in geo distributed data centers . . . 12
2.3 Optimization of resource scheduling algorithms . . . 14
2.4 Virtual machine migration . . . 17

3 Data Analysis 21

3.1 Day ahead and real time energy prices . . . 21
3.2 Characteristics of energy markets . . . 23
3.3 Energy price case study . . . 28

4 Forecasting 33

4.1 Introduction . . . 33
4.2 Methodology . . . 34
4.3 Model generation . . . 43
4.4 Model selection algorithm . . . 49
4.5 R / Java Simulation Framework . . . 52
4.6 Forecast model evaluation . . . 57

5 Simulation Framework 65

5.1 Architecture of simulation framework . . . 65

5.2 Modeling migration energy 70
5.3 SLA management 72
5.4 Cost optimization based on utility function 73

6 Evaluation and Results 79
6.1 Definition of simulation scenarios 79
6.2 Utility function optimization 91
6.3 Simulation Results 100

7 Conclusion and Future Work 119
7.1 Future Work 120

A Forecast Evaluation Results 123
A.1 Result tables 123
A.2 Result graphs 128

B Simulation Results 145
B.1 Result tables 145
B.2 Result graphs 148

C Source Repository 181
C.1 Projects 181

Bibliography 183

List of Figures

1.1 Data center lifetime costs as function of PUE [104] 2

2.1 The dependence between the hourly log prices and hourly log system loads in California for the period January 1 – July 2, 2000. [101] 8
2.2 Taxonomy of price forecasting models [2] 10
2.3 Two level scheduling model [34] 15
2.4 Illustration of scheduling power savings [106] 16
2.5 WWS curve for a complete run of SPEC CINT2000 (512MB VM) [23] 18
2.6 Live migration algorithm performs pre-copying in iterative rounds [58] 19
2.7 Migration cost savings by using proposed models [58] 19

3.1 Day ahead vs real time energy prices, Maine 4th Aug, 2015 22
3.2 Day ahead and real time prices, ISO-NE, ME 23
3.3 One sided and two sided auctions in power markets [97] 26
3.4 EEX - hourly spot prices [67] 27
3.5 EEX levels [67] 28
3.6 Nord Pool levels [67] 28
3.7 Day ahead energy market prices 29
3.8 Real time energy market prices 30
3.9 Histograms of day ahead markets for years 2012 to 2014 31
3.10 Histograms of real time markets for years 2012 to 2014 32

4.1 Periodogram of hourly day ahead prices of Nord Pool Spot, Helsinki from July 7th to July 21st in 2014 35
4.2 STL decomposition of hourly day ahead time series, Nord Pool Spot, Helsinki from July 7th to July 21st in 2014 36
4.3 Manual ARIMA model generation process [85] 44
4.4 Two weeks of hourly day ahead prices with ACF and PACF plots (Nord Pool Spot, Helsinki from July 7th to July 21st in 2014) 45
4.5 Intermediate ARIMA(0,0,0)(1,1,0)[24] model with seasonal difference 46
4.6 Residuals of ARIMA(2,0,0)(1,1,0)[24] 47
4.7 Class diagram of Energy Price Management Application 53
4.8 Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 60
4.9 Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 61

5.1 Architecture Block Diagram 67
5.2 Component Diagram 68

6.1 Map of data center locations for day ahead price simulations 79
6.2 Map of data center locations for real time price simulations 80
6.3 Normalized cost vs power values based on the energy utility scenario 93
6.4 Total resulting costs per scheduler for the energy utility scenario 94
6.5 Cloud metrics calculated for the energy utility over the simulation time period . . . 94
6.6 Normalized cost vs power values based on the cost utility scenario 96
6.7 Total resulting costs per scheduler for the cost utility scenario 96
6.8 Cloud metrics calculated for the cost utility over the simulation time period 97
6.9 Normalized cost vs power values based on the best utility scenario 98
6.10 Total resulting costs per scheduler for the best utility scenario 99
6.11 Cloud metrics calculated for the best utility over the simulation time period 99
6.12 Day ahead energy prices over 5 weeks in 2013, in $ per MWh 103
6.13 Normalized power and costs for DA Summer simulation scenario 104
6.14 Total resulting costs for DA Summer simulation scenario 104
6.15 Cloud metrics resulting from the DA Summer simulation scenario 104

6.16 Day ahead energy prices over 4 months in 2013, in $ per MWh 105
6.17 Normalized power and costs for DA Spring simulation scenario 106
6.18 Total resulting costs for DA Spring simulation scenario 107
6.19 Cloud metrics resulting from the DA Spring simulation scenario 107
6.20 Real time energy prices over 5 weeks in 2013, in $ per MWh 108
6.21 Normalized power and costs for RT Summer simulation scenario 109
6.22 Total resulting costs for RT Summer simulation scenario 110
6.23 Cloud metrics resulting from the RT Summer simulation scenario 110
6.24 Real time energy prices over 4 months in 2013, in $ per MWh 111
6.25 Normalized power and costs for RT Spring simulation scenario 112
6.26 Total resulting costs for RT Spring simulation scenario 113
6.27 Cloud metrics resulting from the RT Spring simulation scenario 113
6.28 Energy prices, energy price forecasts and utilization levels per location for simulation RT Summer and scheduler BCU_MF 115

A.1 Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 128
A.2 Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 129
A.3 Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 129
A.4 Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 130
A.5 Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 130
A.6 Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina 131
A.7 Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 132
A.8 Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 133
A.9 Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 133
A.10 Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 134
A.11 Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 134
A.12 Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain 135
A.13 Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, ISO NE, Portland 136
A.14 Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, ISO NE, Portland 137

A.15 Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, ISO NE, Portland 137
A.16 Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, ISO NE, Portland 138
A.17 Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, ISO NE, Portland 138
A.18 Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, ISO NE, Portland 139
A.19 Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, PJM, Richmond 140
A.20 Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, PJM, Richmond 141
A.21 Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, PJM, Richmond 141
A.22 Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, PJM, Richmond 142
A.23 Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, PJM, Richmond 142
A.24 Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, PJM, Richmond 143

B.1 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BFD 149
B.2 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF 150
B.3 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF_F 151
B.4 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF_IF 152
B.5 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_M 153
B.6 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_MF 154
B.7 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_MIF 155
B.8 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BFD 157
B.9 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF 158
B.10 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF_F 159
B.11 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF_IF 160

B.12 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_M 161

B.13 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_MF 162

B.14 Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_MIF 163

B.15 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BFD 165

B.16 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF 166

B.17 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF_F 167

B.18 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF_IF 168

B.19 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_M 169

B.20 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_MF 170

B.21 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_MIF 171

B.22 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BFD 173

B.23 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF 174

B.24 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF_F 175

B.25 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF_IF 176

B.26 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCU_M 177

B.27 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCU_MF 178

B.28 Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCU_MIF 179

List of Tables

3.1 List of selected energy markets 28
3.2 Day ahead energy market summary statistics 30
3.3 Real time energy market summary statistics 30

4.1 Model names and parameters 49
4.2 AICc and Ljung Box values 49
4.3 List of all conducted forecast simulations 58
4.4 Forecast evaluation results based on RMSE for a training period of 2 weeks, Nord Pool Spot, Hamina 59
4.5 Forecast evaluation results based on RMSE for a training period of 3 weeks, Nord Pool Spot, Hamina 59
4.6 Forecast evaluation results based on RMSE for a training period of 4 weeks, Nord Pool Spot, Hamina 59
4.7 Results of evaluation for Hamina, Nord Pool Spot (DA) 62
4.8 Results of evaluation for St.Ghislain, Belpex (DA) 62
4.9 Results of evaluation for Portland, ISO-NE (RT) 63
4.10 Results of evaluation for Richmond, PJM (RT) 63

5.1 Parameters used in the migration algorithm 71
5.2 VM downtime vs SLA penalties 72
5.3 List of utility criteria 74
5.4 Parameters used in equations 74

6.1 List of day ahead energy markets and locations 80
6.2 List of real time energy markets and locations 81
6.3 List of cloud configuration parameters in the simulation 82
6.4 Existing scheduling scenarios 83
6.5 Bandwidth connections in day ahead market locations 91
6.6 Existing scheduling scenarios 91
6.7 List of all measured cloud metrics 92
6.8 List of energy utility criteria with associated weights 93
6.9 List of cost utility criteria with associated weights 95
6.10 List of best utility criteria with associated weights 98
6.11 Utility evaluation results for all scenarios (in $) 99
6.12 Definition of simulation scenarios 100
6.13 Bandwidth connections in day ahead market locations 100
6.14 Bandwidth connections in real time market locations 101
6.15 Additional cloud parameters for simulation scenarios 101
6.16 Cloud parameters for simulation scenario DA Summer 102

6.17 Cloud parameters for simulation scenario DA Spring 106
6.18 Cloud parameters for simulation scenario RT Summer 109
6.19 Cloud parameters for simulation scenario DA Spring 112
6.20 Average cloud utilization and runtime per simulation and scheduler 116
6.21 Total costs for each simulation scenario and cloud scheduler (in $) 117
6.22 Normalized total cost reductions for different scenarios (in %) 117

A.1 Forecast evaluation results based on RMSE for a training period of 2 weeks, Nord Pool Spot, Hamina 123
A.2 Forecast evaluation results based on RMSE for a training period of 3 weeks, Nord Pool Spot, Hamina 123
A.3 Forecast evaluation results based on RMSE for a training period of 4 weeks, Nord Pool Spot, Hamina 124
A.4 Forecast evaluation results based on RMSE for a training period of 2 weeks, Belpex, St.Ghislain 125
A.5 Forecast evaluation results based on RMSE for a training period of 3 weeks, Belpex, St.Ghislain 125
A.6 Forecast evaluation results based on RMSE for a training period of 4 weeks, Belpex, St.Ghislain 125
A.7 Forecast evaluation results based on RMSE for a training period of 2 weeks, ISO NE, Portland 126
A.8 Forecast evaluation results based on RMSE for a training period of 3 weeks, ISO NE, Portland 126
A.9 Forecast evaluation results based on RMSE for a training period of 4 weeks, ISO NE, Portland 126
A.10 Forecast evaluation results based on RMSE for a training period of 2 weeks, PJM, Richmond 127
A.11 Forecast evaluation results based on RMSE for a training period of 3 weeks, PJM, Richmond 127
A.12 Forecast evaluation results based on RMSE for a training period of 4 weeks, PJM, Richmond 127

B.1 Cloud metrics for simulation „DA Summer“ 145
B.2 Cloud metrics for simulation „DA Spring“ 146
B.3 Cloud metrics for simulation „RT Summer“ 146
B.4 Cloud metrics for simulation „RT Spring“ 146
B.5 Normalized cloud metrics for simulation „DA Summer“ 147
B.6 Normalized cloud metrics for simulation „DA Spring“ 147
B.7 Normalized cloud metrics for simulation „RT Summer“ 148
B.8 Normalized cloud metrics for simulation „RT Spring“ 148

C.1 List of URLs linking to the projects 182

Listings

4.1 Function evaluateModels 55
4.2 Method runSimulation 56
4.3 Method runSimulation web service interface 56
4.4 Method evaluateModels 56
4.5 Simulation name definition 57

CHAPTER 1

Introduction

1.1 Motivation

In times of cloud computing where computer resources are expected to be readily available from anywhere and with an ever growing global network of interconnected devices cloud providers need to cope with increasing energy consumption in data centers as requests need to be handled at a rapid pace [19]. As energy consumption is directly mapped to energy costs, cloud providers aim to reduce power consumption and thus electricity costs within data centers.

Energy consumption, related energy costs and high energy prices may slow down or hinder advancements in cloud computing. Thus, cloud providers must ensure that delivering their services remains profitable. Lower energy costs increase the competitive ability of cloud providers and bigger investments are possible. Likewise customers may benefit from more affordable cloud services.

In a study from 2011 [54] global data center electricity consumption amounted to about 1.3% of the world's total electricity use. This amounts to a total energy usage of 198.8 Terawatt Hours by data centers in 2010. Surprisingly, Google's energy consumption in data centers added up to less than 1% of that amount. This may be due to Google's constant improvements in energy efficiency [41]. When comparing the time period from 2000 to 2005 with the period from 2005 to 2010 data center electricity use increased only moderately due to applied virtualization and other efficiency measures [54, 55]. Still energy costs of large distributed systems are significant and further actions need to be taken to ultimately reduce these costs [81].

Studies of regional data center energy consumption come to similar conclusions [13].Through improvements in hardware and cooling systems energy consumption can be further reduced. In addition „Green IT" Scenarios can be applied which depend on energy aware routing mechanisms and virtual machine migrations to decrease power usage and costs [59].

Another approach to reduce energy related cost is to reduce cooling costs, since they represent a significant part of the overall energy expenses [77]. This is typically done by installing a more efficient cooling system or by building data centers at locations with low ambient temperature. Cooling through natural powers may also be achieved by building data centers at rivers and using the water to cool down the hardware infrastructure.

In general energy cost reductions in data centers can be addressed by utilizing one of two different approaches. One common approach is to increase energy efficiency to reduce energy consumption and therefore energy related costs as well [10, 12, 18, 57]. This is done e.g. by utilizing more energy efficient hardware, implementing an energy efficient cooling system, virtualization of servers or energy aware load distribution policies [12,59,77]. It additionally aims at minimizing the Power Usage Effectiveness (PUE) of data centers which is a term for describing the ratio of the total amount of power consumed by a data center to the power consumed solely by IT equipment [8]. Despite the importance of increasing energy efficiency to reduce costs and environmental load this approach is limited as a data center's lifetime cost may not be further reduced when capital expenditures (CAPEX) exceed possible savings in operational expenditures (OPEX) (Figure 1.1).

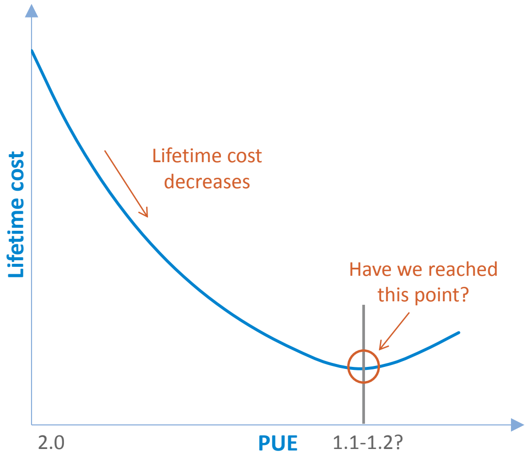


Figure 1.1: Data center lifetime costs as function of PUE [104]

The second approach is directed at reducing costs by applying intelligent scheduling mechanisms in geographically dispersed and interconnected data centers [44, 81]. Thus resources are placed at or migrated to locations where the estimated resulting energy costs are lowest. An emerging and interesting field of application are cost reductions in data centers connected to wholesale power markets [82]. These markets provide power prices that change frequently and exhibit high volatility with price differences of factors up to ten between two consecutive observations. Electricity price behavior of different power markets may differ substantially as each power market is run by independent operators and exhibits location dependent demand and supply patterns [67]. In addition possibly involved seasonality patterns may lead to substantial locational price differences. For example, considering daily seasonal patterns and data centers located at different time zones these patterns may be exploited to route requests to more cost effective locations.

These characteristics of locational and temporal varying energy prices may be exploited by intelligent scheduling algorithms and cost aware resource management routines that place requests at facilities currently exhibiting low energy prices [82]. In addition there is the possibility of migrating resources for sufficiently long running jobs to data centers that operate below peak load and facing low energy prices [15].

The focus of this thesis is to go beyond intelligent scheduling of resources by integrating sophisticated forecasting methods that predict price time series behaviors into the near future. This measure is expected to further optimize resource distribution across data centers by comparing and analyzing current and predicted energy price time series to find the best fit of resources in the near future to result in maximum possible energy cost reduction.

1.2 Problem Statement

A number of challenges arise for cloud providers running a cloud consisting of multiple distributed data centers. Cloud providers may need to both improve energy efficiency due to environmental regulations and reduce total costs within a cloud environment. The focus of this thesis has been laid on energy cost reductions in geo-distributed clouds as opposed to improvements of energy efficiency and reduction in power consumption which are only considered in the context of reducing energy costs.

The following challenges will be addressed in this thesis:

1. Cloud providers have to deal with high energy costs within a set of geo-distributed data centers. As cost reductions based on improving energy efficiency may not be possible for highly optimized data center environments (Figure 1.1) alternative approaches to reducing energy costs are needed.

2. Energy prices in power markets exhibit high volatility and may change rapidly within a short period of time. Thus in order to utilize power market data for energy cost reduction schemes efficient solutions have to be provided that take into account energy price characteristics.

3. Different energy price forecasting methods may be applied to power market prices for reducing costs, however forecast accuracy may vary across different methods and datasets. A solution has to be found for choosing the most suitable forecasting methods for any given dataset.

4. Cloud providers may have different requirements regarding load distribution, cost reductions and prevention of SLA penalties (Service Level Agreements). In particular the latter two requirements are contradictory to each other as aggressive cost reduction techniques may lead to higher SLA penalties and vice versa. Therefore a flexible approach of dealing with these requirements is needed.

5. When dealing with energy prices from different power markets a central management interface for the import and retrieval of energy price data is required as a basis for extended

data analysis. This interface might then be used by cloud scheduling mechanisms to optimize load placement across data centers.

Since energy prices change dynamically a flexible approach of moving data center resources to more cost-effective locations is needed. This work is based on the scenario of a company running data centers distributed across several countries. Depending on the current and estimated energy price for a certain region resources may be relocated to other data centers located at more cost-effective regions.

The key technology to implement the relocation of data center resources is virtual machine (VM) migration [69]. All data and services are located on servers and belong to specific virtual machines within a data center. Since a server may host several VMs that operate completely independent from each other, they can be relocated (migrated) to other servers, possibly over networks without affecting VMs running on the same server. VM migration may be applied across large geographical distances (geo migrations) by leveraging connections across dedicated networks. It is also possible to apply live migrations on large scale networks without noticeable service interruption [21] [58] [3].

Various load distribution policies exist to optimize the load factor and distribution of tasks within data centers [66,78,105]. A workload manager has to make sure that there are still servers having enough resources and no service level agreement (SLA) is violated on the assignment of a new job. A service level agreement is a contract between a cloud provider and a cloud user which provides guarantees to the user in terms of minimum availability of provided services [64].

In order to meet the challenges described above possible solutions to these challenges are presented in the next section.

1.3 Main contributions

The goal of this thesis is to devise possibilities of energy cost reductions for data center operators in charge of geographically distributed and interconnected data centers by exploiting VM migrations and the variability of energy prices in wholesale power markets. This approach presumes that data center operators are charged directly by wholesale power markets, which is becoming a more attractive option since it provides a possibility for large cloud providers to ultimately reduce energy costs [104]. Given the fact that data centers consume a huge amount of energy comparable to the energy consumption of mega cities [81] energy market integration may constitute a significant cost saving opportunity for cloud providers.

The contributions of this work are defined by meeting the challenges described in the last section:

1. A cost aware scheduling algorithm is proposed for maximum energy cost reduction in a multi electricity market environment. Combined with energy price forecasts resources may be scheduled intelligently such that price changes within the near future can be detected and resources are placed with respect to the estimated prices. In addition resources may be rescheduled to consider changing energy prices within the current point in time and the near future.

2. Handling of energy price characteristics such as price spikes and high volatility are addressed by generating models based on energy price data and application of these models for future price estimations. As prices can be modeled more accurately scheduling mechanisms can be improved for further energy cost optimizations.

3. An automated model selection procedure is proposed that investigates the underlying dataset, generates models and validates those models based on the given data. The best resulting model can then be used for forecasting within a defined forecast window.

4. A utility function considering defined criteria and associated weights is introduced to be able to adjust parameters on demand and handle the tradeoff between possibly contradictory criteria. Depending on the defined weights this function emphasizes criteria associated with highest weights to e.g. prioritize cost reductions over SLA penalties.

5. A framework has been implemented for convenient import of energy price data from different power markets and querying data over an extended time period. This framework is capable of setting up schedulers to automatically retrieve data from defined sources and generate forecasting models based on that data. These models may then be used for energy price predictions in scheduling algorithms.

1.4 Methodological Approach

As preparation for the simulation and forecasting scenarios different types of electricity price data from various locations is analyzed and relevant characteristics are discussed. Also different energy markets are introduced where each market has its own unique characteristics that can be used for building forecasting models. Also a case study about energy price time series used in this thesis is presented to see differences in real world energy price data of different types and in different locations.

A large scale evaluation of forecasting methods is performed in order to make qualified assumptions about suitable models for both real time and day ahead energy price data. We will utilize ARIMA models in an algorithm for automatic model selection which determines the most suitable models based on a given training dataset. The resulting models are then evaluated based on a weighted function of corrected Akaike information criterion (AICc) and Ljung box values to determine the most suitable model for the given data.

Models are trained based on different training data sets and exposed to various forecast horizons where the resulting models are compared by a set of error measures (e.g. MAPE, RMSE). Thus the best model can be determined by comparing resulting forecast errors where the model with the least overall forecast error and most stable distribution is taken as basis for further simulations.

A simulation scenario will be established where different types of workload is placed on several geographically distributed and interconnected data centers and results are evaluated based on the given workload, energy prices and data center capacities. In addition, energy price forecasts will be integrated to predict significant changes in energy price levels at different energy markets. This leads to a more cost efficient scheduling approach where workload is placed at data centers that provide best energy price conditions for current and/or future points in time.

The simulation will be based on a well established cloud simulator [62] with a set of parameters to simulate predefined scenarios. The results for different scenarios and parameter settings (e.g. with or without forecasts) will be compared and evaluated such that the possible benefits of each scenario are revealed.

The core of the simulation consists of the energy price scheduler which implements different scenarios and a utility function for simultaneously evaluating different criteria to determine the VMs that are most suitable for migration at the current point in time. The weights of each utility value may be adjusted such that different scenarios with emphasis on different criteria may be simulated.

1.5 Structure of the work

The structure of this thesis is divided into the following sections:

In the *introduction* a general outline is described comprising the purpose of the thesis, the problem statement and how these problems are solved within the thesis. Short insights are given into proposed methodologies and how they are expected to provide improvements to the outlined scenario.

State of the art discusses relevant literature in the field with a comprehensive collection of related articles and journals. It is examined what kind of advancement in the different fields have been made in the past that relate to this thesis and how they can be used as a basis for further investigations.

The chapter about *data analysis* provides insights about real world electricity price data, characteristics and differences of energy markets and an electricity price case study is presented to evaluate the type of data that is dealt with in this thesis.

In the *forecasting* chapter different forecasting models are discussed where the basic model building process is described and an automated model selection algorithm is introduced. Furthermore different models are evaluated across a wider range of electricity price data and compared by forecast accuracy measures.

Then the *simulation framework* is introduced where details about the architecture and implementation of the simulation framework are revealed. This includes the cloud simulator and scheduler to perform different simulations for defined scenarios.

In *Evaluation and results* various simulation scenarios are introduced and results are shown for each type of scenario. It will be shown that significant cost reductions can be achieved by applying the proposed approach.

Finally the thesis concludes with a discussion and future work.

CHAPTER 2

State of the Art

2.1 Electricity price forecasting

A lot of research has been going on in electricity price forecasting in power markets. Different strategies and models have been developed of which the most relevant regarding this thesis will be discussed.

Forecasting spot electricity prices with time series models

In [101] the authors state that forecasting electricity prices with accuracy levels comparable to electricity load forecasting is hard to achieve, as different seasonality patterns (e.g. daily, weekly, annually) and exogenous variables (such as loads and network constraints) have to be taken into account.

Three basic types of models for electricity price forecasting have been identified [17, 101]:

- parsimonious stochastic models
- structural or fundamental models
- non-parametric models

The first class of models aims to capture the data's statistical characteristics and extrapolate these patterns into the future. Energy price characteristics such as mean reversion and price spikes are modeled to more accurately resemble the actual price time series. Structural or fundamental models take a different approach as they take into account external factors (fundamentals) such as load or weather data combined with historical price series to derive accurate forecasts. As the name implies, non-parametric models do not depend on parameter estimations and properties of the time series but define complex algorithms to map input variables to outputs that represent electricity price forecasts for a specific period of time.

In [101] forecast models are developed to model the next 24 hours of the following day (day ahead) by utilizing historical prices and demand as well as day ahead load predictions. In order to model the daily variation of demand, costs and operational constraint models have been built for each hour separately which is inspired by studies of demand forecasting [16].

Data was taken from the Californian power market from before and during the market crash in 2001 [101]. Preprocessing has been done by removing outliers (substitution by the arithmetic average of neighbouring values), mean removal (center data around zero) and logarithmic transformations of load ($z_t = \log(Z_t)$) and price ($p_t = \log(P_t)$) time series. The latter approach is used to obtain a more stable variance of the time series data.

To achieve seasonal autoregression models the log price p_t was made dependent on the same hour in previous days and weeks as well as on an aggregate function based on the previous day prices. The system load has been used as the fundamental variable as strong correlations between loads and prices have been revealed (Figure 2.1).

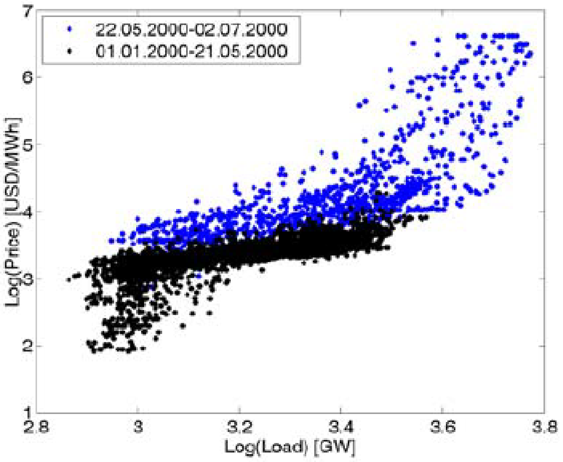


Figure 2.1: The dependence between the hourly log prices and hourly log system loads in California for the period January 1 – July 2, 2000. [101]

As can be observed the dependence between loads and energy prices is almost linear except for very low and very high loads, where electricity prices tend to jump to lower and higher values, respectively.

Results show that for the proposed dataset ARX models (AR with exogenous variables) exhibit lower mean weekly errors (MWE) than AR models which in turn had significantly lower errors than standard ARIMA models. The difference is the separate modeling of individual hours for the ARX and AR models as opposed to an 48 hour aggregate modeling for ARIMA models.

ARIMA models to predict next-day electricity prices

The authors in [27] propose the application of ARIMA models for day ahead energy price forecasting based on data in the Spanish and Californian power markets. These models are based on time series analysis and results show that accurate electricity price forecasting for the aforementioned power markets is possible. Depending on the market and selected time range average weekly prediction errors range from 5% to 10%. These results are quite reasonable given that prices from the Californian market in the year 2000 have been selected which has been a very volatile and unstable time period due to the market crash in early 2001 [97].

The described model generation process in [27] consists of five steps:

0) A class of models is formulated assuming certain hypotheses.

1) A model is identified for the observed data.

2) The model parameters are estimated.

3) If the hypotheses of the model are validated, go to Step 4, otherwise go to Step 1 to refine the model.

4) The model is ready for forecasting.

This process is taken from the Box Jenkins methodology for forecast model generation [45]:

Step 0) consists of determining a suitable class of models that can be applied to the time series based on observed characteristics, e.g. high frequency, nonconstant mean and variance, or seasonality patterns.

Step 1) selects a trial model for which data has to be made stationary. Through application of a logarithmic transformation a more stable variance can be achieved whereas (seasonal) differencing of data may result in a more stable mean. In order to find the appropriate order of the resulting ARIMA model autocorrelation (ACF) and partial autocorrelation (PACF) plots may be consulted to retrieve a first fit of the model.

Step 2) consists of estimation of parameters which is typically done by applying the maximum likelihood function with respect to the parameters. In order to optimize forecasting performance outlier detection methods may be applied to minimize the impact of single „spikes“ in the data.

Step 3) evaluates the residuals (forecast errors) of the resulting model based on desired characteristics. Residuals should exhibit zero mean, constant variance, follow a normal distribution and be uncorrelated. Suitable tests for these properties are portmanteau tests (Ljung box, Box Pierce) and examination of ACF and PACF plots to find dependencies within the data.

Step 4) is concluding the model selection process. The resulting model of step 2) is used for forecasting based on trained data.

Notably only five hour training periods have been used to model forecasts for the next consecutive hour for Spanish power prices, only two hour training periods have been applied for modeling Californian energy prices. Through outlier detection the resulting forecasts could have been slightly improved but as stated in [27] this would result in significantly increased computation time.

Classification of forecast models applicable to electricity price forecasting

Electricity price forecast models in general can be classified in three different sets of models: game theory models, time series models and production cost or simulation models [2,40] (Figure 2.2).

Game theoretic models study the evolution of energy prices by analyzing the strategic behavior of the agents [40]. They aim to model the strategies of market participants (agents) as formalized games and identify solutions to those games [2]. Specifically equilibrium models analyze the strategic market equilibrium which describes the set of strategies such that no player in the market can improve its position if its rivals maintain their strategies [40].

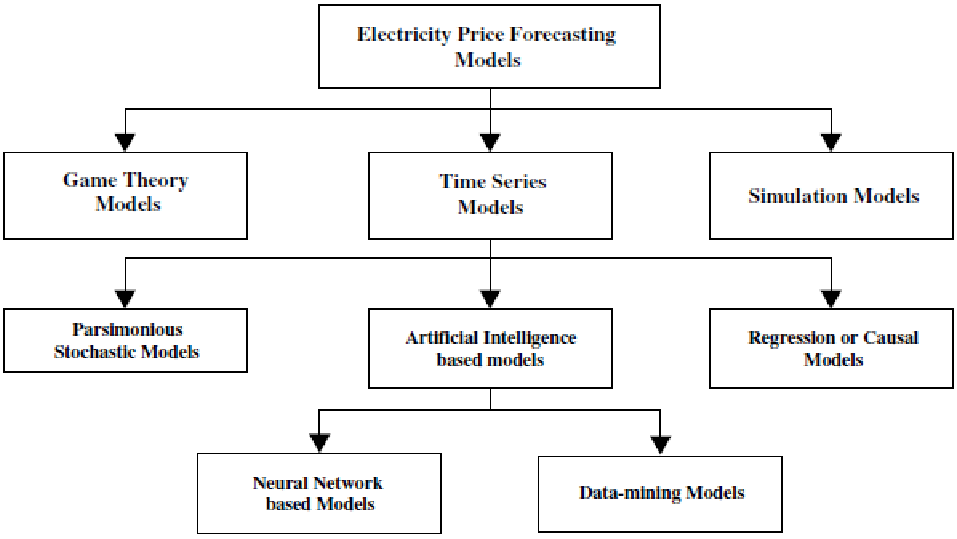

Electricity Price Forecasting Models
Game Theory Models
Time Series Models
Simulation Models
Parsimonious Stochastic Models
Artificial Intelligence based models
Regression or Causal Models
Neural Network based Models
Data-mining Models

Figure 2.2: Taxonomy of price forecasting models [2]

Production cost or simulation models build an exact model of the system and emulate the underlying process [2]. They simulate the operation of power systems as well as agents' strategic behavior impact on market prices [40, 61]. Based on the system model mathematical models are generated by the simulation method which are then solved for price forecasting. Due to their advanced modeling of the system these models require detailed system operation data and exhibit high computational costs [2].

Time series models on the other hand are based on historical price data and statistically model price evolution but do not model the underlying system's behavior in detail. Time series models have the advantage to exhibit a simpler structure and require less information than production cost models or game theory based models [61]. A range of different types of time series models is available, from „black box“ models that are only based on historical price data to complex structural models that include various explanatory variables (e.g. load demand or fuel prices).

Time series models can in turn be classified into classical time series models and models based on machine learning techniques [61]. The benefit of classical time series models is that they exhibit a less complex structure and therefore less computation time than machine learning based models. However it can be harder to obtain accurate forecasts for non-linear time series such as energy prices [61].

Parsimonious stochastic models are inspired by financial research to model characteristics of energy prices. Stochastic models are either based on stationary or non-stationary time series [40]. Stationary denotes the characteristic of a time series to exhibit zero mean and constant variance [85]. The first type of models can only be based on stationary time series (e.g. AR, MA and ARMA models). The latter type can handle non-stationarity by transforming data to a stationary series within the model generation process (e.g. ARIMA, GARCH models) [2, 85].

Regression or casual models handle the relationship between a dependent variable (electricity price) and a number of independent variables (e.g. load or weather conditions). By evaluating the correlations between independent and dependent variables the explanatory variables for the model can be identified [2].

Finally artificial intelligence based methods can be divided into neural network based models and data mining models [2]. Both of these types of models can be used to discover non-linear relationships that might be difficult to obtain by statistical models. By capturing the autocorrelation structure of the time series, neural network models extrapolate patterns discovered in historical price data into the future [40]. In constrast data mining methods use data mining techniques based on categorization of data, e.g. Bayesian categorization or closest k-neighborhood categorization [2].

When comparing statistical time series models with neural network based models a number of benefits and drawbacks can be discovered related to energy price forecasting.

Extensive research has been made regarding stochastic time series models whereby models may be applied to single [25, 28, 37, 71, 91, 98, 102] or complex multiple seasonality patterns [30, 43, 107]. Studies show that univariate time series models may be improved significantly by modeling each hour of the day separately [28, 102]. However this requires model building separately for each hour which increases the amount of effort needed especially in dynamic forecasting environments. It is also stated that inclusion of non-linear phenomena would drastically increase computation time for time series models [28].

Regarding machine learning approaches, artificial neural networks (ANN) have been studied extensively in relation to electricity price forecasting [4, 20, 31, 38, 76, 90, 95]. Results show that ANN models have the potential to outperform statistical models such as Autoregressive (AR) or Autoregressive Integrated Moving Average (ARIMA) models [20, 76]. This may be due to their ability to more accurately model the underlying system behavior while they excel at modeling non-linear relationships in the dataset.

However, in order to successfully apply forecasting based on neural network models a set of requirements have to be met [20]: Availability of sufficient amounts of data, adequate selection of input/output samples, an appropriate number of hidden units and sufficient computational resources. In case these requirements can not be met or it is not possible to appropriately parametrize the model statistical models provide a good alternative for modeling energy prices.

2.2 Cost optimization in geo distributed data centers

A Study of Electricity Price Features on Distributed Internet Data Centers

A comprehensive study about optimization of energy costs under different operational and energy cost prediction regimes is presented in [29].

The focus of that paper is to leverage price differences across different energy markets in a network of geo distributed data centers to minimize energy costs by migrating resources to currently cheaper locations. It takes into account the impact of several metrics including the level of price variability, time lag between locations, reconfiguration delay and accuracy of energy price predictions.

To provide a forecasting model based on the given data the Box-Jenkins method is used to build an ARIMA model characterizing the underlying electricity price data:

$$ARMA(p,q) : P_t = \epsilon_t + \sum_{i=1}^{p} \phi_i P_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$$

where p and q are the number of autoregressive and moving average terms, ϕ_i and θ_j denote the model coefficients and ϵ_t denotes the error term at time t. To accurately model seasonality in the data an $SARIMA(p,d,q)(P,D,Q)[s]$ model has been used which includes the number of terms p, q and P, Q as non-seasonal and seasonal parameters and d and D for non-seasonal and seasonal differencing, respectively.

Price time series have been generated with different levels of variability $\sigma^2_{Price} \in \{50, 126.6, 300, 500\}$. In addition, prediction errors have been used to simulate different forecast accuracy levels, $\sigma_{pred} \in \{2.0, 5.0, 10.0\}$.

Thus, $P = \{p_1, \ldots, p_T\}$ denotes a price time series and $\hat{P} = \{\hat{p_1}, \ldots, \hat{p_T}\}$ a simulation of forecast values such that $\hat{P_t} = \mathcal{N}(P_t, \sigma_{pred}), \forall t \in T$. As a measure of accuracy the mean squared error (MSE) has been used: $MSE(P, \hat{P}) = \frac{1}{T} \sum_{t \in T} (P_t - \hat{P_t})^2$.

The decision of workload assignment at a particular time t is based on the number of active servers at location i and the corresponding forecast prices $\hat{P_i}(t)$. The total cost at any given moment is defined by $\hat{C_t} = \sum_{i=1}^{N} m_i \times \hat{P_i}(t) \times Po_i$ where Po_i denotes the power used by a server running at location i and m_i defines the number of running servers at location i.

Results for different metrics achieved different cost savings [29]. When the number of locations involved in the simulation is increased, cost savings up to 10% have been reported when going from 1 up to 20 locations. The more locations are considered the greater benefit can be drawn from price differences and time lags between different locations.

Price variability has been simulated with $\sigma^2_{price} \in \{0, 5, \ldots, 500\}$ and resulting costs have been estimated. It was shown that up to 15% of savings could be achieved when comparing the maximum and minimum simulated price variabilities.

Concerning time lags it is stated that the most benefit could be drawn from locations in timezones that are sufficiently distant from each other due to the daily seasonality observed in energy prices. For this scenario up to 15% cost savings have been achieved for locations that were perfectly „out of phase“.

Finally results show that forecast errors have a major impact on resulting costs. In order to quantitatively measure the impact of forecast deviations σ_{pred} has been increased stepwise in the range $0, \ldots, 5$ by steps of 0.1. The resulting MSE is quadratically rising with the level of σ_{pred}, thus when errors are at maximum a 10% revenue loss is expected. Therefore accurate forecasts are very important to achieve major cost savings in a multi cloud environment.

Cutting Down the Energy Cost of Geographically Distributed Cloud Data Centers

This paper aims to minimize energy costs in geographically distributed data centers by taking into account spatio-temporal variation in electricity prices and the outside weather temperature [44]. The scenario is modeled as a linear programming problem which is solved by presenting various heuristic solutions to the problem.

While providing algorithms for minizing energy costs resulting SLA penalties as well as cooling related costs are taken into account. For each data center a performance coefficient (CoP) is calculated which is based on outside weather temperature where the coefficient increases linearly with rising temperature. A data center may have different types of servers exhibiting different performance metrics and capacities.

Energy costs are calculated as the sum of server power consumption over a specific time range multiplied by the applicable energy prices during that time [44]. The total costs for data center i are defined as $E_i^{total} = E_i^{IT} \times CoP(T_i)$ where T_i denotes the temperature at location i.

The actual time a job j needs when running on a type k server s is defined as

$$T_{j,s} = \frac{c_j \times f_j \times l_j}{c_k \times f_k}$$

where c and f denote the number of cores and frequency available / needed by a server / job and l denotes the expected runtime of job j. A penalty occurs if the total runtime of a job exceeds the expected runtime when running on server s.

The goal is to minimize the sum of total energy and penalty costs over all data centers

$$\min \sum_{i=1}^{N} E_i^{total} + Pen_i$$

where Pen_i denotes the total penalty cost for data center i.

Based on the proposed linear optimization problem two types of scheduling algorithms are introduced, immediate and delayed. Immediate scheduling algorithms are defined as CheapestDC and CheapestS where the former places resources randomly on hosts located at the currently cheapest data center while the latter includes outside weather temperature as well in the calculation. Delayed scheduling algorithms additionally consider historical energy prices to decide whether jobs should be delayed to a future time stamp.

Simulation runs over six data centers at different locations handling three types of workloads were conducted to evaluate different scheduling algorithms. Results show that delayed scheduling techniques on light workloads achieve over 5% on overall cost savings compared to the baseline random scheduler. Also, spatial electricity price variations have the most impact on cost savings followed by temporal electricity price changes and reduced cooling costs due to outside weather temperature.

Studies for cost optimizations in multi-electricity-market environments

Different studies have been developed for optimizing energy costs in multi market environments.

In [15] the basic tradeoff between energy and bandwidth costs is discussed for migrating virtual machines between geographically distributed data centers. Three different algorithms have been evaluated on a set of 30 US locations covering three years of next day electricity data. Out of the two greedy algorithms and the online migration algorithm the latter provided the best results with only 4 to 6 % off the optimal offline solution.

The authors in [82] aim to minimize energy costs while guaranteeing quality of service by referring to a constrained mixed integer programming model. The approximated linear programming model is converted to a minimum cost flow problem which can be solved fast and efficiently. With five frontend web portal servers targeting three internet data centers cost reductions up to 30% are achieved within the simulation.

An extensive study on electricity price features and its implications for existing cloud systems is given in [81]. While taking into account server energy elasticity and bandwidth constraints a maximum cost savings of 45% could be achieved when running a simulation over 39 months including 29 locations. However, in order to achieve these results bandwidth constraints are relaxed and large distances to clients have to be permitted.

A different approach is examined by [89] where dynamic pricing schemes of different cloud providers are taken advantage of to reduce investment of users. The assumption is that prices for each cloud provider change according to demand and thus virtual cluster placements can be optimized across different cloud providers. As infrastructure is dynamically distributed without notice of the user it provides an efficient and convenient way of reducing costs for cloud users.

Cost based multi cloud schedulers have been investigated in [56] and [94]. A cloud scheduler for simplified management of virtual machines has been developed in [6], a green cloud scheduler in [63]. Also, schedulers based on genetic algorithms were proposed by [33] for cost based multi QoS optimizations and [39] for improving delay scheduling policy.

2.3 Optimization of resource scheduling algorithms

A task scheduling algorithm based on load balancing in cloud computing

In [34] a two level scheduling model has been implemented (Figure 2.3).

A two stage process is described where the first scheduler creates the task description of a virtual machine s.t. the task will have a perfect fit on a virtual machine whereas the second scheduler is responsible to find appropriate resources (hosts) for the provisioned virtual machine. Thus an optimized resource scheduling is presented with focus on load balancing among different clouds.

Tasks have a defined structure where task t_i out of the set of all Tasks T is defined as

$$t_i = \{tId, tRr, tSta, tData, tVmch, tVm\}$$

- tId is the task identification
- tRr denotes the required resources

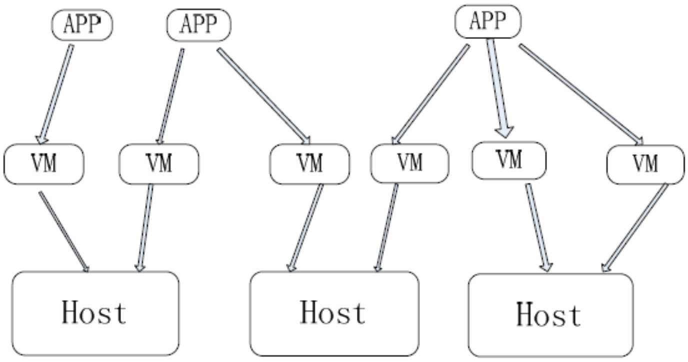


Figure 2.3: Two level scheduling model [34]

- $tSta$ describes the state of the task where $tSta = \{tFree, tAllo, tSche, tWait, tExec, tComp\}$
- $tData$ defines the relative data of the task (computational load, input and output data)
- $tVmch$ denotes the virtual machine specification where $tVmch = \{tId, tRr, tData\}$
- tVm defines the assigned virtual machine of the task where $tVm = \{tId, thId\}$

Hosts on the other hand are defined as $h_j = \{hId, hTcap, hFcap, hData\}$, where

- hId denotes the host identification
- $hTcap$ is the total host capacity (set of resource capacities)
- $hFcap$ is the freely available host capacity
- $hData$ denotes the relative data of the host (including input and output bandwidth)

The scheduling algorithm is based on the following equations:

$$HL_i = \frac{\sum_{j=1}^{n} VL_j}{n}$$

$$avgl = \frac{\sum_{i=1}^{m} HL_i}{m}$$

$$B = \frac{\sqrt{\sum_{i=1}^{m} (HL_i - avgl)^2}}{m}$$

where VL_j is the predicted resource load for VM j, HL_i is the average load for host i, $avgl$ denotes the average load across all running hosts and B denotes a metric for load balancing.

The scheduler iterates over the set of hosts in each timestamp and determines the value of the load balancing metric B. In case the value of B exceeds a predefined threshold B_0 or the predicted resource requirements for a virtual machine exceed the currently allocated resources the VM is migrated to a host occupying the least resources for optimal load balancing.

Results show that considering dynamic resource requirements load is distributed across all machines leading to stable level of cloud utilization.

Efficient resource management for cloud computing environments

In [106] power aware and live migration scheduling techniques are proposed as a green cloud approach to increase overall system efficiency with minimal performance overhead.

A new greedy based algorithm is presented to minimize power consumption. It is based on the findings that power consumption of a server does not increase linearly with the number of cores utilized. Instead, most power is drawn by one utilized core whereas power increases only slightly when comparing a server utilizing 7 to 8 cores. Thus a new VM scheduling algorithm is proposed that minimizes resulting power consumption within a data center.

Hosts are assigned to a pool of resources where a pool can be initialized by a priority based evaluations system to either maximize performance or further minimize power consumption [106]. Each VM request is checked against its resource requirements and available capacities in the pool and assigned to a host to maximize utilization and therefore minimize overall power consumption.

In addition, live migration of virtual machines is applied to further optimize power consumption by shifting load away from under utilized hosts (Figure 2.4). When load increases, Wake on Lan (WOL) technology is used to power servers back up again. Furthermore, a focus has been to reduce VM image size and boot time to optimize virtual machines for a server environment.

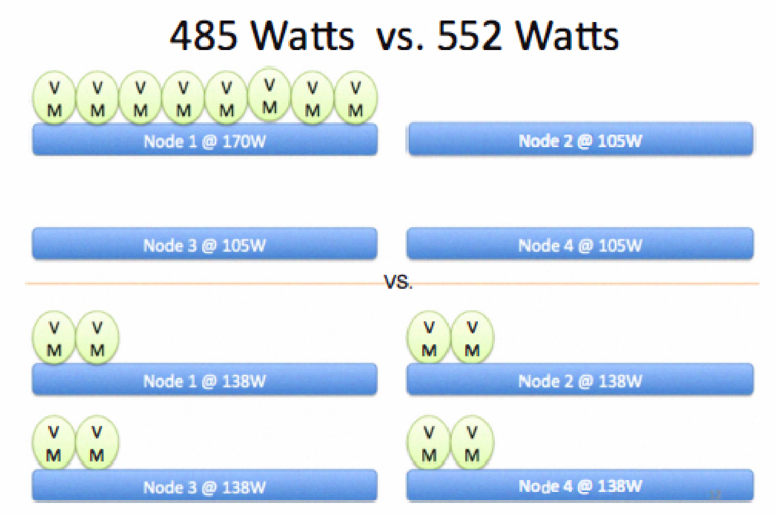


Figure 2.4: Illustration of scheduling power savings [106]

Power aware resource scheduling has been simulated on an OpenNebula [35] pool of 4 servers, equipped with 8 cores each. Power elasticity (power range from idle to peak power of a server) for this type of server is 61% where a total reduction in power consumption of 12% could be achieved. As the results are scalable also to large clusters significant power and cost savings are possible using the proposed approach.

2.4 Virtual machine migration

Live Virtual machine migration is defined as follows:

> A source virtual machine (VM) hosted on a source server is migrated to a destination VM on a destination server without first powering down the source VM [69].

Thus efficient methods for transferring VM memory pages from the source to the destination host are needed which will be evaluated in the following sections.

Live migration of virtual machines

In [23] an efficient approach for live migration of virtual machines is presented. This paper targets the migration of entire OS within a virtual machine without noticeable service interruption.

The applied method for migration should minimize both downtime and total migration time for the virtual machine. The first metric denotes the actual downtime of the VM where it is completely suspended and unaccessible to users. The second metric is defined by the total time between the start of the migration process until its end [23].

Three methods have been identified that have been used previously. The *pure stop-and-copy* approach halts the source VM, copies all memory pages to the destination host and resumes execution at the destination. Both downtime and total migration time are proportional to the amount of memory to be transferred which can lead to unacceptable downtimes.

The *pure demand migration* method is defined where essential data for running the VM is transferred to the destination host after which the destination VM is booted. Any missing memory pages create page faults and trigger a synchronous transfer from the source host on first use. This leads to poor performance until a sufficient amount of memory pages have been transferred.

The *pre-copy* approach combines an iterative push-phase with a very short stop-and-copy phase to minimize downtime. Any pages dirtied during a transfer of memory are re-sent in the next iteration until a sufficient amount of memory pages have been transferred to the destination host. In a final stop-and-copy phase the VM is suspended on the source host, remaining memory pages are copied to the destination and the VM is resumed at the destination.

An important efficiency measure for VM migrations has been identified as the writable working set (WWS) [23]. It is defined as a set of memory pages that are updated very frequently such that it is not worth keeping them consistent at the source and destination hosts. These set of pages should be copied only in the final stop-and-copy phase. An accurate measure of the WWS for the SPEC CINT2000 benchmark is depicted in Figure 2.5.

In this graph the number of 4KB pages of memory dirtied within a corresponding 8 second interval are shown for each of the programs run in the benchmark. It is clearly visible that the number of pages contained within a WWS depends greatly on the behavior of the selected

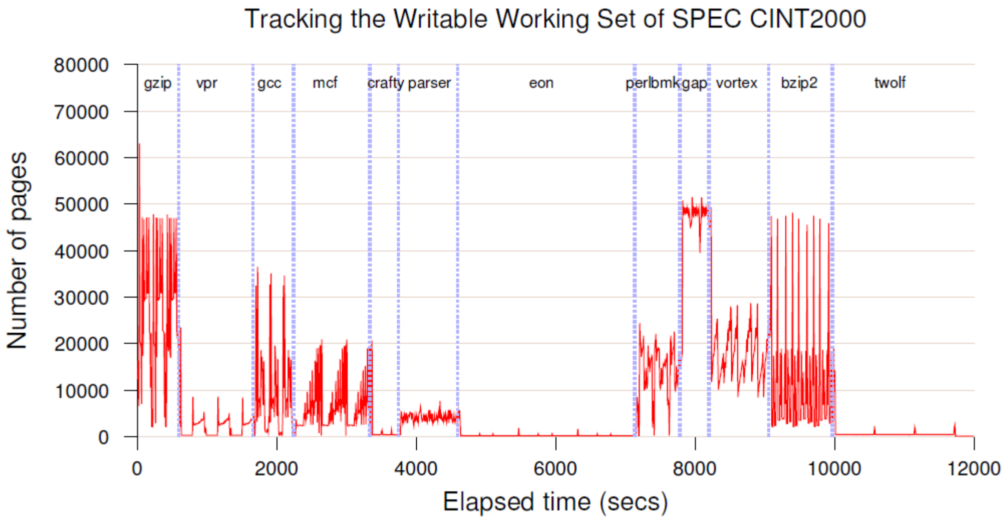


Figure 2.5: WWS curve for a complete run of SPEC CINT2000 (512MB VM) [23]

program. Workloads exhibiting a high amount of frequently updated pages such as *gap* will provide only a small number of pages sensible for inclusion in a pre-copying phase. Thus the downtime and cost resulting from a VM migration depends to a great extent on the type of workload being migrated and the exact moment of the start of the migration.

Results for different kind of workloads indicate that even in the worst case the pre-copy approach significantly reduces downtime compared to the stop-and-copy method. As bandwidth has a direct impact on migration downtime performance can be further increased by raising the bandwidth assigned to the VM migration.

Migration downtimes have been evaluated for different sized VMs running various kinds of workloads. Resulting downtimes range from only 60 ms for small sized VMs (64 MB) and low dirty page rate to 3.5 seconds for large VMs (512 MB) with frequently updated dirty page rates. It is stated that realistic workloads can be migrated with a downtime as low as 210 ms in a virtualized cluster environment.

Performance and energy modeling for live Migration of Virtual Machines

A thorough evaluation of migration performance and resulting migration costs is given by [58].

The authors provide methods to quantitatively predict migration performance and resulting energy cost. A refined pre-copy approach is used to efficiently transfer memory pages in iterative rounds where in each round the memory pages dirtied in the previous round are re-sent (Figure 2.6).

The pre-copying phase is terminated when any of the following conditions are satisfied:

1) memory dirtying rate exceeds memory transmission rate

2) the remaining dirty memory falls below a predefined threshold

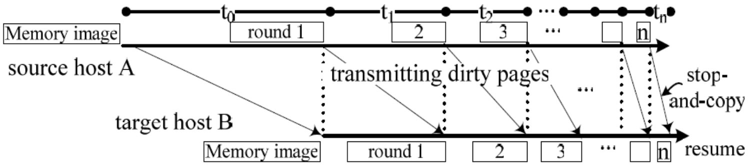


Figure 2.6: Live migration algorithm performs pre-copying in iterative rounds [58]

3) the number of pre-copying iterations exceeds a defined maximum
4) total network traffic exceeds a multiple of the VM memory size

Findings in this work include that with incremental data transmission rate (bandwidth) the power consumption progressively increases while the migration latency decreases. Also, power consumption during migrations increase for both source and destination hosts from which the resulting migration energy can be derived. In addition the migration load and time needed for each pre-copy iteration as well as the total migration load and latency are calculated.

Besides the defined base model a refined migration model is specified which takes into account the size of a workload's writable working set, that is the number of pages dirtied most frequently. The latter model has been empirically evaluated by measuring the WWS on a Xen virtual machine and deriving the maximum sensible pre-copy iteration while continuously checking termination conditions.

Workloads with different sizes for writable working sets have been evaluated in simulations including two hosts running 8 virtual machines. The resulting migration downtime, migration latency, network traffic and energy consumption could be reduced by 72.9%, 93.5%, 74.5% and 73.6% compared to a random selection technique, respectively (Figure 2.7).

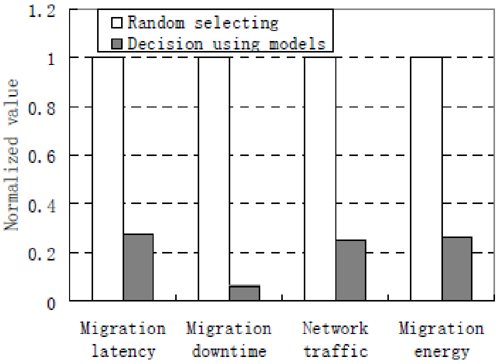


Figure 2.7: Migration cost savings by using proposed models [58]

CHAPTER 3

Data Analysis

3.1 Day ahead and real time energy prices

Real time and day ahead energy prices are traded within deregulated wholesale power markets for the current or the following day. Thus day ahead and real time markets are classified as short-term markets within deregulated power markets [47].

In day ahead or spot markets bids are placed for each consecutive hour of the following day. Prices may change substantially from one hour to the next, up to a factor of ten [48, 100]. Therefore market operators need to apply profound risk management techniques to alleviate such price spikes.

The real time market, also called intraday or balancing market on the other hand is used to compensate demand variations during the day which may result in additional changes in energy prices [9]. Due to this characteristic the real time market prices exhibit an even greater volatility than those of the day-ahead market.

A direct comparison of day ahead and real time prices is depicted in Figure 3.1.

What can be noted is that both day ahead and real time prices show a clear trend over one day denoting a daily seasonality pattern, with real time prices exhibiting more variation during the day. Thus prices start to rise at 5 a.m. and start falling again at around 5 p.m. This pattern is highly related to energy demand which reaches its peak during the day mainly due to industrial power demands.

Price time series spanning a whole month can be seen in Figure 3.2. It shows day ahead and real time markets (top to bottom) from Portland, Maine belonging to the ISO New England power market which is located in Northeast USA. It covers the whole month of August 2015 to display energy price characteristics over an extended time range. Some characteristics of energy prices can be spotted in both day ahead and real time markets such as daily and weekly seasonality, trends, price volatility and price spikes (see Section 3.2).

Daily and weekly seasonality are most evident in day ahead markets. Apart from the daily variations of energy prices a weekly seasonality is clearly visible where prices are higher at the start of a week and decline towards the end of the week. This may be explained by the fact that

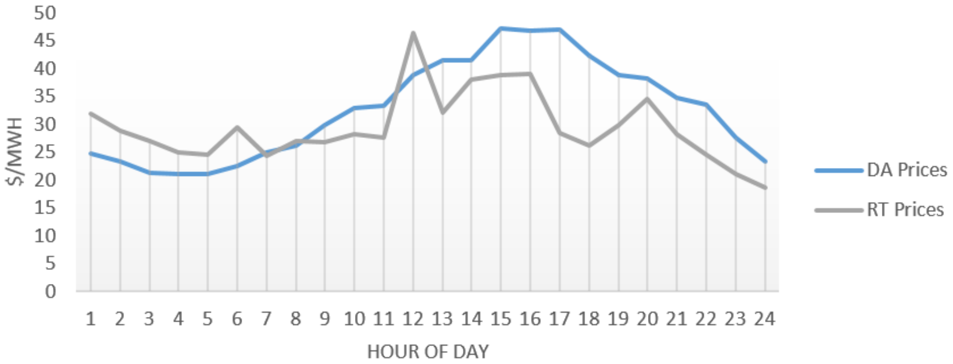


Figure 3.1: Day ahead vs real time energy prices, Maine 4th Aug, 2015

power demand is most present at the start of the week and is usually decreased on weekends due to the weekly lifecycle of industrial facilities and established companies. Even though the real time market shows much higher volatility in energy price levels the daily and weekly seasonality are still present.

Price trends can be observed during the second week of this investigation where prices continuously rise and hit a peak at the beginning of the third week. Weekly seasonality might therefore be masked by the increasing trend however it can still be spotted as the price peak of the second day is surpassed by the one of the previous day. In the third and fourth weeks prices continue to decline to return to their former levels.

Price volatility is most evident in the real time market where variations in energy price levels is more common than in the corresponding day ahead market. Seasonality is less clear and prices tend to change their levels more often and to a greater extent than those at the day ahead market. This can be observed in the middle of the third week where prices experience a sudden drop in price level and variation.

An important characteristic of energy prices in deregulated markets are the so called *price spikes* [103]. A price spike is defined as an extreme price increase (more than three times the standard deviation of the price series) from one observation time stamp to the next where the price tends to rapidly return to its normal level again. A price spike appears e.g. when there is a sudden surge in power demand or a power outage occurred. In the lower part of Figure 3.2 the real time market shows several price spikes. Major spikes can be observed on the 15th, 18th, 19th, 24th and 25th of August. As the spikes reached maximum values of up to $ 700 the range of the y axis has been limited to be able to accurately observe relevant price characteristics.

Another more subtle characteristic of energy prices in wholesale power markets is the occurrence of negative prices. This phenomenon can occur when an excess of electricity is produced which cannot be easily distributed to consumers as electricity demand is low. In this situation market participants are actually paid to get electricity from the market in order to stabilize supply and demand. As electricity is not storable at a larger scale it has to be consumed as soon as it is produced. In Figure 3.2 a negative price occurs at the end of the fourth week on the 28th

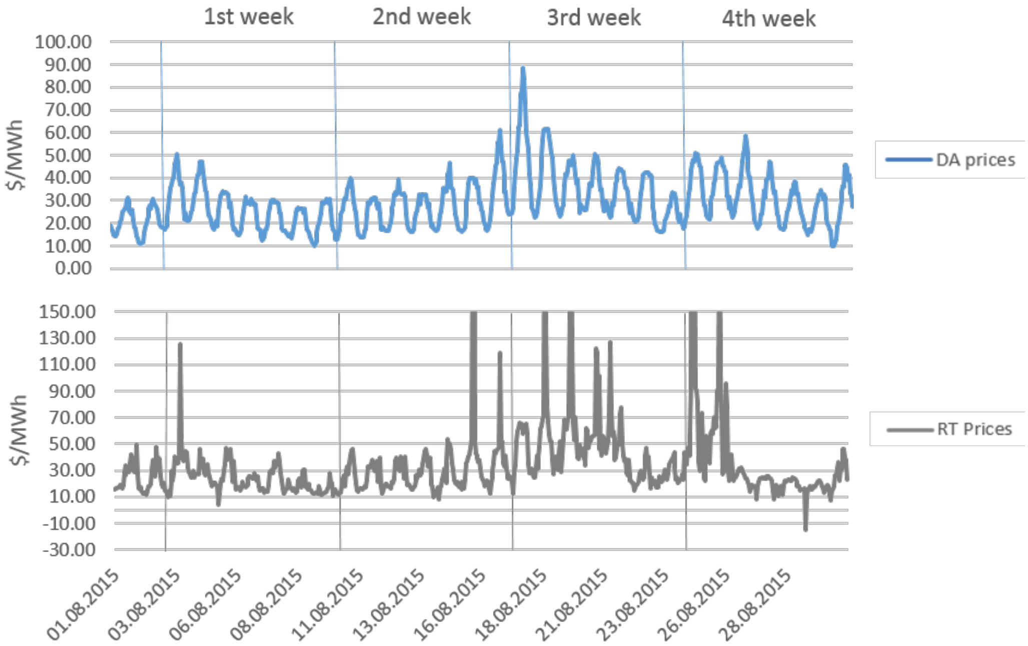


Figure 3.2: Day ahead and real time prices, ISO-NE, ME

of August where prices and demand tend to be low which finally leads to a small negative price spike.

3.2 Characteristics of energy markets

In this section different studies of power market characteristics are presented to distinguish features that may be used in building forecasting models.

The main differences between electricity power markets and other financial markets are price volatility, mean reversion and price jumps or „spikes“ [97, 99, 103]. In addition to these characteristic features electricity prices may exhibit strong seasonality patterns on a daily, weekly and annual basis [103].

Volatility and price variations are inherent characteristics of deregulated power markets as energy cannot be stored but has to be delivered immediately. This results in constant balancing of electricity demand vs. supply to ensure the stability of the electrical grid which has a direct impact on energy prices. When building energy price models the impact of price variations can be reduced by applying logarithmic transformations to input data or utilizing Box Cox transformations [14, 101].

Energy prices in general exhibit strong mean reversion which denotes the characteristic that prices tend to return to their mean levels after a significant deviation from the mean [97, 103]. To discover mean reversion or long range dependencies in time series different models can be utilized of which the most important are rescaled range analysis, Detrended Fluctuation Analysis, periodogram regression and Average Wavelet Coefficient [97]. By applying these methods to electricity price data strong mean reversion could be detected in contrast to typical financial stock indices where this behavior can not be recognized.

Price jumps or spikes are a distinctive feature of electricity markets due to the non-storable nature of electricity prices and occasional gaps in supply vs. demand caused by transmission congestion, plant failure or unanticipated high power consumption [97, 103]. Considering price spikes is critical as price values may increase tenfold from one hour to the next [48]. These phenomenons are called spikes as prices tend to rapidly return to their previous level after the sudden increase. In order to model price spikes a combination of mean reversion and „downward jumps“ can be applied. Another possibility is to model positive and negative jumps through the application of *jump diffusion models* [100]. These kind of models are able to accurately model price spikes.

The actual root cause of price spikes lies in the applied bidding strategy (Section 3.2) [97]. Since electricity is an essential commodity for most market participants they are willing to pay almost any price to ensure a continuous supply of electric power. Thus bids on demand side are regularly placed at the highest possible price (price cap) which results in suppliers raising their bids as well since they are aware of these bidding strategies. Since bids are placed for all 24 hours of the following day market participants have to stick with high priced periods for at least one day before they can choose to search for cheaper alternatives.

An outline of a possible implementation of a competitive energy market is depicted in [47] which is still valid as of today in many respects. A competitive market captures the advantage of market participants being able to actively shaping the final clearing price applicable to both producers and consumers in the market. Thus the market price more accurately resembles actual market conditions in contrast to a defined price dictated in regulated markets.

Electricity market models

Two main models exist for the exchange and types of trading of energy prices in power markets which are bilateral trading and electricity pooling, also called loose pool and tight pool models [9, 22, 46, 73].

Bilateral trading

In bilateral trading or loose pool models utility operators (producers) and energy consumers establish bilateral contracts to determine the terms and conditions applied for trading energy [22, 73]. Generators or producers may also buy electricity from other suppliers in case their own electricity generation does not fit current demand. The exact amount, price and time when tradings take place are negotiated on the contract. Since electricity demand may vary from the negotiated terms the resulting imbalances have to be taken care of by the system operator.

Electricity pooling

Electricity pooling in tight pool models provide a way for utility operators to offer a certain amount of energy at a set price. Each generator places a bid containing the quantity and expected price [9]. In return consumers place bids on how much they are willing to pay for a given amount of energy. The intersection of the aggregated demand and supply curves defines the energy price for that hour. Customers, brokers and aggregators have the choice to either establish long term contracts of electricity in bilateral contracts or rely on the short-term market [46].

Forward Capacity Market (FCM)

In contrast to day ahead and real time markets which are classified as short term markets (Section 3.1) other types of markets exist for mid- and long term planning to ensure sufficient capacities over an extended period of time [9, 72, 83]. One of the most important of these markets is the Forward Capacity Market which is introduced shortly.

A forward capacity market is used to ensure having enough capacity for a specific amount of time into the future [42]. A capacity commitment period (CCP) ensures that a certain amount of capacity will be available during that period. For example, the CCP at ISO-NE is set as one year ranging from June 1st until May 31st.

The market has to ensure that the *Installed Capacity Requirements (ICR)* for the corresponding region are met. These requirements are defined for each capacity commitment period and define the amount of capacity needed to meet estimated peak and reserve demands. Important measures to define the ICR are the local sourcing requirements (LSR) and maximum capacity limits (MCL) that define the constraints given by market participants.

The actual procurement of resources for each capacity commitment period is determined by a *Forward Capacity Auction (FCA)* which aims to meet the defined ICRs for the given period. This auction is carried out three years ahead of the related CCP to ensure that enough resources will be available during that period. *Reconfiguration Auctions (RA)* are then executed annually until the start of the CCP and continued monthly afterward. During an RA capacity resources may be amended to adapt to potential changes in capacity zones [42].

At the FCM of ISO New England there is also the option of making a composite offer where different capacity resources may join their capacity offers (useful e.g. in case of single season capacities) to result in a single resource offer at the market.

Power pools vs power exchanges

Two types of power markets have been established, power pools and power exchanges. The main difference between those types is the type of auction that is applied. In power pools one sided auctions are conducted whereas in power exchanges two sided auctions are performed [97]. The difference lies in the fact whether demand bids are conducted for each hour of the following day or power demand is estimated based on current demand in the market.

The market clearing price (MCP) is the final settled price that results from demand and/or supply bids for a specific hour of the following day. This price can be taken as reference price for the corresponding hour that is to be paid by consumers. The market clearing volume (MCV) is the estimated energy demand (volume) for the corresponding hour that is to be generated by suppliers [97].

For one sided auctions power demand is estimated for all consumers for one specific hour which results in the market clearing volume. The market clearing price is determined by the

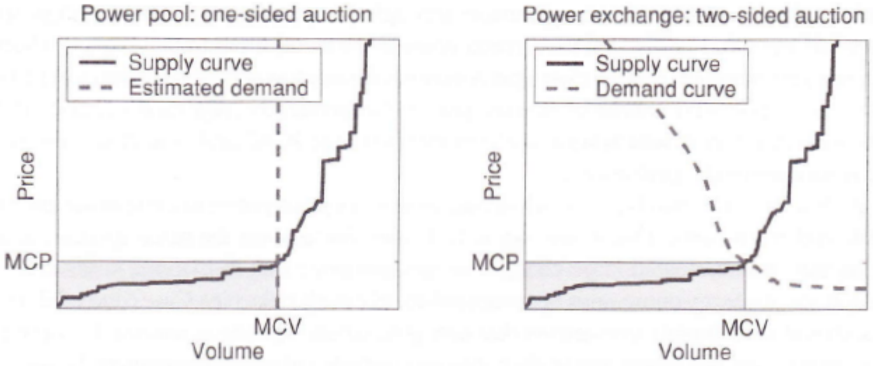


Figure 3.3: One sided and two sided auctions in power markets [97]

intersection between the supply curve (consisting of aggregated supply bids) and the market clearing volume. For two sided auctions the demand curve is established by aggregated demand bids and the MCP is determined by the intersection between the supply and demand curves (Figure 3.3) [97].

The sudden increase in slope of the supply price curve indicates that when market volume reaches a certain threshold a significantly higher price has to be paid due to the activation of more expensive power generation utilities. For example, a market area may depend on renewable energy (hydro generation utilities, wind turbines) for times with low electricity demand, however additional gas and coal power plants may need to be activated when electricity demand increases.

All power markets considered in this work are part of the category *power exchanges* and thus conduct two sided auctions. The following markets have been investigated: Nord Pool Spot, EPEXSpot, Belpex, ISO New England, PJM (Pennsylvania, New Jersey, Maryland). Despite its name, *Nord Pool Spot* is not a power pool but is considered a power exchange.

Bidding strategies

In deregulated power markets two bidding strategies have evolved over time, pay-as-bid and uniform pricing [93, 97]. In [93] it is discussed why pay-as-bid models are not necessarily more cost effective than uniform pricing models even though they seem to offer greater benefits at the first glance.

Pay-as-bid model

In a pay-as-bid auction suppliers place bids according to the desired compensation ($ per MWh) for quantities sold in the market [93]. Suppliers are paid exactly the amount they bid for the quantity transacted, there is no uniform market price which applies to all market participants. Therefore suppliers aim to maximize their revenue by placing bids more closely to the expected average bid price such that they „win“ a greater amount of offers. An offer is won when a bid is placed at or below the offer price of the last bidder whose supplies are needed to meet customer demand.

Uniform pricing model

In contrast to a pay-as-bid model the uniform pricing model demands suppliers to place bids that reflect their actual marginal costs rather than a desired compensation [93]. Supply bids are ranked according to their bid prices, such that supplies exhibiting the least costs are considered first. Suppliers are then dispatched in this order until the total customer demand is met. The supply bid of the last considered supplier is set as the uniform market clearing price that has to be paid by all market participants which placed demand bids at or above the market clearing price.

Electricity price characteristics in European Power Markets

In [67] different European power markets have been investigated to reveal major differences in energy price behaviour. The EEX, Nord Pool Spot and Polish power markets have been evaluated whereby the markets are responsible for the Mid-Europe, Northern Europe and Polish regions respectively.

As electricity prices depend on energy demand which changes due to climate conditions (temperature and number of daylight hours) electricity prices exhibit a seasonal component as well (Figure 3.4) [101].

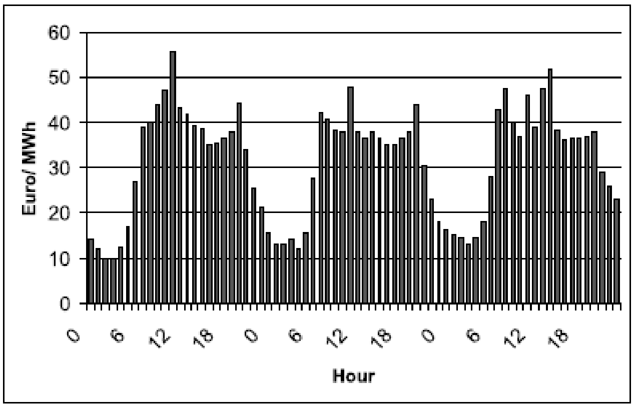


Figure 3.4: EEX - hourly spot prices [67]

The daily seasonality for hourly spot prices at EEX (Figure 3.4) can be clearly spotted where prices start rising at 6 a.m. to 8 a.m. and begin to fall at around 10 p.m. Daily variations of prices are caused by reduced electricity demand and power consumption at nights.

Even though price seasonality and trend seem to be stable over a short time range (Figure 3.4) they can show significant fluctuations over a longer time range. In [67] data related to each examined energy market was fitted to a stable Paretian distribution as well as a normal distribution. The result showed that mature markets as EEX or Nord Pool Spot exhibit high volatility, heavy tails, high kurtosis and asymmetrics in the energy price data which was best modeled by the Paretian distribution. In contrast the Gielda Energii SA market in Poland shows a much more stable energy price behavior which can be modeled by a Gaussian distribution.

The variation in energy price levels for the two power markets EEX and Nord Pool Spot are shown in Figures 3.5 and 3.6.

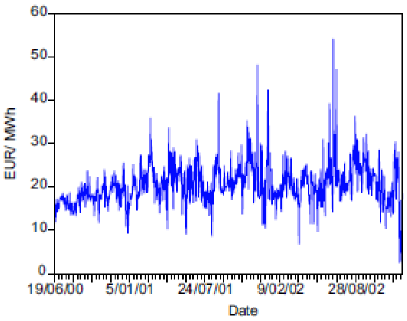


Figure 3.5: EEX levels [67]

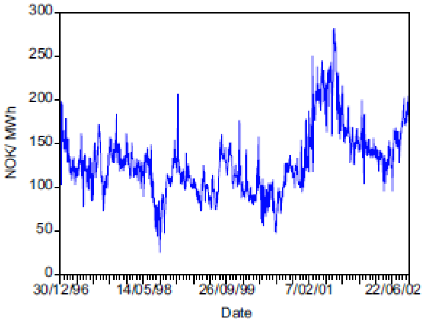


Figure 3.6: Nord Pool levels [67]

By comparing these graphs by scale over a longer period of time (150 NOK equal to about 20 Euros in 2002) it is clearly visible that both energy price levels and amount of price variation can be very different across power markets.

3.3 Energy price case study

In this section the aforementioned characteristics of electricity prices in power markets will be investigated for both day ahead and real time energy price data over an extended period of time.

Evaluation of electricity market price characteristics

Electricity price data used in this thesis is taken from five different energy markets (Table 3.1).

Energy market	Type of data used	Description
Nord Pool Spot (NPS) [7]	Hourly DA (FI)	A power market operating in the region of Northern Europe including Finland, Sweden, Norway and the baltic states
EPEX Spot [87]	Hourly DA (DE)	A market responsible for countries in mid-Europe (Germany, Austria, France and Switzerland)
Belpex [11]	Hourly DA (BE)	Belgian energy market operating in the Belgian region
ISO New England (ISO NE) [50]	Hourly DA and RT (ME, MA)	As the name implies this market operates in the New England region of North-East USA
PJM (Pennsylvania, New Jersey, Maryland) [52]	Hourly RT (VA, MI, PA, WI, OH)	A real time market spanning countries around those included in the acronym

Table 3.1: List of selected energy markets

All of the energy markets listed in above table provide hourly day ahead data (?). In addition, ISO New England and PJM provide extended real time energy price data which can be retrieved from their website.

Through display of energy price time series over an extended time range the general behavior of different energy markets can be observed. Day ahead and real time energy price time series have been collected for the years 2012 to 2014 which are displayed in figures 3.7 and 3.8.

The title displays the name of the energy market followed by the country abbreviation and an indicator whether this market is a day ahead or real time market (DA or RT). Strongest price variations are experienced in real time markets (Figure 3.8) where price spikes of almost 2000 $/MWh occur.

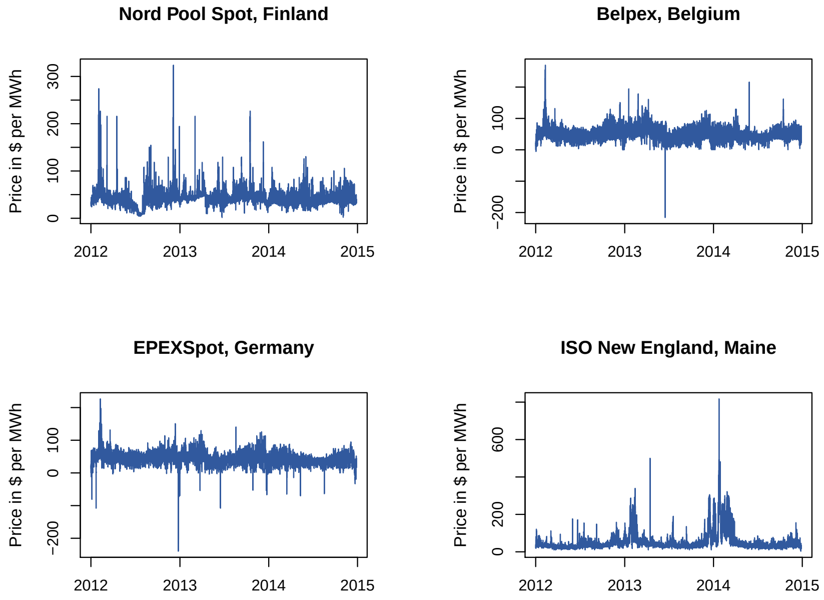


Figure 3.7: Day ahead energy market prices

What can be noted is that markets Belpex and EPEXSpot appear to be the most stable in terms of energy price level and variations except some occasional negative spikes throughout the dataset. The aggregated statistics for each market are outlined in tables 3.2 and 3.3.

The inter quartile range (range between 1st quartile and 3rd quartile) of the market prices lies between 20 $/MWh and 60 $/MWh whereas the minimum and maximum values tend to lie far outside this range. Real time markets exhibit significantly greater values for skewness and kurtosis which indicates a higher probability of extreme price spikes. When comparing maximum price values of real time markets to those of day ahead markets this fact is supported as well. The stable variability of Belpex and EPEXSpot is also emphasized through a low absolute value for the skewness which indicates an almost symmetrical distribution. In contrast, other markets are heavy tailed and experience far more „outliers“ indicated by a higher value for

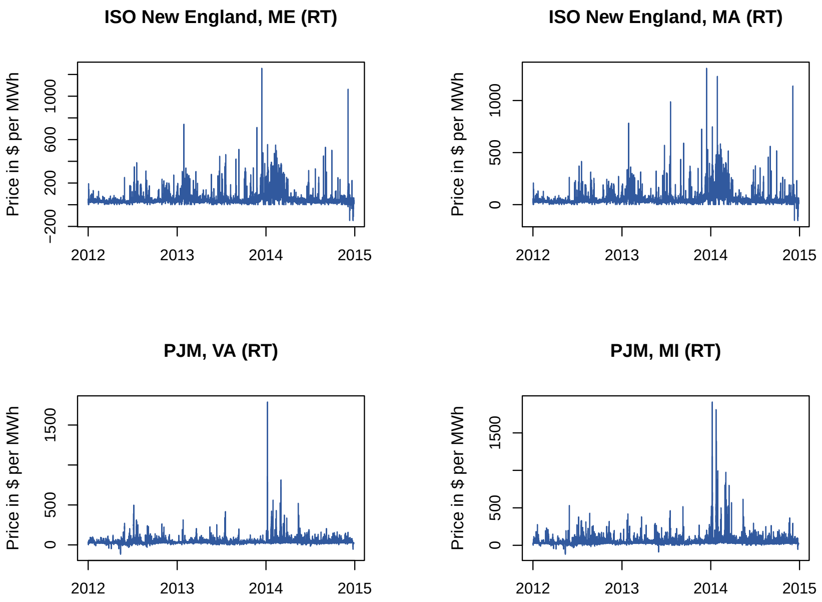


Figure 3.8: Real time energy market prices

skewness.

	Min	1st Qu.	Median	Mean	3rd Qu.	Max	Skew.	Kurt.
NPS, FI (DA)	1.49	33.08	39.18	40.91	46.33	324.00	3.60	40.02
Belpex, BE (DA)	-216.00	37.14	48.60	48.66	59.49	270.00	0.08	13.35
EPEXSpot, DE (DA)	-239.70	31.20	39.34	40.73	51.16	226.80	-1.06	25.29
ISO NE, ME (DA)	3.00	28.94	37.18	50.78	50.41	817.70	3.62	23.62

Table 3.2: Day ahead energy market summary statistics

	Min	1st Qu.	Median	Mean	3rd Qu.	Max	Skew.	Kurt.
ISO NE, ME (RT)	-147.00	26.42	34.83	49.21	48.90	1257.00	4.24	42.21
ISO NE, MA (RT)	-153.90	27.24	35.99	52.23	50.92	1310.00	4.76	48.87
PJM, VA (RT)	-118.60	26.33	30.34	35.85	36.93	1789.00	24.53	996.02
PJM, MI (RT)	-120.80	27.76	32.82	43.83	41.52	1913.00	14.12	316.98

Table 3.3: Real time energy market summary statistics

In order to visualize the aforementioned statistical properties histograms have been generated for each market over the whole time range (Figures 3.9 and 3.10).

As outlined before the Belpex and EPEXSpot markets exhibit the most stable distribution of prices which is visible in the histograms as these markets show the most symmetric distributions. They exhibit an almost gaussian like distribution whereas other markets are clearly heavy tailed to the right. In order to see the distribution of prices more distinctly the histograms where cut off at 250 $/MWh to be able to accurately compare the price distributions.

The previously shown results have major implications for the generation of forecasting models. Various models have different requirements regarding statistical properties of the data, for example statistical time series models demand a constant mean and variance in the dataset to capture the underlying time series characteristics. When observing the presented energy price time series these characteristics can not be found for any of the discussed time series over an extended period of time. Therefore, in order to accurately model energy prices data preprocessing steps might have to be taken before applying statistical models. An in-depth discussion about forecasting models and data preprocessing is provided in Chapter 4.

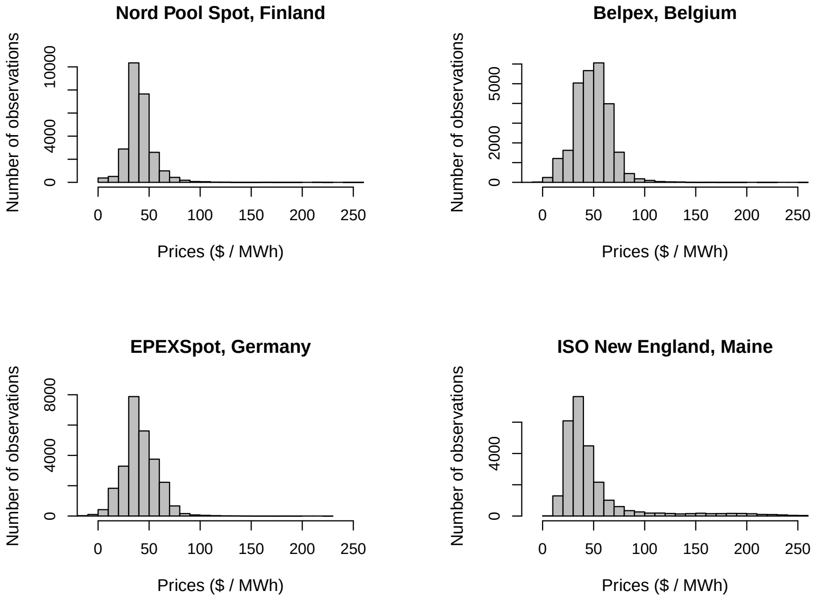


Figure 3.9: Histograms of day ahead markets for years 2012 to 2014

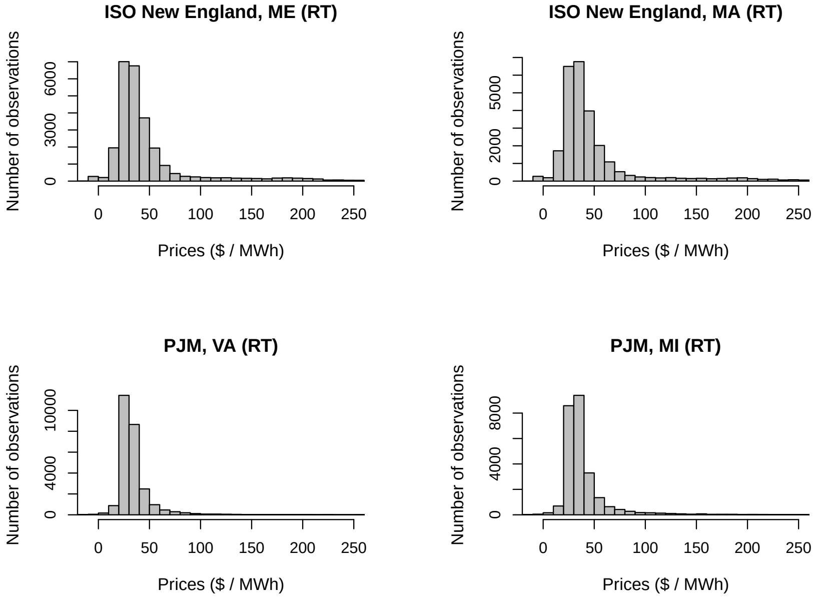


Figure 3.10: Histograms of real time markets for years 2012 to 2014

CHAPTER 4

Forecasting

4.1 Introduction

As outlined in 2.1 different types of models have been investigated for forecasting spot electricity prices. In this work forecasting models should be used that exhibit the following characteristics:

- computationally lightweight
- capable of modeling seasonality
- capable of recognizing trends in the dataset
- good out-of-sample accuracy based on day ahead and real time electricity prices
- capable of being part of an automated model generation process
- accurate modeling of different energy price characteristics

Statistical models have been proven to provide good results and at the same time keeping the computational effort low [2, 17]. In contrast structural or non-parametric models require additional input data and perform more complex operations such that they might not be the optimal choice for dynamic forecast environments. Therefore models in this work have been chosen from the well known set of parsimonious stochastic models.

Different stochastic models are able to generate forecasts while considering existing seasonality in the data [30, 43]. Examples of models exhibiting these characteristics are SARIMA, HoltWinters and TBATS, all of which are investigated in this work to evaluate their capability of forecasting on different settings and datasets. The best model is then chosen to provide forecasts for all simulations described later in this work.

In general time series may be decomposed into seasonal, trend and cycle components which can be made use of by forecast models [30]. Some models such as ARIMA and TBATS can handle existing trends in the dataset by applying transformations in a preprocessing step or by decomposing and separately modeling time series components. More simplistic models such as Simple Exponential Smoothing (SES) or pure Autoregressive (AR) models require the dataset to be stationary, i.e. to exhibit a constant mean and variance [85, 97]. In this case the dataset would have to be transformed before applying those models.

Models should provide good out-of-sample accuracy which means that they need to be trained and tested on different datasets [85]. For example, considering training data from a training period of four weeks of energy price data and test data from a subsequent test period of one week a model is trained based on that training data but tested on the provided test data. Thus real life forecasting scenarios can be established and models are evaluated based on the accuracy on test datasets. A large scale evaluation of forecasting models based on different training and test datasets is presented later in this section.

Building an automated model generation process is important for automatic model evaluation and model selection for simulations running over a longer period of time. As extensive simulations including forecasts represent a core contribution of this work an automated model generation process has been defined including preprocessing steps and model evaluation to find the best suited model for a given dataset.

Inclusion of energy price characteristics such as mean reversion and price spikes into forecast model generation has been studied extensively considering different types of forecast models [2, 17, 102]. Mean reversion denotes the characteristic that a time series returns to its mean after significant deviation. This characteristic is modeled by a AR(1) process and can be incorporated e.g. into an ARIMA model. For handling price spikes three approaches have been defined in literature [102]: 1) Using models that allow spikes in input datasets 2) Remove price spikes completely from the data 3) Recognize and dampen observed spikes to mitigate their impact on out-of-sample forecasts. For simplicity reasons price spikes have not been modeled explicitly in this work meaning they are not dampened or removed from the datasets.

4.2 Methodology

In this section the various methodologies used in the forecasting framework are discussed.

Seasonality estimation and periodogram

Estimating seasonality is an important pre-processing step when building forecasting models. As most models are not able to automatically detect seasonality in the given data it is vital to determine possible seasonality cycles beforehand.

One way of detecting seasonality in the data is by doing a spectral analysis for exploration of cyclical patterns. During the process the data is decomposed into underlying sinusoidal (sine and cosine) functions with particular wavelengths [97]. The wavelength is commonly described as frequency which is the number of cycles per unit time.

The frequency ω and the period T have a reciprocal relationship $\omega = \frac{1}{T}$. Thus the period T denotes the number of unit time stamps required to complete one period which in case of daily seasonality in a time series of hourly observations can be 24 time stamps, i.e. 24 hours.

In order to visualize common frequencies a *periodogram* may be generated which can be regarded as a tool for retrieving the most common frequencies from the dataset. As existing seasonality patterns in the data are likely to be detected by the periodogram as high valued frequencies it can be used to extract these frequencies and calculate the reciprocal as the seasonal period T.

The formula of a periodogram for a vector of observations $\{x_1, \dots, x_n\}$ is defined as [97]:

$$I_n(w_k) = \frac{1}{n} \left| \sum_{t=1}^{n} x_t e^{-i(t-1)\omega_k} \right|^2$$

where $\omega_k = 2\pi(k/n)$ denote the Fourier frequencies in radians per unit time, $k = 1, \dots, [n/2]$ and $[x]$ describes the largest integer value less than or equal to x.

Figure 4.1 shows a periodogram of two weeks of hourly day ahead prices from the Nord Pool Spot power market. On the x-axis frequencies from 0 to 0.5 are displayed, on the y-axis each frequency's corresponding value is depicted, proportional to the number of occurrences of this frequency. In case of a random signal the frequencies should be uniformly distributed across the frequency scale. In this case clearly two frequency values stand out where the values are of magnitudes $\omega_1 = 0.0416$ and $\omega_2 = 0.0833$. This results in periods $T_1 = 24$ and $T_2 = 12$, which means that the underlying series exhibits a strong daily seasonality of 24 hourly prices with a so called *harmonic* of 12 periods which denotes a multiple of the 24 hour period.

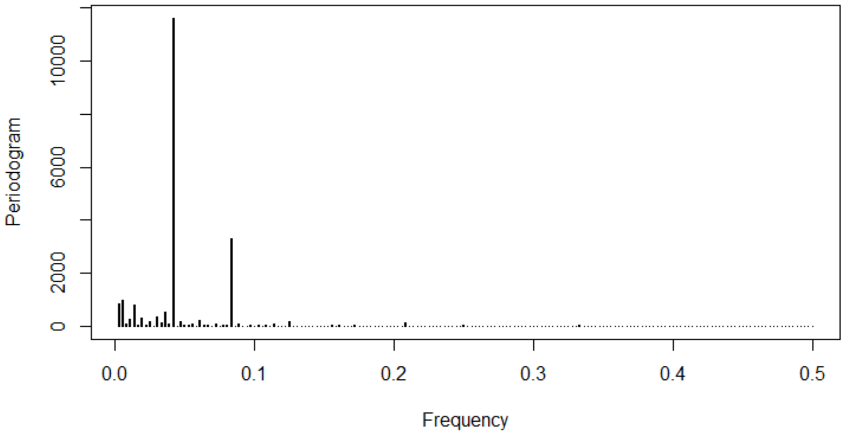


Figure 4.1: Periodogram of hourly day ahead prices of Nord Pool Spot, Helsinki from July 7th to July 21st in 2014

After successfully determine meaningful seasonalities and their resulting periods from the dataset this information can be used to build forecast models that consider the respective seasonal periods in the model generation process.

Seasonal decomposition

Seasonal decomposition describes the process of decomposing a given time series into its components which are trend, seasonal or cyclic and irregular components. Decomposition is done by applying the STL model to the time series and extracting the above mentioned components where STL is defined as Seasonal-Trend decomposition procedure based on Loess [24].

The decomposition of a time series can be formalized as follows [24]:

$$Y_v = T_v + S_v + R_v$$

where Y_v denotes the original time series, T_v the trend component, S_v the seasonal component and R_v the remainder component. $v \in \{1, \ldots, N\}$ denotes an individual time stamp in the series where N is the number of unit time stamps in the time series.

A sample seasonal decomposition of hourly time series is depicted in Figure 4.2.

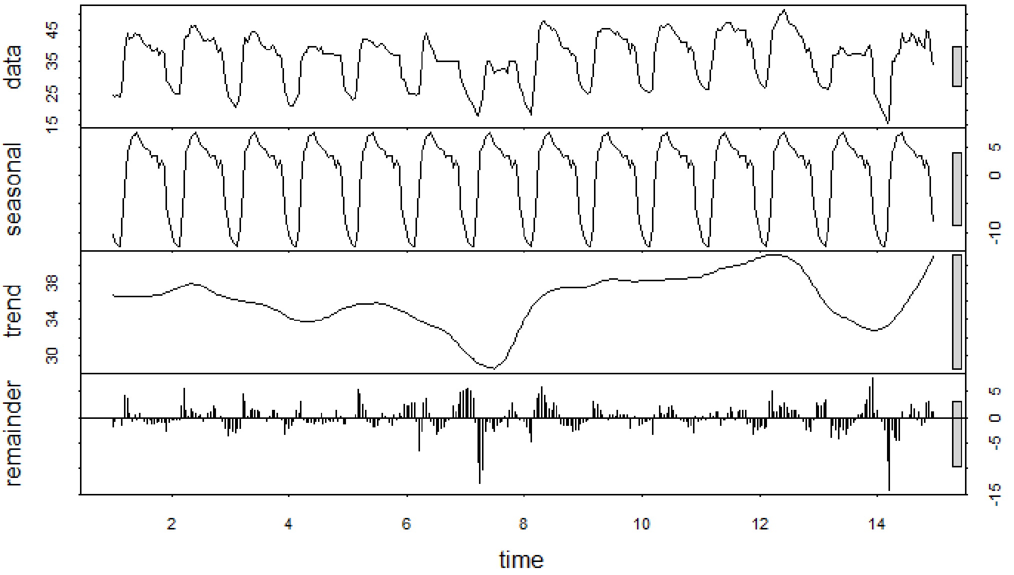


Figure 4.2: STL decomposition of hourly day ahead time series, Nord Pool Spot, Helsinki from July 7th to July 21st in 2014

The plot consists of four rows with each row containing a different time series component. The first row shows the original time series with hourly prices over a time period of two weeks. There are evident daily and weekly seasonal periods visible in the time series. Daily seasonality can be observed by the repetitive pattern of highs and lows over each day whereas weekly seasonality is evident due to the recognizable decline of prices towards the end of each week.

The second row represents the extracted daily seasonal component which depicts the daily variation in prices. STL is only capable of extracting a single seasonality pattern. In case both daily and weekly seasonal periods should be extracted a BATS or TBATS model can be used (these models are discussed later in this section).

In the third row the trend component is displayed. It shows a seemingly repetitive pattern over one week, which denotes the weekly seasonal period observable from the original time series.

Finally in the last row the remaining irregular component is shown which contains all remaining variations in the time series.

The grey bars at the right hand side of each plot aim to provide a relative comparison of scales over all plots, i.e. the height of the bars show the same value range in different scales. Thus the whole value range of the trend component is equivalent with the value range of about one third of the original time series.

Forecast models

Different statistical forecast models have been chosen to be investigated on a large scale. The selection of forecast models includes simpler models such as Simple Exponential Smoothing and more advanced models such as ARIMA and TBATS and variants thereof. The purpose of the evaluation is to provide insights into the performance of different forecast models when applied to electricity price time series.

Mean forecast

The mean forecast is a simple model based on previous observations of a time series. It calculates the mean over a training data set and uses this mean as forecast for all future values [85].

It can be formally defined as in equation 4.1

$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{i=1}^{T} y_i \tag{4.1}$$

where T denotes the number of observations in the training dataset, y_i denote historical values and $\hat{y}_{T+h|T}$ denotes a forecast of h values into the future beginning at the value of y_T.

Even though the mean forecast only includes simple calculations it can be effective to forecast random or near random time series or time series that exhibit strong volatility. As it is the simplest of the proposed models it should serve as a baseline model to which other models' performance can be compared.

Simple Exponential Smoothing

The Simple Exponential Smoothing model (SES) belongs to the category of exponential smoothing models which calculate the weighted average over historical observations where weights are exponentially decreased for observations in the past.

As the name implies this model is the simplest of exponential smoothing models which is designed to provide forecast for time series without trend or seasonal components. Similar to the mean forecast model past values of a time series are processed but with given weights that decrease exponentially over time [85, 97].

The corresponding formula is depicted in Equation 4.2

$$\hat{y}_{T+1|T} = \sum_{i=0}^{T-1} \alpha(1-\alpha)^i y_{T-i} \tag{4.2}$$

where α with $0 \leq \alpha \leq 1$ denotes the smoothing parameter that determines the influence of past observations through setting the corresponding weights. Thus by modifying values of α the resulting impact of past values to forecasted values can be defined.

Holt's Exponential Smoothing

Holt's Exponential Smoothing (Holt's ES) is a linear trend method which extends Simple Exponential Smoothing by including trends in the model generation process [85]. The model consists of two separately modeled components which are level and trend components. These are combined to obtain forecasts for a given time series. It can be formalized by the following equations:

$$\hat{y}_{t+h|t} = l_t + hb_t \tag{4.3}$$

$$l_t = \alpha y_t + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4.4}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{4.5}$$

Equation 4.3 denotes the forecast value $\hat{y}_{t+h|t}$ which is forecasted h steps into the future beginning from time stamp t. It consists of a combination of the level l_t at time t and the trend b_t at time t continued over the next h time intervals.

The level component is described in Equation 4.4 which denotes a linear combination of the actual value y_t at time t and the level forecast constructed from previous time stamp's level l_{t-1} and trend b_{t-1} components.

Equation 4.5 depicts the trend component which is a linear combination of the difference of the current and previous levels (l_t, l_{t-1}) and the estimated trend component b_{t-1} from the previous point in time.

α and β exhibit characteristics $0 \leq \alpha \leq 1$ and $0 \leq \beta \leq 1$ and denote the smoothing parameters for the level and trend components, respectively.

Seasonal HoltWinter's model

The Seasonal HoltWinter's model is the first of the discussed models which is capable of modeling seasonality in the dataset. It further extends the capabilities of the SES and Holt's ES models by an additional seasonal component. Thus besides the level and trend components l_t and b_t a seasonal component s_t is introduced, with smoothing parameters α, β, γ, respectively [85].

The corresponding equations are defined as

$$\hat{y}_{t+h|t} = l_t + hb_t + s_{t-m+h_m^+} \tag{4.6}$$

$$l_t = \alpha(y_t - s_{t-m}) + (1-\alpha)(l_{t-1} + b_{t-1}) \tag{4.7}$$

$$b_t = \beta(l_t - l_{t-1}) + (1-\beta)b_{t-1} \tag{4.8}$$

$$s_t = \gamma(y_t - l_{t-1} - b_{t-1}) + (1-\gamma)s_{t-m} \tag{4.9}$$

where m denotes the period of seasonality (e.g. $m = 12$ for monthly data when one year is the base unit) and $h_m^+ = \lfloor (h-1) \mod m \rfloor + 1$ represents the additional number of steps ahead required to model the corresponding seasonal period m. Therefore $s_{t-m+h_m^+}$ denotes the previously occurred seasonal period which is added to the forecast equation in Equation 4.6.

The level component in Equation 4.7 is adjusted by the level of the last occurred seasonal period s_{t-m} to consider seasonal differences. The equation for the trend component b_t is taken directly from the one of Holt's ES in Equation 4.5 whereas the seasonal component is defined in Equation 4.9.

The seasonal component is denoted as a weighted sum of the current value y_t substracted by previous level and trend components and the value of the seasonal component exactly m periods before.

The smoothing parameters α, β and γ are estimated by minimizing the squared one-step prediction error to appropriately weigh the different components to yield best results [92].

ARIMA models

ARIMA models or Auto Regressive Integrated Moving Average models are highly adjustable time series models that can incorporate a number of features from the dataset. They are capable of modeling correlations in the data as well as provide a smoothing method to dampen the impact of extreme outliers.

ARIMA models may consist of both autoregressive (AR) and moving average (MA) terms to accurately model the underlying dataset [85, 97].

AR model The autoregressive component is used to model dependencies in the data such that correlations within the dataset are reduced. It is defined as a weighted sum of past values of an observation. An AR(p) model is shown in Equation 4.10.

$$y_t = c + \sum_{i=1}^{p} \phi_i y_{t-i} + e_t \tag{4.10}$$

where y_t is modeled as the sum of the past p values of y plus an additional error component e_t and a constant c. AR coefficients $\phi_1, \ldots, \phi_p$ are used to weight past observations and need to be estimated in the model generation process.

MA model The moving average component models a time series as a moving average of past error terms. Equation 4.11 gives the formal definition:

$$y_t = c + e_t + \sum_{i=1}^{q} \theta_i e_{t-i} \tag{4.11}$$

where e_t denotes the forecast error at time t and the error terms are weighted by MA coefficients $\theta_1, \ldots, \theta_q$.

ARIMA model An ARIMA model consist of both AR and MA terms. In addition ARIMA models are able to model non-stationary time series by applying differencing operations to the dataset. ARIMA(p,d,q) denotes a model consisting of p number of AR terms, q MA terms and d number of differences. It can be described as the sum of AR and MA terms:

$$y_t = \sum_{i=1}^{p} \phi_i y_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t + c \tag{4.12}$$

When applying the model the possible applied differences during data preprocessing are reversed by integrating the results which refers to the *Integrating* part of the model.

ARIMA models can be enhanced to model seasonal periods which are defined as ARIMA $(\mathrm{p,d,q})(\mathrm{P,D,Q})_\mathrm{m}$ also denoted as SARIMA models. In addition to non-seasonal parameters p, d and q the seasonal parameters P, D and Q denote the seasonal AR, differencing and MA components, respectively.

BATS and TBATS models

Multiple seasonal periods can be applied by models such as BATS and TBATS [92]. This can be done by applying a BATS or TBATS model to the time series, identifying different seasonal and trend components and modeling each component separately [30].

Both BATS and TBATS models are able to handle multiple seasonalities in the data, i.e. considering both daily and weekly seasonal periods where TBATS models are capable of modeling non-integer periods as well (e.g. 365.25 for annual seasonality).

BATS model BATS stands for Box-Cox transform, ARMA errors, Trend, and Seasonal components. TBATS can then be regarded as the trigonometric version of BATS.

The definition of BATS is outlined in the following equations:

$$y_t^{(\omega)} = \begin{cases} \frac{y_t^{\omega}-1}{\omega}, & \omega \neq 0 \\ \log y_t, & \omega = 0 \end{cases} \tag{4.13}$$

$$y_t^{(\omega)} = l_{t-1} + \phi b_{t-1} + \sum_{i=1}^{T} s_{t-m_i}^{(i)} + d_t \tag{4.14}$$

$$l_t = l_{t-1} + \phi b_{t-1} + \alpha d_t \tag{4.15}$$

$$b_t = (1-\phi)b + \phi b_{t-1} + \beta d_t \tag{4.16}$$

$$s_t^{(i)} = s_{t-m_i}^{(i)} + \gamma_i d_t \tag{4.17}$$

$$d_t = \sum_{i=1}^{p} \Phi_i d_{t-i} + \sum_{i=1}^{q} \theta_i e_{t-i} + e_t \tag{4.18}$$

where $y_t^{(\omega)}$ denotes the Box-Cox transformed value with parameter ω where y_t is the current value at time t (See also Box-Cox transform in 4.2).

l_t describes the level, b denotes the long term trend, b_t represents the trend, e_t is gaussian white noise with zero mean and constant variance and $s_t^{(i)}$ denotes the i-th seasonal component. ϕ denotes the damping parameter for the trend and defines the impact of short and long term trends. All of these components are adjusted by a weighted ARMA component d_t and coefficients α, β and γ, respectively.

A BATS model is parameterized by BATS(ω,ϕ,p,q,m_1,...,m_T) with the Box-Cox parameter ω, damping parameter ϕ, ARMA parameters p and q and seasonal periods $m_1, \ldots, m_T$. A double seasonal model (e.g. daily and weekly) with an AR(1) component can be described as BATS(1,1,1,0,m_1,m_2) with default Box-Cox and damping parameters.

TBATS model As BATS models possibly result in a large number of states and only pure integer seasonal periods may be modeled, the TBATS model has been developed [30]. This model enhances the BATS model with trigonometric expressions:

$$s_t^{(i)} = \sum_{j=1}^{k_i} s_{j,t}^{(i)} \tag{4.19}$$

$$s_{j,t}^{(i)} = s_{j,t-1}^{(i)} \cos \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \sin \lambda_j^{(i)} + \gamma_1^{(i)} d_t \tag{4.20}$$

$$s_{j,t}^{*(i)} = -s_{j,t-1} \sin \lambda_j^{(i)} + s_{j,t-1}^{*(i)} \cos \lambda_j^{(i)} + \gamma_2^{(i)} d_t \tag{4.21}$$

where $s_{j,t}^{(i)}$ describes the stochastic level of the i-th seasonal component and $s_{j,t}^{*(i)}$ represents the stochastic growth in the level of the i-th seasonal component which can be described as the change of value of the seasonal component over a period of time. k_i is the number of harmonics required for the ith seasonal component, $\gamma_1^{(i)}$ and $\gamma_2^{(i)}$ are smoothing parameters and $\lambda_j^{(i)} = \frac{2\pi j}{m_i}$ the trigonometric parameter which depends on the corresponding seasonal period m_i.

Finally the model can be obtained by replacing $s_t^{(i)}$ in equation 4.17 by the trigonometric expression in equation 4.19 and replacing $s_{t-m_i}^{(i)}$ in equation 4.14 by $s_{t-1}^{(i)}$.

Box Cox transformation

Box cox transformations are used to transform non-normal data to exhibit a normal distribution like behavior [14]. Since most statistical time series models (e.g. ARIMA) require data to have constant variance this technique can be used as a preprocessing step before applying data to forecasting models [68].

The formula for box cox transformations is as follows:

$$y^{(\lambda)} = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log y, & \lambda = 0 \end{cases} \tag{4.22}$$

$$y^{(\lambda)} = \begin{cases} \frac{(y+c)^\lambda - 1}{\lambda}, & \lambda \neq 0 \\ \log(y+c), & \lambda = 0 \end{cases} \tag{4.23}$$

where conditions $y > 0$ and $y > -c$ apply to equations 4.22 and 4.23, respectively.

Since the value of λ can be less than one y has to be positive and thus a constant term $c > 0$ is required in case $y < 0$ (equation 4.23).

In [68] the autors show that it might not be always possible to transform a series to show a normal distribution and that the application of the procedure can be costly. Still some improvements to forecasts can be achieved by applying Box-Cox transformation to non-normal distributions as a preprocessing step to model generation.

Forecast accuracy measures

Different forecast accuracy methods can be applied to investigate model performance. These methods are based on estimating the overall forecast error for a given model and forecast horizon [85,97].

The forecast errors are computed as a function of residuals which are defined as the differences of actual values to forecasts [85]:

$$e_i = y_i - \hat{y_i} \tag{4.24}$$

where $\hat{y_t}$ denotes the one step ahead forecast based on a series of past values $\{y_1, \ldots, y_{t-1}\}$.

Scale-dependent error measures

Scale dependent errors are absolute error measures depending on the scale of the examined dataset. Therefore when comparing these types of error measures they should be applied to data showing the same scale.

Mean error The mean error (ME) denotes a measure of the mean value of forecast errors over a given period of time. It can be seen as indication of the symmetry of the forecast error distribution.

$$\frac{1}{n}\sum_{i=1}^{n} \hat{y_i} - y_i \tag{4.25}$$

where n is the number of observations, $\hat{y_i}$ is the forecast for observation i and y_i denotes the actual value for observation i.

Mean absolute error The mean absolute error (MAE) shows the mean of all absolute errors over the forecast horizon where the absolute error is defined by the absolute difference between a value of the dataset and its forecasted value. The mean absolute error provides a means for retrieving a measure proportional to actual forecast errors.

$$\frac{1}{n}\sum_{i=1}^{n} |\hat{y_i} - y_i| \tag{4.26}$$

Root mean squared error The root mean squared error (RMSE) takes the root of the sum of the squared forecast errors which puts more emphasis on possible outliers in residual values.

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2} \tag{4.27}$$

Percentage based error measures

Percentage based error measures have the advantage of providing scale independent measures for comparing forecast errors across different time series. However a significant disadvantage of percentage errors is their inability to handle zero values in time series. Also numerical compuations become unstable when values get closer to zero.

Mean percentage error The mean percentage error (MPE) is dependent on the symmetry of the forecast error distribution as positive and negative value differences might cancel each other out. This can be compared to the mean error which shows the same behavior as a scale dependent measure.

$$\frac{1}{n}\sum_{i=1}^{n}\frac{\hat{y}_i - y_i}{y_i} \tag{4.28}$$

Mean absolute percentage error The mean absolute percentage error (MAPE) shows similar to the mean absolute error the mean of all absolute errors but as percentage error relative to the corresponding actual value.

$$\frac{1}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \tag{4.29}$$

4.3 Model generation

In this section the procedure of generating an ARIMA model is described. Since ARIMA models are classical time series models which showed reasonable results in energy price forecasting [2, 101] they have been investigated in detail to provide a better understanding of the model generation process.

In addition to the well known Box Jenkins approach (see section 2.1) a custom model selection process is proposed for manual ARIMA model selection [85]. This process is outlined in Figure 4.3.

In this case a manual model generation process is conducted. The data based on which a model will be generated together with diagnostic plots (ACF, PACF) is shown in Figure 4.4.

The dataset consists of two weeks of hourly day ahead prices amounting to a total of 336 hours. Figure 4.4 shows three plots, at the top a time series plot showing the energy price time series is depicted, on the bottom left an autocorrelation plot (ACF) is shown and on the bottom

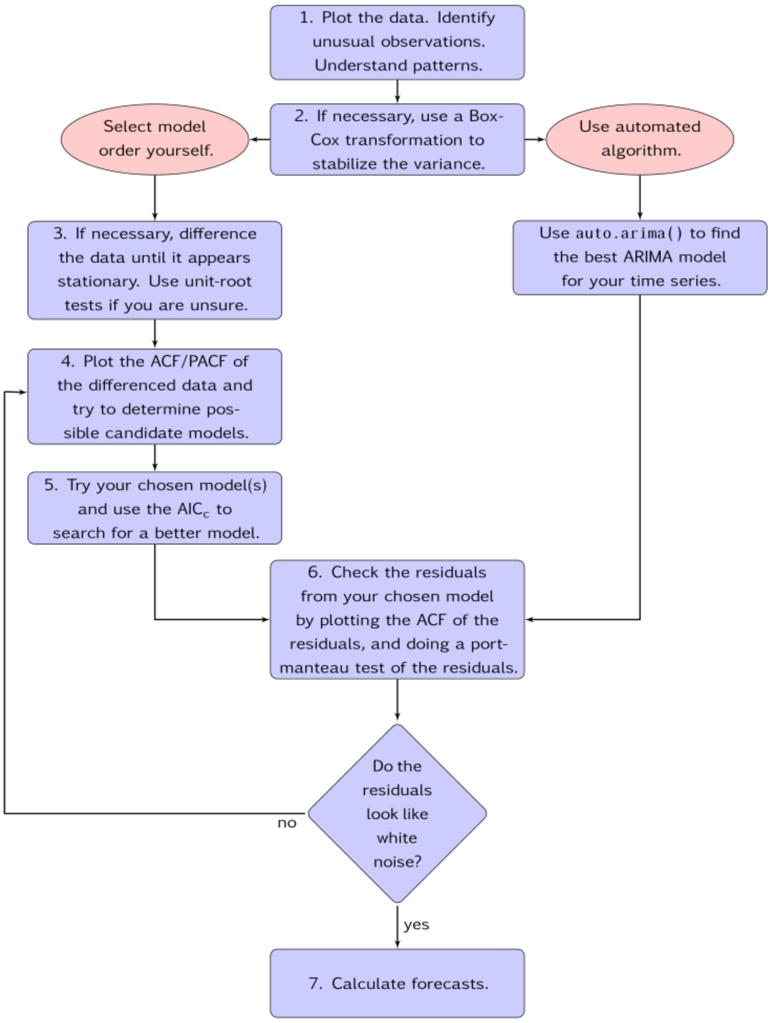

1. Plot the data. Identify unusual observations. Understand patterns.
2. If necessary, use a Box-Cox transformation to stabilize the variance.
Select model order yourself.
Use automated algorithm.
3. If necessary, difference the data until it appears stationary. Use unit-root tests if you are unsure.
Use auto.arima() to find the best ARIMA model for your time series.
4. Plot the ACF/PACF of the differenced data and try to determine possible candidate models.
5. Try your chosen model(s) and use the AICc to search for a better model.
6. Check the residuals from your chosen model by plotting the ACF of the residuals, and doing a portmanteau test of the residuals.
Do the residuals look like white noise?
no
yes
7. Calculate forecasts.

Figure 4.3: Manual ARIMA model generation process [85]

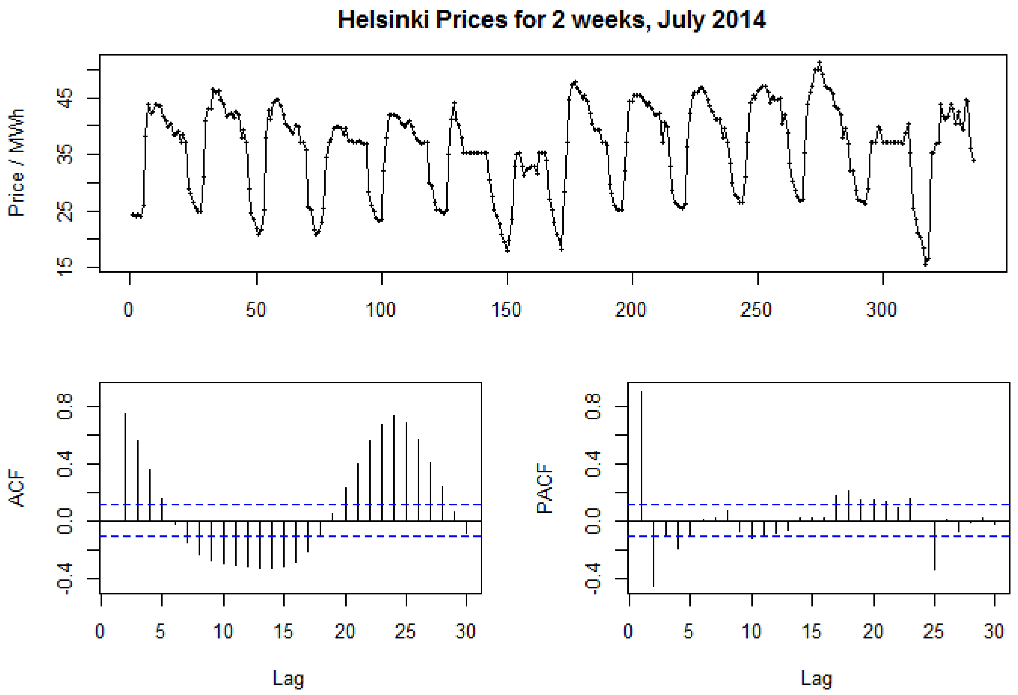


Figure 4.4: Two weeks of hourly day ahead prices with ACF and PACF plots (Nord Pool Spot, Helsinki from July 7th to July 21st in 2014)

right a partial autocorrelation plot (PACF) is printed. As already discussed in section 4.2 this data exhibits daily and weekly seasonality which is clearly visible on the time series plot.

As ARIMA models require data to be stationary (i.e. it exhibits no trend or seasonality) the proposed method to transform a series into a stationary series is to apply one or more levels of differencing or seasonal differencing. The resulting residuals of the model should show no autocorrelations and should exhibit a zero mean [85]. Ideally residuals have constant variance and exhibit a normal distribution as well.

ACF and PACF plots are diagnostic plots to examine correlations within the dataset [70, 86]. Both ACF and PACF plots display individual lags on the x-coordinate while on the y-coordinate the values of the autocorrelation and partial autocorrelation functions are displayed for ACF and PACF plots, respectively. The dashed blue lines denote the 95% confidence interval for white noise, i.e. lag values contained within this interval do not reject the white noise hypothesis. Values exceeding the confidence bounds are „significant“ regarding correlations in the data.

The ACF plot displays the “coefficients of correlation between a time series and lags of itself” whereas the PACF plot shows the “partial correlation coefficients between the series and lags of itself” [86]. Correlation coefficients for autocorrelations are said to be interdependent which means that the value of an autocorrelation at lag h depends on correlations of all previous lags $1, \ldots, h-1$. In contrast, partial autocorrelations only include correlations specific to lag h without considering correlations at lower order lags.

According to the model generation process in Figure 4.3 data should be differenced if necessary to make the series stationary. In this case a seasonal difference might be appropriate due to the obvious seasonality in the time series.

The ACF plot above shows a decay of significant autocorrelations whereas the PACF plot depicts several significant spikes with most significant ones at lags 2 and 25. As the spike at lag 25 in the PACF plot is close to the assumed seasonal period of 24 this may suggest adding a seasonal AR(1) term with a period of 24. The peak value at lag 24 in the ACF plot may be another indicator of seasonal correlation. Therefore, the suggested intermediate ARIMA model is ARIMA(0,0,0)(1,1,0)[24] with a seasonal AR term and one level of seasonal differencing with a period of 24.

The resulting model residuals are depicted in Figure 4.5.

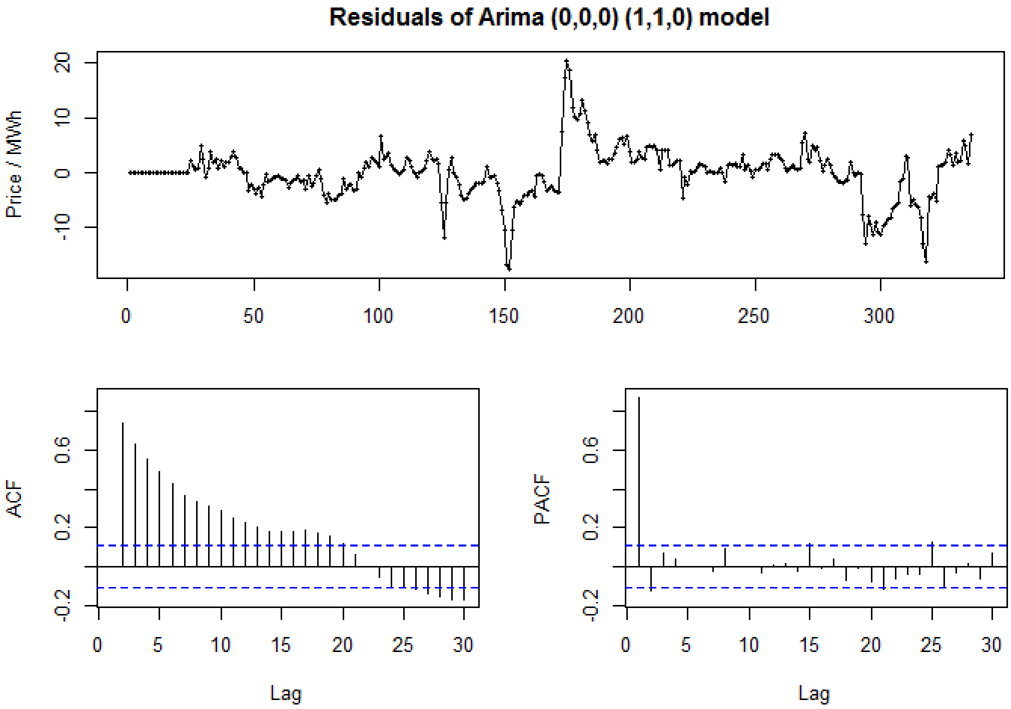


Figure 4.5: Intermediate ARIMA(0,0,0)(1,1,0)[24] model with seasonal difference

Figure 4.5 depicts residuals with removed hourly seasonality and zero mean. However the series can not be regarded as stationary as significant deviations exist from the mean in the time series. Apart from the weekly seasonality further correlations can be identified.

According to [86] a cut-off (rapid decrease of correlation values) at the PACF plot indicates adding an AR term with an order corresponding to the number of the last significant lag of this occurrence. Conversely a cut-off at the ACF plot indicates the addition of a MA term with the number of the last significant lag in the ACF plot.

Since Figure 4.5 shows a slow decay in the ACF and a small but significant coefficient at lag 2 in the PACF we recognize a cut-off at this lag in the PACF and add a (non-seasonal) AR(2) term to the model. Thus the resulting model is ARIMA(2,0,0)(1,1,0)[24] which is shown in Figure 4.6.

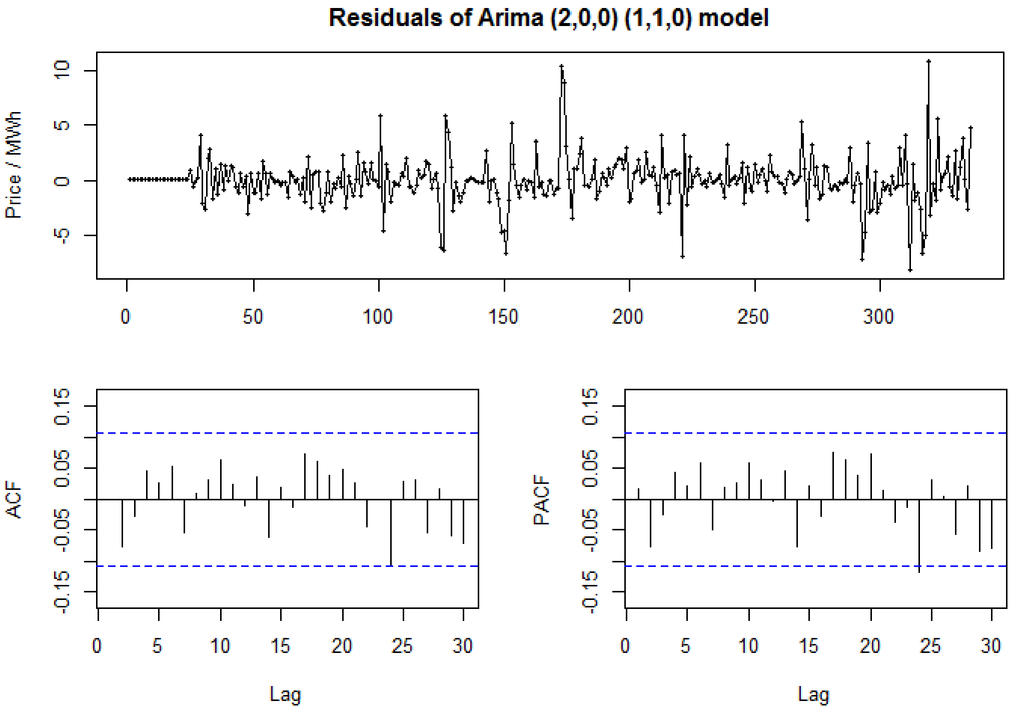


Figure 4.6: Residuals of ARIMA(2,0,0)(1,1,0)[24]

The time series of residuals above can be assumed to be stationary as no significant deviations from the mean are visible.

There is only one significant spike at lag 24 remaining which might indicate still some seasonal information in the data, however it could be as well regarded as outlier as 1 in 20 spikes is said to be significant by chance alone [85].

Model validation

The corrected Akaike Information Criterion (AICc) is a well known goodness of fit measure for stochastic models [97]. It provides a relative quality measure for models such that models with lower AICc values are assumed to better capture the characteristics of the underlying data. Equation 4.30 defines the metric.

$$AICc = -2\log\mathcal{L} + \frac{2dn}{n-d-1} \tag{4.30}$$

with $\log\mathcal{L}$ being the log-likelihood, d the model size (number of parameters) and n denotes the sample size. Note that the model size d is present in order to penalize models having too many parameters. The log-likelihood function estimates how well the model fits the data.

After a model has been chosen by evaluating the AICc value the model residuals should be checked for existing correlations. As shown in the last section this can be done by verifying the ACF and PACF plots but tests are available for automated testing against randomness [85,97].

One of these tests is the Ljung Box test which tests a given number of autocorrelations if they fall outside the significance bounds [97]. The test is defined in Equation 4.31.

$$Q = n(n+2)\sum_{j=1}^{h}\frac{\hat{\rho}^2(j)}{n-j} \tag{4.31}$$

with n as the sample size, h as number of lags and $\hat{\rho}^2(j)$ as the squared autocorrelation at lag j. Its distribution can be approximated by the χ^2 distribution with h degrees of freedom. The white noise hypothesis is rejected if $Q > \chi^2_{1-\alpha}(h)$ with a defined significance level α with $\chi^2_{1-\alpha}$ being the $(1-\alpha)$ quantile of the χ^2 distribution with h degrees of freedom.

Model evaluation

In order to assess the quality and goodness of fit of the models generated in section 4.3 they are validated and compared to a model generated by an automatic model generation procedure.

In addition to the manual model selection procedure shown in the last section automatic model generation procedures exist to detect the model with best goodness of fit parameters by iterating over a set of candidate models.

For generating ARIMA models the *auto.arima* function may be used [85,92].

It is defined by the following steps [85]:

1. Determine the number of differences d by applying a unit root test (e.g. KPSS test).

2. A set of candidate models is generated starting with the following models:

   ARIMA(2,d,2)
   ARIMA(0,d,0)
   ARIMA(1,d,0)
   ARIMA(0,d,1)

   Model parameters p and q are modified by adding ± 1 where the best model is set as reference. Repeat the last line until no model with a lower AICc value can be found.

Model Name	Model Parameters
Intermediate Model	ARIMA(0,0,0)(1,1,0)[24]
Final Model	ARIMA(2,0,0)(1,1,0)[24]
Automatic Model	ARIMA(2,0,3)(1,0,2)[24]

Table 4.1: Model names and parameters

A model has been generated using auto.arima based on the same data as used for manual model selection in section 4.3. The resulting models with corresponding model parameters are outlined in table 4.1.

AICc and Ljung Box values have been calculated for each model defined above. For Ljung Box tests a 95% confidence interval has been used such that resulting p-values greater than 0.05 indicate white noise properties of the residuals. Thus a high Ljung Box test value gives high probability of a series resembling white noise.

The AICc and Ljung Box values are outlined in table 4.2.

	Intermediate Model	Final Model	Automatic Model
AICc	1868.01	**1400.22**	1457.26
Ljung Box	< 2.2e-16	0.1523	**0.9927**

Table 4.2: AICc and Ljung Box values

Results show that it is possible for manually selected models to outperform automatically generated models, i.e. the Final Model shows a lower AICc value than the Automatic Model. This is presumably due to the manual model using explicit seasonal differencing which the automatic nodel does not use. However the Ljung Box test values indicate a different result where the Automatic Model results in a higher p-value and thus exhibits less correlations than the Final Model. As both models exceed the significance bounds of 0.05 this is only of minor importance as both exhibit white noise characteristics. If in doubt the AICc value should be given precedence over Ljung Box test results.

4.4 Model selection algorithm

An automated model selection algorithm has been implemented to provide aid in finding suitable ARIMA models for a given training dataset. The process consists of several data preprocessing steps with generation and comparison of different ARIMA models and model evaluation based on a weighted function of AICc and Ljung Box test values.

The model selection algorithm consists of three separate functions each contributing a different part to model generation. These functions are *GenerateArimaModel* as the base function for model generation, *AutomatedBoxTest* for determining the right parameters for the Ljung Box tests and *SeasonalPeriods* for estimation of possibly existing seasonal periods within the data.

Function GenerateArimaModel

1. Get trainingdata
   a) Read time series of energy prices from given location
   b) Define training period by start and end date
2. Determine seasonality
   a) Call *SeaonalPeriods* to retrieve seasonal periods from the data
3. Create time series objects
   a) For each seasonal period found create a time series object based on that period
4. Calculate the Box-Cox transformation parameters
   a) Compute the Box Cox parameters for each of the created time series objects
5. Create models
   a) Create model(s) without BoxCox transformation
      i. Generate model with auto.arima for each of the created time series objects
   b) Create model(s) with BoxCox transformation
      i. Generate model with auto.arima and previously defined lambda (Box-Cox) parameter for each of the created time series objects if Box-Cox parameter is not equal to 1 (then it would have no effect)
6. Execute Ljung Box test for each of the models
   a) Call *AutomatedBoxTest* to retrieve the Ljung Box p-values for each model
7. Saving models, boxtests, AICc values and p-values in vectors
8. Model Evaluation
   a) Check model goodness of fit via AICc value comparison
      i. $p_{aicc_i} = \frac{1}{|AIC_{min} - AIC_i| + 2}$
      ii. “+2” -> moderate the decrease of values due to the inverse function
   b) Compare p-values of Ljung Box tests to check residual characteristics
      i. p-values range from 0 to 1 -> determine difference in relation to full range
      ii. $p_{ljung} = \frac{p_{value} - 0.05}{1 - 0.05}$
9. Model selection
   a) Calculate weighted result based on goodness of fit values with user defined weights
   b) $F_i = w_{aicc_i} p_{aicc_i} + w_{ljung_i} p_{ljung_i}$ for each model $i \in M$
10. Return model with the highest goodness of fit value F_i

SeasonalPeriods

1. If target period is specified and enforced
    a) return a list of (targetPeriod, defaultPeriod)
2. Else
    a) Get the x most frequent periods sorted by number of occurrences descending, where x is a user defined number (apply periodogram)
    b) If a max period limit has been specified, discard any periods above this limit
    c) If the number of occurrences of a period falls below the white noise threshold, discard the period
    d) If the target period is specified and has been found
        i. return a list of (targetPeriod, defaultPeriod)
    e) Otherwise
        i. add the defaultPeriod (=1) to the list of periods and return the list

AutomatedBoxTest

1. Determine the most suitable value for the lag based on the sample size and whether seasonal periods are existing in the dataset.
    a) The following rules of thumb have been established to determine a suitable value for the lag of the test [49]:
        i. for non-seasonal time series a lag value of $h = min(10, T/5)$ should be used
        ii. for seasonal time series a lag value of $h = min(2m, T/5)$ should be used, where T denotes the sample size and m the seasonal period
    b) The number of determined lags is reduced by the number of parameters in the model
    c) The box test is executed and result is returned

Discussion

The core of the model selection algorithm are steps 8 and 9 of Function GenerateArimaModel where relative measures for both AICc and Ljung Box values have been developed to be able to integrate them into a weighted function. Thus it is possible to define weights to define the impact of each validation measure on forecast model selection. As mentionend before (Section 4.3) AICc value results should in general be given precedence over Ljung Box values since AICc values are considered more stable.

Another thing to note is the so called *target period* in function *SeasonalPeriods*. This is a user estimated period which is assumed to be contained in the dataset. If it is found it is given precedence over other periods returned. There is also the possibility of “enforcing” the period in which case it is taken without calculating other periods.

4.5 R / Java Simulation Framework

The R / Java Simulation Framework is responsible for preparation and execution of a large scale simulation of generating and evaluating forecast models over an extended period of time. In this section the architecture and implementation of the simulation framework is presented as well as the interfaces existing between R and Java.

Architecture of the simulation framework

The framework consists of a Java application server connected to a separately running R server on which R commands are executed. R is a statistical program which is able to model complex statistical functions and includes a considerable amount of statistical packages [36]. It became the de-facto standard for stochastic processing and is available free of charge as well as for various platforms.

The Java application server has been set up as WildFly 8 [51] with Java Enterprise Edition 7 [74] and an Oracle Database [75] which provides a scalable architecture including the possibility for modeling web service interfaces.

In this work a collection of web services has been set up for simple access of data for external applications. This can be used e.g. by the cloud simulator outlined in section 5.1. The Java application server wraps methods for retrieval and storage of energy price data, forecast model generation and large scale simulations. It utilizes R for complex statistical processing which is included into advanced methods for simulations and model generation.

Class diagram

In order to visualize the most important entities of the framework a class diagram is outlined in Figure 4.7.

A basic entity for energy price retrieval is the *Resource* which wraps the retrieved contents of an energy price source. Data retrieval happens over a generic *DataFetch* interface which is able to fetch data from an URL (e.g. web service) or directly from a file. The parsing of the energy price data is achieved by a generic *Parser* interface which can be implemented by Parsers for different file types (currently XLS and CSV).

A *ResourceType* stores a reference to a specific DataFetch and Parser implementation. As source types and formats of energy data will be different for each energy market at least one DataFetch and Parser implementation has to be provided for each market. The *ResourceManager* keeps track of all resources associated with a registered energy market. It is implemented by resource managers for day ahead and real time markets, respectively.

The *MarketData* entity handles all the logic necessary for actually retrieving energy price data for a given Resource. It provides methods to retrieve data from registered energy sources and subsequently parsing and importing the data into the database. Instances of MarketData are used by *RTPricesResourceRESTService* and *DAPricesResourceRESTService* that provide web service interfaces for data retrieval. In turn, *RManagerResourceRESTService* utilizes these web services to retrieve energy price data for model generation and forecast simulations.

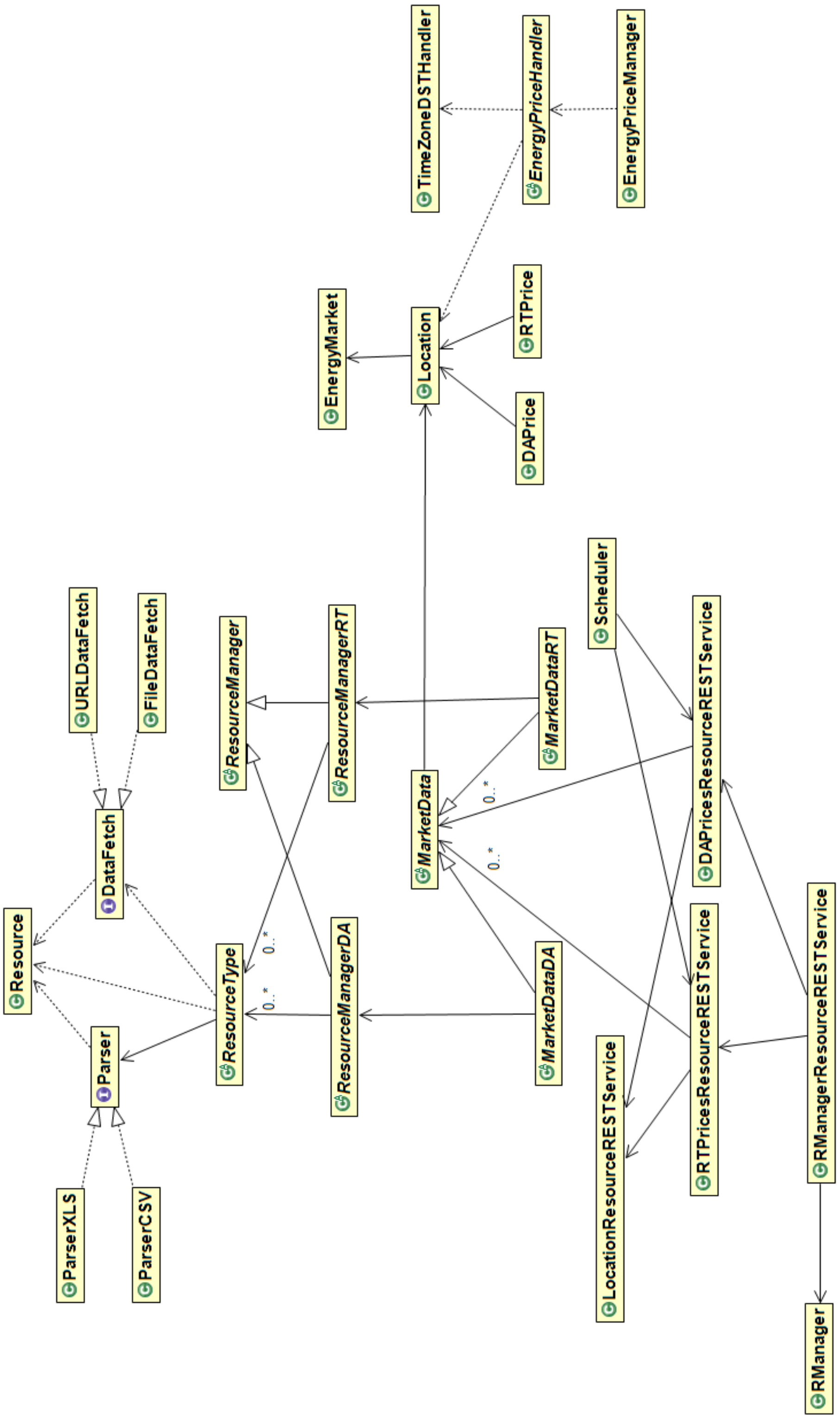


Figure 4.7: Class diagram of Energy Price Management Application

The *Scheduler* utilizes Java EE scheduling mechanisms to automatically retrieve energy prices from specific locations at defined time intervals (e.g. each day at 3 pm). Thus for continuous operation a scheduler can be set up for a location and dates when data should be retrieved regularly from defined interfaces.

Each MarketData instance stores a reference to a specific *Location* of an *EnergyMarket*. Instances of *DAPrice* and *RTPrice* reference the location from which they were retrieved where these price entities each resemble exactly one energy price entity stored in the database.

Finally the *EnergyPriceHandler* provides a generic interface for parsing energy prices with location specific DST (daylight saving time) changes. This interface is extended by specific implementations of energy markets to manage DST dates. The *TimeZoneDSTHandler* provides a method to retrieve exact DST dates for each location and year. Thus energy prices are stored with dates correctly considering DST changes specific to each location.

Web Service interfaces

The *Energy Price Management Application* (EPMA) on the application server provides different types of web service interfaces.

Web services are grouped by type:

- *DAPricesResourceRESTService* for managing day ahead prices
- *RTPricesResourceRESTService* for managing real time prices
- *LocationResourceRESTService* for managing locations
- *EnergyMarketResourceRESTService* for managing energy markets
- *RManagerResourceRESTService* for managing R resources

Each web service interface is defined by a base path (e.g. /rtprices), the respective web service name (e.g. /importall) and mandatory or optional web service parameters.

R Forecast evaluation

The forecast model evaluation discussed in Section 4.6 is based on implementation of an R function which is called and processed by respective methods on the application server.

The corresponding R function is named *evaluateModels* and takes energy price data as input separated into a training and test data sets. It generates different forecast models based on the given training set considering possible seasonal periods in the data (see Section 4.4) and returns aggregated accuracy measures for each model and forecast horizon.

Function evaluateModels

Input: *pricesTraining* - a list of energy prices taken as training data set
pricesTest - a list of energy prices taken as test data set
Output: *modelList* - a list of models generated within the function
accuracyList - a list of accuracy measures calculated for each model

```
1  evaluateModels <- function(pricesTraining, pricesTest)
2  {
3    period <- getSeasonalPeriod()
4    pricesTrainingTs <- generateTimeSeries(pricesTraining, period)
5    modelList <- generateModels(Mean,Ses,Holt,HoltWinters,
6                                        Arima,Tbats)
7    accuracyList <- list()
8    forecastHorizonList <- (1,3,6,12,18,24,36,48,96,168)
9    for( m in modelList ) {
10     for( h in forecastHorizonList ) {
11       fc <- getForecastForHorizon(m,h)
12       accuracyList(m,h) <- getAccuracyMeasures(fc,pricesTest)
13     }
14   }
15   return list(modelList,accuracyList)
16 }
```

Listing 4.1: Function evaluateModels

In Listing 4.1 the function *evaluateModels* is defined. In lines 3 and 4 the most prevailing seasonal period (if any) is determined and a time series object is generated based on that period. In line 5 a series of forecast models is generated that are investigated in the algorithm which are Mean forecast model, SES model, Holt (SES with trend), HoltWinters (SES with trend and seasonality), ARIMA (model generation described in Section 4.3) and TBATS (seasonal model).

Line 8 sets a list of forecast horizons in hours, i.e. forecasts are generated for horizons from 1 up to 168 hours. In line 11 forecasts are produced for each model and forecast horizon separately and in line 12 the generated forecasts are validated by getting accuracy measures for out-of-sample forecasts based on the test data set. Line 15 returns a compound object of the list of generated models and accuracy measures.

Java implementation on application server

A large scale evaluation of forecasting methods should be conducted where models and corresponding accuracy measures are generated for different training data sets over an extended period of time. Thus the application server implements methods for performing customizable forecasting simulations.

A java method *evaluateModels* in class *RManager* has been implemented which calls the previously introduced R function *evaluateModels* (Listing 4.1) and saves the result in an internal folder on the application server. This method is in turn called by an identically named method in *RManagerResourceRESTService* which provides the REST service interfaces to external applications for retrieval of results.

In order to conduct automated large scale simulations additional methods have been implemented in *RManagerResourceRESTService* to perform customizable simulations. The method *runSimulation* performs a simulation based on a number of parameters. The method signature is depicted in Listing 4.2.

```
runSimulation(String priceType, Long locationId,
              String simulationStart, String simulationEnd,
              String trainingPeriod, String testPeriod, String intervalPeriod
                  )
```

Listing 4.2: Method runSimulation

The *priceType* defines whether the simulation should be based on day ahead or real time prices. Accordingly the parameter is set to either “da” or “rt”. The *locationId* expects an Id of a registered location within the application server. The parameter *simulationStart* expects a date time String containing the start date from where the simulation should be started. The parameter *simulationEnd* describes the end date of the simulation as a date time String, respectively. *trainingPeriod* is the training period for which energy prices should be loaded for model generation, *testPeriod* is the test period against which the models are to be tested and *intervalPeriod* denotes the time interval the simulation should be advanced in each step.

The corresponding web service interface is given in Listing 4.3.

```
/r/simulation/{type}/{loc_id}/{simulationStart}/{simulationEnd}/
                {trainingPeriod}/{testPeriod}/{intervalPeriod}
```

Listing 4.3: Method runSimulation web service interface

The base path of the web service interface is */r* which is an indicator for web services that are based on R implementations. The name of the web service is *simulation* and parameters have already been described for Listing 4.2. This method or interface is responsible for executing a single simulation with defined start and end dates, training-, test- and interval periods.

A common format has been defined to distinguish different periods and intervals. A regular expression for the format could be described as *^\d+[hdw]$* which is a combination of at least one digit and one of *h*, *d* or *w* which means *hour*, *day* and *week*, respectively. Therefore a training period of 2w, 1d or 48h means two weeks, one day or 48 hours. The same holds true for test- and interval periods.

The method *runSimulation* calls the method *evaluateModels* which performs a single model evaluation for a given training and test period. Its method signature is depicted in Listing 4.4.

```
evaluateModels(String simulationName, String priceType, Long locationId,
              String startTraining, String endTraining,
              String startTest, String endTest)
```

Listing 4.4: Method evaluateModels

This method is responsible for conducting a model evaluation by calling the respective same-named function on the *RManager*. This method is called for each simulation run triggered by the *runSimulation* method, i.e. it is called for different training-, test- and interval periods. In order to distinguish different simulations where each simulation is saved in a folder named after the simulation name the simulationName has been normed (Listing 4.5):

```
Definition of the simulation name:

<priceType>_sim_<locationId>_<trainingPeriod>_<testPeriod>_<intervalPeriod>
```

Listing 4.5: Simulation name definition

where *priceType* is the type of energy price (“da” or “rt”), *sim* is a delimiter common to all simulations, *locationId* is the id of the location for which to conduct the simulation, *trainingPeriod* is the encoded form of the training period, the same holds true for *testPeriod* and *intervalPeriod*. All parameters are delimited by underscores to clearly distinguish different parameters.

Therefore a valid simulation name would be *da_sim_1_2w_1w_1w* denoting a simulation on day ahead prices for location with Id 1, 2 weeks training period, 1 week test period and 1 week interval period.

The method *runSimulation* calls the method *evaluateModels* repeatedly over the whole simulation time period (simulation start to simulation end) in intervals specified by the *intervalPeriod* with training and test periods specified by the *trainingPeriod* and *testPeriod* parameters.

The results of each call of evaluateModels are saved in a separate file under a folder named the same as the simulation. Each file has a naming convention of <*simulationName*>_<*start Training*> where *startTraining* denotes the start date of the training period for this simulation.

4.6 Forecast model evaluation

A large scale forecast model evaluation is performed over an extended period of time in order to identify models that show good performance across different sets of energy price time series. Different metrics and forecast horizons are investigated for a thorough evaluation of model performance to derive the best model from the simulations.

Four simulations have been run based on data from four different energy markets, namely *Nord Pool Spot*, *Belpex*, *ISO New England* and *PJM*. A time period of three years of energy price data has been chosen to be able to make a meaningful statement about forecast performance on energy prices.

For each of these locations three different training periods have been evaluated: two weeks, three weeks and four weeks. The simulations have been run with intervals of 1 week. The (maximum) test period has been set to 1 week as well.

The resulting simulations according to the defined format in Section 4.5 are outlined in Table 4.3. Each of these simulations and models are evaluated for five different accuracy measures to get a broader view of the model performances (see also Section 4.2):

- Mean error (ME)
- Mean absolute error (MAE)
- Root mean squared error (RMSE)
- Mean percentage error (MPE)

- Mean absolute percentage error (MAPE)

In addition, each of these error measures has been evaluated for ten different forecast horizons (in hours): 1, 3, 6, 12, 18, 24, 36, 48, 96, 168. Thus one goal of the simulations is to evaluate the performance of models when exposed to different forecast horizons.

Simulation Name	Type	Location	Training Period	Test Period	Interval Period
da_sim_1_2w_1w_1w	day ahead	Hamina	2 weeks	1 week	1 week
da_sim_1_3w_1w_1w	day ahead	Hamina	3 weeks	1 week	1 week
da_sim_1_4w_1w_1w	day ahead	Hamina	4 weeks	1 week	1 week
da_sim_2_2w_1w_1w	day ahead	St.Ghislain	2 weeks	1 week	1 week
da_sim_2_3w_1w_1w	day ahead	St.Ghislain	3 weeks	1 week	1 week
da_sim_2_4w_1w_1w	day ahead	St.Ghislain	4 weeks	1 week	1 week
rt_sim_4_2w_1w_1w	real time	Portland	2 weeks	1 week	1 week
rt_sim_4_3w_1w_1w	real time	Portland	3 weeks	1 week	1 week
rt_sim_4_4w_1w_1w	real time	Portland	4 weeks	1 week	1 week
rt_sim_6_2w_1w_1w	real time	Richmond	2 weeks	1 week	1 week
rt_sim_6_3w_1w_1w	real time	Richmond	3 weeks	1 week	1 week
rt_sim_6_4w_1w_1w	real time	Richmond	4 weeks	1 week	1 week

Table 4.3: List of all conducted forecast simulations

Forecast evaluation results

For conducting the simulations energy price data has been taken from years 2012 to 2014. As mentioned above model evaluations have been done in intervals of 1 week and forecast errors have been measured based on different training and test data sets for various forecast horizons. Each of the forecast error results has been aggregated for the respective training-, test period and forecast horizon over the whole time period of the simulation.

Result tables

Aggregated RMSE results for Hamina belonging to the Nord Pool Spot power market are shown in Tables 4.4, 4.5 and 4.6 for 2, 3 and 4 weeks of trainingdata, respectively.

Results show that forecasts over different horizons exhibit comparable relative values across almost all forecasting models. That is, values behave in a similar way across forecast models when examining different forecast horizons.

Notably values stay much in the same value range except for Holts and HoltWinters models where forecast errors increase non-linearly with an increase of forecast horizon. This may be due to their inability to capture short term seasonality.

From the results it is visible that the forecast periods from 24 hours to 48 hours provide the best results through almost all models. A forecast period of 48 hours seems to provide the best results apart from results for low periods (1 to 6 hours). Small forecast horizons generally exhibit smaller forecast errors as the short period of time does not allow a high value for aggregated

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.30	10.66	10.79	14.78	14.22	13.02	13.21	12.58	12.99	12.18
ses	2.13	3.54	5.06	15.83	16.38	14.98	14.97	14.64	15.13	13.30
holts	1.91	3.36	9.48	30.83	38.85	44.08	58.79	74.25	137.44	228.66
holtwinters	3.22	7.15	13.18	26.22	36.32	46.07	65.31	85.66	167.14	289.51
arima	2.11	3.08	4.65	13.66	13.89	12.67	12.87	12.61	13.68	12.98
tbats	2.16	2.89	4.35	10.68	10.89	10.02	10.19	9.94	10.70	10.33

Table 4.4: Forecast evaluation results based on RMSE for a training period of 2 weeks, Nord Pool Spot, Hamina

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.40	10.75	10.90	15.03	14.46	13.25	13.44	12.81	13.27	12.42
ses	2.14	3.56	5.06	15.84	16.38	14.98	14.97	14.63	15.14	13.31
holts	1.91	3.38	9.51	30.88	38.90	44.15	58.90	74.39	137.77	229.23
holtwinters	3.44	7.61	14.06	27.95	38.22	47.77	67.07	87.51	169.44	292.43
arima	2.12	2.95	4.37	13.02	13.12	11.89	11.99	11.63	12.37	11.50
tbats	2.15	3.04	4.73	11.23	11.41	10.51	10.57	10.33	11.05	10.67

Table 4.5: Forecast evaluation results based on RMSE for a training period of 3 weeks, Nord Pool Spot, Hamina

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.52	10.88	11.04	15.22	14.63	13.42	13.58	12.94	13.41	12.57
ses	2.13	3.57	5.06	15.84	16.38	14.97	14.96	14.61	15.13	13.30
holts	1.89	3.37	9.53	31.00	39.07	44.34	59.18	74.75	138.54	230.59
holtwinters	5.58	11.05	17.33	33.52	44.92	54.87	76.25	98.63	189.26	327.09
arima	1.98	2.85	4.39	12.69	12.79	11.60	11.73	11.35	12.06	11.12
tbats	2.20	3.09	4.64	10.98	11.19	10.31	10.45	10.20	10.91	10.36

Table 4.6: Forecast evaluation results based on RMSE for a training period of 4 weeks, Nord Pool Spot, Hamina

deviations from the series. Therefore it can be stated that forecast error results can only be meaningfully compared among forecast windows of a minimum of 12 hours.

When comparing performance of forecast models the TBATS model clearly exhibits the lowest forecast errors when considering forecast windows from 12 hours upwards. It is followed by the ARIMA model and, suprisingly by the Mean forecast model. For the 2 weeks training period and $>= 48$ forecast periods it even outperforms the ARIMA model. This is possible when the forecasts from advanced models such as ARIMA are less exact and data is closely moving around the mean which benefits mean forecasts.

Result graphs

Results for different forecast error measures are shown for ARIMA and Mean models and different training data periods in Figures 4.8 and 4.9. Absolute values for different forecast error measures can be very different as depicted in figures below. What becomes apparent is that percentage error measures exhibit significantly higher absolute values than other methods. This can be attributed to their instability when facing low time series values.

When only comparing RMSE errors (the second bar in each group of bars) the values of ARIMA and Mean forecasts overall appear much in the same time range as already pointed out by the discussion in 4.6. Also for both models values for forecast horizons 24h to 48h appear lower than for other horizons for all forecast error measures which is also consistent with the observation in the result tables.

All results can be looked up in the Appendix (result tables: A.1, result graphs: A.2).

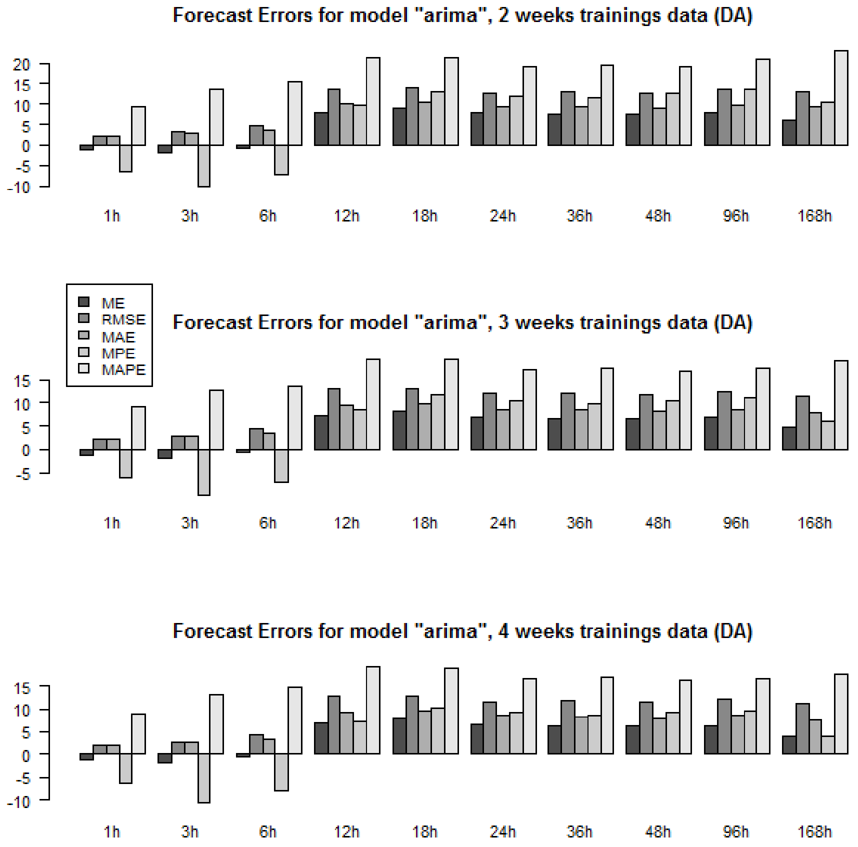
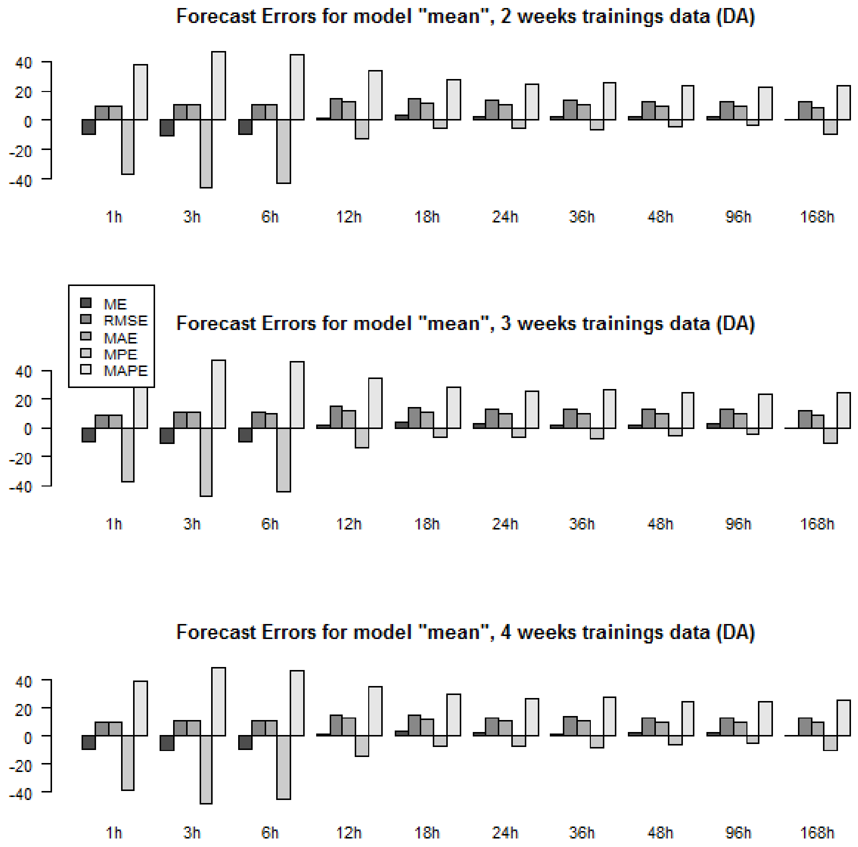


Figure 4.8: Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina


Forecast Errors for model "mean", 2 weeks trainings data (DA)

Forecast Errors for model "mean", 3 weeks trainings data (DA)

Forecast Errors for model "mean", 4 weeks trainings data (DA)

Figure 4.9: Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

The result tables as well as graphs emphasize the observations made previously about model behavior.

What can be observed by investigating the large scale forecast evaluation over all locations is that very different behaviors and scales of forecast errors exist for the same model over different locations (i.e. datasets). An extreme example is the difference in forecast error distributions of the Nord Pool Spot and Belpex energy markets (tables A.1 and A.1, graphs A.2 and A.2).

However, the seemingly extreme difference in forecast error distribution is mainly caused by exorbitant values of the MPE and MAPE error measures. This may be attributed to the occurrence of many low valued energy prices (close to zero) in the Belpex energy market which causes very high values for these methods.

When comparing models TBATS, ARIMA followed by Mean and SES forecast models achieve the best results. Remarkably Mean forecasts outperform forecasts of SES models for almost all data sets of day ahead energy markets regarding forecast horizons $>= 12$. This behavior is reversed for real time markets where SES generally provides better performance.

TBATS models exhibit the least forecast errors when they are able to capture seasonality appropriately. Otherwise these models show exponential growth of forecast errors over all measures. Due to this instability ARIMA models may be preferred as they exhibit stable forecast errors over most data sets and forecast horizons.

Another note to error distribtions is that for the same forecast model similar patterns can be observed despite differences across energy markets. Concerning different training periods it can be observed that for most methods errors decrease with increasing training period, i.e. four weeks of training data exhibits the least errors in most cases.

Also different distributions are apparent for different error measures, e.g. RMSE tend to increase with increasing forecast horizon whereas percentage errors (MPE and MAPE) tend to hit their lowest values for forecast horizons of 24h to 48h apart from horizons below 12h.

Concerning the forecast error measures a focus has been laid on evaluating Root Mean Squared Errors (RMSE) since this measure has been proven to provide the most stable results. Percentage error measures such as Mean Percentage Error (MPE) and Mean Absolute Percentage Error (MAPE) tend to become numerically unstable for low values. Difference based measures such as Mean Error (ME) with non-absolute value differences provide little insights to the actual occurred forecast errors as positive and negative values might cancel each other out.

Aggregated Results

Results have been aggregated over all forecast horizons and for each simulation which are shown in Tables 4.7, 4.8, 4.9 and 4.10. Best results have been formatted in bold in the tables.

ARIMA and TBATS models showed superior results over all forecast horizons and training periods. TBATS models show best results when data can be modeled appropriately but exhibit anomalies in other cases. For different training periods a trend can be detected where forecast errors increase with increasing duration of the training period. However, ARIMA models show opposite results where forecast errors consistently decrease with increasing training period.

	mean	ses	holts	holtwinters	arima	tbats
2 weeks	12.37	11.60	62.76	73.98	10.22	**8.21**
3 weeks	12.57	11.60	62.90	75.55	9.50	**8.57**
4 weeks	12.72	11.60	63.23	85.85	9.26	**8.43**

Table 4.7: Results of evaluation for Hamina, Nord Pool Spot (DA)

	mean	ses	holts	holtwinters	arima	tbats
2 weeks	15.77	15.23	127.13	186.82	**13.49**	1.27E+50
3 weeks	16.07	15.27	128.05	190.86	12.94	**11.61**
4 weeks	16.25	15.30	127.03	186.37	**12.87**	2.49E+36

Table 4.8: Results of evaluation for St.Ghislain, Belpex (DA)

	mean	ses	holts	holtwinters	arima	tbats
2 weeks	26.42	23.61	235.84	417.39	**24.11**	5.45E+16
3 weeks	26.36	23.56	238.92	686.95	**22.26**	1.64E+18
4 weeks	26.84	23.32	268.10	437.85	**21.83**	2.18E+32

Table 4.9: Results of evaluation for Portland, ISO-NE (RT)

	mean	ses	holts	holtwinters	arima	tbats
2 weeks	17.41	15.17	148.54	333.32	**15.02**	7.57E+13
3 weeks	17.36	15.23	147.77	318.61	14.14	**13.46**
4 weeks	17.34	15.29	148.67	357.82	**14.11**	5.44E+108

Table 4.10: Results of evaluation for Richmond, PJM (RT)

Conclusion of results

From the discussion of the large scale forecast evaluation in the previous section various conclusions may be drawn.

Model performance Concerning model performance ARIMA models showed the best results over different energy markets, training periods and forecast horizons. They showed consistently low forecast errors and appeared very stable throughout the simulations. Thus among the observed models ARIMA models may be the best choice considering forecast performance and statbility of results.

Forecast Horizon Concerning forecast horizons it has been observed that horizons from 24 hours to 48 hours provide the best results when considering all models and training periods. For low horizons up to 12 hours different models may be suggested, i.e. Holts model provide considerably lower forecast errors for forecast windows of 1 and 3 hours. For other than very short-term forecasts a forecast window of 24 to 48 hours provides best results.

Training periods Different model training periods provide different results. Forecast errors have been observed to increase with the duration of the training period most of the time except for ARIMA models where the opposite behavior has been detected. Thus for ARIMA models a training period of 4 weeks is suggested for model generation.

Forecast error measures As discussed before different forecast error methods provide different results for the same dataset. The ME provides an indication of the symmetry of the error distribution but little information about actual forecast performance. The MAE and RMSE provide a way of reliably model forecast errors where they achieve very similar results while the RMSE emphasizes outliers in the data. Percentage error measures have not been found to provide a reliable source of error measurements due to their instability for low values. Thus MAE and RMSE are considered the preferred way of modeling forecast errors.

CHAPTER 5

Simulation Framework

5.1 Architecture of simulation framework

The simulation framework described in this section encompasses already described components in Section 4.5 but enhances the framework by the cloud simulator. The cloud simulator should be able to assess the outcome of different scenarios encompassing data provided by the application server and utilizing forecast models generated on the application server in a previous step.

The goal was to deliver a complete framework that can handle multiple data sources and is extendable for future work with possibly different data sets and an optional connection to a real cloud environment which could replace the simulated cloud environment that is connected to the framework in this implementation.

The implementation consists of three parts which will be presented separately below.

Data management The first part represents the data handling and management interfaces presented in part already in Section 4.5. The purpose of this platform is to have a generic means of parsing and fetching data from various sources that can be defined in advance. For example, data may be fetched from local files that were previously retrieved from energy markets or remote web services that provide energy data of that respective energy market. In addition parsers for different file formats may be defined to automatically parse data and put it into the database for later retrieval. From there arbitrary queries may be executed to retrieve and aggregate data in a specific fashion and execute further tasks based on that data (e.g. model generation).

Forecast generation The second part consists of a collection of statistical methods that should assist in making accurate forecasts based on the previously collected data. A large scale evaluation of forecast methods has been presented in Section 4.6 to determine the best forecast models for data from various energy markets. The purpose behind these statistical examinations is to provide meaningful and accurate forecasting models that can be utilized by the simulation framework. Eventually the performance of forecasts within the simulation should be examined and compared with approaches based solely on currently available data. It is expected that the application of accurate forecasts to energy price time series within the simulation improves overall performance and thus leads to a reduction in total energy costs.

Cloud simulation The third part constitutes the actual cloud simulation which is based on a previously existing and sophisticated cloud simulator written in Python that incorporates cost models for energy and cooling expenses and is easily extendable by integrating custom schedulers [62]. In this work custom settings have been applied to allow for a simulation that meets the needs of the framework to integrate different scenarios. Various parts of the simulator needed to be extended and a completely new scheduler has been built to schedule both cost aware and non-cost aware scenarios. It works together well with other parts of the framework s.t. data can be retrieved from interfaces of the application server and integrated into the cloud simulation. The components are decoupled such that each of them may be replaced by other components easily provided that the same interface is used.

Architectural outline

The architectural outline provides an overview of all components involved in the presented simulation framework and how they work together. All components may interact with one another via defined interfaces s.t. either component may be replaced by a similar component with possibly different implementation details but same interfaces.

Figure 5.1 depicts a block diagram of the simulation framework that shows which components are included and how they interact with one another. Data retrieval from the energy markets is done by the *Import* module as the first action in the process. Several energy markets and their interfaces may be registered beforehand to retrieve data from these interfaces.

From the *Import* module the retrieved data is handed over to the *Data Processing and Consolidation* module which transforms the data to a common format such that it can be fed to the database. This process is similar to the well known ETL process (Extract Transform Load) [96] which extracts data from various sources, transforms it into a defined data format and loads it into a data warehouse or database. The applied process in this work is similar as parsers for different file formats may be defined that can be configured to parse data from different sources. After the data transformation process the retrieved energy prices are stored in the database from where they may be retrieved for simulation purposes.

The *Data Interface* contains all methods for data retrieval of prices stored in the database. Thus data is provided to web service interfaces for arbitrary time periods and both day ahead and real time prices. In addition it is possible to query for local or DST corrected bidding dates or retrieving data adjusted to a predefined currency (e.g. dollars).

The *Forecast Interface* handles all requests related to model generation and forecasting which is provided by the *R server*. It contains methods defined for large scale forecast evaluation, automated model generation and calculating forecasts for generated models.

The *R server* does not directly connect to the database but retrieves energy data from the *Forecast Interface* which triggers the generation of forecasting models. The interface used to

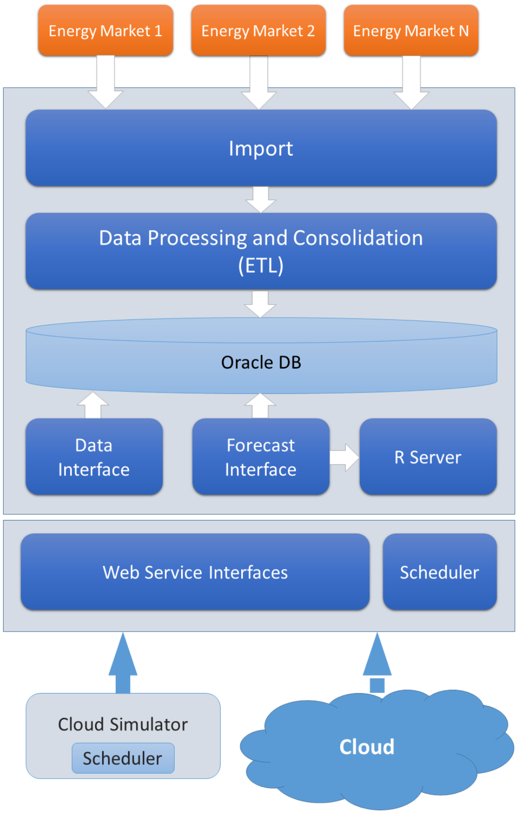


Figure 5.1: Architecture Block Diagram

communicate from Java to R (and vice versa) is called *rJava* [84] which provides a wrapper to directly execute R code in Java.

The *Scheduler* may be configured to trigger data import from specific energy markets at a defined time interval (e.g. every hour or once per day). In this process it may also trigger the generation of forecasting models as the model generation process can take a considerable amount of time depending on the amount of training data provided to the model.

Finally the *Web Service Interfaces* are a means of providing data to external components such as the simulator or a real cloud. These interfaces provide data retrieval and forecast interfaces as described previously. They can be used to get historical data from different energy markets and locations as well as time periods for the purpose of getting customized data for simulations. In addition they provide a convenient way of retrieving forecast data for specific locations and periods of time. As soon as forecasting models have been generated for a specific period of time forecasts can be provided instantly for that period.

Components and Interfaces

In this section a more detailed view on the various components of the framework is provided and the interfaces that exist between them. Figure 5.2 shows this view in form of a component diagram.

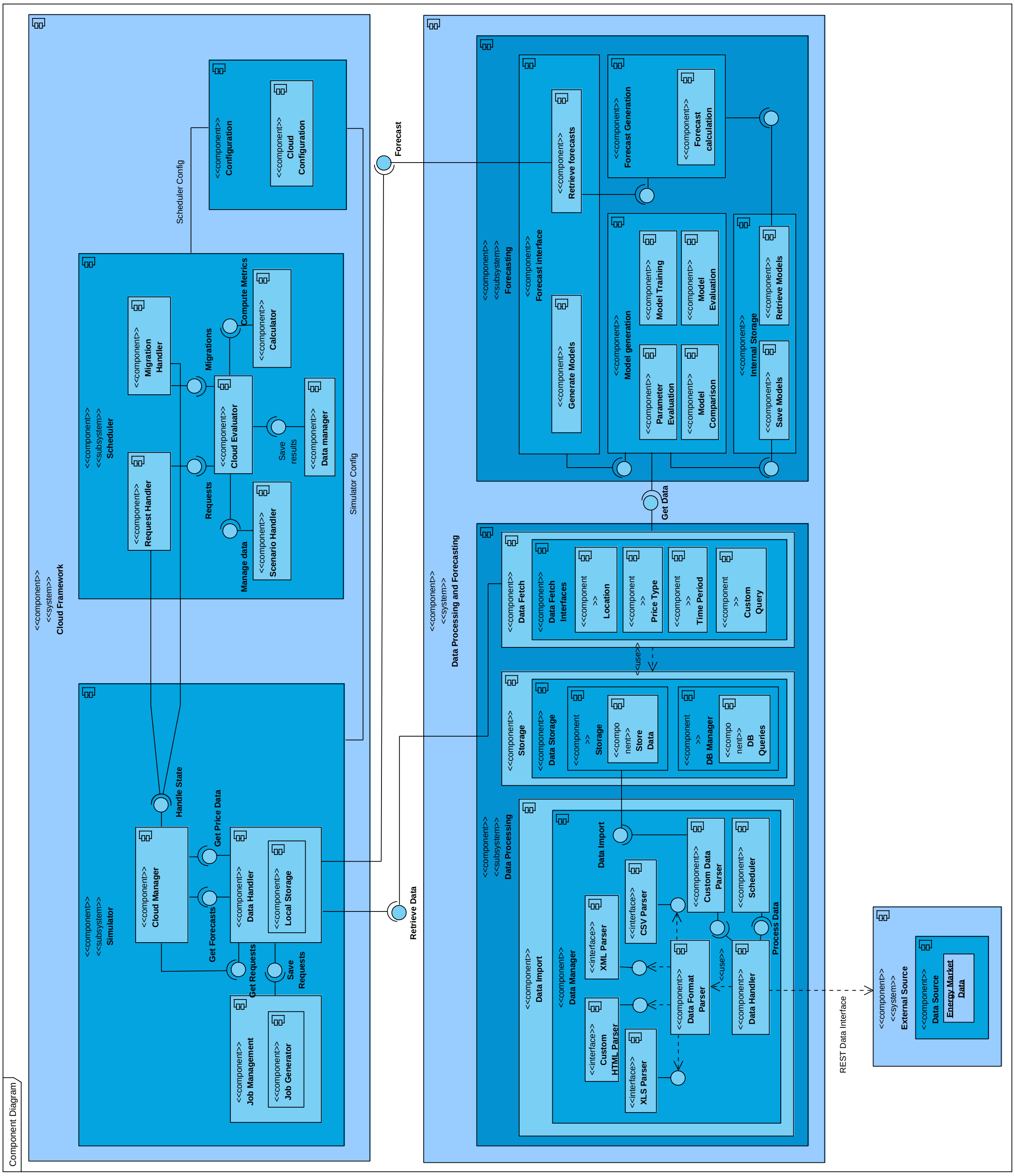


Figure 5.2: Component Diagram

The diagram basically consists of two parts, the *Cloud Framework* and *Data Processing and Forecasting* components. The Cloud Framework contains the *Simulator* and *Scheduler* and is implemented in python [62]. It is responsible for the actual simulation and application of scheduling algorithms. The Data Processing and Forecasting part consists of the *Data Processing* and *Forecasting* components and is implemented on the Java application server. This component is responsible for data management and forecast operations.

Data Processing consists of *Data Import*, *Storage* and *Data Fetch* components.

The *Data Import* component is responsible to retrieve data from previously registered energy markets as indicated by the dashed line connection to the external data source. The data import can as well be triggered by the scheduler which calls the respective services at the Data Handler. The Data Handler uses appropriate parsers to extract data into a common processable format. When the processing of data is finished the data is fed to a *Custom Data Parser* which handles peculiarities such as DST time changes and missing energy price data. The cleared data is then handed on to the Storage component which saves the data to the database.

The *DB Manager* in the Storage component provides methods for data retrieval which is used by the *Data Fetch* component. Different interfaces are provided for executing queries of stored energy price data, e.g. by location, price type or custom queries. This can in turn be used by the simulator to retrieve energy price data.

The *Forecasting* component provides interfaces to *Generate Models* and *Retrieve Forecasts*.

The *Generate Models* component refers to the *Model Generation* component with *Parameter Evaluation*, *Model Training*, *Model Evaluation* and *Model Comparison*. Before model generation it refers to the Data Fetch component for retrieval of energy price data of the desired time period. After the model(s) have been generated they are saved to the *Internal Storage*, i.e. in the file system of the application server.

For retrieving forecasts the *Retrieve Forecasts* interface is queried which in turn calls the *Forecast Generation* component. Thus forecasts may be retrieved from models which have been generated and stored previously within the Internal Storage component.

fetches data from the database and does some data preprocessing that is used for forecasting. After the data has been processed and formatted appropriately it is ready to be analyzed by statistical methods to determine which forecasting model suits best for the selected time series. Different models are trained and compared and the best model is chosen. It is then selected by the Forecast Module and the actual forecasts are calculated which can be queried via a provided public interface that is exposed to external systems.

The *Cloud Framework* uses the historical price data provided by the *Data Fetch* component to run simulations for various scenarios. It is read into a local storage by a *Data Handler* for further processing of the price data in simulations. In addition, forecasts may be retrieved by the Data Handler at this stage to prevent having an open connection to the application server during simulations. Job requests are generated by the *Job Management* component which reflect certain characteristics or requirements of tasks that should be processed by the Cloud Framework. Requests and energy prices are read by the *Cloud Manager* that keeps track of the cloud's current state. The state is retrieved and modified by the scheduler which retrieves data from *Request* and *Migration Handlers* which is then fed to the *Cloud Evaluator*. The evaluator decides on needs for migrations depending on the given scenario defined by the *Scenario Handler*. It uses a *Calculator* component to compute metrics such as migration energy and VM downtime.

At each step the Cloud Evaluator gets current request and migration demands and checks metrics for VM migration. If it retrieves a positive result the resources are moved to the respective server. Results are handled by a Data Manager component after the simulation finishes. Both scheduler and simulator components are configurable by a config file defined by the *Cloud Configuration* component. Thus new configuration options are easily added and can be changed at any time to adjust the output of simulations.

5.2 Modeling migration energy

Besides the structural outline presented in the previous section important methodologies used by the cloud simulator and scheduler are investigated in detail in the following sections.

Since the goal of the simulation presented in this work is to reduce energy costs by intelligently migrating resources across geo-distributed data centers an important metric to consider is the migration energy and migration costs. An outstanding paper on this topic has been presented in [58] where the energy and downtime related to a VM migration are examined and formalized in detail. This paper has been briefly outlined in Section 2.4.

The impact of the *writable working set* (WWS) of an application on VM migration time and total downtime is described in [23] (see also Section 2.4). It is the set of most frequently updated memory pages in a running application which is dirtied in each pre-copy round of an VM migration [23, 58]. Thus a large WWS can increase migration downtime significantly with a high amount of pages dirtied in each pre-copy iteration (dirty page rate) that ultimately have to be sent at a time at the final stop-and-copy phase.

With increasing number of iterations more data has to be transferred and transmission costs rise. The final stop-and-copy phase is reached when any of the following conditions has been met (as implemented in this work):

(1) memory dirtying rate exceeds memory transmission rate

(2) the remaining dirty memory falls below a predefined threshold

(3) the number of pre-copying iterations exceeds a defined maximum

From [58] the proposed base model for VM migration has been implemented. It is based on a list of parameters defined in Table 5.1.

Migration energy, load and downtime metrics have been implemented based on the following equations:

$$\lambda = \frac{D}{R} \tag{5.1}$$

$$V_i = D\frac{V_{mem}}{R}\lambda^{i-1} = V_{mem}\lambda^i \tag{5.2}$$

$$T_i = \frac{DT_{i-1}}{R} = \frac{V_{mem}D^i}{R^{i+1}} = \frac{V_{mem}\lambda^i}{R} \tag{5.3}$$

$$V_{mig} = \sum_{i=0}^{n} V_i = V_{mem} \frac{1-\lambda^{n+1}}{1-\lambda} \tag{5.4}$$

$$T_{mig} = \sum_{i=0}^{n} T_i = \frac{V_{mem}}{R} \frac{1-\lambda^{n+1}}{1-\lambda} \tag{5.5}$$

$$n = \left\lceil \log_\lambda \frac{V_{thd}}{V_{mem}} \right\rceil \tag{5.6}$$

$$T_{down} = T_n + T_{resume} \tag{5.7}$$

$$\begin{aligned} E_{mig} &= E_{sour} + E_{dest} \\ &= (\alpha_s + \alpha_d) V_{mig} + (\beta_s + \beta_d) \\ &= \alpha V_{mig} + \beta = 0.512 V_{mig} + 20.165J \end{aligned} \tag{5.8}$$

$$V_n \leq V_{thd} \Leftrightarrow V_{mem}\lambda^n \leq V_{thd} \tag{5.9}$$

Parameter	Description
V_{mem}	The total size of the VM memory
V_{mig}	The total migration load (network traffic) during migration
T_{mig}	Total migration time
V_i	Migration load for the i-th iteration
T_i	Migration transfer time for the i-th iteration
T_{down}	Effective downtime of migration
T_{resume}	Time to resume operation of VM on other host
E_{mig}	Total migration energy
α, β	Model parameters for migration energy
R	Memory transmission rate or bandwidth
D	Dirty page rate of application
V_{thd}	Threshold of remaining memory to end pre-copy phase
λ	Convergence coefficient of VM migration
n	The index of the last pre-copy iteration
n_{max}	The maximum number of pre-copy iterations

Table 5.1: Parameters used in the migration algorithm

Equation 5.1 defines λ as the convergence coefficient of the migration since it states how fast the migration converges to the final stop-and-copy phase.

V_i and T_i in Equations 5.2 and 5.3 denote the migration load and time in pre-copy iteration i whereby these metrics depend greatly on the convergence coefficient λ. If $D < R$ then $\lambda < 1$ and migration load will eventually fall below the defined threshold V_{thd} (assuming a writable working set less than V_{thd}).

V_{mig} and T_{mig} in Equations 5.4 and 5.5 describe the total migration load and time which is defined as the sum of migration load and time for all iterations i up to the last pre-copy iteration n. Equation 5.6 defines the index of the last iteration after which the stop-and-copy phase is executed.

Equation 5.7 describes the effective downtime of the migration consisting of the time needed for the last iteration (stop-and-copy phase) and the time needed to resume operation of the newly created VM. Equation 5.8 formalizes migration energy as the sum of source and destination energy (E_{sour} and E_{dest}). Model parameters α and β can each be defined separately for source and destination (e.g. α_s and α_d). However homogenous parameters have been assumed and estimated as described in [58] for both source and destination (last line of Equation 5.8).

The condition for reaching the defined memory threshold is depicted in Equation 5.9. As defined before n is the index of the last pre-copy iteration where remaining memory should be below threshold V_{thd} (see definition of condition (2)). In case this condition is not met and $n = n_{max}$ the pre-copy phase is aborted and all remaining memory is transferred to the destination host (condition (3)). When condition (1) is detected the stop-and-copy phase is executed immediately.

5.3 SLA management

In this work the Service Level Agreement (SLA) is specified as the guaranteed availability of VMs. Cloud providers such as Google or Amazon provide SLAs for different levels of guaranteed availability [80,88]. In this work the 99.95% availability contract as depicted on the Google Compute Cloud [80] has been chosen as a standard metric valid for all VMs in simulations. The cloud provider establishes this contract in accordance with the customer which states that a running VM will have an uptime duration of at least 99.95% of the total runtime of the VM. If the total runtime exceeds one month (which is the billing period) the percentage of uptime duration is related to this period only. Thus the total downtime of a VM over the applicable time period is not allowed to exceed 0.05% of that time period (e.g. the maximum allowed downtime per hour would be $3600s * 0.0005 = 1.8s$).

In case this criterion can not be met the cloud provider is obliged to pay penalties, depending on the total downtime. The amount of penalties and the relation to the total downtime is depicted in Table 5.2.

Uptime Percentage	Percentage of penalty relative to a user's total cost
99.00% - < 99.95%	10%
95.00% - < 99.00%	25%
< 95.00%	50%

Table 5.2: VM downtime vs SLA penalties

For a VM having an uptime of below 99.95% but above 99% the user has to be paid a penalty of 10% of the total amount the user would have originally paid. Analogously if uptime falls below 99% but above 95% the user is paid back 25% and for uptimes below 95% of the time the user is refunded 50% of the price.

Different SLA availability contracts are available, i.e. Google storage provide only a reduced availability of 99.9% or even 99% [79]. However these have not been considered in this work.

5.4 Cost optimization based on utility function

Utility function definition

Problems comprising multiple criteria with different associated weights have been researched in different contexts [5,32]. These kind of problems are commonly referred to as *Multiple Criteria Decision Aid* (MCDA) where for different possibly contradicting criteria the best solution should be determined considering custom criteria weights [32].

Utility functions have been proposed as a way of solving multi criteria decision problems [1, 5]. For such problems a so called *Decision Maker* (DM) manages a set of choices or alternatives $A = \{a, b, c, \ldots\}$ and a fixed set of n criteria $G = \{g_1, g_2, \ldots, g_i, \ldots, g_n\}$, where $g_i : A \rightarrow \Re$. In [5] a *marginal weak preference relation* is further defined as $\succsim_i, i = 1, \ldots, n$ on the set of alternatives A such that $\forall a, b \in A, a \succsim_i b$. The meaning of the relation is "a is at least as good as b regarding criterion g_i". Thus for each $g_i \in G$ and $\forall a, b \in A$ it holds that $g_i(a) \geq g_i(b) \Leftrightarrow a \succsim_i b$.

A comprehensive weak preference relation $a \succsim b$ is defined as $\forall a, b \in A, a \succsim b$ and means "a is globally at least as good as b". A utility model may be defined from that relation in case it is a complete preorder [5]:

$$\forall a, b \in A : U(g(a)) \geq U(g(b)) \Leftrightarrow a \succsim b \tag{5.10}$$

with $U : \Re^n \rightarrow \Re$ and $U(g(.)) = U(g_1, g_2, \ldots, g_i, \ldots, g_n)$

As stated in [5] an additive utility function may be derived from the relation in Equation 5.10:

$$U(x) = \lambda_1 u_1(g_1(x)) + \lambda_2 u_2(g_2(x)) + \ldots + \lambda_n u_n(g_n(x)), x \in A \tag{5.11}$$

with $\lambda_1, \lambda_2, \ldots, \lambda_n \in \Re^+$ and u_i defined as non-decreasing marginal utility functions.

Based on the utility function definition in Equation 5.11 a utility function has been introduced to the Cloud Scheduler with $\lambda_1, \lambda_2, \ldots, \lambda_n \in [0, 1]$ denoting the *weights* and u_i denoting functions that normalize criteria values g_i such that $g_1, g_2, \ldots, g_i, \ldots, g_n \in [0, 1]$.

Cloud Scheduler and utility function

The goal of the Cloud Scheduler is to optimize resource scheduling with respect to current and future energy prices. In order to take into account different criteria and choose VMs exhibiting the best fit for migration a utility function has been implemented that evaluates the current cloud state and provides aid in finding VMs with best conditions for migration.

The utility function definition is outlined in Equation 5.12.

$$U(v) = \sum_{i=1}^{k} w_i c_i(v) \tag{5.12}$$

where w_i denote the weights and c_i denote the normalized criteria values which are λ_i and $u_i(g_i(x))$ in Equation 5.11, respectively. Criteria values (and thus the utility function as well) are defined s.t. a value of 0 denotes the worst result whereas a value of 1 is considered the best result, i.e. VMs associated with an utility value near 1 are very likely to be migrated.

In Table 5.3 all criteria that have been defined in the Cloud Scheduler are displayed.

Criteria	Name	Description
c_1	probability of SLA penalty	probability that an SLA penalty will occur after migration (based on experienced downtime)
c_2	estimated migration energy	the expected migration energy depending on VM memory, bandwidth and dirty page rate
c_3	remaining VM duration	number of unit time spans the job or VM is still running
c_4	data center load	load of DC where VM is located, balance load across DCs
c_5	estimated cost benefit	expected migration benefit (cost savings) given the current conditions

Table 5.3: List of utility criteria

Each of the criteria above describes one metric by which a virtual machine running within the cloud simulation can be evaluated such that values for different VMs may be compared and VMs having a utility value greater than a defined threshold may then be considered for migration. Thus a utility threshold U_{thd} is defined s.t. VMs are scheduled for migration if $U(v) \geq U_{thd}$.

Criteria definition

In the following, each criterion is formalized and described in detail.

Probability of SLA penalty As outlined in Section 5.3 it is vital for cloud providers to ensure that VM downtimes only occur rarely during their execution. Thus the *probability of an SLA penalty* is an important metric to consider when performing migrations as they may lead to extensive downtimes.

All parameters used in the following equations are outlined in Table 5.4.

Parameter	Description
v_{dur}	Total expected duration of VM v
v_{SLA}	SLA availability percentage agreed for VM v
$v_{SLA_{TH}}$	Threshold for SLA violation for VM v
loc	set of locations included in the simulation
$down_{acc}(v, t)$	accumulated downtime for VM v
$cond_{pred}(v, l, t)$	condition for prediction of down time of VM v
$dt_v(t)$	function to determine if a down time occurred for VM v at time t
$p_{SLA}(v, l, t)$	probability of SLA penalty for VM v when migrated to location l at time t
$pen_{SLA}(v, l, t)$	adjusted probability of SLA penalty when migrating VM v to location l at time t
$best_{SLA}(v, t)$	minimum probability of SLA penalties when considering all locations

Table 5.4: Parameters used in equations

The corresponding equations are outlined below.

$$dt_v(t) = \begin{cases} 1, & \text{if VM } v \text{ was down at time } t \\ 0, & \text{otherwise} \end{cases} \tag{5.13}$$

$$down_{acc}(v,t) = \sum_{z=0}^{t} dt_v(z) \tag{5.14}$$

$$v_{SLA_{TH}} = v_{dur}\left(1 - \frac{v_{SLA}}{100}\right) \tag{5.15}$$

$$p_{SLA}(v,l,t) = \frac{down_{acc}(v,t) + T_{down}(v,t)}{v_{SLA_{TH}}} \tag{5.16}$$

$$cond_{pred}(v,l,t) = down_{acc}(v,t) + T_{down}(v,t) - v_{SLA_{TH}} \tag{5.17}$$

$$pen_{SLA}(v,l,t) = \begin{cases} 1, & \text{if } cond_{pred}(v,l,t) > 0 \\ p_{SLA}(v,l,t), & \text{otherwise} \end{cases} \tag{5.18}$$

$$best_{SLA}(v,t) = \min_{l \in loc, l \neq v_{loc}} pen_{SLA}(v,l,t) \tag{5.19}$$

$$c_1(v,t) = 1 - best_{SLA}(v,t) \tag{5.20}$$

Equations 5.13 and 5.14 show formulas of aggregated downtime of VM v up to time t. $v_{SLA_{TH}}$ is the threshold in time periods that describes up to which downtime duration can be tolerated considering the SLA assigned to this VM (v_{SLA}).

A simple probability measure for experiencing an SLA penalty is shown in Equation 5.16 where the sum of accumulated downtimes $down_{acc}$ and the predicted downtime T_{down} is set in relation to the SLA threshold. If $p_{SLA} \geq 1$ an SLA penalty will occur considering the predicted downtime. The corresponding conditional equation to test expected downtime against SLA threshold is depicted in Equation 5.17.

pen_{SLA} restricts the outcome of the SLA probability p_{SLA} to the range $[0,1]$. It denotes the probability of SLA penalty when migrating VM v to location l at time t (Equation 5.18). The $best_{SLA}$ metric in Equation 5.19 calculates the min of pen_{SLA} for all locations except the one where VM v is located to evaluate the best (lowest) probability of SLA penalty at the current point in time.

Finally the criterion c_1 is described by inverting the value of $best_{SLA}$ regarding the range $[0,1]$ (Equation 5.20). The purpose of that is to provide values closer to one for “good” utility values and values close to zero for “bad” utility values.

Estimated migration energy The *estimated migration energy* for live migration of virtual machines is closely related to the total migration load that is transferred during the process [58]. The results are depending greatly on the amount of VM memory to transfer, bandwidth between source and destination hosts and dirty page rate of the application to be transferred.

This metric uses the formula for migration energy already defined in Section 5.2. The corresponding equations for this metric are outlined below.

$$E_{rel}(v,l) = \frac{E_{mig}(v,l)}{\max_{v_{mig}\in VM, v_{mig_{loc}}\neq l}(E_{mig}(v_{mig},l))} \tag{5.21}$$

$$E_{best}(v,t) = \min_{l\in loc, l\neq v_{loc}} E_{rel}(v,l) \tag{5.22}$$

$$c_2(v,t) = 1 - E_{best}(v,t) \tag{5.23}$$

where VM denotes the set of VMs active in the simulation.
The metric E_{rel} (Equation 5.21) computes the relative migration energy where the calculated energy for the current VM is related to the maximum of expected migration energy of all VMs when migrated to location l.

E_{best} denotes the minimum relative migration energy when considering migration to all locations (Equation 5.22).

c_2 is then the additive inverse of E_{best} within the interval $[0,1]$ (Equation 5.23).

Remaining VM duration The *remaining VM duration* is a simple metric to have the possibility to favor those VMs that are running for a longer period of time, e.g. to mitigate SLA penalties.

It is defined as follows:

$$remaining(v,t) = \frac{v_{rem}(t)}{\max_{vm\in VM} vm_{rem}(t)} \tag{5.24}$$

$$c_3(v,t) = remaining(v,t) \tag{5.25}$$

where v_{rem} denotes the remaining duration for VM v at time t.

Equation 5.24 is a relative measure of a VM's remaining duration in relation to the maximum of remaining durations of currently running VMs. The result criterion c_3 in Equation 5.25 is just assigned the metric itself as longer running VMs should be favored.

Data center load The *data center load* is a global metric to achieve better load balancing across data centers, i.e. VMs are favored for migration from data center locations exhibiting very high load.

$$load(v,t) = \frac{load_{DC_v}}{\max_{vm\in VM} load_{DC_{vm}}} \tag{5.26}$$

$$c_4(v,t) = load(v,t) \tag{5.27}$$

where $load_{DC_v}$ is the current load at the data center where VM v is currently located.

The relative load of the data center where VM v is currently located in relation to the maximum load is depicted in Equation 5.26. The result criterion c_4 in Equation 5.27 is again assigned the previous metric directly as VMs located at data centers exhibiting high load are favored for VM migrations.

Estimated cost benefit The *estimated cost benefit* can probably be regarded as the most important criterion above all introduced criteria as it allows to optimize VM migrations with respect to resulting costs.

This metric takes into account future energy prices in case forecasts are enabled within the simulation. Thus the estimated cost benefit is not only calculated based on current energy prices but calculates the mean for each forecast horizon and location and returns the maximum evaluated difference of prices between two locations and forecast horizons.

Equations are listed below.

$$fce(i,j,r) = \sum_{h=0}^{r} fc_i(h) - fc_j(h) \tag{5.28}$$

$$max_{fc}(v,t) = \begin{cases} \min(v_{rem}(t), fc_{max}), & \text{if } fc_{en} \text{ is true} \\ 0, & \text{otherwise} \end{cases} \tag{5.29}$$

$$es(v,t) = \max_{i \in loc, i \neq v_{loc}} fce(v_{loc}, i, max_{fc}(v,t)) \tag{5.30}$$

$$cb(v,t) = \frac{es(v,t)}{\max_{v_{mig} \in VM} es(v_{mig}, t)} \tag{5.31}$$

$$c_5(v,t) = cb(v,t) \tag{5.32}$$

where fc_{max} is the maximum forecast horizon taken into account, v_{rem} is the remaining duration for VM v (see paragraph in 5.4) and fc_{en} is a logical variable indicating whether forecasts are enabled or not.

The mean difference (forecast estimation) between each two locations summed up over all forecast horizons up to a defined maximum is given in Equation 5.28. For two given locations i and j the mean difference is calculated up to a maximum of r forecast horizons.

In Equation 5.29 the maximum applicable forecast horizon is determined which is based on the remaining VM duration but can not exceed fc_{max}. The remaining duration is taken into account as beneficial price differences can only be taken advantage of as long as the VM is still active.

Equation 5.30 denotes the maximum estimated cost benefit when migrating VM v at time t. It determines the maximum value across all forecast estimations fce for the given VM. The resulting cost benefit is defined in Equation 5.31 where the estimated cost benefit of VM v is set in relation to the maximum cost benefit of all VMs.

The criterion for the estimated cost benefit is set to the resulting cost benefit since VMs with the highest expected cost benefit should be more likely to be migrated (Equation 5.32).

Discussion

The presented metrics have been chosen to get a diverse set of criteria where different criteria may be emphasized. Some of the criteria are opposite to each other, e.g. minimization of SLA penalties is orthogonal to cost optimization as one cannot be minimized without affecting the other one. The goal was to provide a flexible means of prioritizing criteria to be able to adjust to different circumstances. Additional criteria may be added to the framework at any time.

CHAPTER 6

Evaluation and Results

6.1 Definition of simulation scenarios

In this section the simulation scenarios and settings are described to get a complete picture of the simulated cloud environment.

The simulation scenarios can be divided into two parts which are day ahead and real time simulations based on day ahead and real time energy prices, respectively. For each type of simulation different locations and energy markets have been chosen. The basic scenario types are oulined below.

Day ahead simulation scenario

The simulation scenario for day ahead energy prices is comprised of data centers located in Europe and the USA. A map showing five data centers as a basis for day ahead simulations is outlined in Figure 6.1.

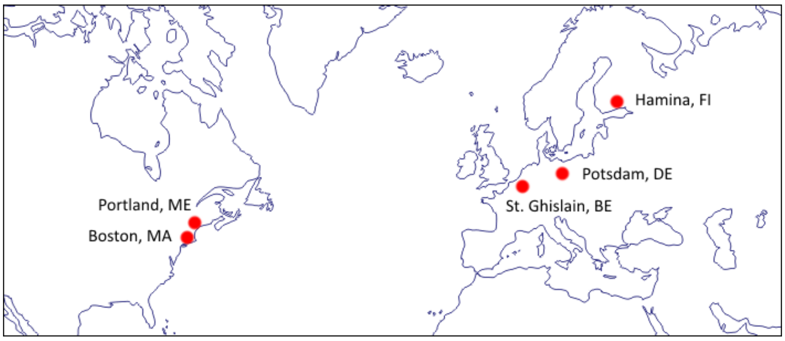


Figure 6.1: Map of data center locations for day ahead price simulations

The locations have been chosen such that a diverse set of data from different energy markets is available. In addition time zone effects should be modeled by placing data centers sufficiently far from each other such that daily variation of energy prices in different time zones can be utilized.

A list of day ahead energy markets and corresponding locations is shown in Table 6.1.

Energy market	Location
Nord Pool Spot	Hamina, Finland
Belpex	St. Ghislain, Belgium
EPEXSpot	Potsdam, Germany
ISO New England	Portland, Maine
ISO New England	Boston, Massachussetts

Table 6.1: List of day ahead energy markets and locations

Real time simulation scenario

For the real time simulation scenario data centers have been chosen exclusively from location within the US. Thus energy prices are more localized and performance of simulations should be evaluated for datasets with similar characteristics in contrast to day ahead simulations.

A map showing data centers for all real time markets and locations is depicted in Figure 6.2.

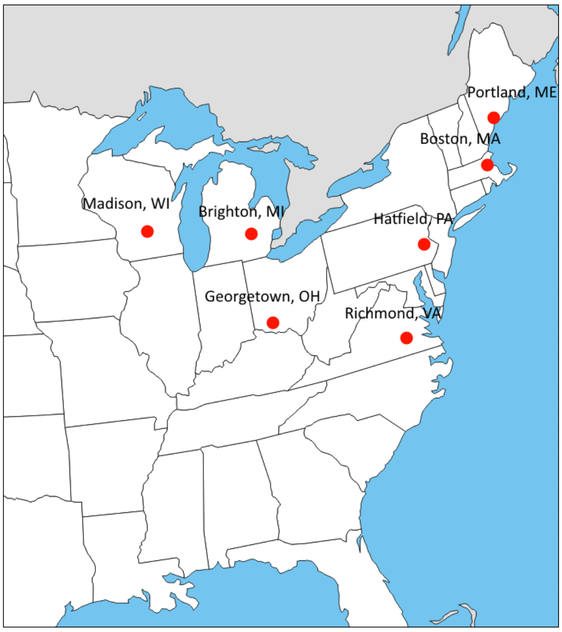


Figure 6.2: Map of data center locations for real time price simulations

A list of real time energy markets and locations is outlined in Table 6.2.

Energy market	Location
ISO New England	Portland, Maine
ISO New England	Boston, Massachussetts
PJM	Richmond, Virginia
PJM	Brighton, Michigan
PJM	Hatfield, Pennsylvania
PJM	Madison, Wisconsin
PJM	Georgetown, Ohio

Table 6.2: List of real time energy markets and locations

Simulation Configuration Parameters

Cloud settings

As the simulator has been built to be highly configurable [62] different cloud parameters can be adjusted to run simulations with different settings. In Table 6.3 the most relevant simulation parameters are listed.

The number of servers and VMs per simulation has been adjusted such that the resulting utilization is reasonable, e.g. with 1000 servers and 2000 VMs in total an average utilization of only about 1 % has been achieved over a simulation period of 5 weeks. Thus raising the number of VMs and in turn reducing number of servers led to a more realistic utilization of 30% to 40%.

The relation of number of resources per server to the amount of VM resources (i.e. number of cpus, memory) has been defined such that capacity constraints will not be an issue in simulations. The same method has been applied to the relation of total number of servers to number of VMs.

The dirty page rate has a direct impact on VM migration performance (see Section 5.2) and therefore different values have been assigned to VMs to simulate different application behavior. In combination with bandwidth this may lead to a high variation in migration downtimes.

The SLA level has been set to the most common value for internet cloud services [80, 88]. With different penalty levels greater downtimes directly impact resulting costs for cloud providers.

A different set of duration values has been assigned to VMs in order to simulate different kinds of workloads. This includes VM durations of 1 to 48 hours for scientific or HPC applications. Long running VMs have been omitted to focus on short term applications.

The VM price most resembles the price for a “n1-standard-1” VM on Google Compute Cloud[1]. As this is one of the smallest VMs available in the Google Cloud this price is still comparably cheap.

Bandwidth is divided into low, medium and high values to simulate real world scenarios where high bandwidth may not always be available. As bandwidth has a direct impact on migration downtime this allows to evaluate the total resulting downtime and costs in different scenarios.

---

[1] https://cloud.google.com/compute/pricing

Metric	Values	Description
number of servers	500	The total number of servers across all locations
number of virtual machines	5000	The total number of virtual machines over the whole simulation time period
server cpus	$[4, 8]$	The number of CPUs per server uniformly distributed over the given interval
server memory	$[8, 16]$	The amount of memory per server uniformly distributed over the given interval
virtual machine cpus	$[1, 4]$	The number of CPUs per VM uniformly distributed over the given interval
virtual machine memory	$[1, 4]$	The amount of memory per VM uniformly distributed over the given interval
dirty page rate	$\{20, 40, 70, 90\}$	One of four dirty page rates in MB/s assigned to each VM
SLA level	99.95%	A fixed SLA availability level applied to each VM
VM duration	$\{1, 2, 5, 8, 12, 24, 48\}$	A fixed set of duration values in hours applied uniformly to the set of VMs
VM price	4 ¢ / h	The price of a VM in cent per hour
bandwidth	$\{400, 800, 1000\}$	One of 3 amounts of bandwidths in MBit/s set between each two locations
bandwidth cost	0.1 ¢ / GB	Bandwidth costs in cent per GB
server peak power	200 W	The power a server draws when its resources are fully utilized
server idle power	100 W	The power a server draws when idle
cpu resource weight	0.7	The weight associated with cpu power when calculating server utilization
memory resource weight	0.3	The weight associated with memory when calculating server utilization

Table 6.3: List of cloud configuration parameters in the simulation

Bandwidth costs have been set to a fixed price per amount of traffic while it is usually paid by fixed price contracts with 95/5 bandwidth constraints [81]. These contracts measure traffic in fixed time intervals and the 95-th percentile of total usage is charged. To simplify the model in this work it has been set to a fixed price per volume.

Server peak and idle power have been set where a server at peak load consumes double the amount of power than when idle. This may be too optimistic as it is stated that servers consume about 60% of power when idle [65]. However studies show that new server technology is able to achieve power reductions up to 75% from peak to idle mode [26]. A greater ratio of peak to idle power results in greater possibility for energy savings. Therefore the idle-to-peak ratio is critical for any scheduling algorithms aiming to reduce costs based on energy.

CPU and memory resource weights have been defined to calculate resulting power consumption from server utilization of each resource. From investigations in [53, 65] it can be deduced that CPU is the most power consuming component where memory power consumption amounts to about one third up to equal amounts of CPU power. Therefore CPU and memory weights have been set to accommodate those findings.

Cloud scheduler

The core of the simulation is represented by the Cloud Scheduler. It is responsible for managing requests and for deciding which VMs should be migrated at which point in time. To make it a flexibel approach and be able to compare different scenarios the scheduler can be configured to run one out of seven different scenarios (Table 6.4).

	Request assignment	**Migration handling**
BFD baseline scheduler	not cost aware, based on load	None
BCF scheduler	cost aware, no forecasts	None
BCF scheduler + FC	cost aware, forecasts	None
BCF scheduler + Ideal FC	cost aware, ideal forecasts	None
BCU scheduler + M	cost aware, no forecasts	cost aware, no forecasts
BCU scheduler + M + FC	cost aware, forecasts	cost aware, forecasts
BCU scheduler + M + ideal FC	cost aware, ideal forecasts	cost aware, ideal forecasts

Table 6.4: Existing scheduling scenarios

The first scheduler is a *Best Fit Decreasing* (BFD) Scheduler that does no investigations on energy prices at all. It is based on a load balancing approach which aims to assign requests to the least occupied data center and utilize as few servers as possible to save energy costs. The BFD Scheduler is outlined in algorithm 6.1.

The BFD scheduler iterates over the set of VM requests at the current point in time and aims to optimize placement of VMs by considering resource utilization of VMs and PMs for maximum consolidation. VM requests are sorted by resources in descending order to process VMs exhibiting large resource requirements first (line2). Locations are sorted by utilization ascending to assign requests first to under utilized locations (line 3). Lines 4 and 5 denote sorting of PMs by (free) capacities to consider most occupied servers first.

```
1  Function BFD Scheduler:
       Data: Input Requests
       Result: Requests assigned according to BFD Schedule
2      sort VM requests with resource utilization in descending order;
3      sort locations by utilization in ascending order;
4      sort available PMs by location and free capacities in ascending order;
5      sort inactive PMs by capacities in ascending order;
6      foreach vm in VM requests do
7          while vm not assigned and size(available PMs) > 0 do
8              foreach pm in available PMs do
9                  if vm fits on pm then
10                     assign vm to pm;
11                     remove vm from available PMs;
12                     break;
13                 end
14             end
15             if vm not assigned and size(inactive PMs) > 0 then
16                 add first of inactive PMs to available PMs;
17                 sort available PMs by location and free capacities ascending;
18             else
19                 break;
20             end
21         end
22     end
```

Algorithm 6.1: Best fit decreasing scheduler

The BFD algorithm iterates over the set of requests and tries to assign VMs to hosts as long as non-fully occupied servers are available (line 7,8-14). In case a VM could not be assigned a server from the set of inactive hosts is activated and the aforementioned procedure is executed again (lines 15-17). In case no suitable host could be found or no further hosts are available the loop is exited with the current VM not being assigned (lines 18-19).

As discussed before Table 6.4 describes further scenarios with different scheduling algorithms. These algorithms are based on the *Best Cost Fit* (BCF) and *Best Cost Utility* (BCU) Scheduler with different variations to include migrations (denoted by *M*), forecasts (denoted by *FC*) and ideal forecasts (denoted by *ideal FC*).

The next discussed algorithm is the one from the Best Cost Fit scheduler which aims to maximize cost savings when assigning requests utilizing price forecasts. The corresponding algorithm is outlined in 6.2.

The best cost fit scheduler takes three inputs: The VM requests, a flag denoting whether forecasts should be enabled and a second flag for possible inclusion of ideal forecasts. In line 2 requests are sorted descending by their resource requirements which resembles the step in the BFD scheduler.

```
1  Function BCF Scheduler:
       Data  : Input Requests
               flag for forecasts (FC)
               flag for ideal FC (idealFC)
       Result: Requests assigned according to BCF Schedule and given scenario
2      sort VM requests with resource utilization in descending order;
3      maxH <- 1;
4      if FC then
5          maxH <- get max fc horizon from config;
6      end
7      foreach vm in VM requests do
8          dur <- get remaining duration for vm;
9          if dur <= 1 then
10             FC <- false;
11             idealFC <- false;
12         else
13             maxH <- min(dur,maxH);
14         end
15         fcData <- get sorted price and forecast data (FC, idealFC);
16         foreach data in fcData do
17             h <- get forecast horizon (data);
18             loc <- get location (data);
19             assign to host BFD (vm, loc);
20             if assigned then
21                 block vm for the next h time periods;
22                 break;
23             end
24         end
25     end
```

Algorithm 6.2: Best cost fit scheduler

Lines 3-6 determine the maximum forecast horizon that should be taken into account when considering future energy prices (in case forecasts are enabled). From lines 7-11 the set of VM requests is iterated and forecasts are enabled based on the remaining duration. Line 13 determines the forecast horizon that will be applied through a minimum function of the remaining VM duration and the maximum forecast horizon defined previously.

In line 15 a procedure is called to retrieve locations and best estimated forecast windows for these locations. Thus for each location and forecast horizon the average estimated energy price is calculated and results are sorted by price ascending such that the location with the least expected costs regarding a specific forecast horizon is placed first.

In lines 16 to 24 the resulting dataset from line 15 is iterated and the forecast horizon and location from the cheapest to more expensive locations are retrieved. For each of these data it

```
Function BCU Scheduler:
    Data  : Input Requests
            Current cloud state
            flag for forecasts (FC)
            flag for ideal FC (idealFC)
    Result: A list of VMs with resulting utility value above the utility threshold
    [slaPen,migEnergy,maxMigEnergy,remDur,maxRemDur,util,estimatedSavings] <-
      prepare utility function (FC, idealFC);
    uResult <- [];
    foreach vm in VM requests do
        currentLoc <- getLocation(vm);
        uValue <- [];
        foreach loc in locations do
            if loc != currentLoc then
                slaPenalty <- 1 - slaPen[vm][loc];
                migrationEnergy <- 1 - (migEnergy[vm][loc] / maxMigEnergy);
                remainingDur <- remDur[vm] / maxRemDur;
                dcLoad <- getRelativeDCLoad (util, currentLoc);
                costSavings <- estimatedSavings[vm][loc];
                result <- wSLA * slaPenalty + wMig * migrationEnergy + wDur *
                  remainingDur + wLoad * dcLoad + wCost * costSavings;
                uValue [loc] <- result;
            end
        end
        maxUValue <- max(uValue);
        append (uResult, (vm, getLocation(maxUValue), getUtilityValue(maxUValue)));
    end
    sort uResult by utility value descending;
    result <- [];
    uTH <- get utility TH from config ();
    foreach r in uResult do
        if utilityValue(r) > uTH then
            append (result, r);
        end
    end
    return result;
```

Algorithm 6.3: Best cost utility scheduler

is tried to assign the VM to a host from the given location based on a BFD approach. If the VM could be assigned successfully the VM will be blocked for migrations for the next h time periods to prevent oscillation of VMs between different locations.

The last scheduler is the Best Cost Utility Scheduler that takes into account VM migrations between different locations based on a utility function (see Section 5.4 for a definition of the utility function). The algorithm of the BCU scheduler is shown in 6.3.

Line 2 of the algorithm calls the procedure *prepare utility function* that returns a list of metrics depending on the flags *FC* and *idealFC*. Results of utility calculations are stored in *uResult* in line 3.

From lines 4-6 start iterating over the set of VM requests and prepare the current location and a list of utility values. In addition lines 7 and 8 start iterating over the set of locations where the current location is skipped for migration. From line 9 to 13 the different criteria values are normalized from the results of the procedure in line 2. A list of criteria regarding this utility function can be found in Section 5.4.

In line 14 the total utility value is calculated where each normalized criteria is weighted with its corresponding weight. The total utility value is stored in line 15 for the current VM *vm* and location *loc*. Line 18 calculates the maximum utility value for this VM over all locations. That is, the most promising location where VM *vm* could be migrated to is estimated.

In line 19 a tuple consisting of *(vm, loc, uvalue)* is added to the result utility list where *uvalue* is the maximum of the calculated utility values for VM *vm* associated with location *loc*. Line 21 sorts the result utility list by utility value descending and line 22 and 23 are preparing the final result list and the utility threshold, which is taken from a configuration file.

Lines 24 to 28 iterate over the resulting utility list and add values to the final result list in case the corresponding utility value exceeds the predefined utility threshold *uTH*. The algorithm is concluded by returning the list of result VMs with total utility values above the utility threshold in line 29.

Cost models

Cost models and related metrics are an important base for transforming aggregated power values to actual costs. Therefore a number of formulas have been defined to accurately model the resulting power consumption and energy costs.

Utilization of servers The basis for modeling power consumption is to define resource utilization of servers since server utilization relates directly to resulting power consumption [65]. Server utilization based on a set of resources R is defined in Equation 6.1.

$$s_{util}(t) = \sum_{r \in R} w_r \frac{s_{used}(r,t)}{s_{cap}(r)} \tag{6.1}$$

with w_r denoting the weight of resource r, s_{used} is the currently used capacity of server s regarding resource r at time t, s_{cap} denotes the total capacity of server s of resource r and s_{util} being the resulting utilization of the server.

Utilization per location In order to calculate data center load for each location a method of computing the average utilization per location is needed. It is defined as the average utilization of all servers in a given location (Equation 6.2):

$$util_{loc}(t) = \frac{1}{|loc_s|} \sum_{s \in loc_s} s_{util}(t) \tag{6.2}$$

where loc_s being the set of servers at location loc and $util_{loc}$ denotes the current utilization at location loc.

Power per location From the previous metric *utilization per location* the resulting cloud power per location can be derived (Equation 6.3).

$$power_{loc}(t) = util_{loc}(t) \cdot |loc_{active_s}(t)| \cdot (P_{peak} - P_{idle}) + |loc_{active_s}(t)| \cdot P_{idle} \tag{6.3}$$

where $|loc_{active_s}(t)|$ is the number of active servers at location loc at time t, P_{peak} is the peak power of the server and P_{idle} describes the idle power of the server.

Total cloud power The total cloud power is the sum of power consumption over all locations L and time periods T (Equation 6.4).

$$m_{TCP} = \sum_{t \in T} \sum_{loc \in L} power_{loc}(t) \tag{6.4}$$

Migration load This metric serves as the basis for calculating migration energy as it directly depends on migration load. In addition it is used for further metrics such as migration costs. The formula for aggregated migration load per time t is outlined in Equation 6.5.

$$mig_l(t) = \sum_{v \in VM} V_{mig}(v, t) \tag{6.5}$$

where VM denotes the set of all active VMs at time t and $V_{mig}(v, t)$ is the total migration load for vm v for a migration starting at time t. The latter expression is taken from the definition of migration energy in Section 5.2.

Migration energy Modeling migration energy is needed for calculating the migration costs and it may also give a hint for resulting downtime and possible penalty costs due to the amount of migration energy consumed. The formulas for summed migration energy over the set of all VMs VM and total migration energy over a simulation time period are denoted in Equations 6.6 and 6.7.

$$mig_e(t) = \sum_{v \in VM} E_{mig}(v, t) \tag{6.6}$$

$$m_{ME} = \sum_{t \in T} mig_e(t) \tag{6.7}$$

Migration cost The migration costs are defined on the one side by combining resulting migration energy with current energy prices and on the other side by combining migration load and bandwidth costs.

$$p_{mean}(t) = \frac{p_{l1}(t) + p_{l2}(t)}{2} \tag{6.8}$$

$$mig_c(t) = mig_e(t) \cdot p_{mean}(t) \tag{6.9}$$

$$mig_{c_{bw}}(t) = mig_l(t) \cdot c_{bw} \tag{6.10}$$

$$m_{MC} = \sum_{t \in T} (mig_c(t) + mig_{c_{bw}}(t)) \tag{6.11}$$

Equation 6.8 shows the mean price calculated from prices p_{l1} and p_{l2} in locations $l1$ and $l2$ at time t. Equation 6.9 describes the migration costs depending on the migration energy and mean energy price p_{mean} at time t. The bandwidth related costs for migration are outlined in Equation 6.10 with migration load and bandwidth costs combined (i.e. amount of GB multiplied by ¢ / GB). The migration cost metric m_{MC} in Equation 6.11 is then the sum of both energy and bandwidth costs.

Cost of VM The resulting costs v_c for a user of a VM is defined as the price v_p set for the VM multiplied by its duration up to time t (Equation 6.12).

$$v_c(t) = v_p \cdot v_{dur}(t) \tag{6.12}$$

Penalty cost The total penalty costs are the aggregated costs for all penalties and VMs occurred during a simulation. Penalty costs occur when the maximum allowed downtime for a VM regarding an SLA threshold is exceeded. Depending on the threshold that has been surpassed different penalty costs have to be paid. As described in Section 5.3 three different penalty thresholds are defined which result in different penalty costs.

$$SLA_{TH_i}(v,t) = v_{dur}(t) \left(1 - \frac{v_{SLA_i}}{100}\right), i \in \{1,2,3\} \tag{6.13}$$

$$v_{SLA_1} = 99.95, v_{SLA_2} = 99, v_{SLA_3} = 95 \tag{6.14}$$

$$v_{pen}(t) = \begin{cases} 0.5, & \text{if } down_{acc}(v,t) > SLA_{TH_3}(v,t) \\ 0.25, & \text{if } down_{acc}(v,t) > SLA_{TH_2}(v,t) \\ 0.1, & \text{if } down_{acc}(v,t) > SLA_{TH_1}(v,t) \\ 0, & \text{otherwise} \end{cases} \tag{6.15}$$

$$v_{c_{pen}}(t) = v_c(t) \cdot v_{pen}(t) \tag{6.16}$$

$$m_{TPC} = \sum_{v \in VM} v_{c_{pen}}(t_{last}) \tag{6.17}$$

Equation 6.13 defines an SLA threshold for either of the defined SLA levels in Equation 6.14. In Equation 6.15 penalty values in the interval [0,1] are assigned based on the SLA threshold reached. This penalty value is then applied to the costs of the VM in Equation 6.16 resulting in penalty costs for VM v. Equation 6.17 calculates the total penalty costs across all VMs at the last simulation time stamp t_{last}.

Total downtime The total downtime is calculated from the sum of downtimes of all VMs over the whole simulation time period (up to t_{last}, Equation 6.18).

$$m_{TDT} = \sum_{v \in VM} down_{acc}(v, t_{last}) \tag{6.18}$$

Number of migrations The total number of migrations is defined as the sum of migrations at each time stamp t (Equation 6.19).

$$m_{NM} = \sum_{t \in T} mig_{num}(t) \tag{6.19}$$

where $mig_{num}(t)$ denotes the number of migrations at time t.

Total cloud costs The total cloud costs are calculated as the sum of cloud power values multiplied by the price at the respective location and time (Equation 6.20).

$$m_{TCC} = \sum_{t \in T} \sum_{loc \in L} power_{loc}(t) \cdot p_{loc}(t) \tag{6.20}$$

Total cloud power with migrations Calculating the total cloud power including migrations is a simple addition of the total cloud power and migration energy (Equation 6.21).

$$m_{TCPWM} = m_{TCP} + m_{ME} \tag{6.21}$$

Total cloud cost with migrations Analogously to *Total power with migrations* this metric is defined as the sum of total cloud cost and migration cost (Equation 6.22).

$$m_{TCCWM} = m_{TCC} + m_{MC} \tag{6.22}$$

Total power The total power metric encompasses the metric *Total cloud power with migrations* to describe the total power consumption in the simulation (Equation 6.23).

$$m_{TP} = m_{TCPWM} \tag{6.23}$$

Total cost Total cost denote the sum of all cost occurred during simulation which includes the total cloud cost with migrations plus total penalty costs (Equation 6.24).

$$m_{TCCWM} = m_{TCCWM} + m_{TPC} \tag{6.24}$$

6.2 Utility function optimization

As explained in Section 5.4 the scheduler incorporates a utility function to provide a means of flexibel adjustments to different cloud criteria. This section aims to optimize the parameters of the utility function regarding both energy cost savings and SLA penalties. The resulting settings will then be used in large scale simulations which will be presented in Section 6.3.

In the following simulations the scheduler operates on the defined cloud parameters in Section 6.1. For the utility test scenarios outlined in this section the number of servers and VMs is changed to a total of 1000 servers and 2000 VMs over the whole simulation time period. This gives sufficient opportunity to test the utility function but does not require huge computational resources.

The simulation runs in this section are based on 5 weeks of energy price data from day ahead markets with locations defined in Section 6.1. As a common setting to these simulations the following bandwidth connections have been defined for day ahead scenarios between locations (Table 6.5). To simplify the scenario all bandwidth connections are symmetric, i.e. the same in- and outgoing bandwidth connections have been used between each two locations.

Source	Destination	Bandwidth
Hamina	Potsdam	1000 MBit/s
Hamina	St. Ghislain	800 MBit/s
Potsdam	St. Ghislain	800 MBit/s
Boston	Portland	800 MBit/s
Europe	USA	400 MBit/s

Table 6.5: Bandwidth connections in day ahead market locations

The connection Europe to USA denotes connections from any country from Europe to any country in the USA which is mapped to 400 MBit/s.

In order to easily distinguish between different schedulers a characterizing symbol has been introduced to designate each scheduler. The resulting mapping is shown in Table 6.6.

Scheduler name	Symbol
BFD baseline scheduler	BFD
BCF scheduler	BCF
BCF scheduler + FC	BCF_F
BCF scheduler + Ideal FC	BCF_IF
BCU scheduler + M	BCU_M
BCU scheduler + M + FC	BCU_MF
BCU scheduler + M + ideal FC	BCU_MIF

Table 6.6: Existing scheduling scenarios

For a reference of all implemented cloud metrics shown in subsequent graphs these metrics are outlined in Table 6.7 (see also Section 29).

Cloud metric	Symbol	Description
Total cloud power	TCP	The total power consumed by active servers aggregated over all locations and time periods (in kWh)
Total cloud cost	TCC	The total costs resulting from applying energy prices to power consumption over simulation times (in $)
Migration energy	ME	Migration energy summed up from all migrations over the simulation time period (in kWh)
Migration cost	MC	Migration costs summed up from all migrations over the simulation time period (in $)
Total cloud power with migrations	TCPWM	The total cloud power including the migration energy (in kWh)
Total cloud cost with migrations	TCCWM	The total cloud costs including the resulting migration costs (in $)
Total penalty cost	TPC	Total penalty costs occurred during the simulation time period (in $)
Total downtime	TDT	Total downtime aggregated over the simulation time period (in sec)
Number of migrations	NM	Number of migrations triggered over the simulation time period
Total power	TP	The total cloud power including the migration energy (in kWh)
Total cost	TC	The total cloud costs including migration costs and total penalty costs (in $)

Table 6.7: List of all measured cloud metrics

As might be noticed the metrics *Total cloud power with migrations* and *Total power* are defined the same since on the one side no more power values are to be added and on the other side a total metric comparable to the *Total cost* should to be defined.

In the following sections three different scenarios are defined to show the impact of changing utility parameters on resulting costs and SLA penalties.

Scenario1: Energy utility function

The first scenario consists of parameters to emphasize the reduction of migration energy consumption whereby less weights have been put on prevention of SLA penalties. The utility criteria and corresponding weights are outlined in Table 6.8.

We can observe that the *estimated cost benefit* and *estimated migration energy* criteria have been given the highest weights whereas other criteria such as *probability of SLA penalty* are set to lower weights. The utility threshold value of 1.8 is the minimum value that has to be reached by a VM's utility function for the VM to be considered eligible for migration.

Index	Name of criterion	weight
c_1	probability of SLA penalty	0.4
c_2	estimated migration energy	0.7
c_3	remaining VM duration	0.5
c_4	data center load	0.4
c_5	estimated cost benefit	0.8
ut	utility threshold	1.8

Table 6.8: List of energy utility criteria with associated weights

Since all criteria values are normalized to the interval $[0, 1]$ it can be quite easily estimated which values of the different criteria lead to utility values above the utility threshold. A trivial method of estimating utility weights is to set all criteria values to 1 and calculate the sum of their weights. This should result in a value signficantly greater than the utility threshold. One could also put particularly high emphasis on one criterion by keeping the sum of weights of all other criteria below the weight of this criterion. Then by setting the utility threshold to a value at least as high as that specific criterion it is assured that a sufficiently high value of this criterion is required to get above the utility threshold.

The results for normalized power vs cost values are depicted in Figure 6.3.

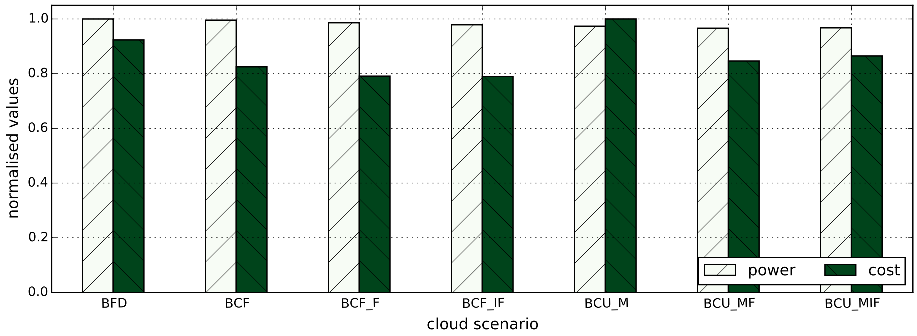


Figure 6.3: Normalized cost vs power values based on the energy utility scenario

This graph outlines the resulting power and costs for this utility scenario across all defined schedulers (see Section 6.1 for a definition of the schedulers). What can be observed is that power values differ only marginally while energy costs differ substantially between different schedulers. The BFD scheduler is taken as the baseline scheduler to which all other schedulers are compared.

In this scenario the BFD scheduler exhibits the highest power consumption among all schedulers but less total costs than the BCU_M scheduler. Overall the BCU schedulers (BCU_M, BCU_MF, BCU_MIF) are not able to reduce costs as much as BCF schedulers (BCF, BCF_F, BCF_IF). Thus the utility function which is applied in BCU schedulers is not well calibrated.

This behavior can be explained as we put on a high weight for the *estimated cost benefit* criteria but a low weight on *probability of SLA penalties* which allows the use of excessive migrations that can result in increased SLA penalty costs.

The total resulting costs including migration costs and SLA penalty costs is shown in Figure 6.4. What is clearly visible are the added penalty and migration costs for the schedulers BCU_M, BCU_MF and BCU_MIF. Therefore even though the cloud cost may be reduced compared to other schedulers the total costs are higher due to these additional costs.

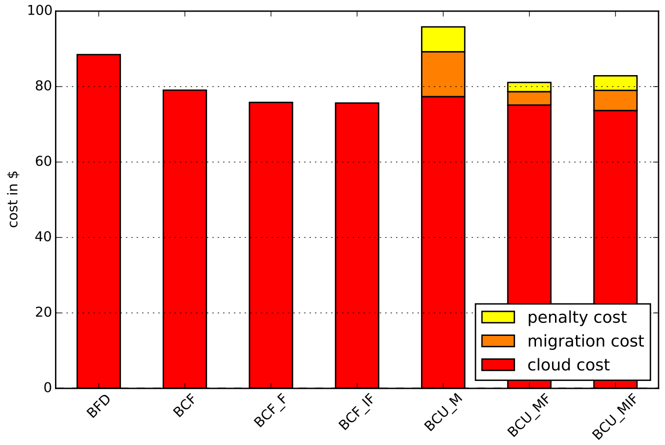


Figure 6.4: Total resulting costs per scheduler for the energy utility scenario

A diagram summarizing all the cloud metrics is depicted in Figure 6.5. For a list of cloud metrics see Table 6.7.

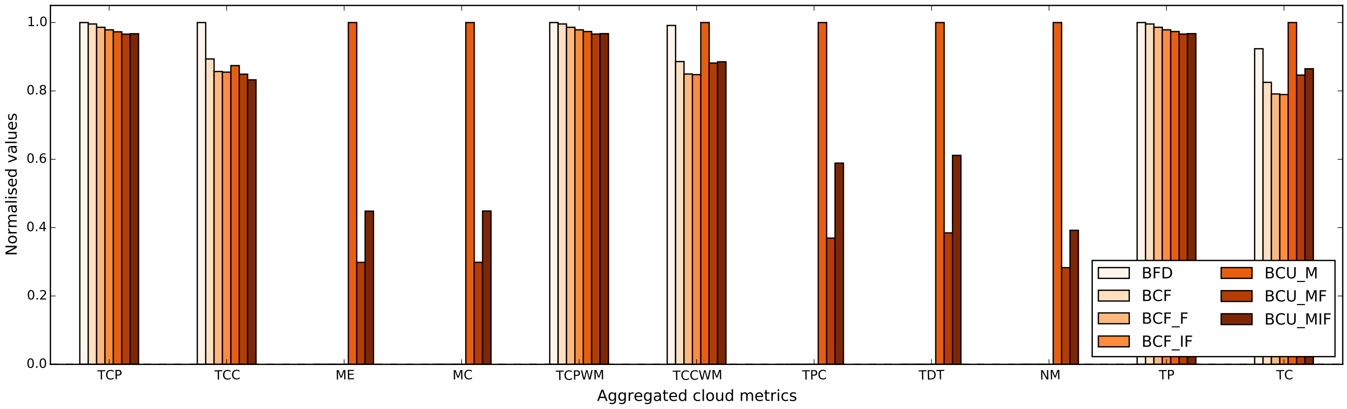


Figure 6.5: Cloud metrics calculated for the energy utility over the simulation time period

This diagram shows normalized values of cloud metrics over the whole simulation time period. Significant deviations of values can be observed for different schedulers. The results coincide with the graphs shown before as total cloud power values decrease only marginally while total cloud costs are reduced significantly (i.e. BFD scheduler compared to BCU_MIF scheduler for metrics TCP and TCC).

Since migration energy and migration costs are directly proportional the normalized values are exactly the same. They are highest for the BCU_M scheduler and decrease for BCU_MF and BCU_MIF schedulers since the latter ones include forecasts in the calculations. Other schedulers do not exhibit any migration energy or costs since no migrations are performed.

The total cloud power with migrations (TCPWM) is almost identical to the TCP metric since migration energy only accounts for a very small fraction of the total cloud power. The total cloud costs with migrations (TCCWM) show a significantly different behavior due to the addition of migration costs based mainly on bandwidth costs.

In the total penalty cost (TPC) metric the BCU_M scheduler exhibits the highest penalty costs. It shows a similar behavior to the TCCWM metric as higher migration costs and therefore a higher number of migrations generally lead to increased penalty costs. In accordance with the penalty costs and migration costs the total downtime shows similar patterns as penalty costs are directly related to VM downtime.

The number of migrations (NM) metric again shows similar behavior as the aforementioned metrics all relate to the total number of migrations. The total power (TP) metric is identical to the TCPWM metric and shown on the right for easier comparison to the total cost (TC) metric.

The total cost metric adds to the pattern of TCCWM as in addition to cloud and migration costs the penalty costs are taken into account. Thus it can be concluded that due to high migration and penalty costs the BCU schedulers result in increased costs compared to the BCF schedulers.

Scenario2: Cost utility function

The cost utility function is defined as utility scenario 2 where the focus has been laid on high cost optimization with other metrics having low priority.

In Table 6.9 the weights for the cost utility function are displayed.

Index	Name of criterion	weight
c_1	probability of SLA penalty	0.3
c_2	estimated migration energy	0.4
c_3	remaining VM duration	0.2
c_4	data center load	0.1
c_5	estimated cost benefit	1.0
ut	utility threshold	1.4

Table 6.9: List of cost utility criteria with associated weights

A very high value has been set for the *estimated cost benefit* criterion equaling to the sum of all other criteria. The utility threshold has been set to a value higher than the sum of all criteria but the cost benefit, therefore this threshold cannot be reached without a significantly high value of the cost benefit.

Probability of SLA penalty and *estimated migration energy* are weighted higher than other criteria but exhibit still a low value compared to the cost benefit. Thus it is expected that costs will be optimized with still a fraction existing for migration and penalty costs.

The normalized cost vs power graph is shown in Figure 6.6.

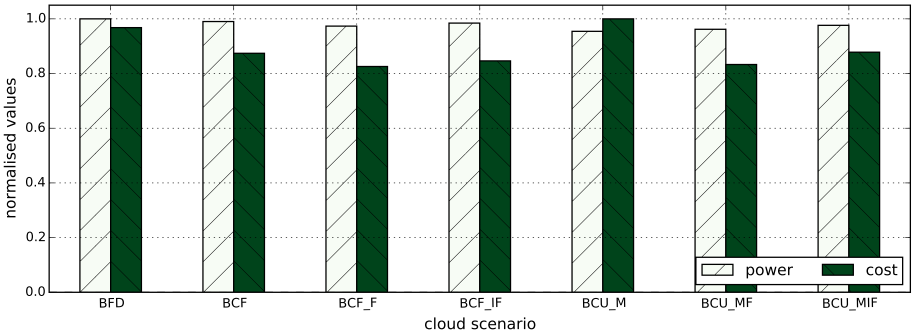


Figure 6.6: Normalized cost vs power values based on the cost utility scenario

This graph shows more evenly distributed cost values than the one from scenario 1 (Figure 6.3). Thus the BFD scheduler almost reaches the maximum cost value but the BCU_M scheduler still shows the highest costs of all schedulers. It can be noticed that the BCU schedulers are at the same levels as BCF schedulers with the BCU_MF scheduler showing only a minor increase in costs in comparison to the BCF_F scheduler exhibiting the least costs.

The total costs are depicted in Figure 6.7. Compared to the total cost chart of scenario 1

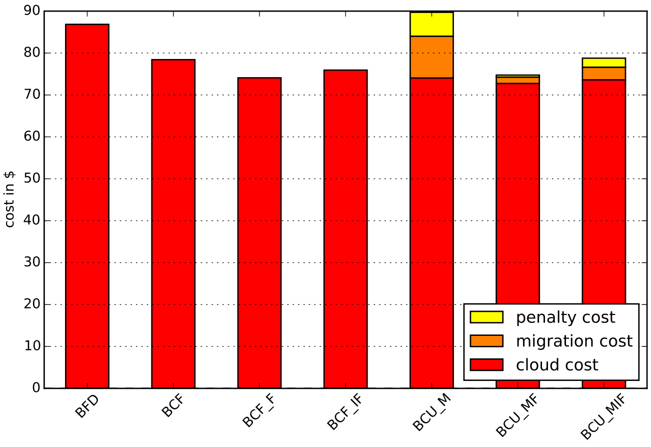


Figure 6.7: Total resulting costs per scheduler for the cost utility scenario

(Figure 6.4) the total costs for all schedulers except the BFD scheduler have been reduced. Also the fraction of migration and penalty costs could be reduced except for the BCU_M scheduler with a slight increase of penalty costs.

In Figure 6.8 the resulting cloud metrics are shown.

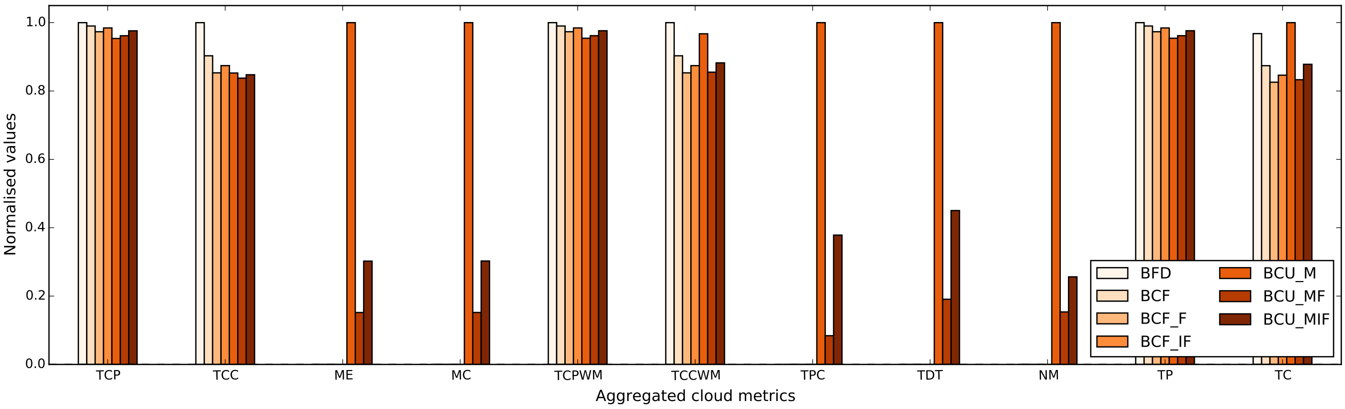


Figure 6.8: Cloud metrics calculated for the cost utility over the simulation time period

The most obvious changes compared to the energy utility scenario are the decreased migration energy, migration cost, total penalty cost, total downtime and number of migration criteria (ME, MC, TPC, TDT and NM) for the BCU_MF and BCU_MIF schedulers in relation to the BCU_M scheduler.

Thus fewer migrations lead to fewer downtimes, SLA violations and penalties and also migration energy and costs. The reason that the latter two BCU schedulers perform significantly better than the BCU_M scheduler is the fact that the *estimated cost benefit* metric has the most impact when applied to an extended time series with forecasts.

Scenario3: Best utility function

The best utility function has been defined as the scenario with the highest cost savings compared to other scenarios. It puts emphasis on both reduction of SLA penalties as well as optimization of energy costs during the simulation.

The defined weights assigned to the different criteria are denoted in Table 6.10.

As can be observed the *probability of SLA penalty* and *estimated cost benefit* metrics are given the highest possible weights while other criteria are weighted almost negligibly. The utility threshold amounts to the sum of the two highest weights making sure that only VMs with high relative values for metrics c_1 and c_5 are considered for migrations.

The power vs cost chart is given in Figure 6.9. The normalized costs are more evenly distributed across the schedulers than for scenarios 1 and 2 (Figures 6.3 and 6.6). The BCU_M scheduler still shows higher costs compared to other schedulers except the BFD scheduler.

Index	Name of criterion	weight
c_1	probability of SLA penalty	1.0
c_2	estimated migration energy	0.1
c_3	remaining VM duration	0.2
c_4	data center load	0.1
c_5	estimated cost benefit	1.0
ut	utility threshold	2.0

Table 6.10: List of best utility criteria with associated weights

Power values also show small deviations of up to 2% where most energy savings can be achieved by the BCU_M and BCU_MF schedulers.

Total costs are depicted in Figure 6.10. No penalty costs have resulted for this scenario which coincides with the adjustments of utility weights where highest weights were put on *probability of SLA penalty* and *estimated cost benefit* criteria. Also total costs could be reduced for schedulers BCU_M, BCU_MF and BCU_MIF.

In Figure 6.11 the resulting cloud metrics are shown.

What can be deduced by this graph is that cost and migration dependent metrics (TCCWM, TPC, TC) show substantially reduced values where no penalty costs are shown at all (TPC). The ME, MC, TDT and NM metrics show lower values as well in relation to the BCU_M scheduler. Other metrics such as power values do not exhibit significant variations however emphasis have been put on cost rather than power reductions.

The resulting cost reductions for all scenarios are shown in Table 6.11. Cost reductions of cost and best utility scenarios in percent compared to the energy utility scenario have been denoted as cost (%) and best (%), respectively. The best utility scenario exhibits the least total costs where substantial cost reductions of up to 15% (BCU_M scheduler) have been achieved. It also provides the highest cost reductions compared to the energy utility scenario with further reduced costs for the BCU_M, BCU_MF and BCU_MIF schedulers.

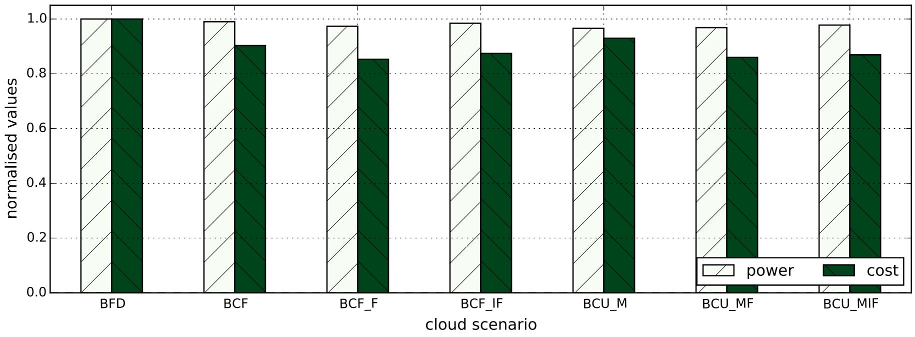


Figure 6.9: Normalized cost vs power values based on the best utility scenario

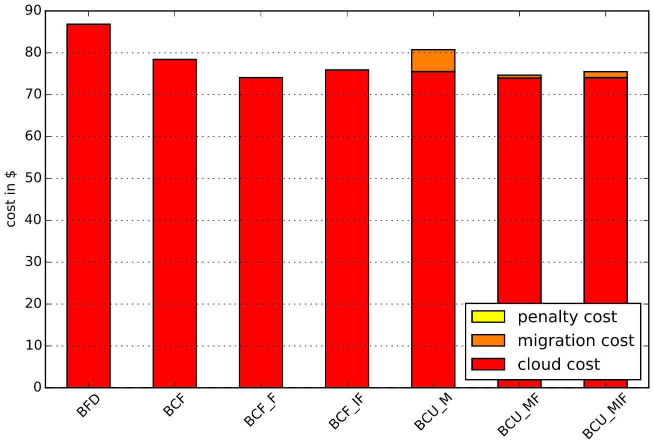


Figure 6.10: Total resulting costs per scheduler for the best utility scenario

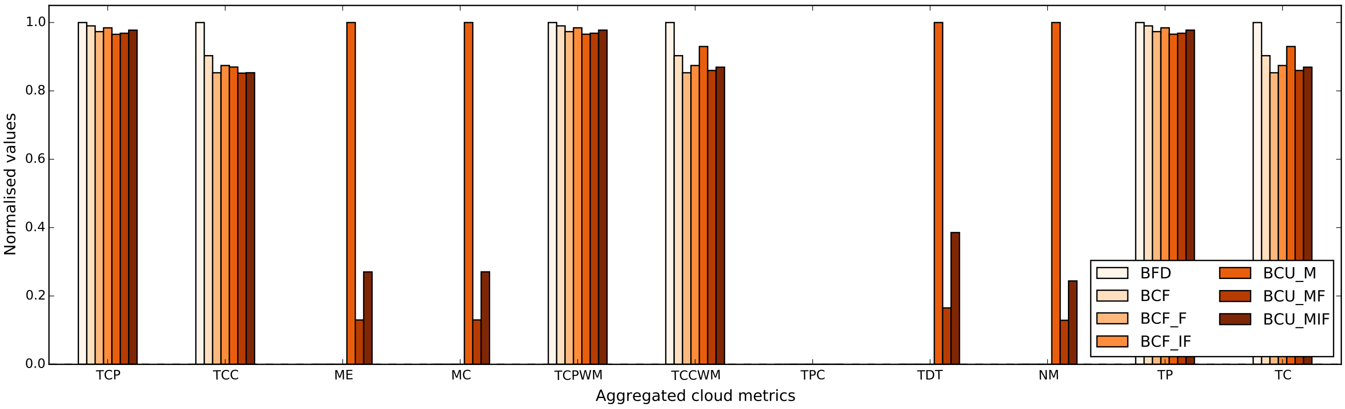


Figure 6.11: Cloud metrics calculated for the best utility over the simulation time period

	energy	cost	cost (%)	best	best (%)
BFD	88.47	86.84	-1.85	86.84	-1.85
BCF	79.05	78.42	-0.80	78.42	-0.80
BCF_F	75.80	74.08	-2.27	74.08	-2.27
BCF_IF	75.64	75.92	0.38	75.92	0.38
BCU_M	95.82	89.72	-6.37	80.75	-15.73
BCU_MF	81.09	74.74	-7.83	74.66	-7.92
BCU_MIF	82.86	78.78	-4.93	75.50	-8.88

Table 6.11: Utility evaluation results for all scenarios (in $)

6.3 Simulation Results

In this section the results of large scale simulations conducted for both day ahead and real time markets are presented. The simulations are based on energy price data from 2013 for two different time periods for each of day ahead and real time scenarios.

An outline of the simulation scenarios with definition of cloud parameters has been given in Section 6.1. A definition of simulation scenarios including corresponding time periods are shown in Table 6.12.

Scenario name	Time period	Description
DA Summer	2013-06-20 to 2013-07-28	A period of 5 weeks of day ahead energy price time series in summer 2013
DA Spring	2013-01-02 to 2013-04-30	A period of 4 months of day ahead energy price time series in spring 2013
RT Summer	2013-06-20 to 2013-07-28	A period of 5 weeks of real time energy price time series in summer 2013
RT Spring	2013-01-02 to 2013-04-30	A period of 4 months of real time energy price time series in spring 2013

Table 6.12: Definition of simulation scenarios

Common cloud parameters

Common parameters have been set for both day ahead and real time markets as presented in Section 6.1. In addition some parameters have been amended to fit the different simulation scenarios. One of those parameters is the bandwidth between locations which is defined as common setting for each of day ahead and real time price simulations, respectively.

Bandwidth mappings defined for both day ahead and real time simulations are outlined in Tables 6.13 and 6.14.

Source	Destination	Bandwidth
Hamina	Potsdam	1000 MBit/s
Hamina	St. Ghislain	800 MBit/s
Potsdam	St. Ghislain	800 MBit/s
Boston	Portland	800 MBit/s
Europe	USA	400 MBit/s

Table 6.13: Bandwidth connections in day ahead market locations

The day ahead bandwidth connections have already been defined in Section 6.1 which are included here for ease of reference. The real time bandwidth connections in the table are defined between different real time markets but do not consider individual locations. Thus if the same

Source	Destination	Bandwidth
ISO New England	ISO New England	1000 MBit/s
ISO New England	PJM	800 MBit/s
PJM	PJM	1000 MBit/s

Table 6.14: Bandwidth connections in real time market locations

market is given for both source and destination it means that connections between locations from within this market exhibit the same bandwidth connection. When different markets for source and destination are given the defined bandwidth is valid for connections between locations of these two markets.

As outlined in Table 6.15 additional parameters have been set that influence the behavior of the simulations. The maximum forecast horizon is the maximum time period for which forecasts will be available. This is important for the metric that averages over current and future energy prices whereby forecasts with a longer time period of e.g. one week will not provide accurate predictions. In this case it has been set to 12 hours ahead as it already provides reasonable forecasts but restricts estimations to allow for a more dynamic forecast environment.

Cloud metric	Value
maximum forecast horizon	12
minimum price threshold	0.1
minimum remaining duration	1
maximum duration VMs	None
VMs with additional SLA levels	None

Table 6.15: Additional cloud parameters for simulation scenarios

The minimum price threshold is given in ¢ / kWh where a VM will not be migrated to a location where price differences fall below the given threshold. It is set to a low value to allow for possibly excessive migrations if not prevented by the scheduler.

The minimum remaining duration is set to the number of hours a VM has to be still running in order to be eligible for migration. Also in this case it has been set to the lowest possible value to not restrict migrations beforehand.

Maximum duration VMs are VMs that run over the whole simulation time period. This has not been taken into account since migration behavior differs substantially when such VMs are included in the simulation which may be investigated in future work.

Applying different SLA availability levels to VMs has not been considered since resulting penalty costs would not be coherent over the whole set of VMs.

The utility values for all subsequent simulations will be taken from the *best utility function* evaluated in the previous section. Since it provided promising results on a small scale cloud it is expected to perform well also in larger cloud environments.

The maximum cloud utilization is calculated based on the total number of virtual machines, the simulation time period, number of servers and the average runtime of VMs. It describes the maximum possible utilization in case all active servers are fully occupied at all times. The corresponding formulas are depicted in Equations 6.25 and 6.26.

$$util_{max} = \frac{|VM| \cdot dur_{avg}}{|T| \cdot |s|} \tag{6.25}$$

$$dur_{avg} = \frac{1}{|VM|} \sum_{v \in |VM|} v_{dur} \tag{6.26}$$

The average VM runtime dur_{avg} is calculated as the mean of all fixed VM duration values where VM assignments to durations follow the uniform distribution. The maximum utilization $util_{max}$ is then the number of VMs multiplied by their average duration in relation to the number of hourly time periods $|T|$ and the number of servers $|s|$ across all locations.

The forecast model chosen to be used in all simulations is the ARIMA model. As outlined in Section 4.6 it provided the best results in forecast evaluation. The simulations have been conducted based on ARIMA models trained for each day of the simulation with 2 weeks of training data. Each model was used to generate forecasts for the next 24 hours to generate a seamless series of forecasts to be used in the simulations.

Each simulation scenario has been run on all schedulers defined in 6.1 where all previously defined cloud metrics are evaluated (Section 6.1). This results in a comprehensive evaluation of the defined schedulers across an extended time period.

In the following sections each simulation scenario is presented in detail together with the results.

Simulation scenario DA Summer

The first simulation scenario is based on a time period of 5 weeks of day ahead energy prices.

The corresponding dataset is shown in Figure 6.12 which exhibits strong seasonal patterns over each day and week. Locations St. Ghislain and Potsdam show the most stable seasonal periods where Hamina exhibits an almost exponential decline. Portland and Boston show regular seaonal periods as well except for one week in mid of July where the energy price levels increase drastically.

The simulation has been run based on this dataset and previously defined cloud parameters. The following parameters have been set specifically for this simulation (Table 6.16).

Cloud metric	Value
number of servers	500
number of servers per location	100
number of virtual machines	5000

Table 6.16: Cloud parameters for simulation scenario DA Summer

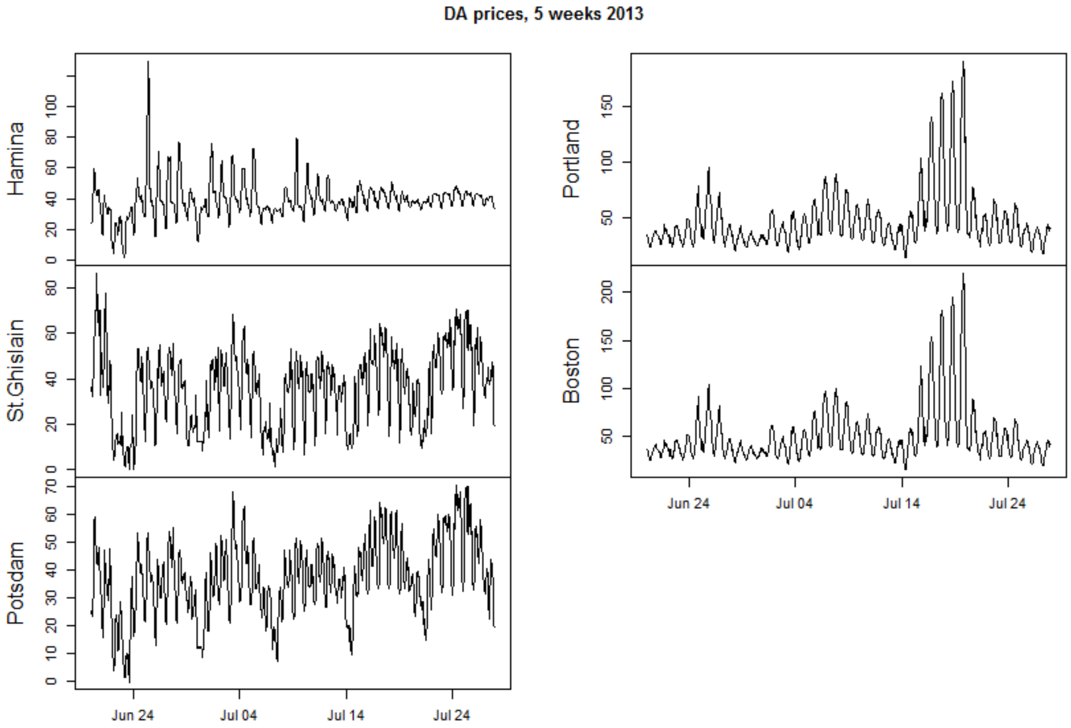


Figure 6.12: Day ahead energy prices over 5 weeks in 2013, in $ per MWh

The set of servers is distributed to locations such that each location is assigned to the same number of servers. Thus the number of servers has been chosen to accommodate the need for equal distribution regarding the number of locations.

In this case the maximum utilization based on the calculation of Equation 6.25 would result in 15.7%. Thus the cloud (number of servers, number of VMs) has been defined such that a sufficient amount of servers are available for VMs at each point in time and a reasonable value of cloud utilization can be reached.

Figure 6.13 shows the results for normalized power vs cost for each scheduler. It can be noted that this shows very similar behavior to the results of the *best utility function* in Section 6.2. However in this simulation the BCU_MF scheduler shows the least resulting costs relative to the baseline scheduler. This indicates a good performance of the utility function based on energy price forecasts which could outperform all other schedulers in terms of resulting costs.

Figure 6.14 confirms these findings by displaying the total resulting costs for this simulation scenario. In this scenario no penalty costs due to migration downtimes have been resulted from any scheduler which attributes to high utility weights regarding SLA penalty and cost metrics.

The resulting cloud metrics are depicted in Figure 6.15. This graph is similar to the cloud metrics of the best utility function with further reduced number of migrations of the BCU_MF scheduler relative to the BCU_M scheduler. Also total costs could be reduced due to less migration costs and less total cloud costs which is caused by a reduction of total power consumption.

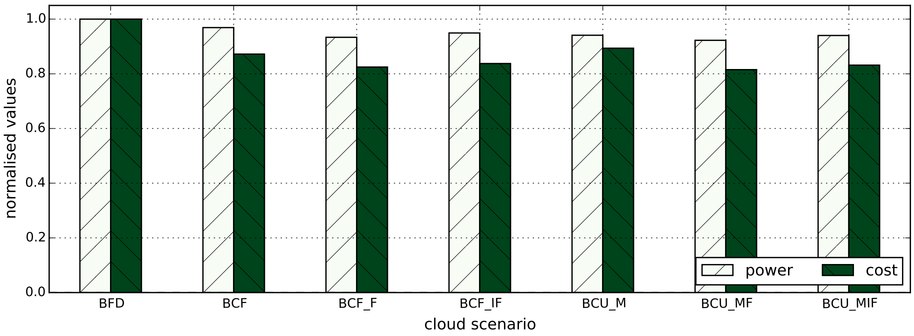


Figure 6.13: Normalized power and costs for DA Summer simulation scenario

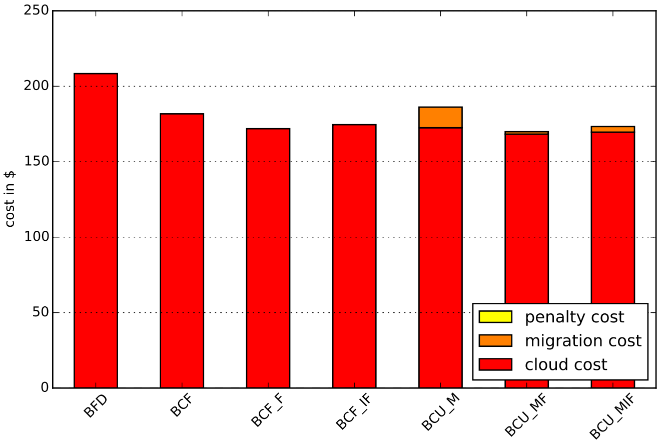


Figure 6.14: Total resulting costs for DA Summer simulation scenario

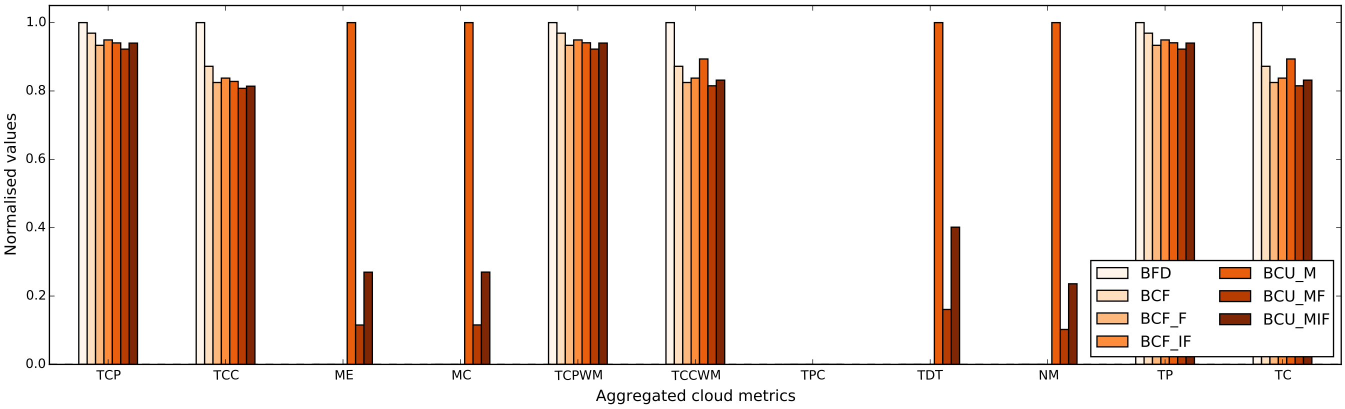


Figure 6.15: Cloud metrics resulting from the DA Summer simulation scenario

Simulation scenario DA Spring

This simulation scenario is based on a total of 4 months of day ahead energy price data which should prove the validity of the schedulers based on an extended simulation time period.

Figure 6.16 shows the underlying dataset on which this simulation is based. Different time series show very different behavior in energy price variations. For example, Portland and Boston exhibit an increase in energy price levels of up to a factor of 10 during the period from January to March compared to prices after this period. Energy prices for Hamina on the contrary show a much more stable mean with just one significant spike at the beginning of March. This should result in a challenging scenario in terms of accurate forecast calculations for each location.

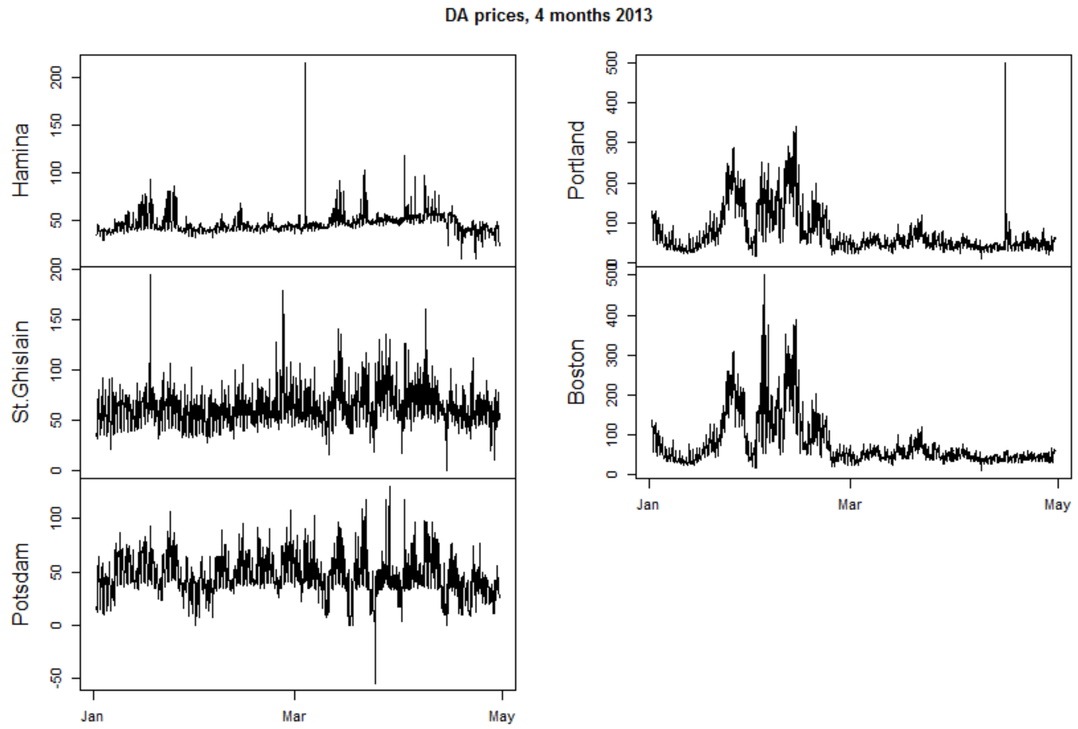


Figure 6.16: Day ahead energy prices over 4 months in 2013, in $ per MWh

Simulation specific parameters have been set according to Table 6.19. Thus the number of servers and servers per location has been defined analogously to the DA Summer simulation scenario but the number of virtual machines has been adjusted to accommodate the extended simulation time period. This results in a maximum utilization (VMs per server and point in time) of 14.9% across the simulation time period.

From the results in Figure 6.17 it is clearly visible that substantial cost savings have been achieved in this scenario compared to the BFD baseline scheduler. Even though the energy price

Cloud metric	Value
number of servers	500
number of servers per location	100
number of virtual machines	15000

Table 6.17: Cloud parameters for simulation scenario DA Spring

dataset has not been optimal for applying forecasting methods the BCU_MF scheduler which triggers migrations based on forecasts again results in the largest cost savings. Surprisingly the scheduler BCU_MIF which is based on ideal forecasts is outperformed by the BCU_MF scheduler which is based on actual forecasts generated for this dataset. This phenomenon can be attributed to the fact that the BCU_MIF scheduler is highly impacted by occurring price spikes which may result in a higher number of migrations and thus resulting costs.

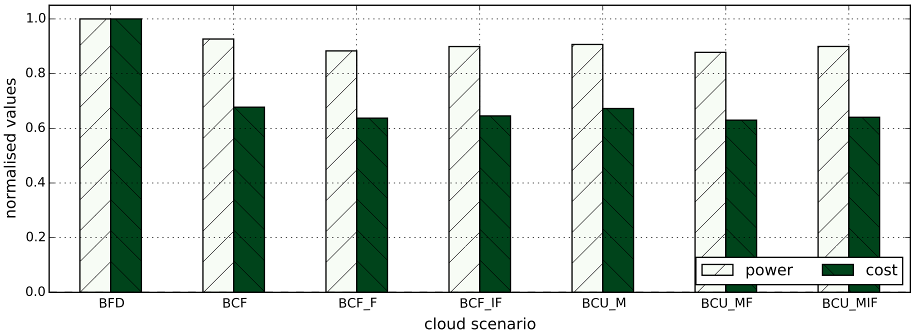


Figure 6.17: Normalized power and costs for DA Spring simulation scenario

This is also emphasized by the total cost chart in Figure 6.18. Due to a fewer number of migrations and different scheduling decisions the BCU_MF scheduler results in less cloud costs attributed to less resulting cloud power. It can also be observed that the migration cost have been reduced to a negligible but still existing fraction of the total costs except for the BCU_M scheduler. Therefore the cost difference between the BCU schedulers is caused due to different scheduling decisions and server consolidation resulting in different power and cost levels.

The graph showing cloud metrics confirms the statements made above (Figure 6.19). Total cloud power as well as total cloud costs could be significantly reduced compared to the BFD scheduler. The number of migrations and therefore migration costs could be greatly reduced as well for BCU_MF and BCU_MIF schedulers. The difference in total costs between the BCF and BCU schedulers is small but still significant. Interestingly the BCF_F scheduler which is based on real forecasts outperforms both BCF and BCF_IF schedulers in the same manner as it has been explained for BCU schedulers above.

Therefore the BCU_MF scheduler provides the best results for this simulation scenario similar to the DA Summer simulation.

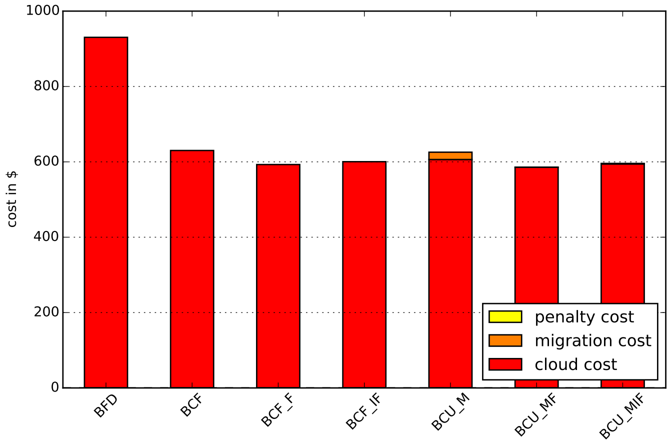


Figure 6.18: Total resulting costs for DA Spring simulation scenario

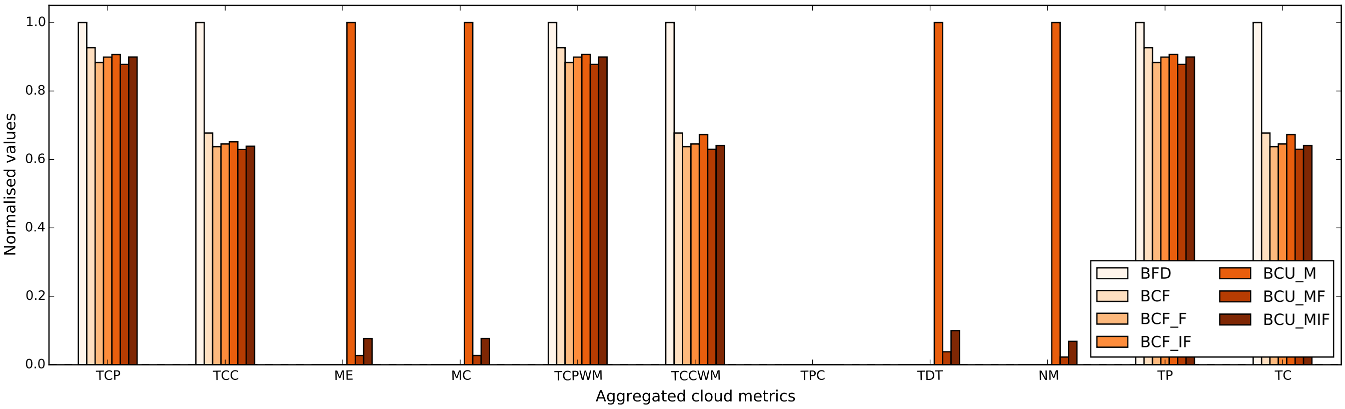


Figure 6.19: Cloud metrics resulting from the DA Spring simulation scenario

Simulation scenario RT Summer

This simulation scenario is again based on 5 weeks on energy price data however prices are retrieved from real time markets defined in Section 6.1.

Figure 6.20 shows the corresponding dataset for this simulation. Similar to the 5 week day ahead dataset it shows obvious daily and weekly seasonality with substantial increase in price levels beginning in mid of July. What is interesting to note is that all locations show similar price behavior over the whole simulation time period. This can be attributed to the fact that most of the locations are bound to the same energy market (PJM) and thus exhibit comparable patterns. Despite the locations Boston and Portland being associated with a different energy market (ISO New England) they show very similar patterns compared to the PJM locations.

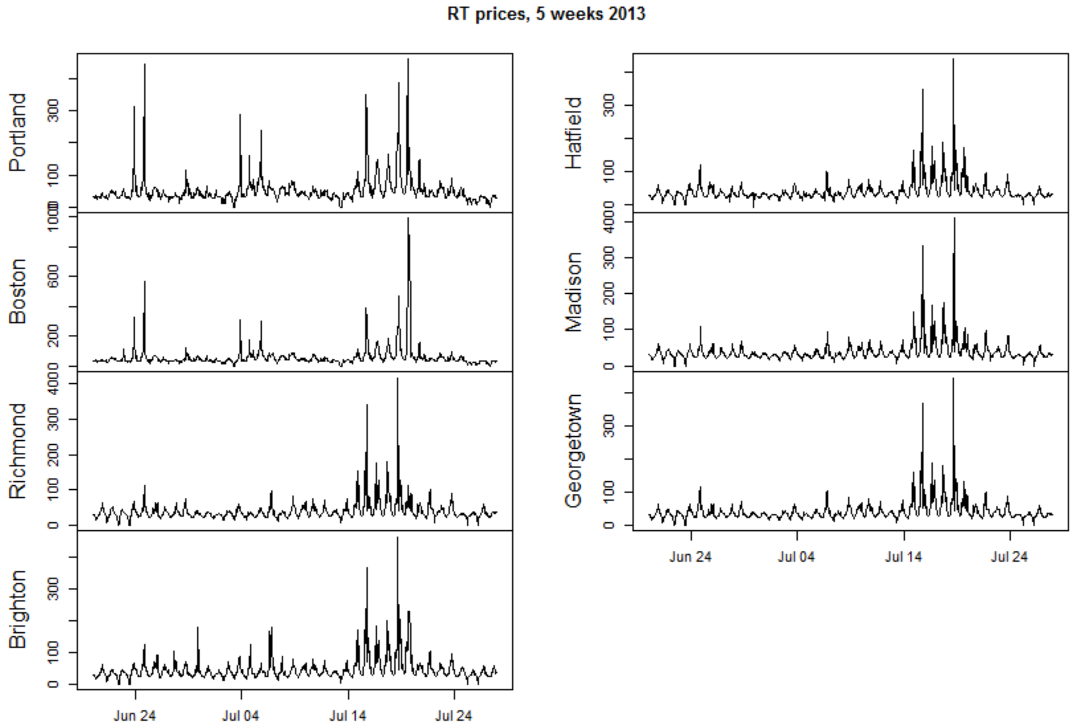


Figure 6.20: Real time energy prices over 5 weeks in 2013, in $ per MWh

The parameters applicable to this scenario are shown in Table 6.18. The number of servers and virtual machines has been adjusted to fit the greater number of locations in this scenario. The resulting maximum utilization according to Equation 6.25 would be 15.7% which is equal to the maximum utilization of the DA Summer simulation.

The normalized cost chart depicted in Figure 6.21 shows slightly different patterns than the simulation scenarios presented for day ahead energy prices. In particular the schedulers

Cloud metric	Value
number of servers	700
number of servers per location	100
number of virtual machines	7000

Table 6.18: Cloud parameters for simulation scenario RT Summer

exhibiting ideal forecasts (BCF_IF and BCU_MIF) provide the best results and the most cost reductions. This may be attributed to the fact that the real time price dataset for this simulation does not exhibit as many "outliers" or price spikes as the dataset associated with the DA Summer simulation over the same time period. Thus in this case the accurate ideal forecasts are not triggering too many unnecessary migrations and VMs do not need to be rescheduled as often leading to less power consumption and costs.

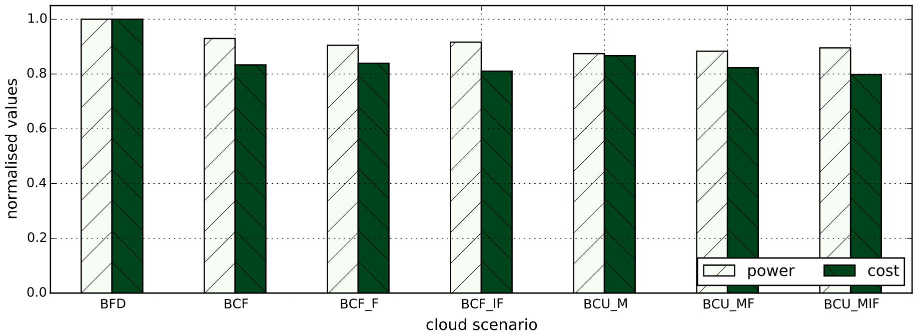


Figure 6.21: Normalized power and costs for RT Summer simulation scenario

The same findings can be derived from the total cost chart in Figure 6.21. The BCU_MIF scheduler results in slightly more migrations compared to the BCU_MF scheduler but results in lower cost due to better utilization of the energy price differences for different locations. When comparing the BCU and BCF schedulers of the same type the former provide better performance than the latter in terms of cost reductions. Therefore it can be deduced that the utility functions operated at the BCU schedulers is well calibrated and provide reasonable results.

The cloud metrics in Figure 6.23 provide a detailed view on the results of the different metrics. Despite the increased power consumption caused by the BCU_MIF scheduler it results in less total costs due to its ability to consider all occurring price differences. It also results in significantly more VM downtimes than the BCU_MF scheduler however accumulated downtimes for individual VMs do not reach the penalty threshold. Therefore increased cost reductions come at the price of increased downtimes which are however not considered significant.

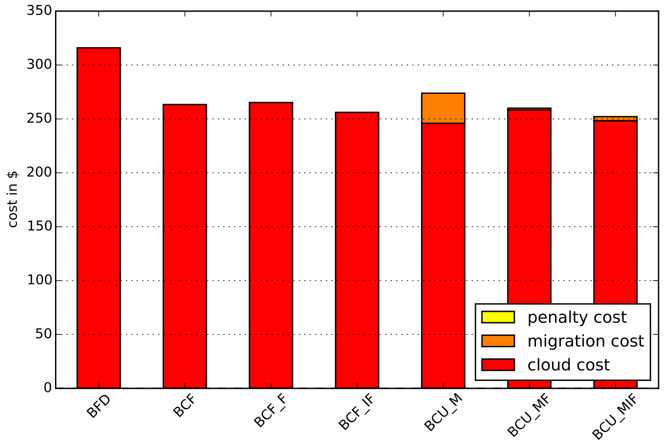


Figure 6.22: Total resulting costs for RT Summer simulation scenario

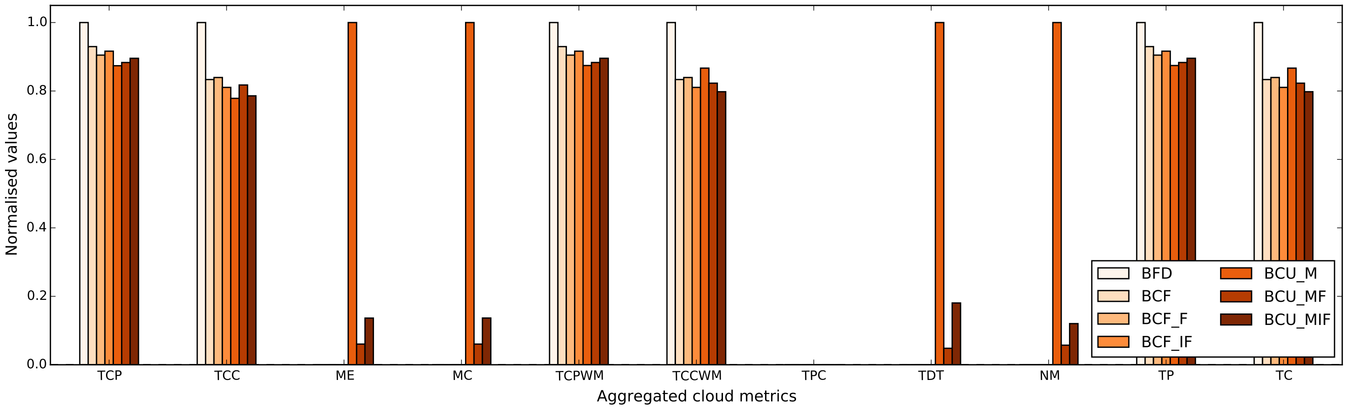


Figure 6.23: Cloud metrics resulting from the RT Summer simulation scenario

Simulation scenario RT Spring

The final simulation scenario is based on 4 months of real time energy prices. Analogously to the day ahead simulation DA Spring it exhibits the same time period but operates on a different dataset. The dataset associated with this simulation is outlined in Figure 6.24.

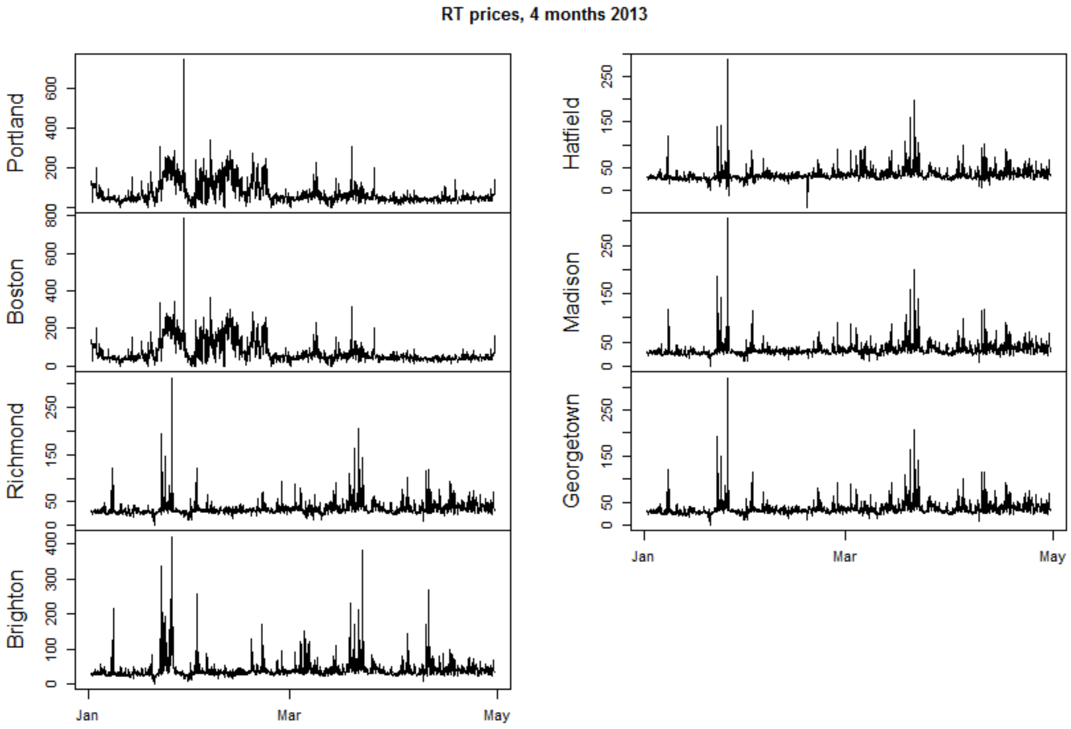


Figure 6.24: Real time energy prices over 4 months in 2013, in $ per MWh

Comparable to the dataset of the RT Summer simulation the locations belonging to the same energy market (see Section 6.1) show very similar patterns over time. The PJM market locations experience a significant price spike at the end of January with another substantial increase in price levels in mid of March. The patterns may be similar but not identical, therefore it may still be sensible to migrate between locations of the PJM market. The locations Boston and Portland show similar patterns as their day ahead counterparts (see Section 6.3). Thus a significant increase in price levels and variation can be observed from mid of January to mid of February in the graph which may result in less accurate forecasts.

The cloud parameters specific to this simulation are depicted in Table 6.19. The relation of number of VMs to number of servers has been set the same as for the DA Spring simulation for a comparable distribution of VMs to servers. Therefore the maximum cloud utilization for this simulation scenario is 14.9% which is the same as for the DA Spring simulation.

Cloud metric	Value
number of servers	700
number of servers per location	100
number of virtual machines	21000

Table 6.19: Cloud parameters for simulation scenario DA Spring

The normalized power and cost graph shows different results from the RT Summer simulation (Section 6.3). In this scenario the schedulers operating on real forecasts provide the best results in contrast to the schedulers based on ideal forecasts in the RT Summer simulation. After the investigations in the previous sections it can be stated that this behavior can be deduced with high probability from the characteristics of the energy price dataset. As energy prices tend to be more volatile in this scenario the schedulers based on real forecasts show an advantage of averaging extreme price values. In the previous simulation the price volatility has not been as high therefore the ideal forecasts exhibited an advantage.

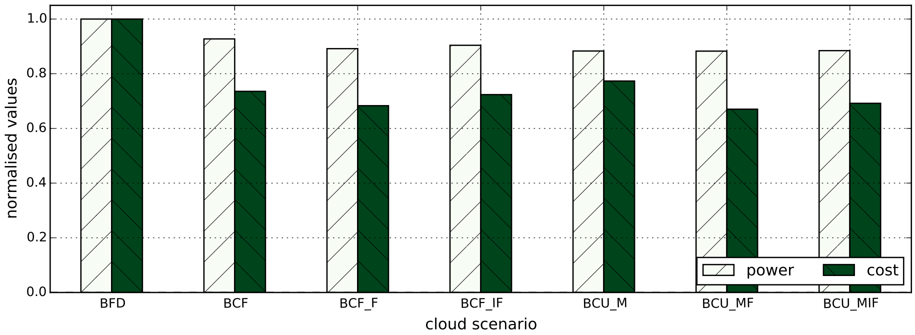


Figure 6.25: Normalized power and costs for RT Spring simulation scenario

The total cost graph emphasizes these observations (Figure 6.26). Compared to the DA Spring simulation (Section 6.3) total costs show higher values for both cloud and migration costs. This can be attributed to the higher number of locations in this simulation which result in increased energy costs over the same simulation time period. As shown above the schedulers based on actual forecasts achieve the best results (BCF_F, BCU_MF). Thus the “real” forecast scheduler is suitable to be applied over a longer period of time with energy prices exhibiting high volatility.

The resulting cloud metrics are depicted in Figure 6.27. Substantial power and cost reductions are clearly visible for total power and cost metrics. The BCU_MF scheduler provides the best results with lowest energy costs and power consumption. In addition it experiences the least number of downtimes from all schedulers that include migrations (BCU_M, BCU_MF, BCU_MIF). Therefore together with the findings earlier in this section it can be concluded that the BCU_MF scheduler provides superior results for highly volatile energy price time series.

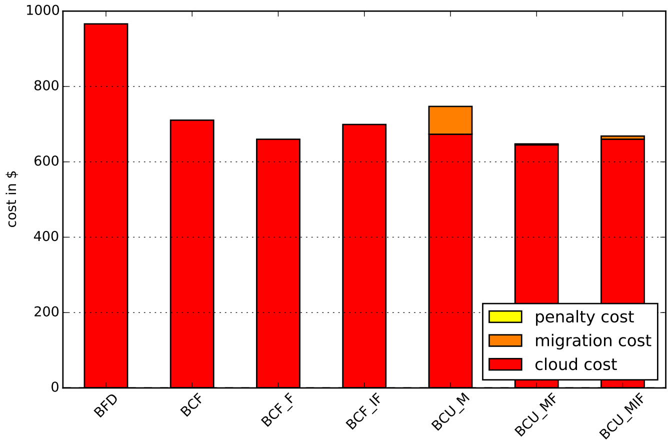


Figure 6.26: Total resulting costs for RT Spring simulation scenario

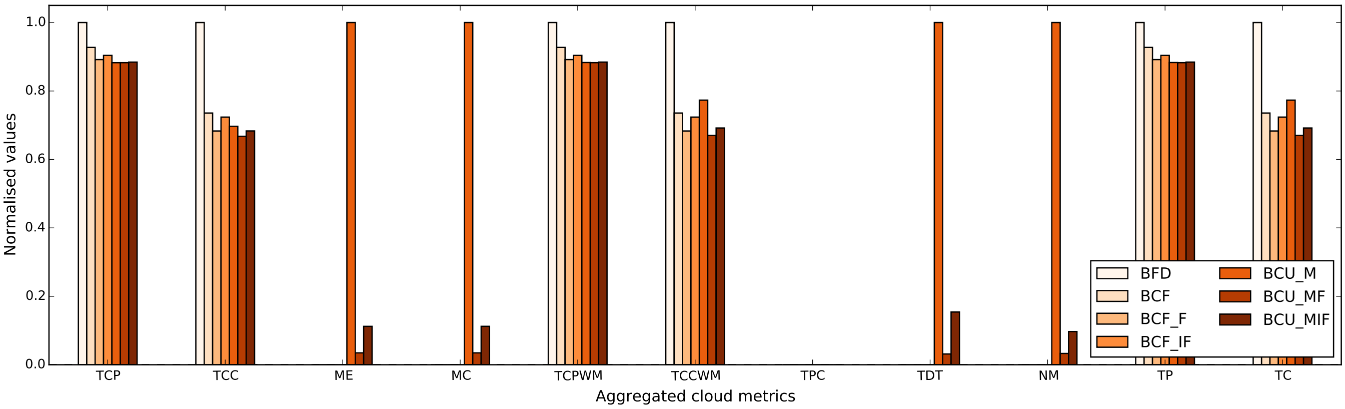


Figure 6.27: Cloud metrics resulting from the RT Spring simulation scenario

Utilization and forecast evaluation

In order to better investigate the results presented in the previous sections the actual price time series vs their utilization levels have been investigated. Thus for each simulation scenario and each scheduler the cloud utilization has been printed together with the energy prices and corresponding forecasts. The results for the simulation RT Summer and the BCU_MF scheduler are shown in Figure 6.28.

In this graph all locations are listed with each location showing a top and bottom graph. The top graph of each location shows the energy price time series over the simulation time period as a green line and the corresponding forecasts as overlay as a red line. The bottom graph shows the changes of utilization over time for each location, that is the actual total server utilization per time.

Results show substantial variations in utilization levels over the simulation time period. The change in utilization is triggered by the actual forecasts (red line) as the BCU_MF scheduler evaluates the utility function based on real forecasts. This can be observed when investigating the location Richmond and its utilization levels.

The utilization at this location shows a drop in utilization levels beginning from 16th of July due to increased forecasted prices at this time. Thus the load is moved to other locations exhibiting a reduced energy price at this point in time. The same behavior can be observed for the locations Hatfield and Georgetown where utilization drops around the same time.

In contrast location Madison shows more stable utilization levels with still a trough around the same time where drops in utilization have been observed for other locations. Locations Boston and Portland have not experienced high utilization on average as the effort of moving resources to these locations is higher due to lower bandwidth connections (see Table 6.14).

However around the same time at mid of July Portland experiences a substantial increase in load which is shifted from other locations. The reason might be the small drop in energy price forecasts at this location on the 16th of July as well as a lower forecasted average price over the next hours.

Utilization levels increase up to 50 percent at times (see location Portland) but average utilization per location varies from very low values to about 25 percent depending on the scenario. As the weights of the utility function (see Section 6.2) have been set to high values for SLA penalties and cost reductions but to a low value for data center load the primary reason for VM migration are to mitigate SLA penalties due to VM downtimes and enhance cost reductions based on cheaper energy prices. Load balancing has not been considered as important metric in the presented simulations.

The resulting graphs for all locations, all simulations and all schedulers have been placed in the Appendix for reference (B.2). The BFD scheduler in the RT summer simulation does not consider any energy price changes in its scheduling decisions and thus load is distributed uniformly over all locations (see B.15). This results in similar utilization at all locations where average utilization levels for this scheduler evolve around 8 to 10 percent.

The BCU_M scheduler only considers current energy prices for migrations which leads to very high number of migrations and high volatility in the utilization (see B.26). For the RT Spring simulation this is best visible for location Portland which exhibits a very high number of

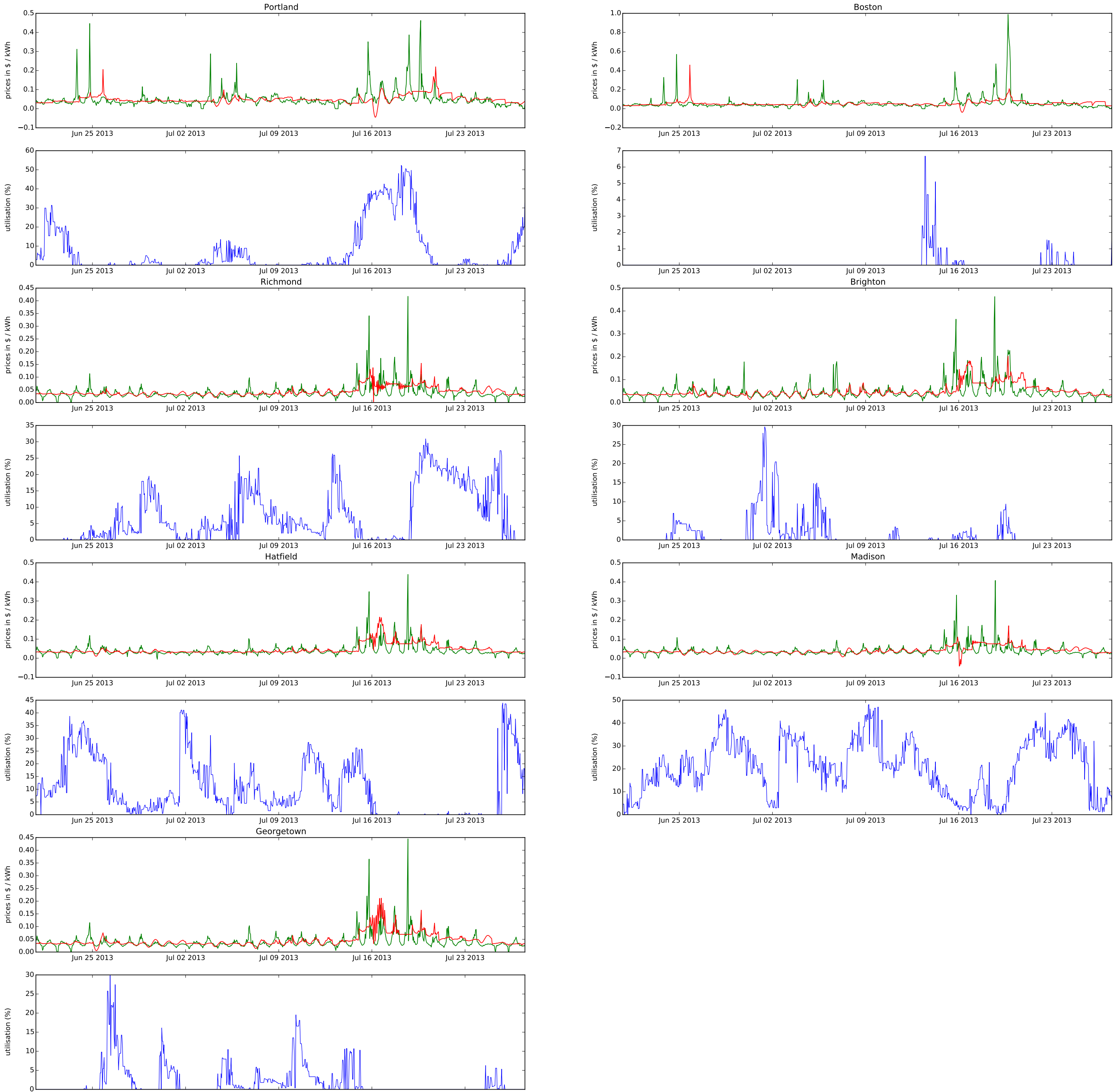


Figure 6.28: Energy prices, energy price forecasts and utilization levels per location for simulation RT Summer and scheduler BCU_MF

"spikes" in utilization. Thus due to high variations in energy prices a high number of migrations have been performed.

When comparing the behavior of the schedulers based on actual and ideal forecasts (BCU_MF, BCU_MIF) it can be observed that the BCU_MIF scheduler exhibits a higher number of migrations (higher volatility of utilization) than the BCU_MF scheduler (see B.13 and B.14 for locations Hamina and Potsdam). This confirms the fact described before that the BCU_MIF scheduler is more sensitive to price changes and tends to trigger migrations more frequently.

When observing simulation results for DA Summer for scheduler BCU_MF (see B.6) the relation of forecasts to actual energy prices can be seen more clearly. It seems that the forecasted energy prices "lag behind" the actual energy prices by a certain amount of time. Daily seasonality can be clearly spotted also in forecasted energy prices with similar patterns as they exist in actual energy prices, best visible for locations Portland and Boston.

However for the real time price scenario shown in Figure 6.28 it can be observed that the forecasts do not model the exact price time series but average out price spikes that may occur in the energy price time series. Therefore schedulers incorporating real forecasts in contrast to ideal forecasts can be seen as an advantage for highly volatile price series as they do not consider each price spike but model energy prices more closely to their mean. Thus ideal forecasts may even lead to sub-optimal scheduling decisions as results showed in DA Spring and RT Spring simulations (see linked Sections 6.3 and 6.3).

Simulation results

The simulation scenarios discussed in the previous sections provide a clear hint for the most suitable scheduling algorithm for a given dataset. In case of datasets exhibiting mid to high range volatility the BCU_MF scheduler results in lowest total costs whereas for datasets with more stable volatility the BCU_MIF scheduler provides best results.

Table 6.20 displays the average utilization and runtimes of each scheduler and simulation. As pointed out before maximum utilization evolves around 15 percent whereas average utilization stays at around 7 percent. This may be due to insufficient server consolidation which may need to be improved to achieve further cost savings.

	DA Summer		DA Spring		RT Summer		RT Spring	
	Util	Runtime	Util	Runtime	Util	Runtime	Util	Runtime
BFD	7.70	250.38	7.67	1375.55	7.86	425.71	7.60	2617.22
BCF	7.31	271.46	7.11	1306.38	7.22	428.04	7.13	2609.53
BCF_F	7.12	282.85	6.89	1394.61	7.06	581.61	6.86	2809.27
BCF_IF	7.20	288.25	6.97	1401.16	7.10	496.77	6.94	3154.17
BCU_M	7.12	342.54	7.00	1556.34	6.95	646.65	6.85	4168.69
BCU_MF	7.09	311.09	6.87	1527.99	6.98	567.82	6.83	3057.39
BCU_MIF	7.14	321.61	6.96	1554.50	7.03	553.67	6.86	3045.05
Total	7.24	5335.41	7.07	39421.84	7.17	10061.16	7.01	76344.82

Table 6.20: Average cloud utilization and runtime per simulation and scheduler

Runtimes (in seconds) are highest for the BCU_M scheduler as it involves excessive migrations. The reason the total runtime shows a higher value than the sum of all runtimes is the fact that the cloud is generated freshly for each simulation and each scheduler.

The tables of all cloud metrics as actual and normalized values can be referred to in the Appendix B.1.

Total costs

The total resulting costs for each scheduler and simulation scenario are outlined in Table 6.21. The lowest costs are marked in bold. These values confirm the fact that the BCU_MF scheduler outperforms other schedulers in terms of costs except for the RT Summer simulation where the BCU_MIF scheduler exhibits the lowest costs.

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
DA Summer	208.36	181.72	171.85	174.53	186.19	**169.86**	173.29
DA Spring	930.41	630.13	592.83	600.42	625.62	**585.97**	595.81
RT Summer	315.96	263.32	265.21	256.08	273.87	259.98	**252.09**
RT Spring	966.04	710.59	659.92	699.13	747.06	**647.57**	668.35

Table 6.21: Total costs for each simulation scenario and cloud scheduler (in $)

Substantial cost savings could be achieved compared to the BFD baseline scheduler which are shown in Table 6.22. It displays cost reductions in percent relative to the BFD scheduler. The maximum cost reductions across all simulations result in an average of 27 percent where the highest cost reduction could be achieved in the DA Spring simulation with cost savings of over 37 percent in relation to the BFD scheduler.

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
DA Summer	0	12.79	17.52	16.24	10.64	**18.48**	16.83
DA Spring	0	32.27	36.28	35.47	32.76	**37.02**	35.96
RT Summer	0	16.66	16.06	18.95	13.32	17.72	**20.21**
RT Spring	0	26.44	31.69	27.63	22.67	**32.97**	30.82

Table 6.22: Normalized total cost reductions for different scenarios (in %)

These results show that significant cost savings are possible using the proposed approach compared to a non-cost aware approach.

CHAPTER 7

Conclusion and Future Work

Today cloud providers offering services on a large scale exhibit significant energy costs which may result in decreased revenue for provided cloud services. Therefore cost reductions in cloud environments are critical for cloud providers to gain maximum profit and ensure continuous operation of the services.

In this work the cost efficiency of geo-distributed clouds has been investigated. Through intelligent load scheduling significant cost savings could be achieved in relation to a baseline approach. Thereby the basic tradeoff between cost reductions and prevention of SLA penalties has been considered in simulations.

Different energy markets and data have been analyzed to evaluate characteristics of different energy price time series that may impact the performance of forecasting methods. Various forecasting models have been introduced that should be compared regarding forecast accuracy on different data sets.

A large scale forecast evaluation has been conducted to find the best suited model for generating forecasts for both day ahead and real time energy prices. Results show that forecast accuracy varies greatly on different datasets and different forecasting models. However common behavior across different simulations could be detected such as consistently lower prediction errors for certain forecast horizons and model training periods. From the evaluated forecasting models the ARIMA model has been chosen as the best model exhibiting the least forecast errors on a consistent basis.

The simulation framework comprises a cloud scheduler which is capable of executing different scenarios that integrate different scheduling algorithms for maximum cost savings. The core mechanism of the scheduler consists of an utility function that decides whether a VM should be migrated at the current point in time based on different criteria. Five criteria have been defined to model specific constraints that should be considered in scheduling decisions. These criteria may be extended at any time by constraints useful to a particular cloud environment.

The utility function has been optimized regarding minimization of SLA penalties and maximum cost reductions across simulations. Results have shown that weights chosen for the different criteria have significant impact on the resulting number of SLA penalties and cost reductions.

Therefore different experiments have been conducted to determine the best suited utility weights for the given cloud setting.

Large scale simulations have been defined and executed on a cloud framework to evaluate the performance of the schedulers on different datasets and cloud conditions. The results of these simulations show that schedulers based on the utility function incorporating energy price forecasts achieve the best results and even outperform schedulers based on ideal forecasts. Through handling the tradeoff between maximum cost reductions and prevention of SLA penalties substantial cost savings could be achieved in relation to a baseline scheduler.

These results show the validity of our approach in utilizing energy price differences across different energy markets. With appropriately calibrated schedulers maximum energy cost savings can be achieved with geographically dispersed data centers connected to different energy markets. Thus for large scale cloud environments with connections to energy markets significant cost savings are possible using the proposed approach.

7.1 Future Work

Various optimizations are possible for the introduced cloud framework. The weights of the utility function have been estimated empirically to find the best fit regarding defined cloud settings. However a more strategic approach may be applied to estimate parameters based on maximum likelihood estimations. Furthermore additional criteria may be defined to model constraints in a cloud environment.

Forecasting models may be improved, or different models may be investigated based on electricity price data. Dynamic Regression (DR) and Transfer function (TF) models have been shown to provide a significantly better performance than ARIMA models [2, 101]. This may be due to the additional regressor variables that are considered in multivariate models (DR and TF) in contrast to models based on a single univariate time series (ARIMA). However the amount of data and computational effort needed for model training increases for multivariate models. Therefore the expected benefit from improved forecasts vs the model generation costs have to be weighted carefully.

Cloud parameters could be amended to model different conditions for cloud environments. Parameters include the price of VMs and bandwidth costs that directly impact the resulting cloud costs and migration costs. Additional SLA levels could be introduced which may result in different penalty costs depending on the scenario. Modeling of long running VMs (web servers) show different resource utilizations and impact the resulting cloud costs and penalty costs.

In [29] it is shown that the cost of optimal assignment is further reduced by increasing the number of locations with data centers assigned to different energy markets and/or by increasing the variability of energy prices. Thus the simulation scenarios may be adjusted to include data from additional locations in different time zones to evaluate the effect of time lag on energy cost reductions.

As Qureshi et al. pointed out in [81] most companies would need to renegotiate their energy contracts in order to utilize the proposed approach. Companies renting space in co-location facilities usually pay a fixed price for energy rather than directly taking part in the price offers of wholesale markets. However for cloud providers running several distributed data centers the proposed approach provides real opportunities for energy cost reductions compared to a fixed price scheme.

Demand response programs provide a way of further decreasing energy costs as electricity consumers agree on a reduced price in exchange for reduced load when it is requested by the grid operator [60]. The program can be incorporated in data center operations as long as the reduction of load requests can be handled without a significant increase of SLA violations. This can be achieved by dynamically shifting load to other locations when requested or by installing local generation facilities that provide additional energy sources on demand.

APPENDIX A

Forecast Evaluation Results

A.1 Result tables

Results for Nord Pool Spot, FI (DA)

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.30	10.66	10.79	14.78	14.22	13.02	13.21	12.58	12.99	12.18
ses	2.13	3.54	5.06	15.83	16.38	14.98	14.97	14.64	15.13	13.30
holts	1.91	3.36	9.48	30.83	38.85	44.08	58.79	74.25	137.44	228.66
holtwinters	3.22	7.15	13.18	26.22	36.32	46.07	65.31	85.66	167.14	289.51
arima	2.11	3.08	4.65	13.66	13.89	12.67	12.87	12.61	13.68	12.98
tbats	2.16	2.89	4.35	10.68	10.89	10.02	10.19	9.94	10.70	10.33

Table A.1: Forecast evaluation results based on RMSE for a training period of 2 weeks, Nord Pool Spot, Hamina

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.40	10.75	10.90	15.03	14.46	13.25	13.44	12.81	13.27	12.42
ses	2.14	3.56	5.06	15.84	16.38	14.98	14.97	14.63	15.14	13.31
holts	1.91	3.38	9.51	30.88	38.90	44.15	58.90	74.39	137.77	229.23
holtwinters	3.44	7.61	14.06	27.95	38.22	47.77	67.07	87.51	169.44	292.43
arima	2.12	2.95	4.37	13.02	13.12	11.89	11.99	11.63	12.37	11.50
tbats	2.15	3.04	4.73	11.23	11.41	10.51	10.57	10.33	11.05	10.67

Table A.2: Forecast evaluation results based on RMSE for a training period of 3 weeks, Nord Pool Spot, Hamina

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	9.52	10.88	11.04	15.22	14.63	13.42	13.58	12.94	13.41	12.57
ses	2.13	3.57	5.06	15.84	16.38	14.97	14.96	14.61	15.13	13.30
holts	1.89	3.37	9.53	31.00	39.07	44.34	59.18	74.75	138.54	230.59
holtwinters	5.58	11.05	17.33	33.52	44.92	54.87	76.25	98.63	189.26	327.09
arima	1.98	2.85	4.39	12.69	12.79	11.60	11.73	11.35	12.06	11.12
tbats	2.20	3.09	4.64	10.98	11.19	10.31	10.45	10.20	10.91	10.36

Table A.3: Forecast evaluation results based on RMSE for a training period of 4 weeks, Nord Pool Spot, Hamina

Results for Belpex, BE (DA)

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	12.96	15.84	18.24	16.96	15.83	15.74	15.56	15.31	15.44	15.79
ses	8.32	11.37	14.02	16.60	16.52	17.05	16.85	17.07	17.46	17.08
holts	7.88	11.28	18.10	45.21	62.73	80.79	113.75	149.46	288.20	493.94
holtwinters	10.77	21.02	34.17	61.39	86.76	113.05	164.22	216.22	424.16	736.39
arima	6.86	7.10	7.82	14.34	15.42	15.41	15.64	16.11	17.79	18.47
tbats	4.86E+49	1.05E+50	1.46E+50	1.45E+50	1.40E+50	1.39E+50	1.38E+50	1.37E+50	1.36E+50	1.35E+50

Table A.4: Forecast evaluation results based on RMSE for a training period of 2 weeks, Belpex, St.Ghislain

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	13.27	16.10	18.41	17.27	16.18	16.06	15.90	15.66	15.75	16.07
ses	8.37	11.41	14.07	16.64	16.56	17.08	16.88	17.10	17.50	17.10
holts	7.80	11.29	18.07	45.41	63.10	81.37	114.61	150.60	290.39	497.90
holtwinters	10.39	18.71	31.85	61.71	86.71	114.91	167.80	221.43	436.30	758.77
arima	6.70	6.61	7.07	13.92	15.02	14.99	15.26	15.67	17.06	17.12
tbats	6.05	6.50	7.12	11.95	13.17	13.40	13.71	14.11	14.98	15.05

Table A.5: Forecast evaluation results based on RMSE for a training period of 3 weeks, Belpex, St.Ghislain

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	13.62	16.34	18.54	17.44	16.36	16.22	16.07	15.82	15.89	16.19
ses	8.38	11.45	14.12	16.68	16.59	17.11	16.91	17.12	17.52	17.12
holts	7.82	11.30	18.05	45.14	62.69	80.72	113.69	149.35	287.92	493.58
holtwinters	9.68	20.86	34.34	60.86	85.91	112.49	164.12	216.17	423.85	735.38
arima	6.82	6.91	7.56	14.06	15.05	14.87	15.09	15.42	16.47	16.41
tbats	2.53E+35	2.17E+36	3.83E+36	2.88E+36	3.16E+36	2.83E+36	2.73E+36	2.63E+36	2.36E+36	2.07E+36

Table A.6: Forecast evaluation results based on RMSE for a training period of 4 weeks, Belpex, St.Ghislain

Results for ISO New England, ME (RT)

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	19.21	21.35	22.93	24.24	28.47	28.56	27.93	29.36	30.92	31.17
ses	7.92	10.71	12.94	21.12	27.62	28.38	28.25	30.32	33.99	34.90
holts	16.74	26.25	42.47	84.48	120.17	149.95	210.04	274.13	527.63	906.49
holtwinters	24.65	54.05	85.92	144.90	203.27	257.96	368.25	482.61	936.53	1615.73
arima	9.78	13.81	16.13	20.09	25.98	26.87	27.18	29.37	34.24	37.68
tbats	9.61E+16	5.92E+16	6.82E+16	7.15E+16	6.03E+16	5.48E+16	4.56E+16	3.97E+16	2.81E+16	2.13E+16

Table A.7: Forecast evaluation results based on RMSE for a training period of 2 weeks, ISO NE, Portland

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	19.55	21.69	23.11	24.16	28.38	28.43	27.71	29.08	30.57	30.94
ses	7.96	10.58	12.86	20.98	27.56	28.33	28.20	30.29	33.93	34.89
holts	16.76	26.55	43.00	85.36	121.44	151.57	212.65	277.82	535.27	918.77
holtwinters	22.72	53.73	107.68	218.76	311.85	412.94	605.67	801.89	1582.44	2751.86
arima	9.42	13.01	15.81	19.23	24.53	25.21	25.17	26.90	30.67	32.61
tbats	2.24E+18	2.61E+18	2.31E+18	1.98E+18	1.76E+18	1.58E+18	1.33E+18	1.16E+18	8.24E+17	6.23E+17

Table A.8: Forecast evaluation results based on RMSE for a training period of 3 weeks, ISO NE, Portland

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	19.92	22.05	23.52	24.76	28.95	28.83	28.13	29.56	31.07	31.61
ses	7.70	10.26	12.61	20.76	27.36	28.11	27.95	30.07	33.72	34.67
holts	16.47	27.14	45.95	93.96	134.70	169.48	239.48	313.17	603.78	1036.84
holtwinters	18.60	43.70	79.50	139.59	204.63	265.58	386.07	508.92	999.22	1732.67
arima	8.58	12.42	15.33	18.73	24.17	24.88	24.88	26.65	30.38	32.33
tbats	3.55E+31	3.00E+32	2.59E+32	2.40E+32	2.34E+32	2.31E+32	2.27E+32	2.25E+32	2.18E+32	2.11E+32

Table A.9: Forecast evaluation results based on RMSE for a training period of 4 weeks, ISO NE, Portland

Results for PJM, VA (RT)

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	11.24	12.47	12.28	15.22	17.40	19.37	21.73	22.65	21.80	19.98
ses	4.62	6.13	6.20	12.82	16.55	18.90	21.05	22.68	22.44	20.27
holts	5.48	11.20	21.74	52.37	75.99	97.15	136.63	178.28	335.33	571.26
holtwinters	13.35	34.52	60.06	110.03	155.78	204.00	297.67	388.74	757.00	1312.02
arima	4.21	6.04	7.16	13.30	16.13	18.27	20.70	21.97	21.96	20.43
tbats	4.88E+13	3.37E+13	6.17E+13	1.04E+14	1.01E+14	8.86E+13	8.59E+13	8.22E+13	7.72E+13	7.47E+13

Table A.10: Forecast evaluation results based on RMSE for a training period of 2 weeks, PJM, Richmond

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	11.31	12.54	12.31	15.20	17.37	19.32	21.65	22.54	21.58	19.77
ses	4.70	6.19	6.26	12.87	16.58	18.95	21.12	22.77	22.50	20.32
holts	5.74	11.54	22.09	52.45	75.77	96.79	136.06	177.31	333.00	566.99
holtwinters	13.84	34.60	56.75	102.56	146.17	193.02	282.89	371.14	725.80	1259.30
arima	3.78	5.49	6.12	12.00	15.03	17.35	19.89	21.26	21.09	19.36
tbats	2.88	4.41	4.90	11.40	14.64	16.89	19.33	20.89	20.56	18.75

Table A.11: Forecast evaluation results based on RMSE for a training period of 3 weeks, PJM, Richmond

	1h	3h	6h	12h	18h	24h	36h	48h	96h	168h
mean	11.34	12.56	12.32	15.08	17.25	19.24	21.64	22.55	21.64	19.82
ses	4.73	6.25	6.29	12.90	16.62	19.01	21.20	22.86	22.59	20.42
holts	5.70	11.42	22.02	52.42	76.01	97.17	136.67	178.24	335.44	571.65
holtwinters	14.30	37.76	64.13	116.26	166.12	219.01	319.68	418.22	813.04	1409.65
arima	3.70	5.44	6.04	12.09	15.08	17.30	19.88	21.16	21.03	19.39
tbats	5.83E+55	2.82E+99	1.36E+109	9.65E+108	7.88E+108	6.82E+108	5.57E+108	4.82E+108	3.41E+108	2.58E+108

Table A.12: Forecast evaluation results based on RMSE for a training period of 4 weeks, PJM, Richmond

A.2 Result graphs

Results for Nord Pool Spot, FI (DA)

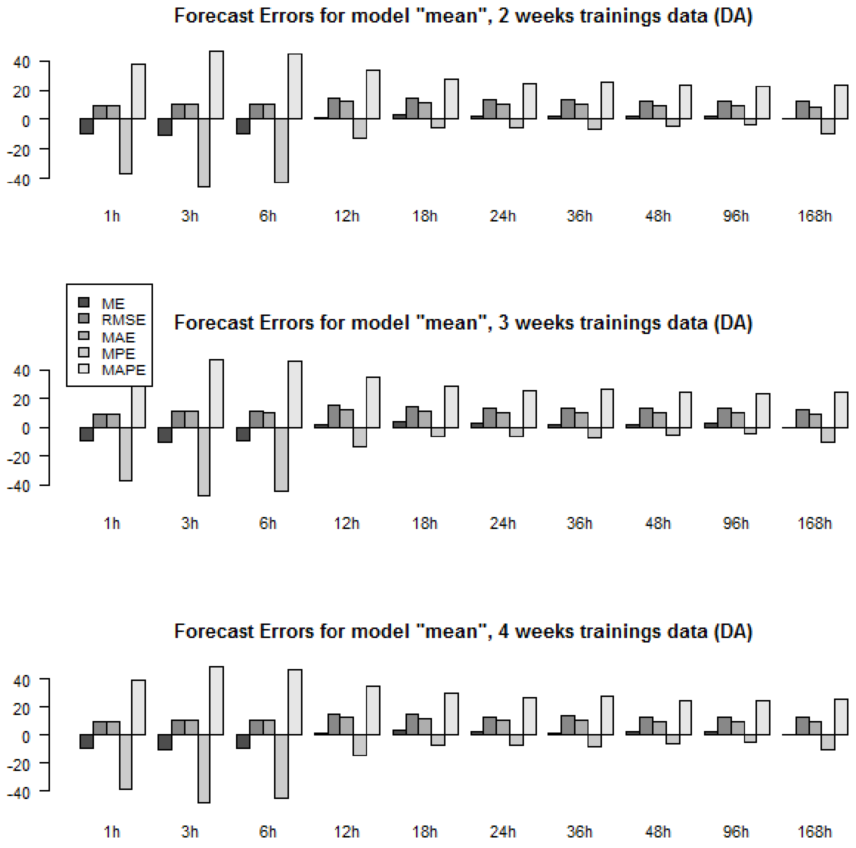


Figure A.1: Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

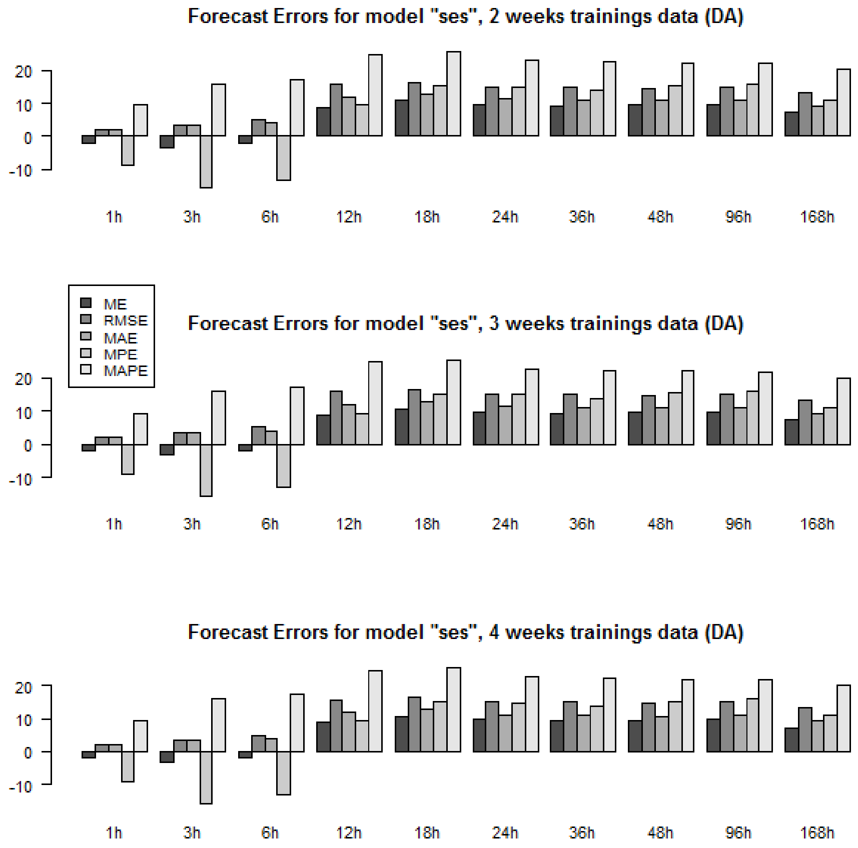


Figure A.2: Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

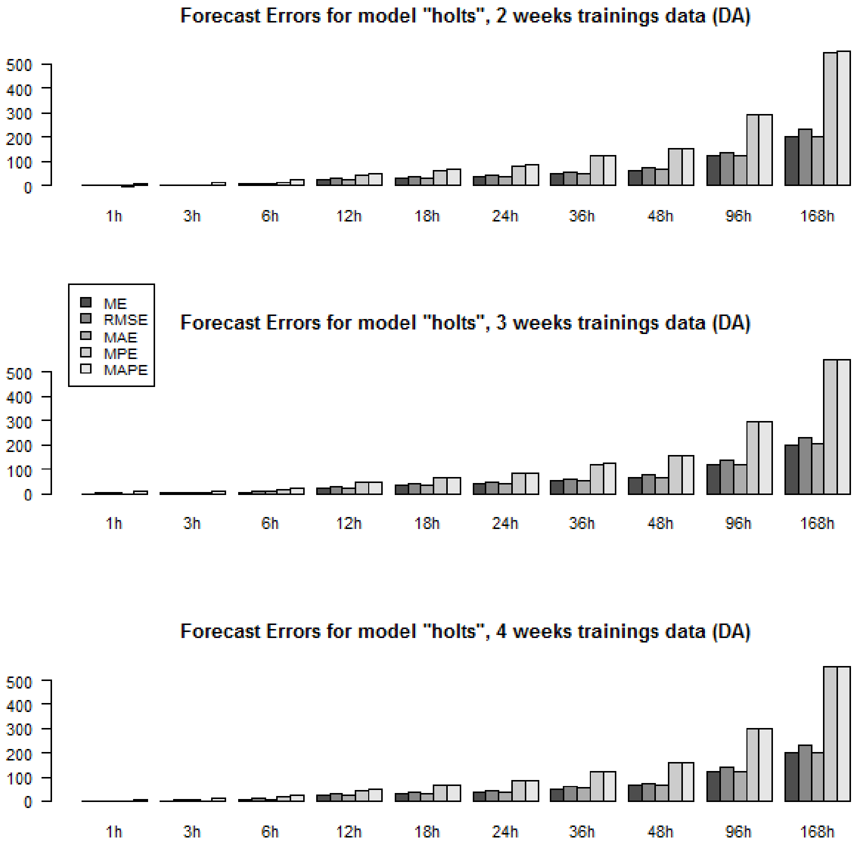


Figure A.3: Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

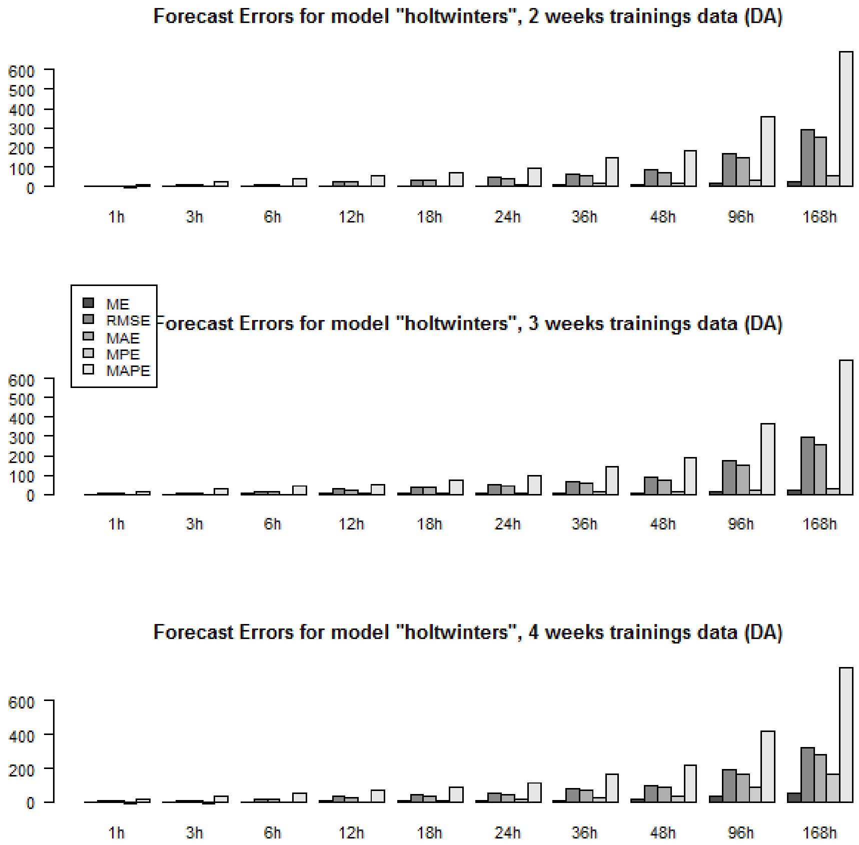


Figure A.4: Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

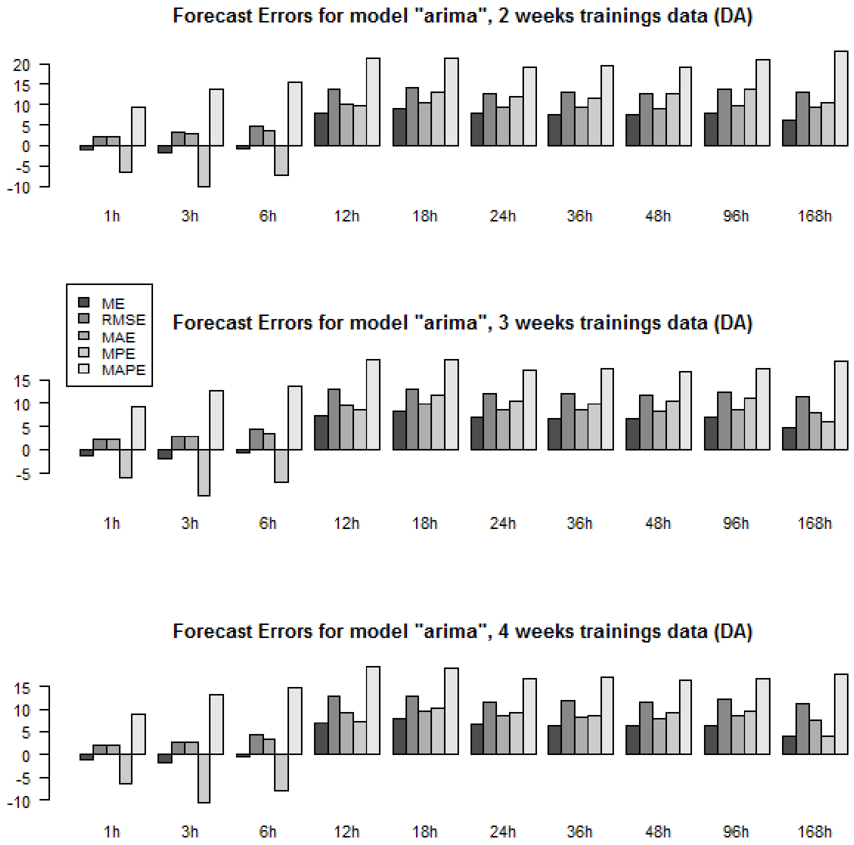


Figure A.5: Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

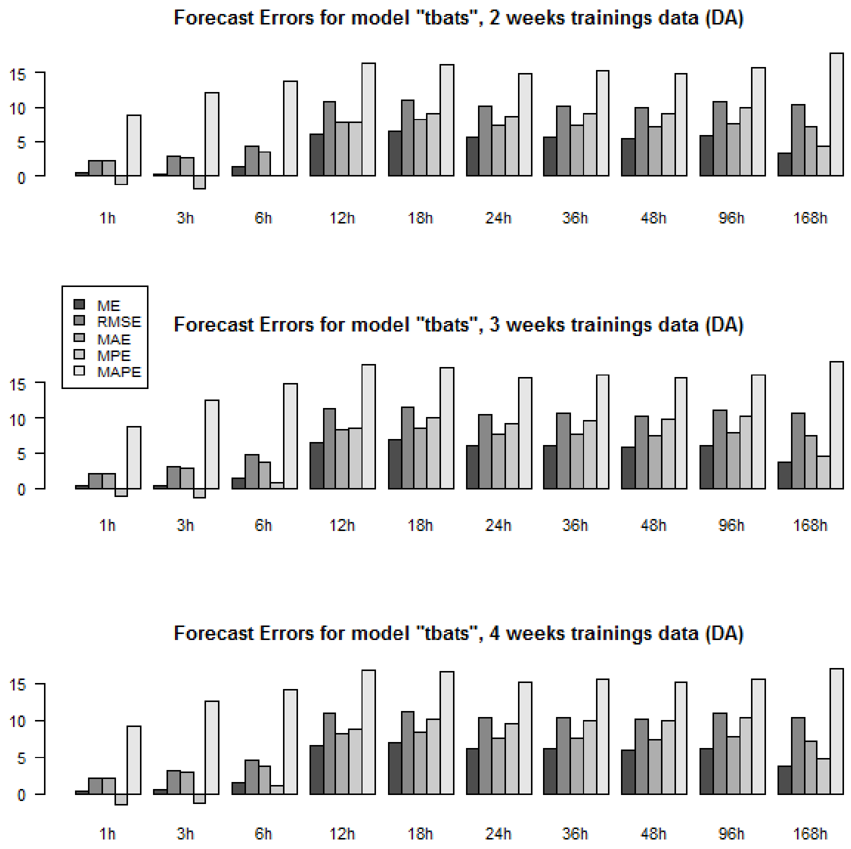


Figure A.6: Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, Nord Pool Spot, Hamina

Results for Belpex, BE (DA)

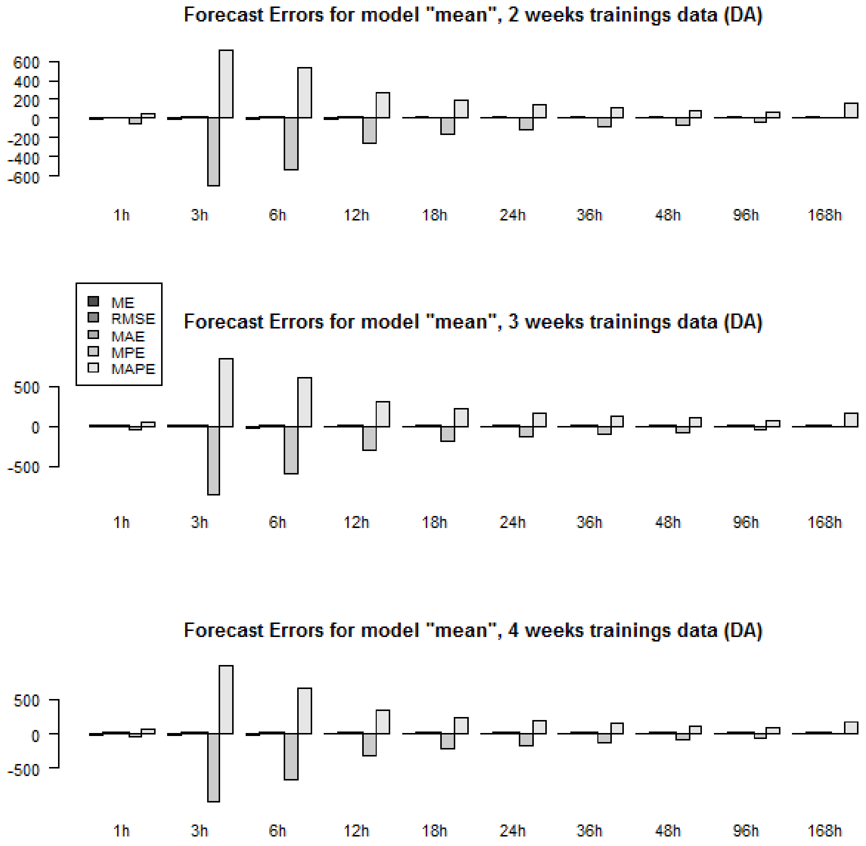


Figure A.7: Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

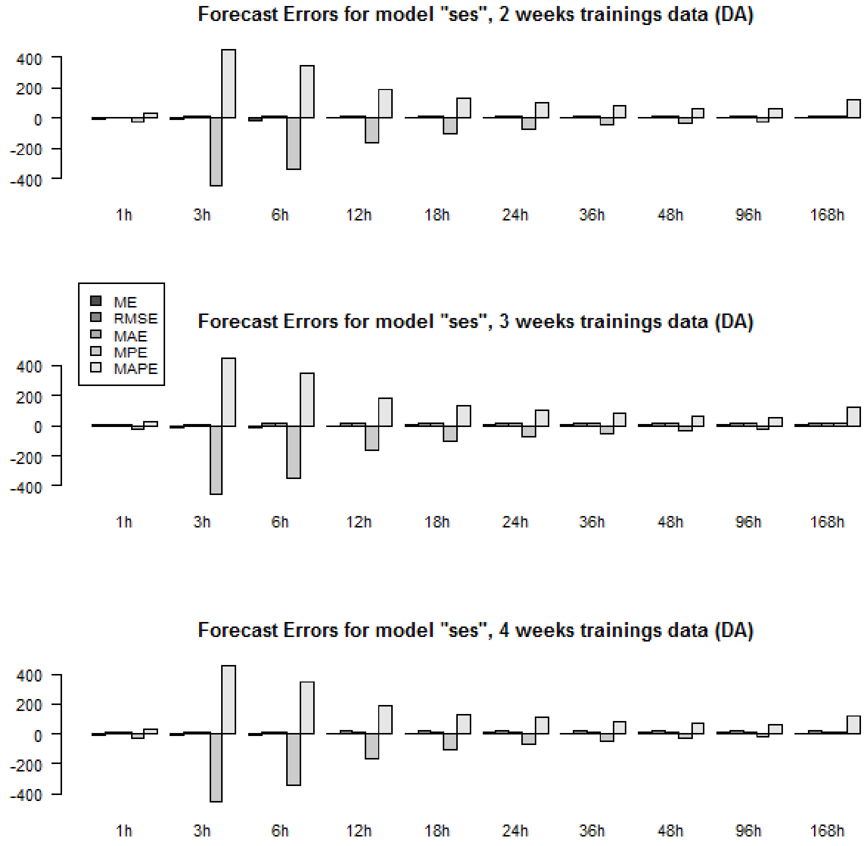


Figure A.8: Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

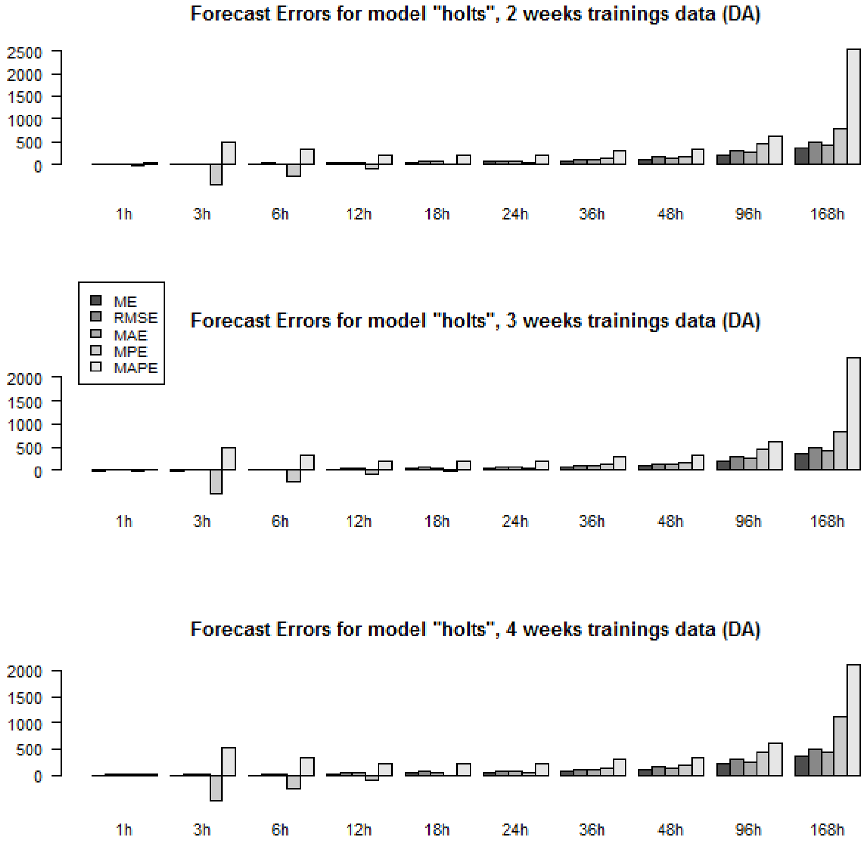


Figure A.9: Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

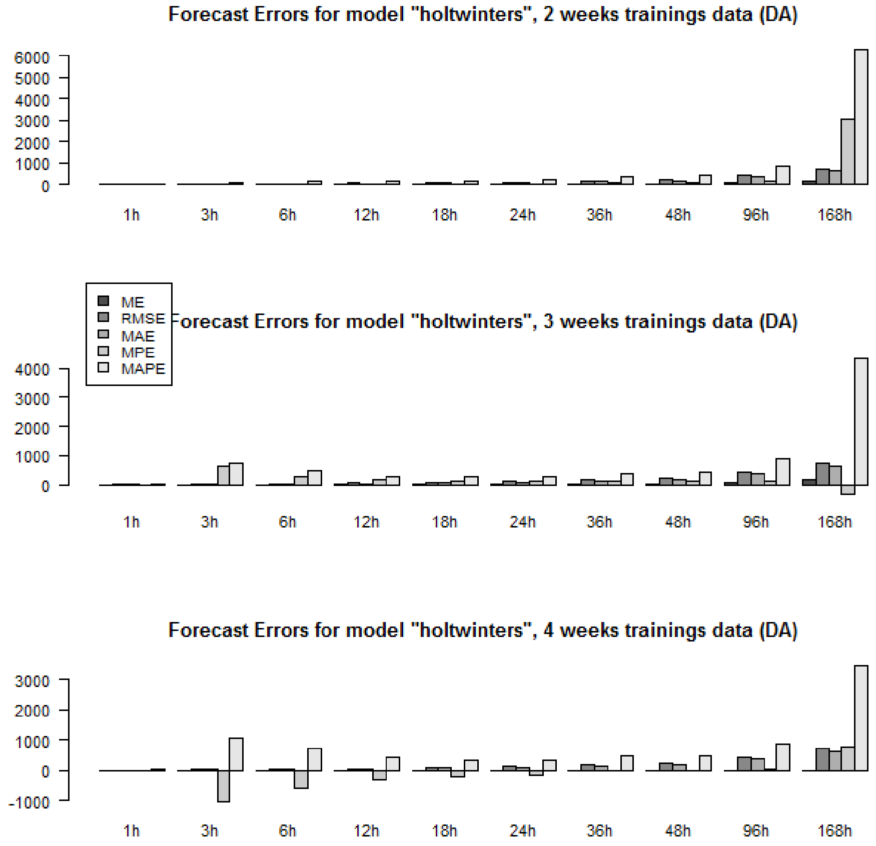


Figure A.10: Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

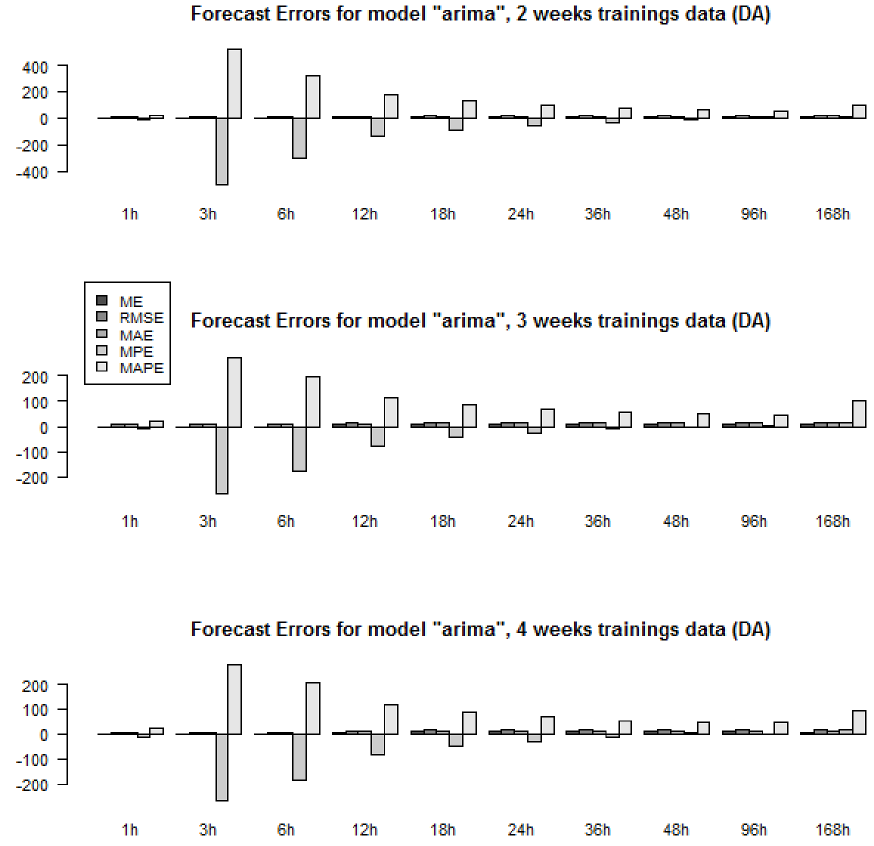


Figure A.11: Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

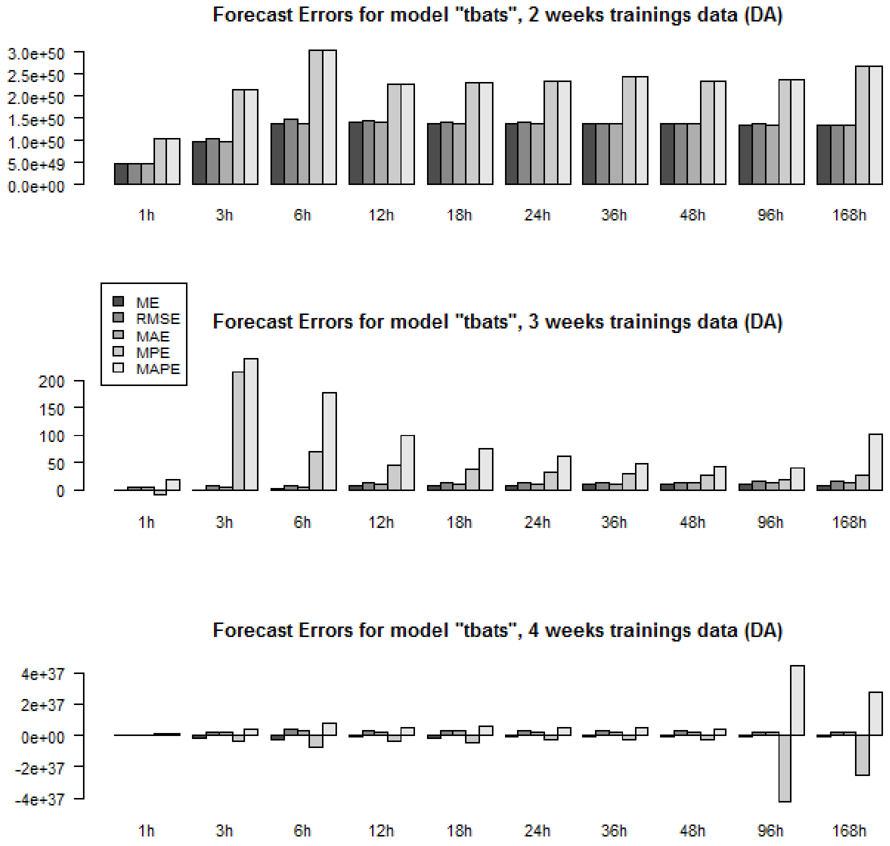
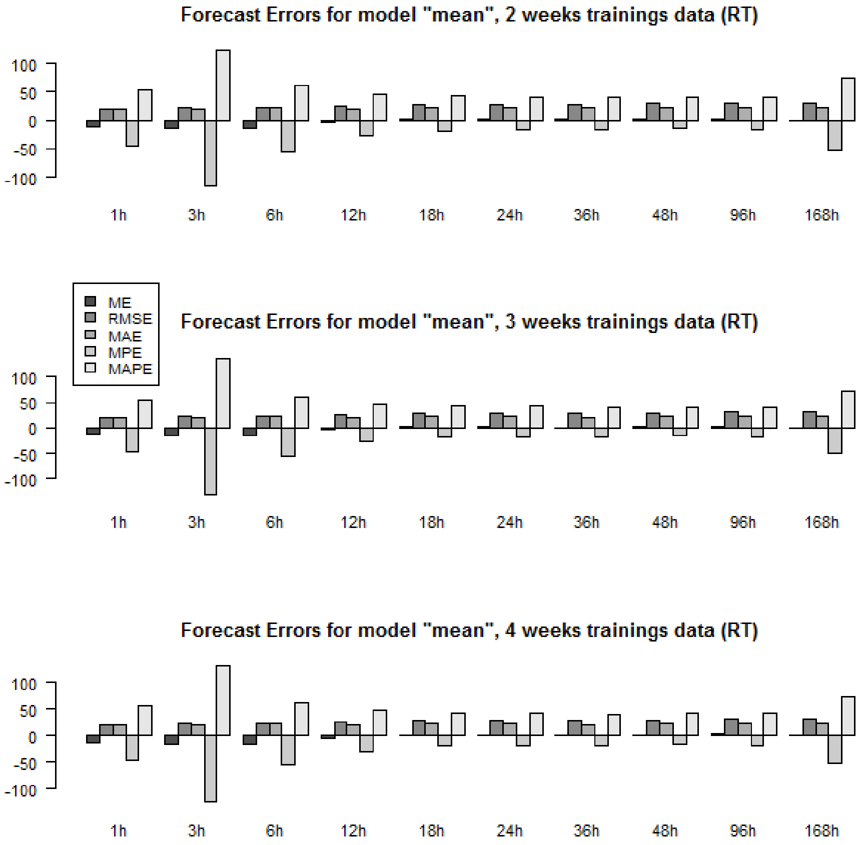


Figure A.12: Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, Belpex, St.Ghislain

Results for ISO New England, ME (RT)

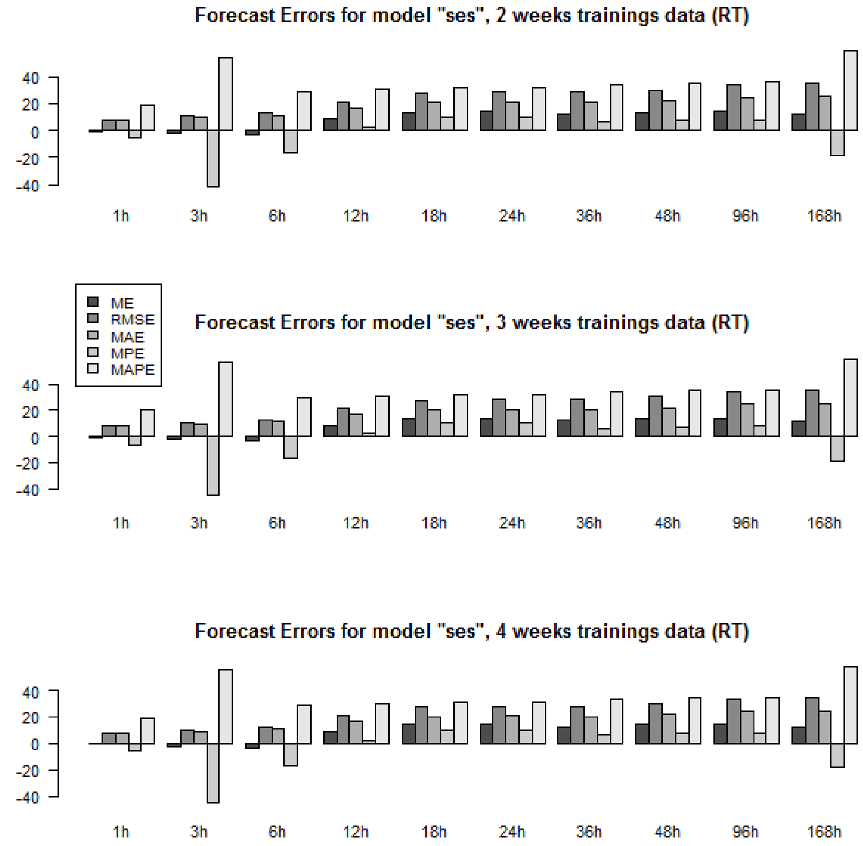


Forecast Errors for model "mean", 2 weeks trainings data (RT)

Forecast Errors for model "mean", 3 weeks trainings data (RT)

Forecast Errors for model "mean", 4 weeks trainings data (RT)

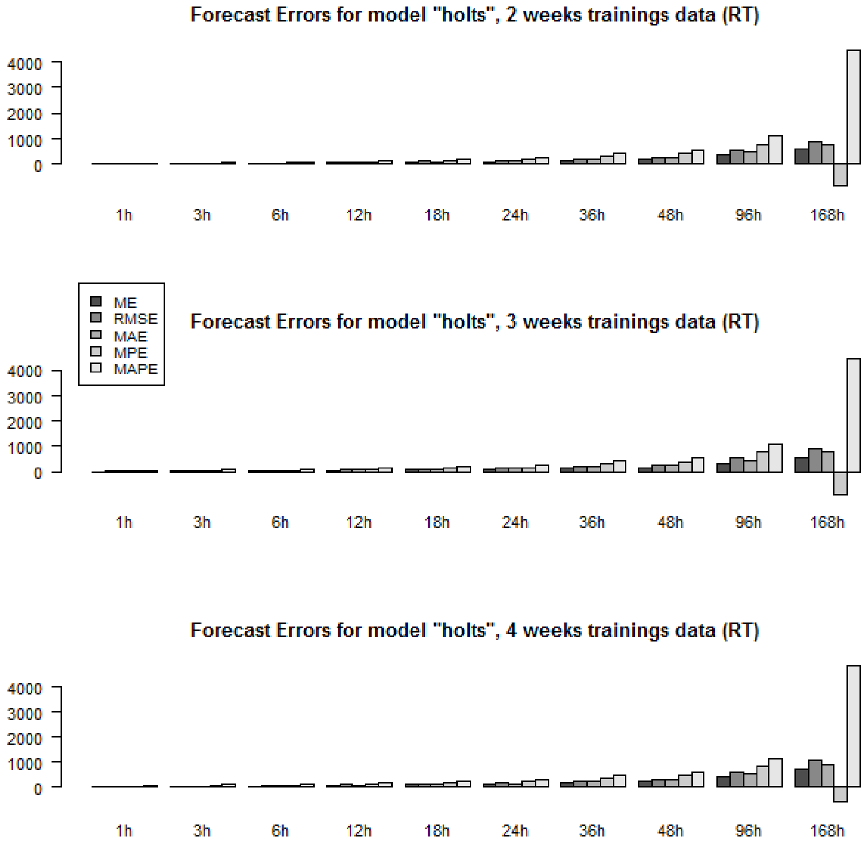
Figure A.13: Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, ISO NE, Portland


Forecast Errors for model "ses", 2 weeks trainings data (RT)

Forecast Errors for model "ses", 3 weeks trainings data (RT)

Forecast Errors for model "ses", 4 weeks trainings data (RT)

Figure A.14: Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, ISO NE, Portland


Forecast Errors for model "holts", 2 weeks trainings data (RT)

Forecast Errors for model "holts", 3 weeks trainings data (RT)

Forecast Errors for model "holts", 4 weeks trainings data (RT)

Figure A.15: Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, ISO NE, Portland

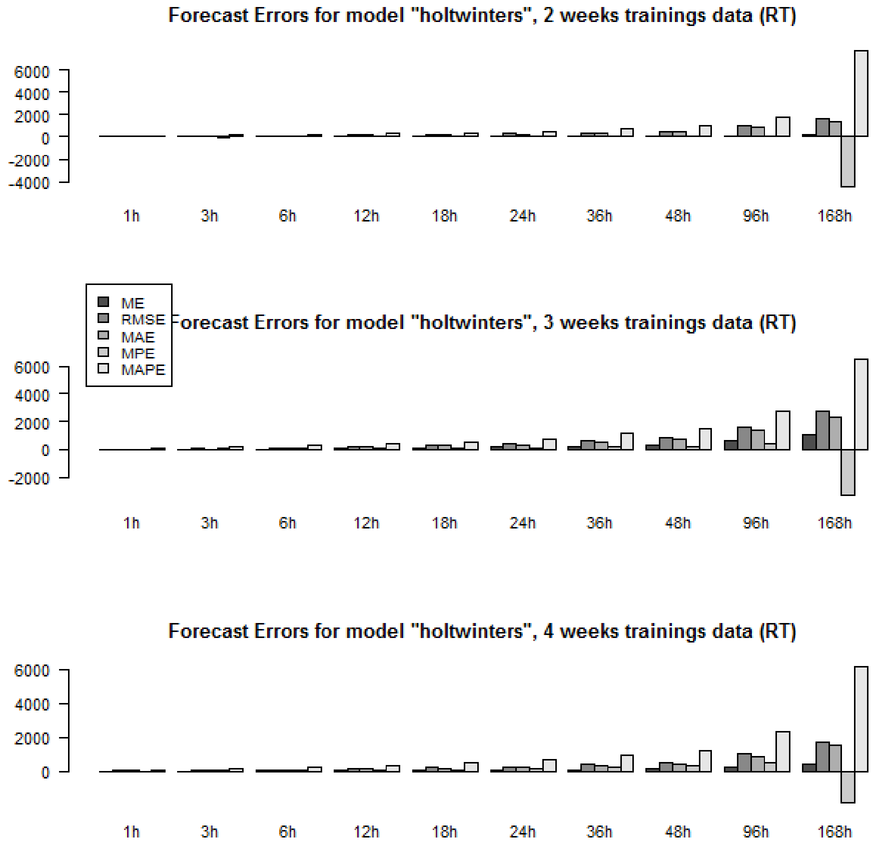


Figure A.16: Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, ISO NE, Portland

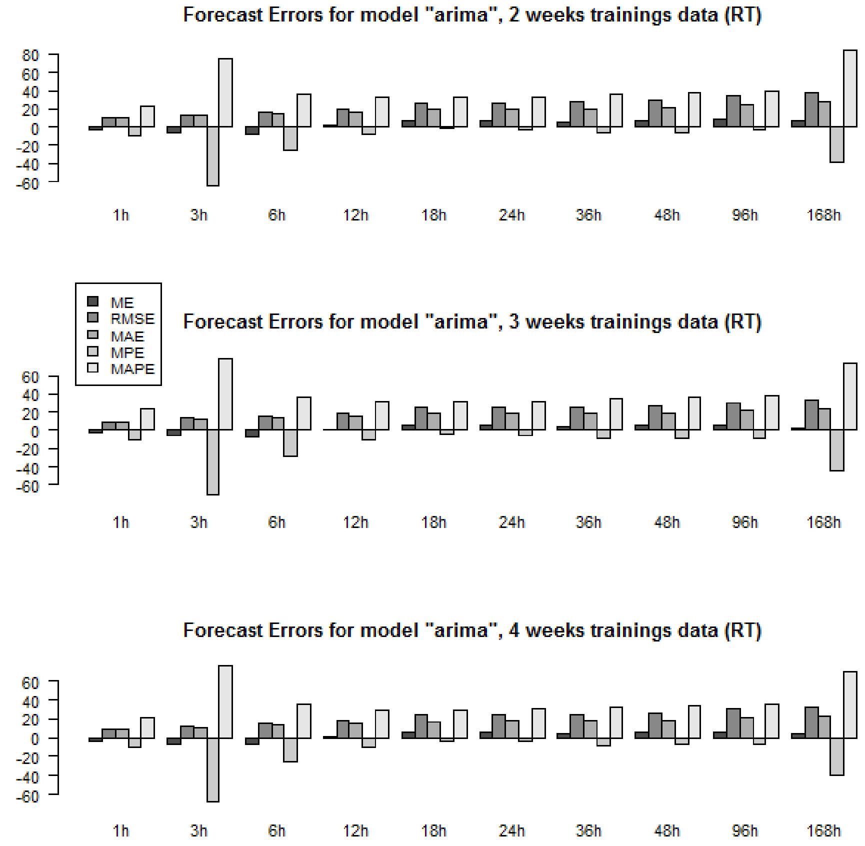


Figure A.17: Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, ISO NE, Portland

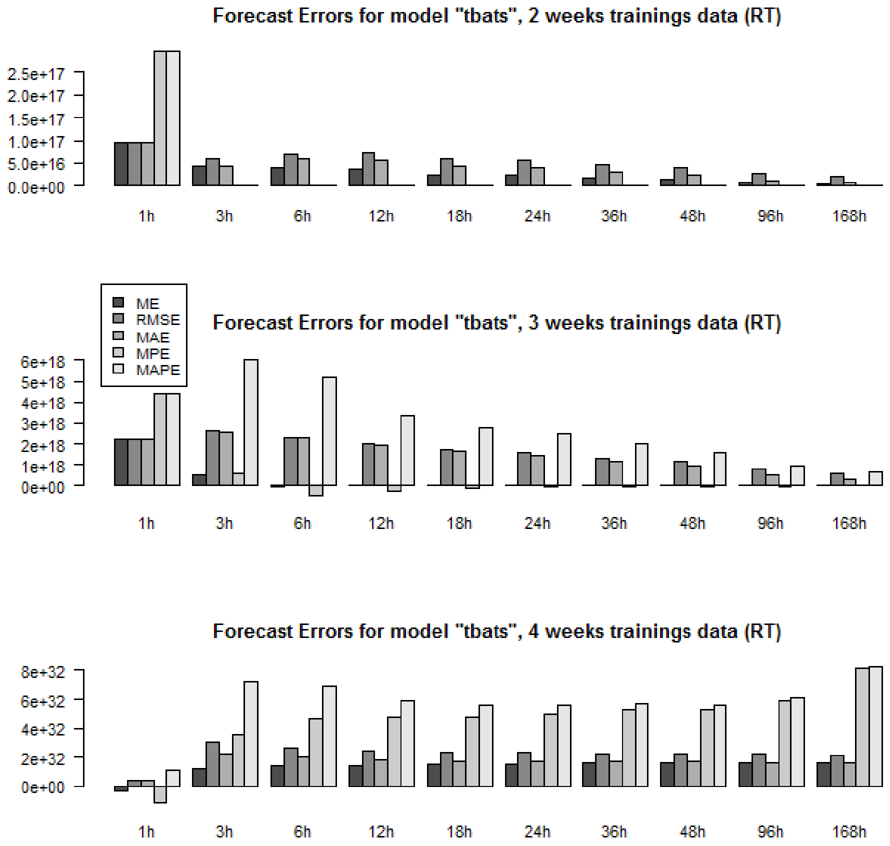


Figure A.18: Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, ISO NE, Portland

Results for PJM, VA (RT)

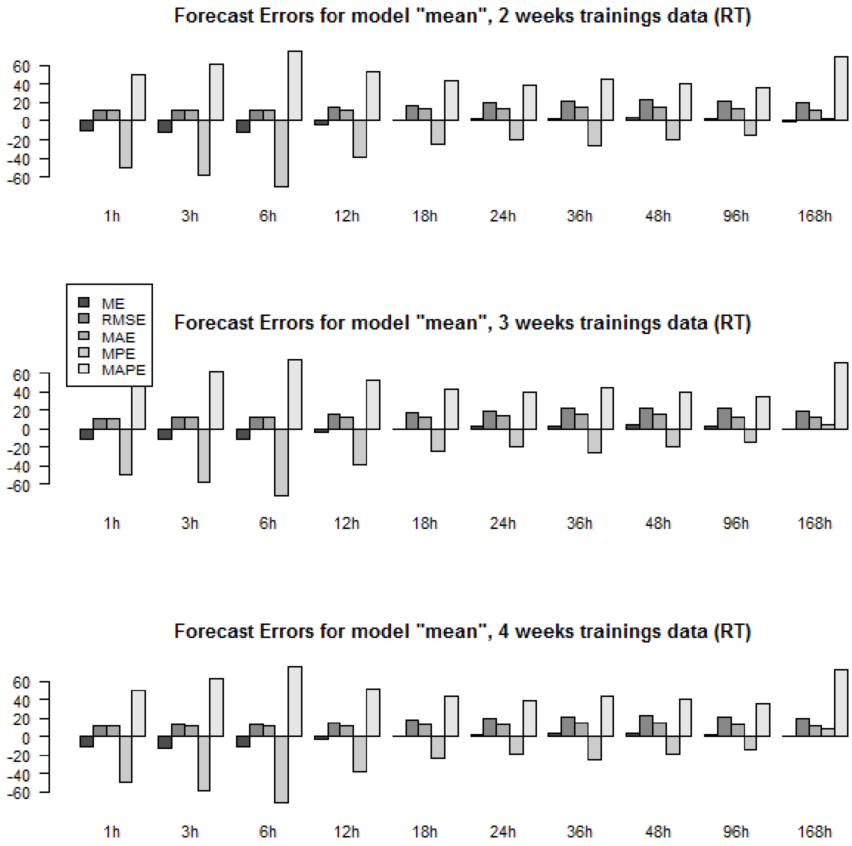


Figure A.19: Aggregated accuracy measures for Mean model and training data of 2, 3 and 4 weeks, PJM, Richmond

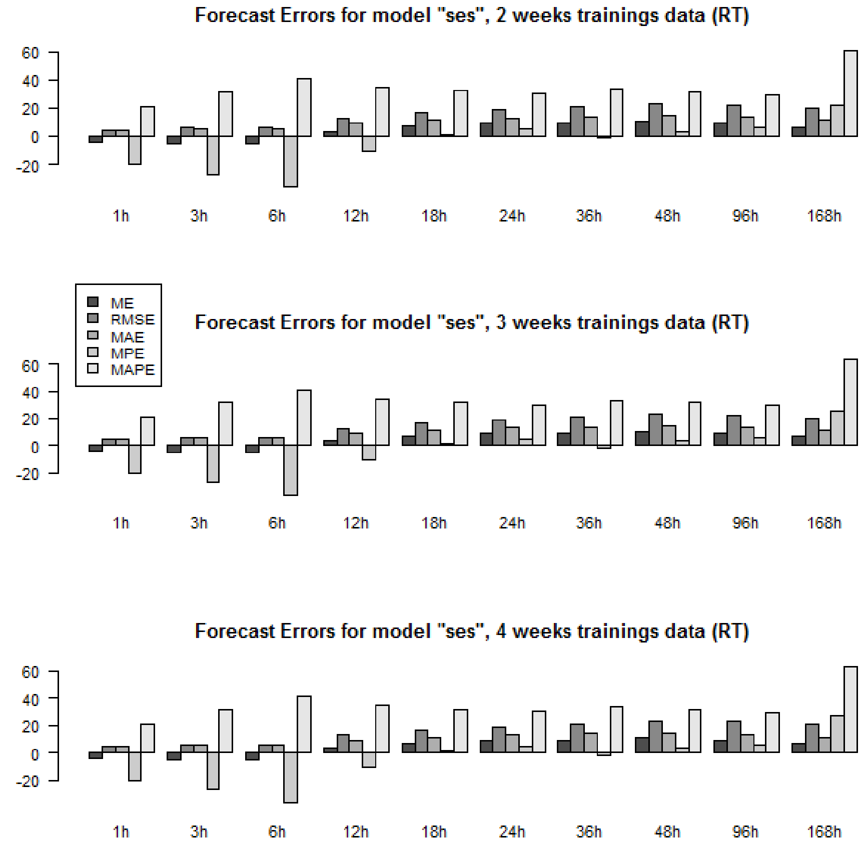


Figure A.20: Aggregated accuracy measures for SES model and training data of 2, 3 and 4 weeks, PJM, Richmond

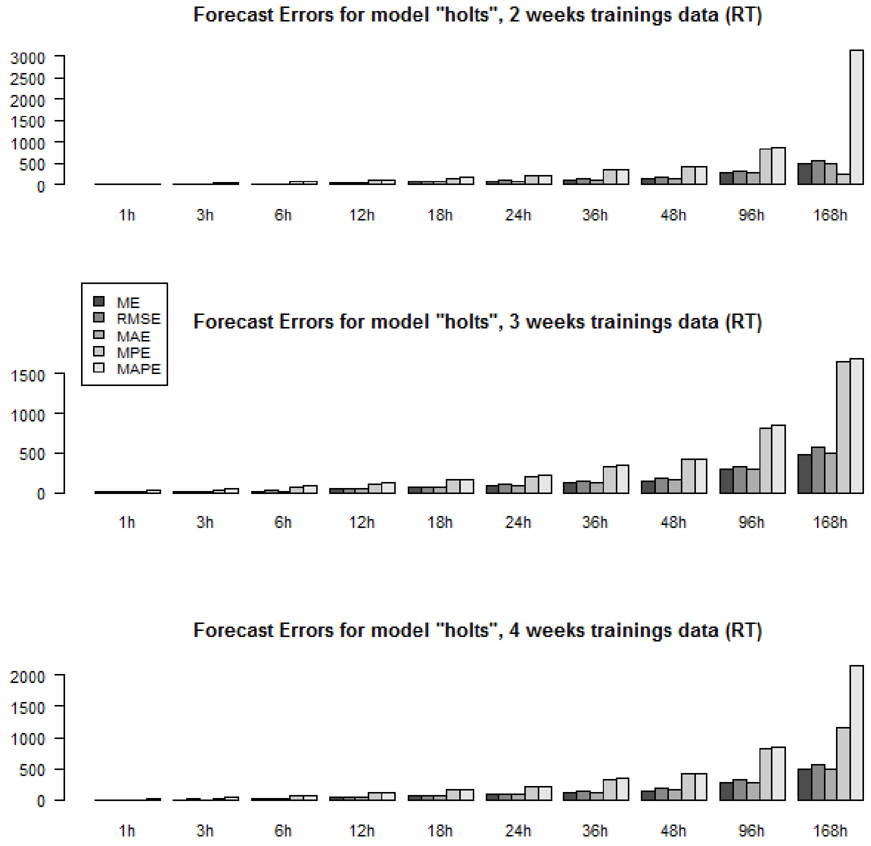


Figure A.21: Aggregated accuracy measures for Holts model and training data of 2, 3 and 4 weeks, PJM, Richmond

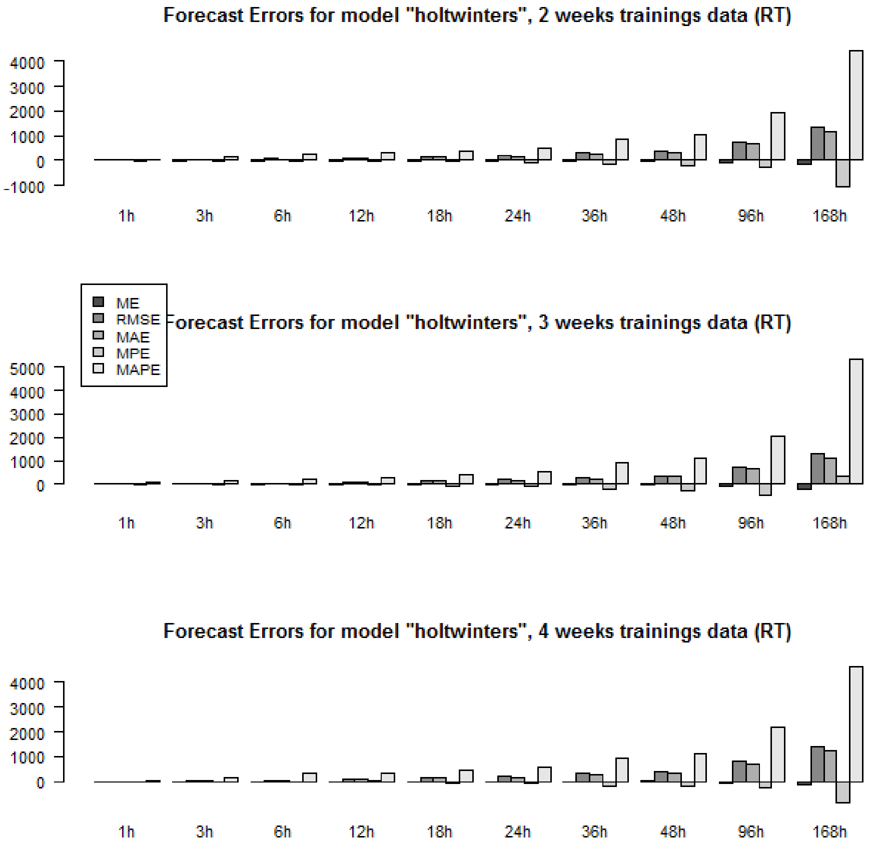


Figure A.22: Aggregated accuracy measures for HoltWinters model and training data of 2, 3 and 4 weeks, PJM, Richmond

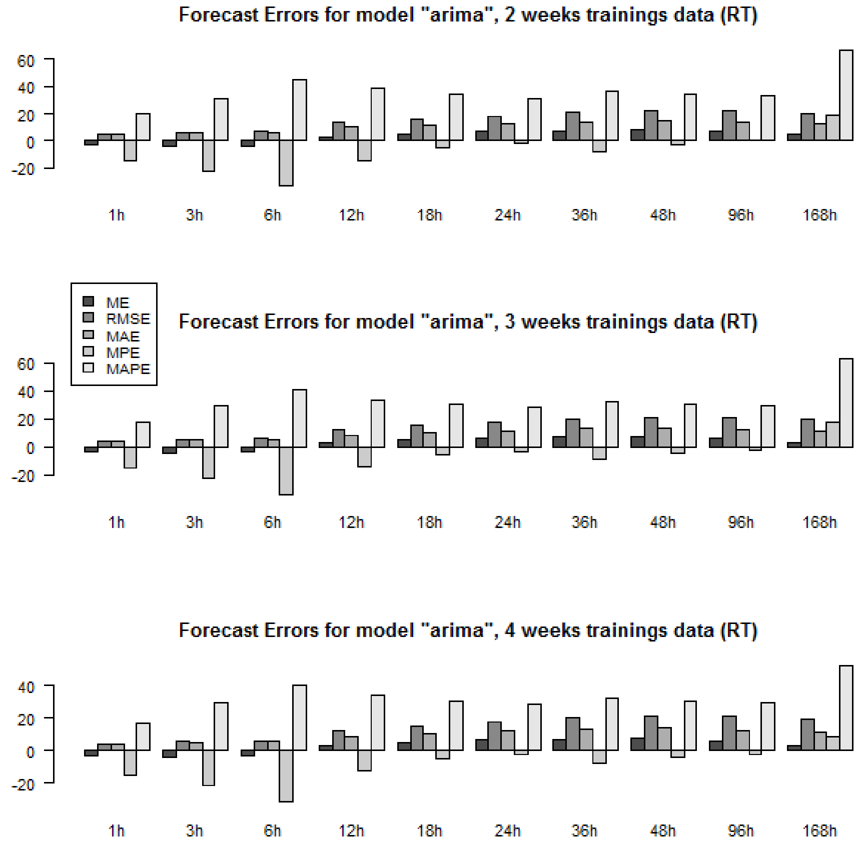


Figure A.23: Aggregated accuracy measures for ARIMA model and training data of 2, 3 and 4 weeks, PJM, Richmond

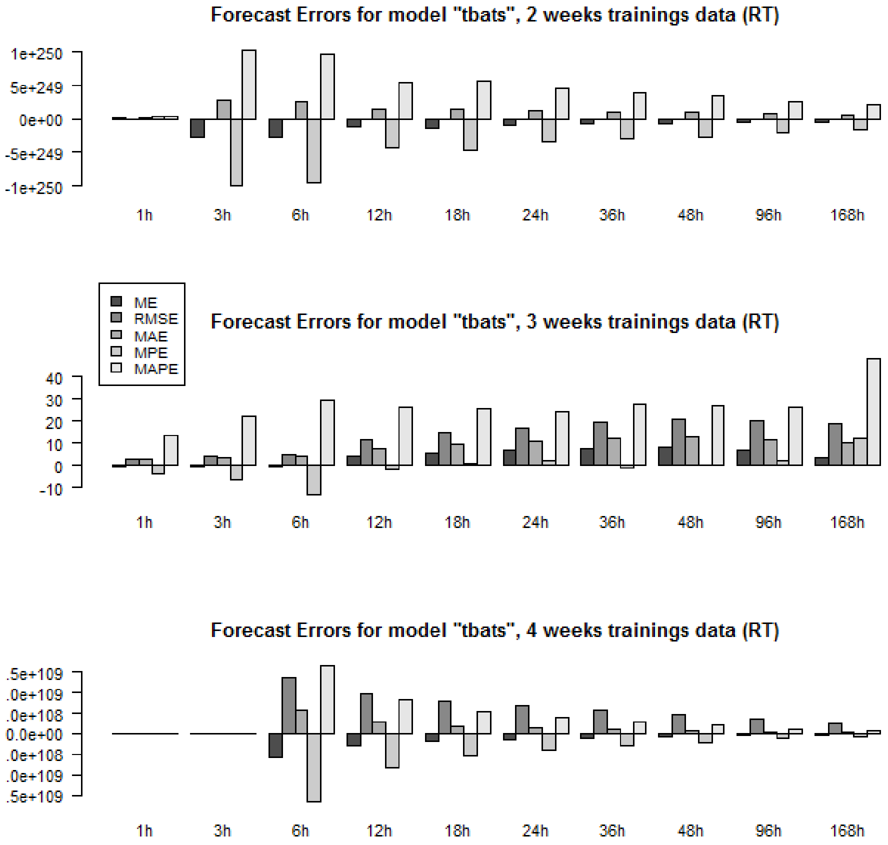


Figure A.24: Aggregated accuracy measures for TBATS model and training data of 2, 3 and 4 weeks, PJM, Richmond

APPENDIX B

Simulation Results

B.1 Result tables

Cloud metrics results

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	4984.39	4830.12	4652.92	4731.71	4688.42	4597.72	4685.54
TCC	208.36	181.72	171.85	174.53	172.50	168.28	169.60
ME	0.00	0.00	0.00	0.00	1.95	0.22	0.53
MC	0.00	0.00	0.00	0.00	13.69	1.58	3.70
TCPWM	4984.39	4830.12	4652.92	4731.71	4690.37	4597.95	4686.07
TCCWM	208.36	181.72	171.85	174.53	186.19	169.86	173.29
TPC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0.00	0.00	0.00	0.00	8716.00	1401.00	3499.00
NM	0.00	0.00	0.00	0.00	2162.00	221.00	510.00
TP	4984.39	4830.12	4652.92	4731.71	4690.37	4597.95	4686.07
TC	208.36	181.72	171.85	174.53	186.19	169.86	173.29

Table B.1: Cloud metrics for simulation „DA Summer“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	15338.31	14211.48	13545.14	13789.94	13903.81	13462.65	13794.13
TCC	930.41	630.13	592.83	600.42	606.20	585.45	594.32
ME	0.00	0.00	0.00	0.00	2.76	0.07	0.21
MC	0.00	0.00	0.00	0.00	19.42	0.53	1.49
TCPWM	15338.31	14211.48	13545.14	13789.94	13906.57	13462.72	13794.34
TCCWM	930.41	630.13	592.83	600.42	625.62	585.97	595.81
TPC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0.00	0.00	0.00	0.00	12728.00	482.00	1269.00
NM	0.00	0.00	0.00	0.00	3332.00	75.00	228.00
TP	15338.31	14211.48	13545.14	13789.94	13906.57	13462.72	13794.34
TC	930.41	630.13	592.83	600.42	625.62	585.97	595.81

Table B.2: Cloud metrics for simulation „DA Spring“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	6922.22	6435.57	6263.45	6343.36	6048.94	6113.84	6198.98
TCC	315.96	263.32	265.21	256.08	245.97	258.30	248.28
ME	0.00	0.00	0.00	0.00	3.97	0.24	0.54
MC	0.00	0.00	0.00	0.00	27.90	1.69	3.81
TCPWM	6922.22	6435.57	6263.45	6343.36	6052.91	6114.08	6199.52
TCCWM	315.96	263.32	265.21	256.08	273.87	259.98	252.09
TPC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0.00	0.00	0.00	0.00	9575.00	461.00	1729.00
NM	0.00	0.00	0.00	0.00	5580.00	319.00	671.00
TP	6922.22	6435.57	6263.45	6343.36	6052.91	6114.08	6199.52
TC	315.96	263.32	265.21	256.08	273.87	259.98	252.09

Table B.3: Cloud metrics for simulation „RT Summer“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	21501.23	19938.71	19173.33	19437.17	18978.17	18979.72	19016.16
TCC	966.04	710.59	659.92	699.13	673.24	645.01	660.06
ME	0.00	0.00	0.00	0.00	10.51	0.37	1.18
MC	0.00	0.00	0.00	0.00	73.82	2.56	8.29
TCPWM	21501.23	19938.71	19173.33	19437.17	18988.68	18980.09	19017.34
TCCWM	966.04	710.59	659.92	699.13	747.06	647.57	668.35
TPC	0.00	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0.00	0.00	0.00	0.00	24578.00	773.00	3787.00
NM	0.00	0.00	0.00	0.00	14401.00	476.00	1398.00
TP	21501.23	19938.71	19173.33	19437.17	18988.68	18980.09	19017.34
TC	966.04	710.59	659.92	699.13	747.06	647.57	668.35

Table B.4: Cloud metrics for simulation „RT Spring“

Normalized cloud metrics results

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	100	96.90	93.35	94.93	94.06	92.24	94.00
TCC	100	87.21	82.48	83.76	82.79	80.76	81.39
ME	0	0.00	0.00	0.00	100.00	11.53	26.98
MC	0	0.00	0.00	0.00	100.00	11.55	27.01
TCPWM	100	96.90	93.35	94.93	94.10	92.25	94.01
TCCWM	100	87.21	82.48	83.76	89.36	81.52	83.17
TPC	0	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0	0.00	0.00	0.00	100.00	16.07	40.14
NM	0	0.00	0.00	0.00	100.00	10.22	23.59
TP	100	96.90	93.35	94.93	94.10	92.25	94.01
TC	100	87.21	82.48	83.76	89.36	81.52	83.17

Table B.5: Normalized cloud metrics for simulation „DA Summer“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	100	92.65	88.31	89.91	90.65	87.77	89.93
TCC	100	67.73	63.72	64.53	65.15	62.92	63.88
ME	0	0.00	0.00	0.00	100.00	2.71	7.67
MC	0	0.00	0.00	0.00	100.00	2.71	7.68
TCPWM	100	92.65	88.31	89.91	90.67	87.77	89.93
TCCWM	100	67.73	63.72	64.53	67.24	62.98	64.04
TPC	0	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0	0.00	0.00	0.00	100.00	3.79	9.97
NM	0	0.00	0.00	0.00	100.00	2.25	6.84
TP	100	92.65	88.31	89.91	90.67	87.77	89.93
TC	100	67.73	63.72	64.53	67.24	62.98	64.04

Table B.6: Normalized cloud metrics for simulation „DA Spring“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	100	92.97	90.48	91.64	87.38	88.32	89.55
TCC	100	83.34	83.94	81.05	77.85	81.75	78.58
ME	0	0.00	0.00	0.00	100.00	6.04	13.64
MC	0	0.00	0.00	0.00	100.00	6.04	13.66
TCPWM	100	92.97	90.48	91.64	87.44	88.33	89.56
TCCWM	100	83.34	83.94	81.05	86.68	82.28	79.79
TPC	0	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0	0.00	0.00	0.00	100.00	4.81	18.06
NM	0	0.00	0.00	0.00	100.00	5.72	12.03
TP	100	92.97	90.48	91.64	87.44	88.33	89.56
TC	100	83.34	83.94	81.05	86.68	82.28	79.79

Table B.7: Normalized cloud metrics for simulation „RT Summer“

	BFD	BCF	BCF_F	BCF_IF	BCU_M	BCU_MF	BCU_MIF
TCP	100	92.73	89.17	90.40	88.27	88.27	88.44
TCC	100	73.56	68.31	72.37	69.69	66.77	68.33
ME	0	0.00	0.00	0.00	100.00	3.47	11.23
MC	0	0.00	0.00	0.00	100.00	3.47	11.24
TCPWM	100	92.73	89.17	90.40	88.31	88.27	88.45
TCCWM	100	73.56	68.31	72.37	77.33	67.03	69.18
TPC	0	0.00	0.00	0.00	0.00	0.00	0.00
TDT	0	0.00	0.00	0.00	100.00	3.15	15.41
NM	0	0.00	0.00	0.00	100.00	3.31	9.71
TP	100	92.73	89.17	90.40	88.31	88.27	88.45
TC	100	73.56	68.31	72.37	77.33	67.03	69.18

Table B.8: Normalized cloud metrics for simulation „RT Spring“

B.2 Result graphs

Simulation results for DA Summer

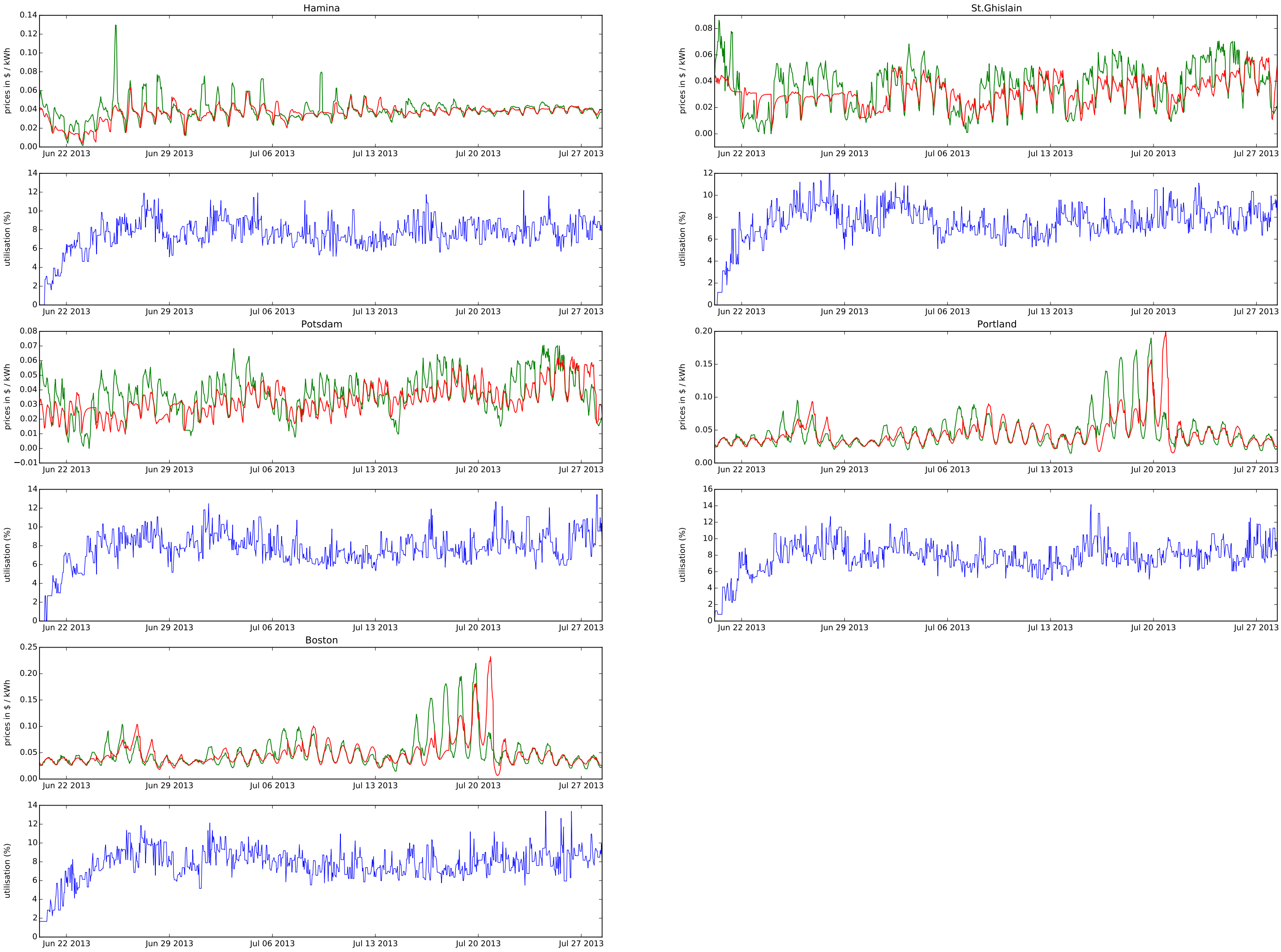


Figure B.1: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BFD

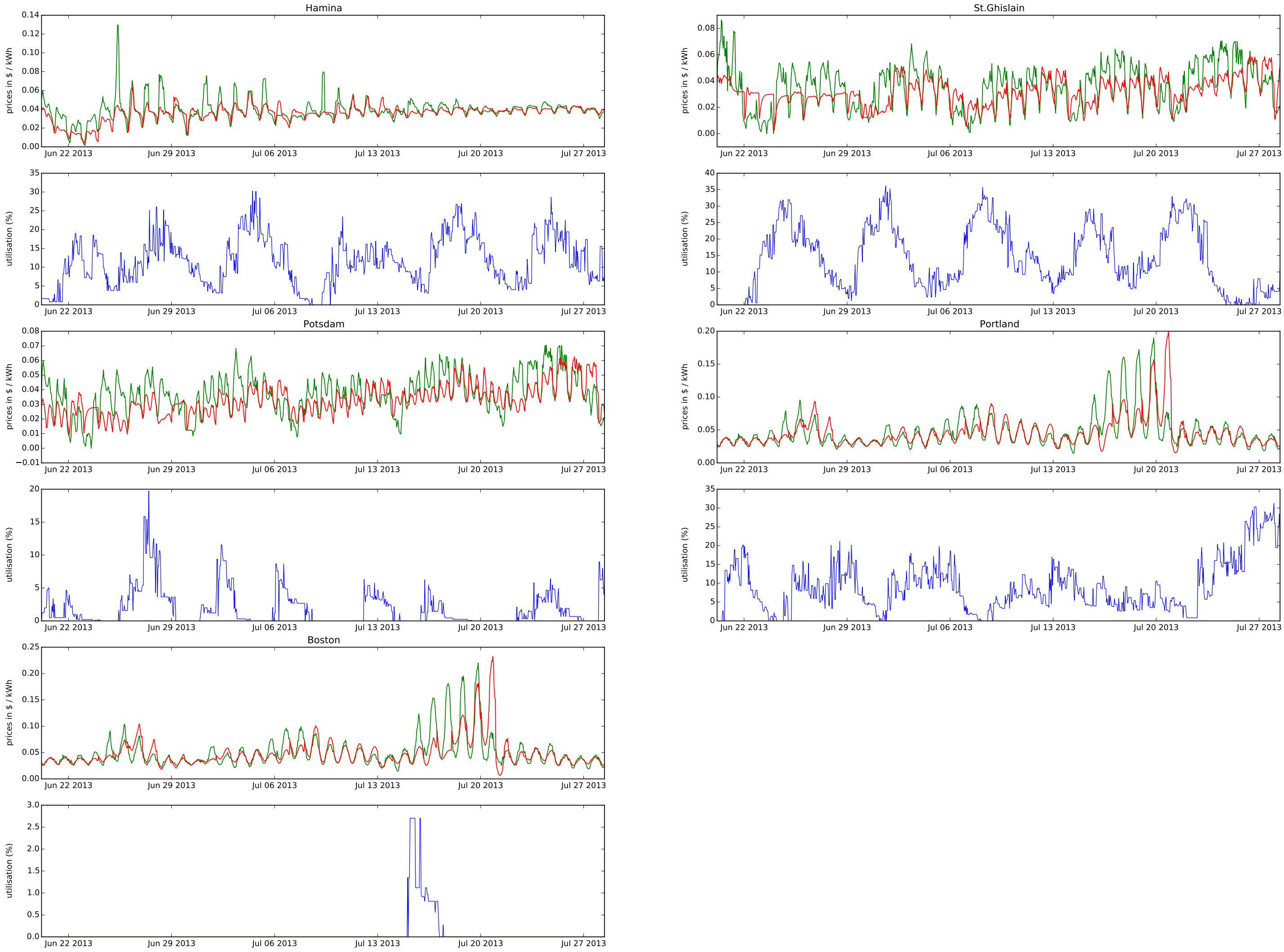


Figure B.2: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF

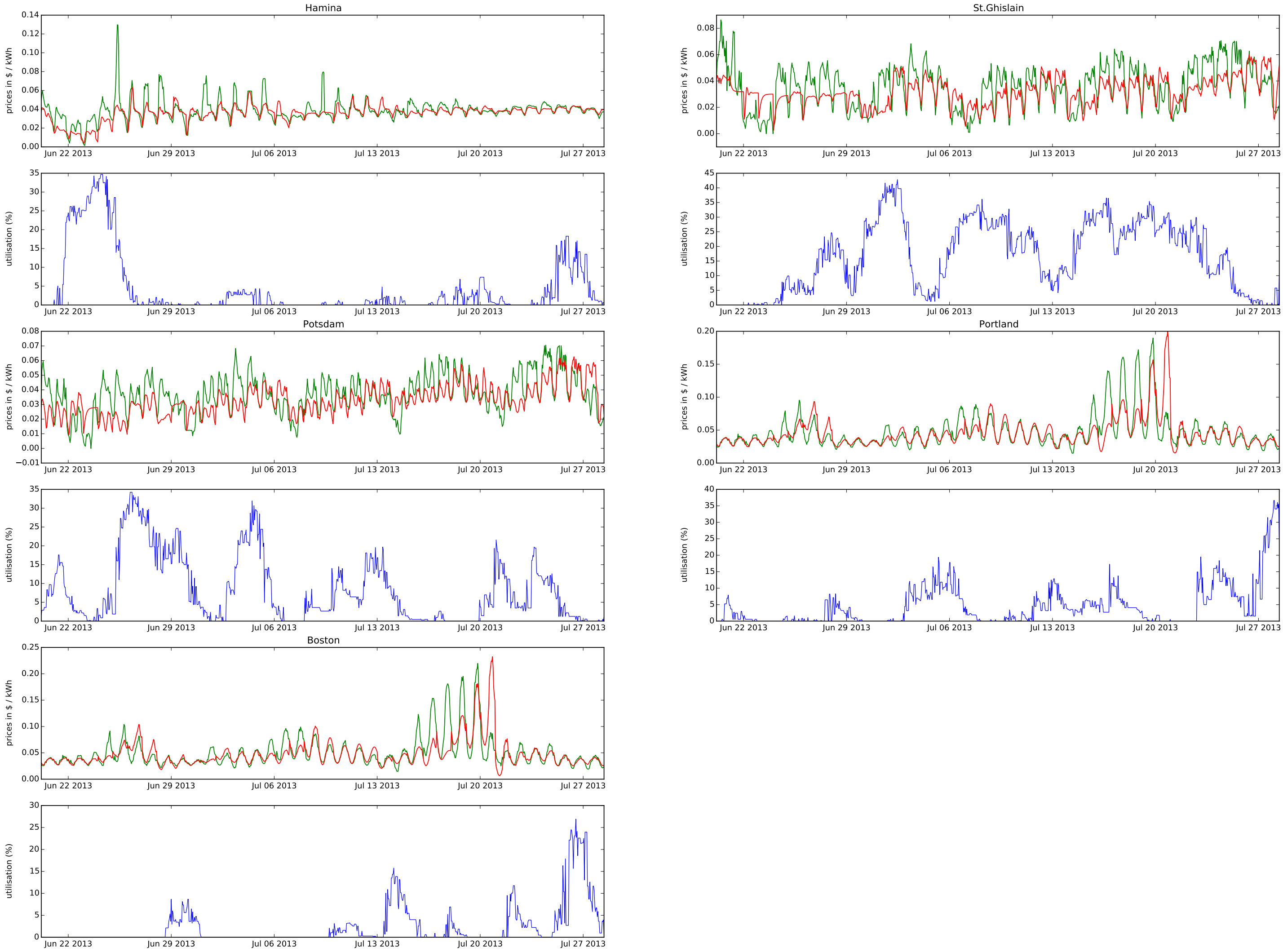


Figure B.3: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF_F

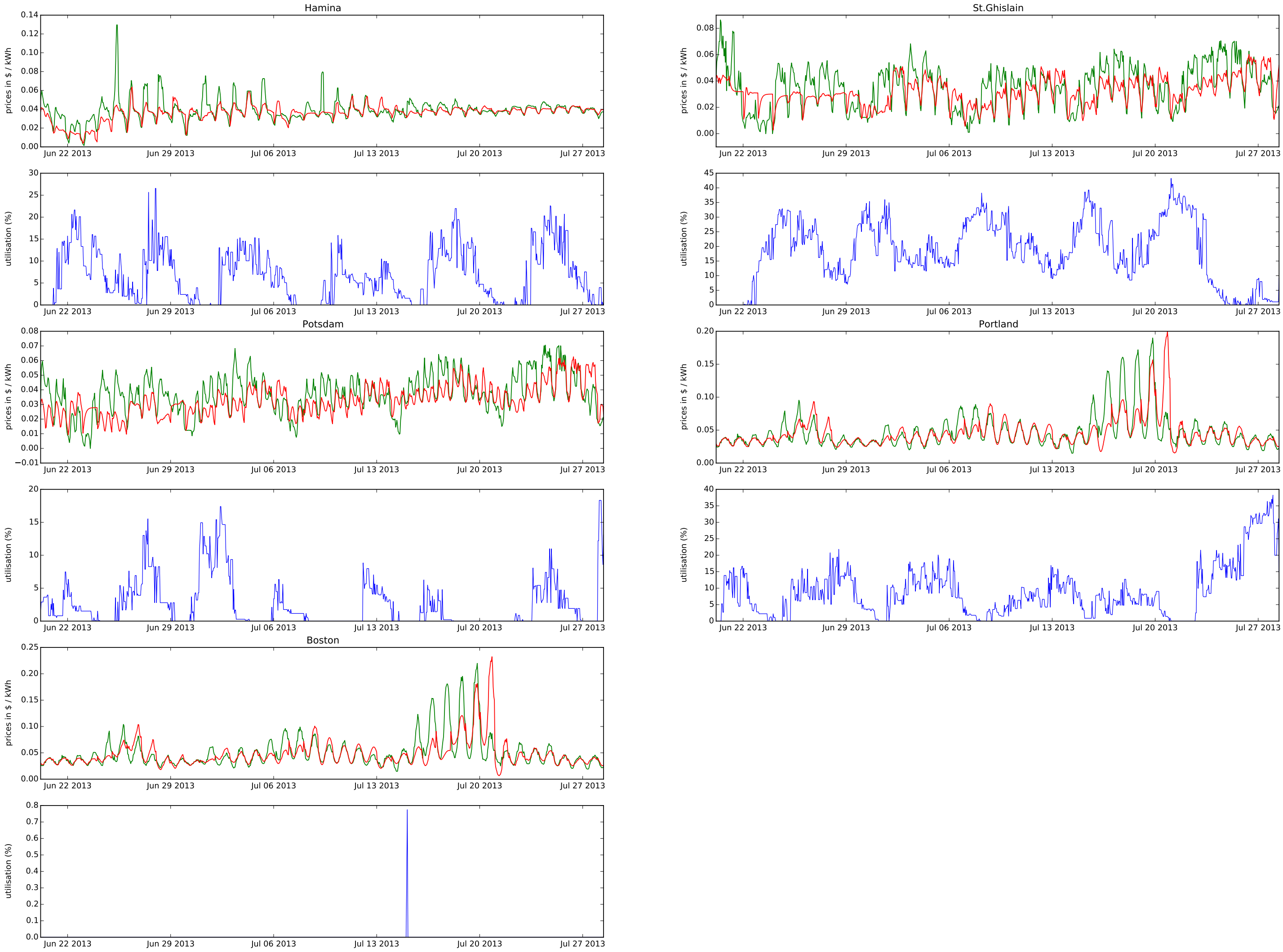


Figure B.4: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCF_IF

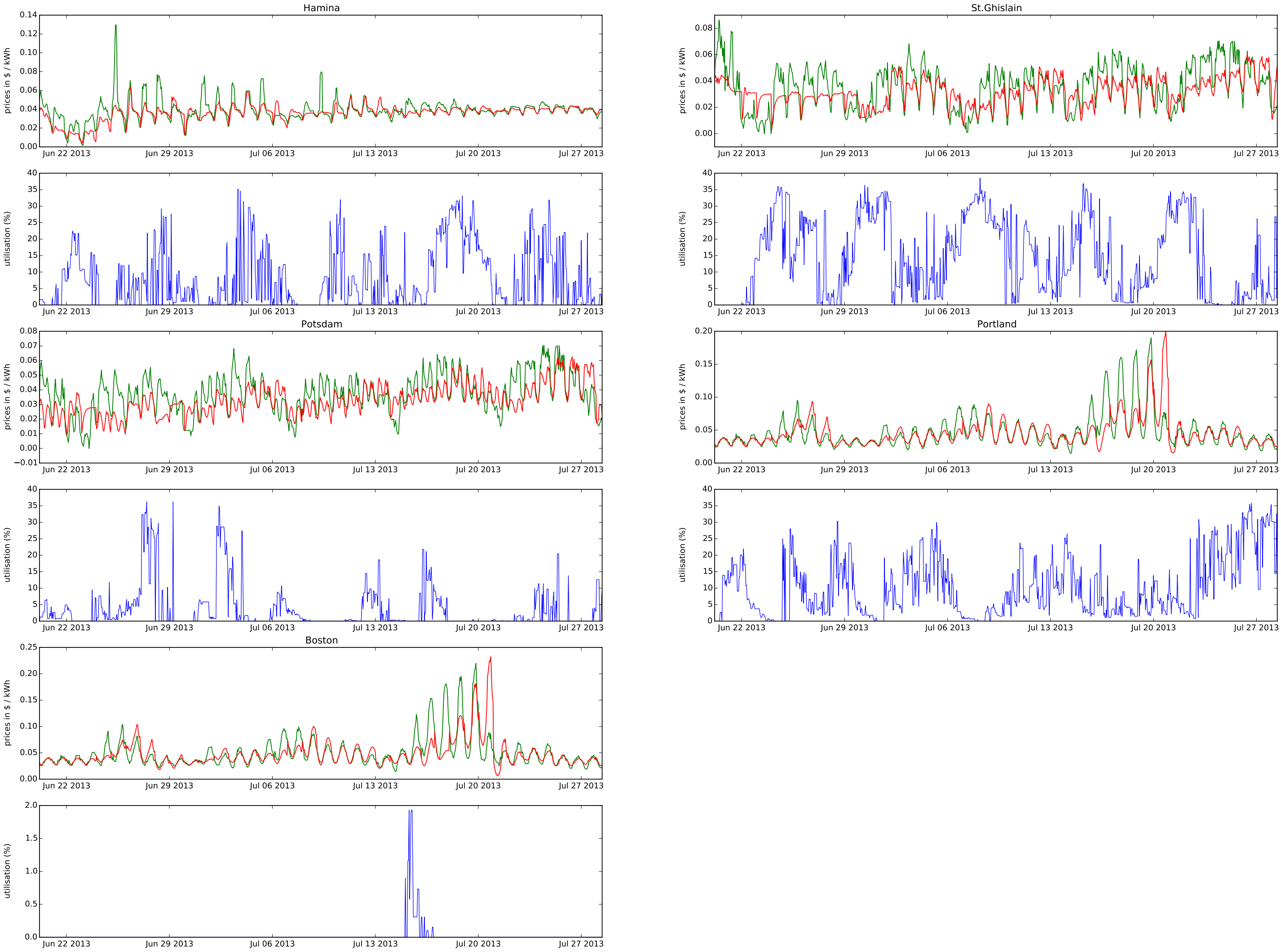


Figure B.5: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_M

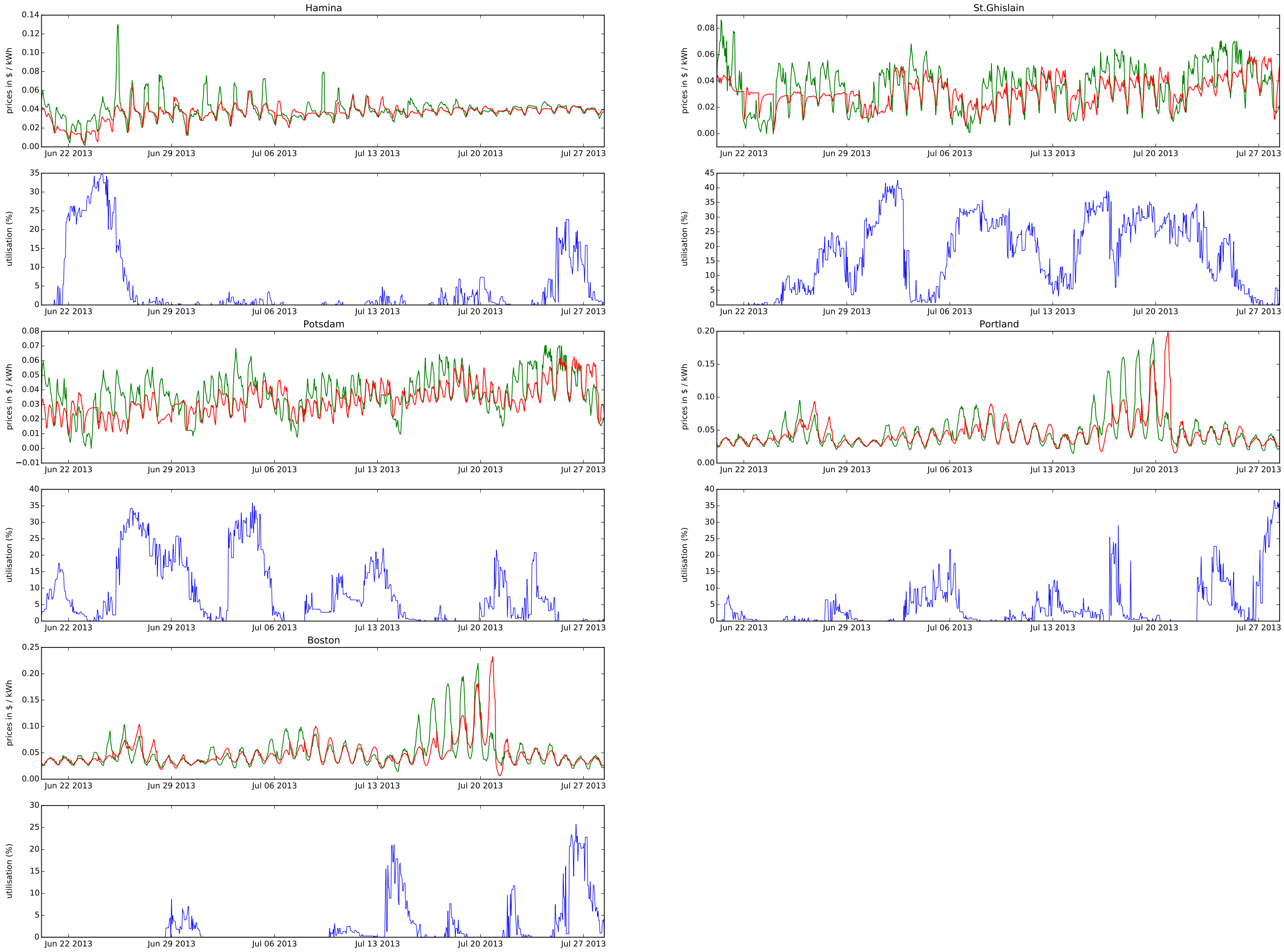


Figure B.6: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_MF

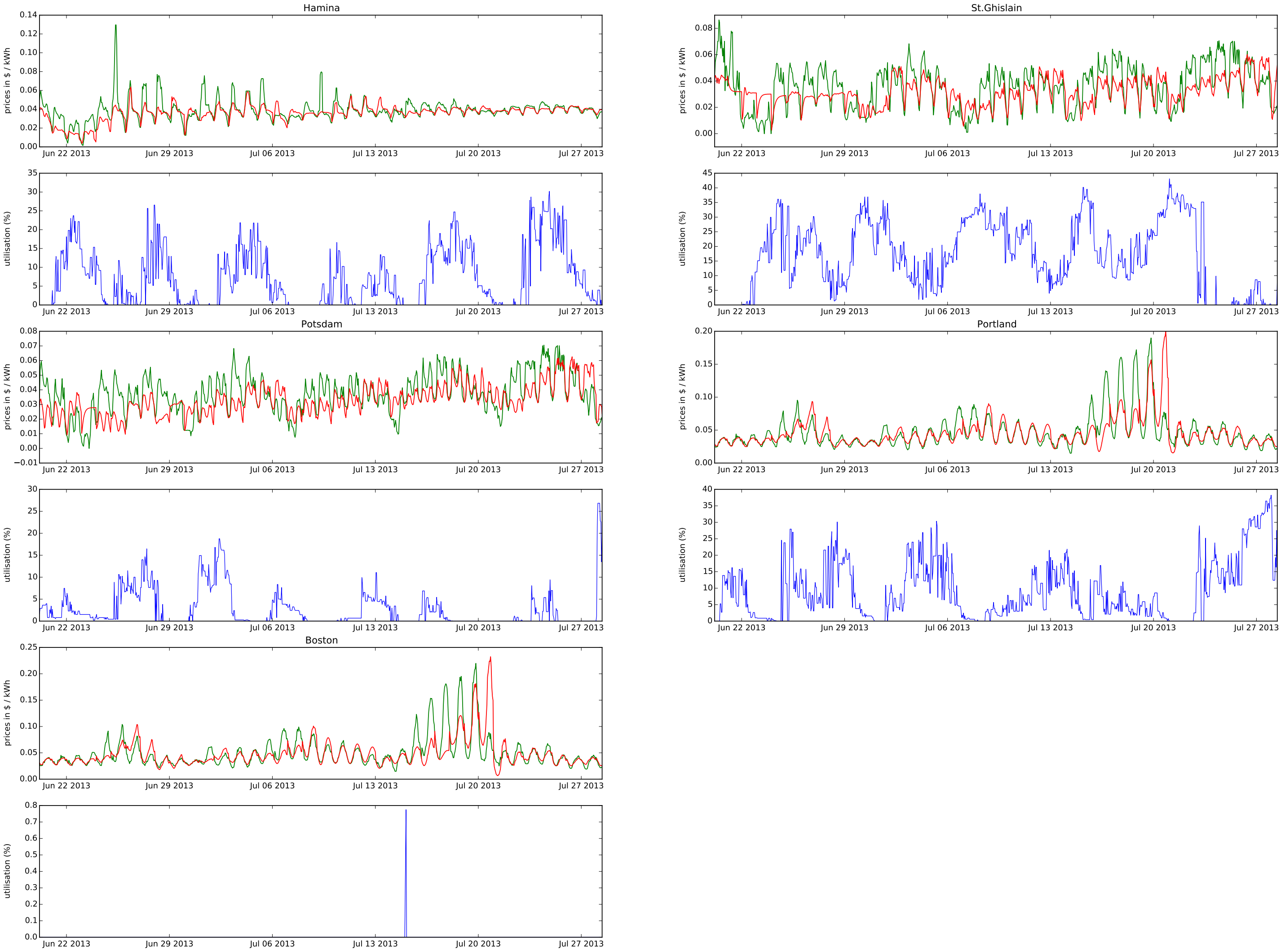


Figure B.7: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Summer” and scheduler BCU_MIF

Simulation results for DA Spring

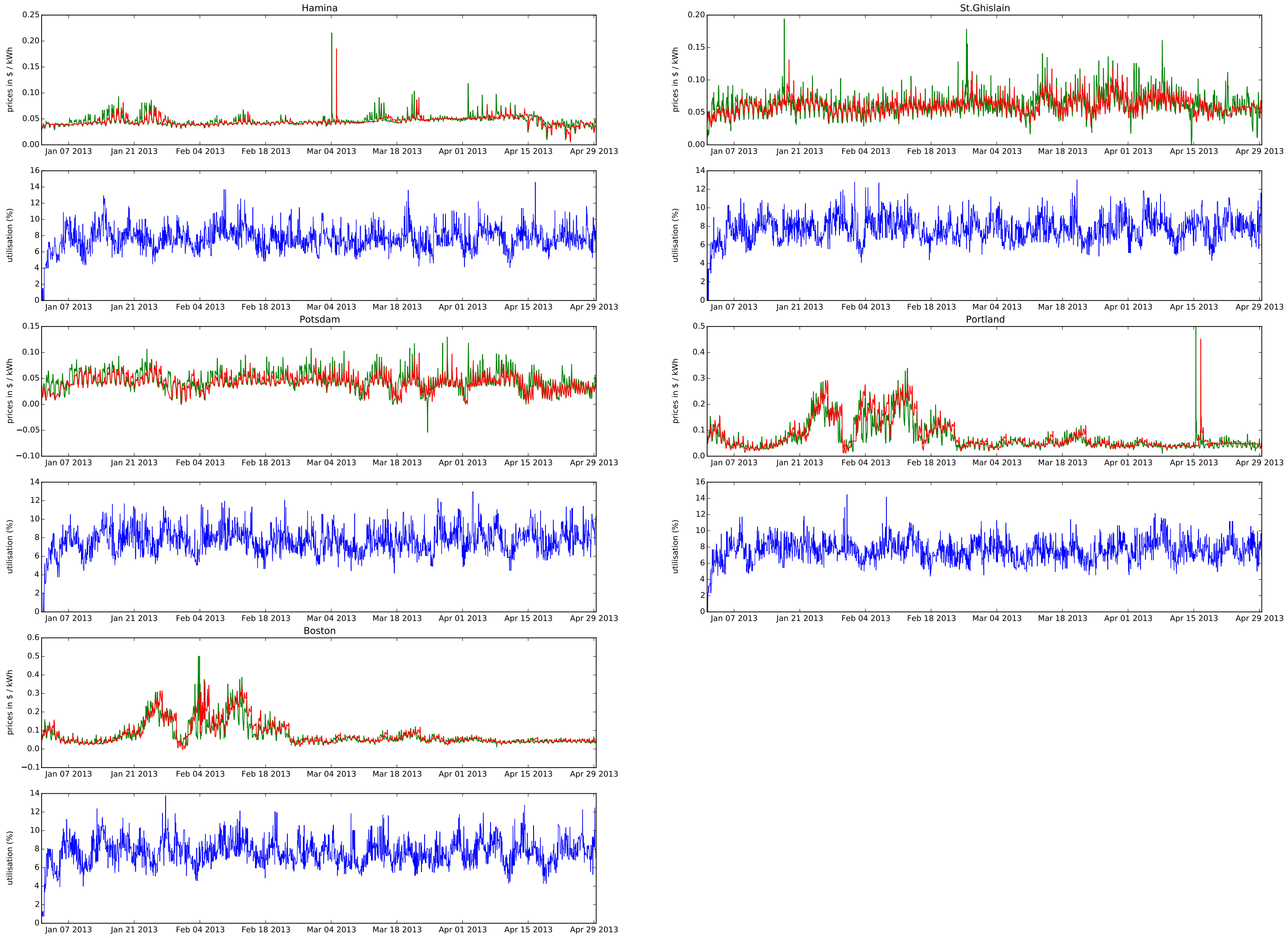


Figure B.8: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BFD

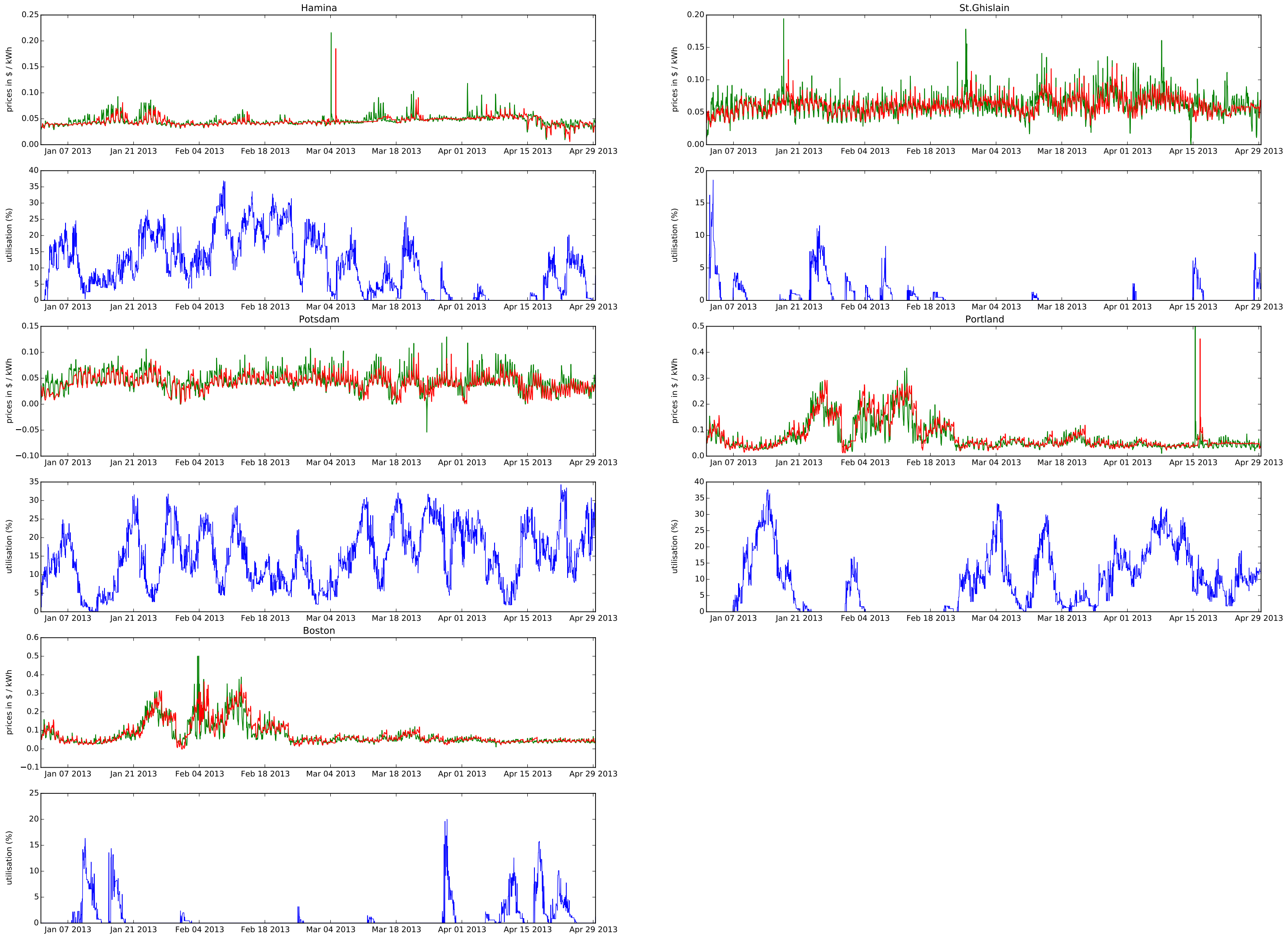


Figure B.9: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF

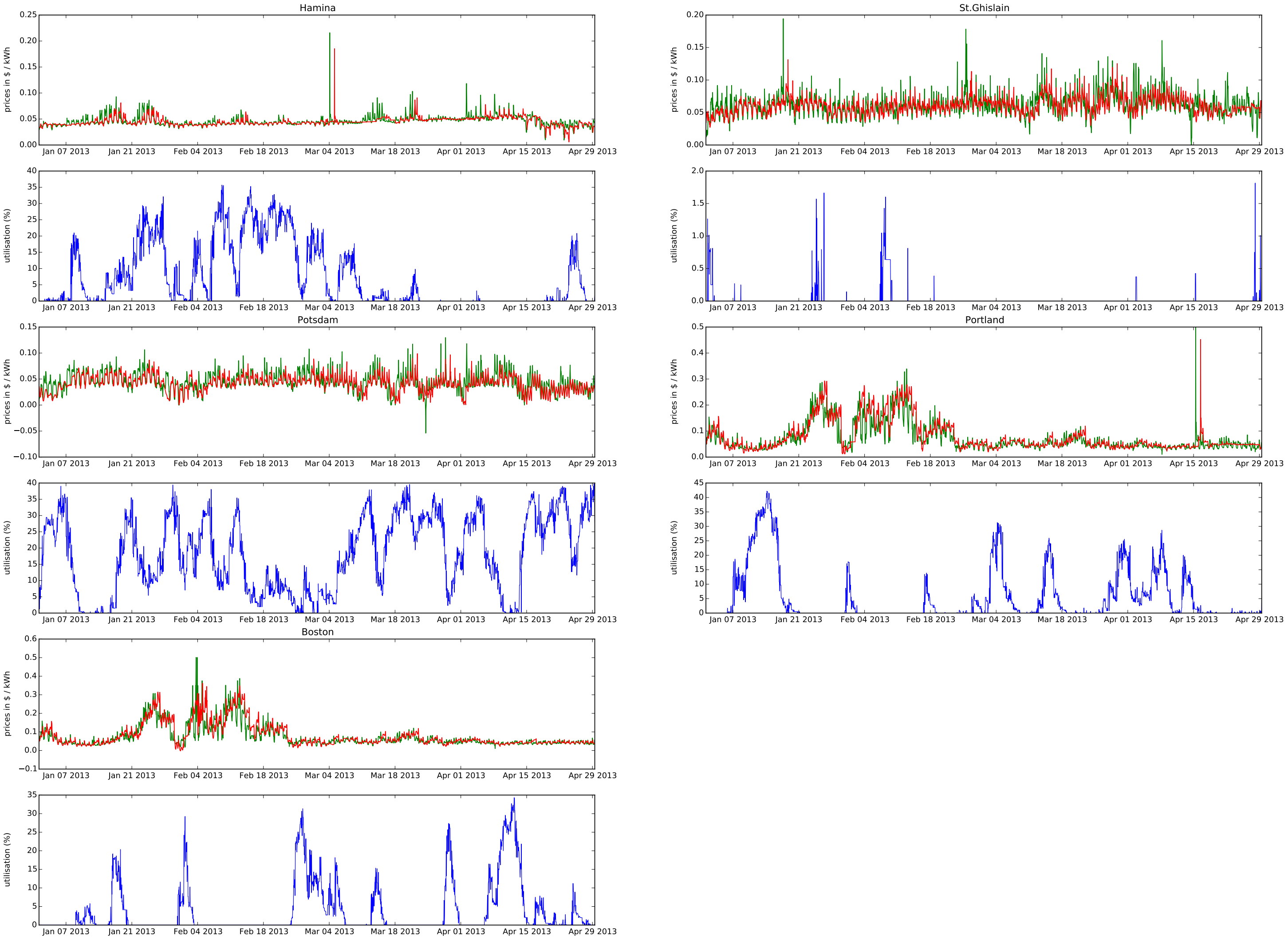


Figure B.10: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF_F

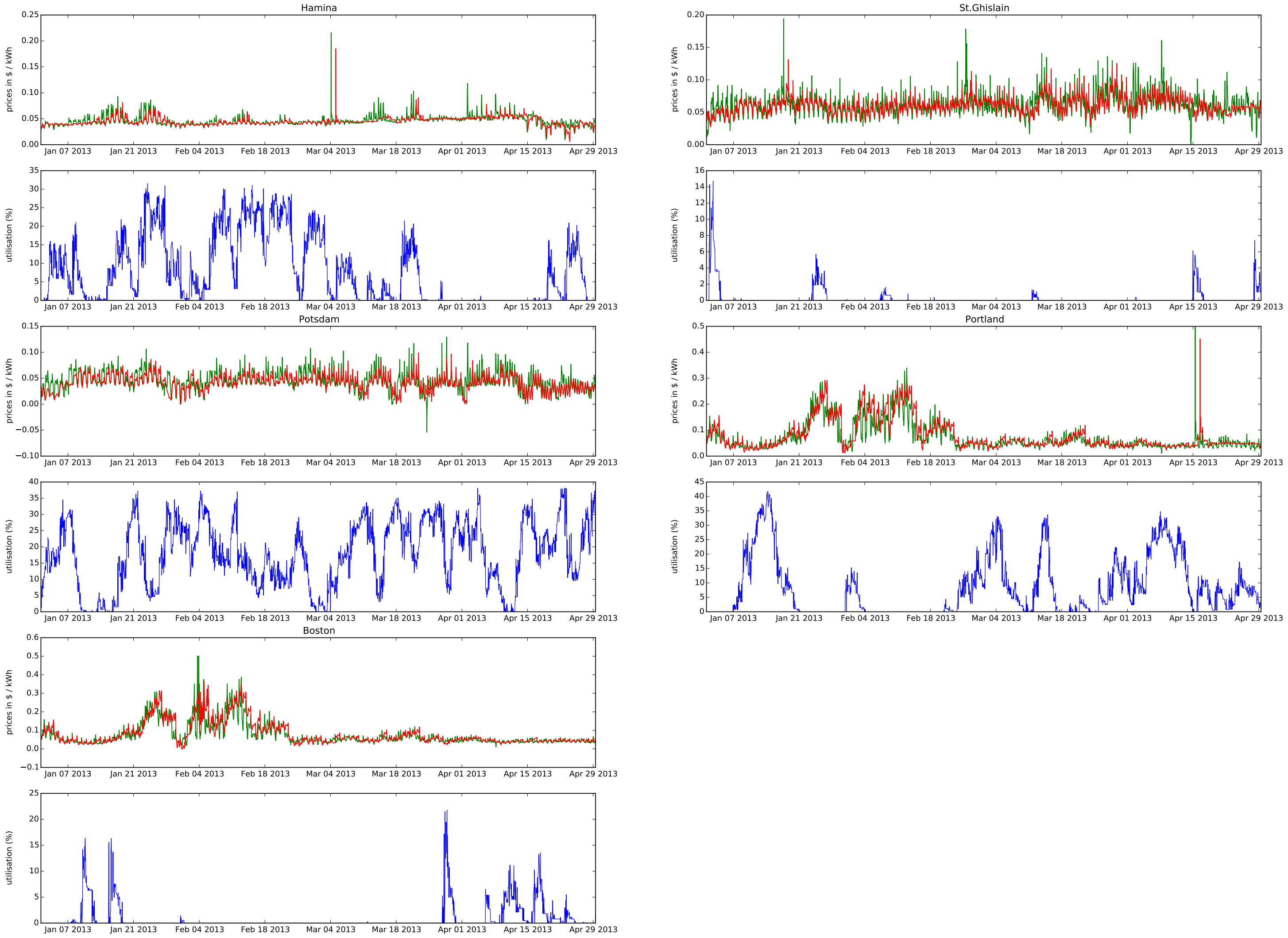


Figure B.11: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCF_IF

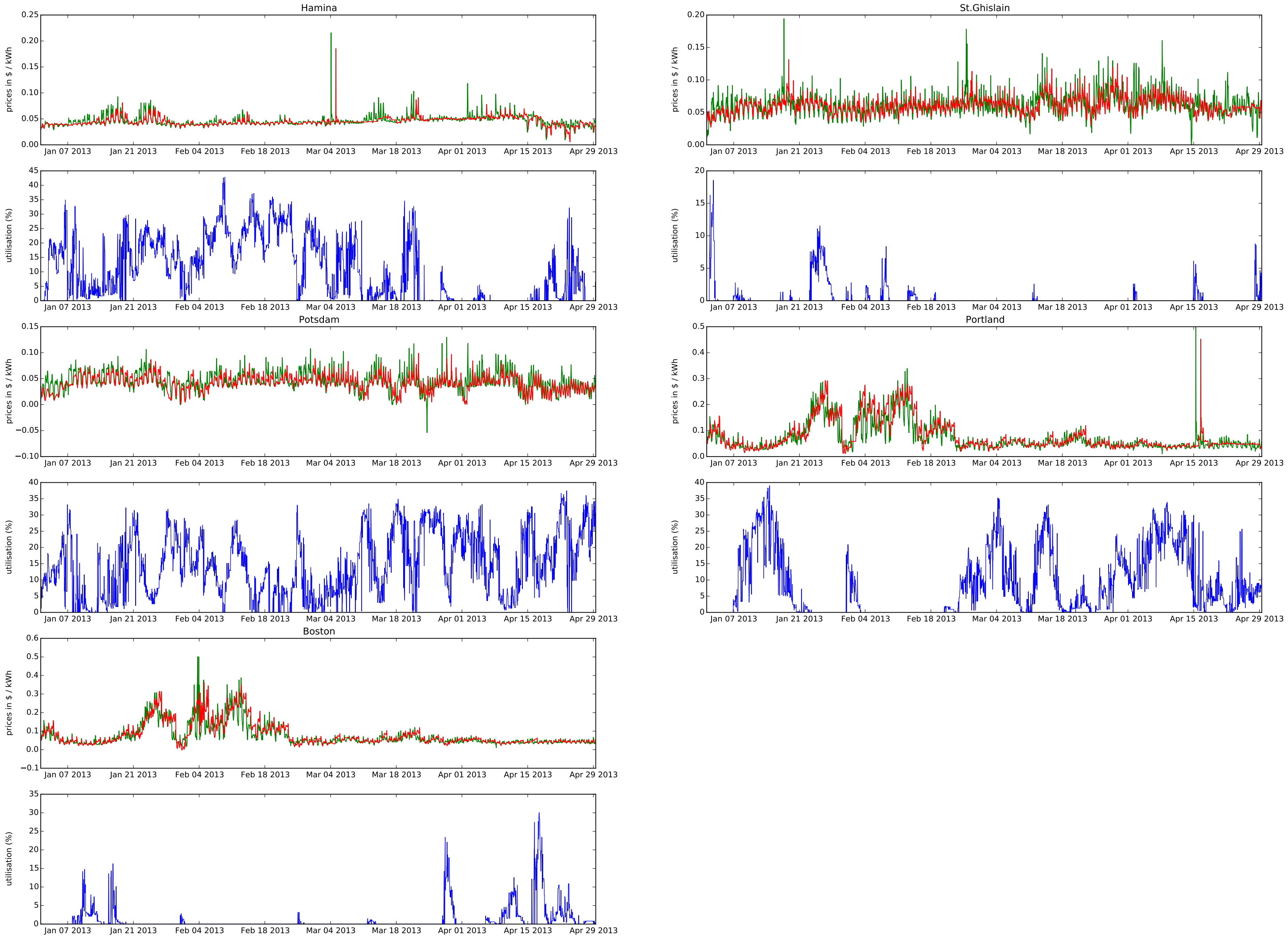


Figure B.12: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_M

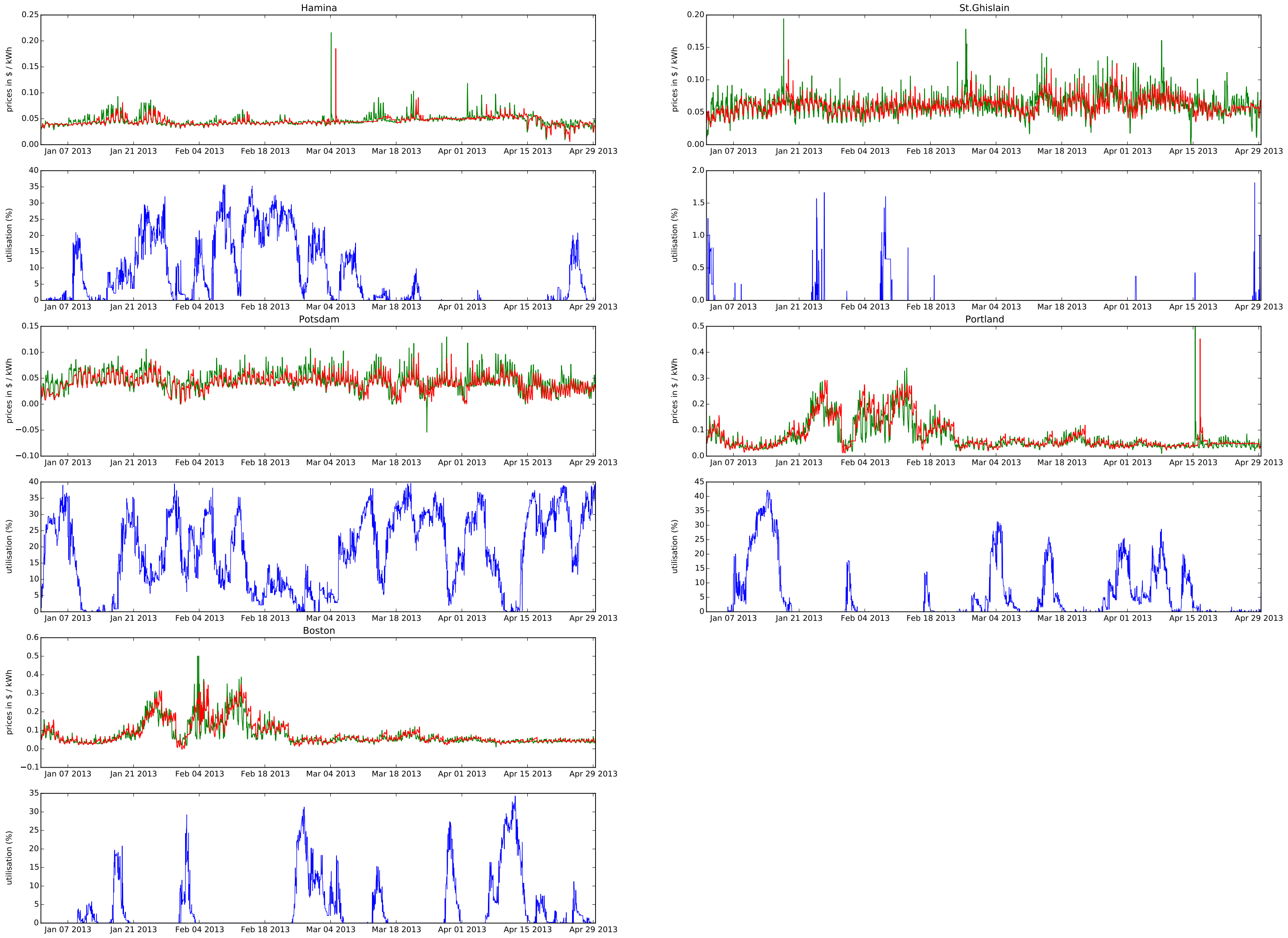


Figure B.13: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_MF

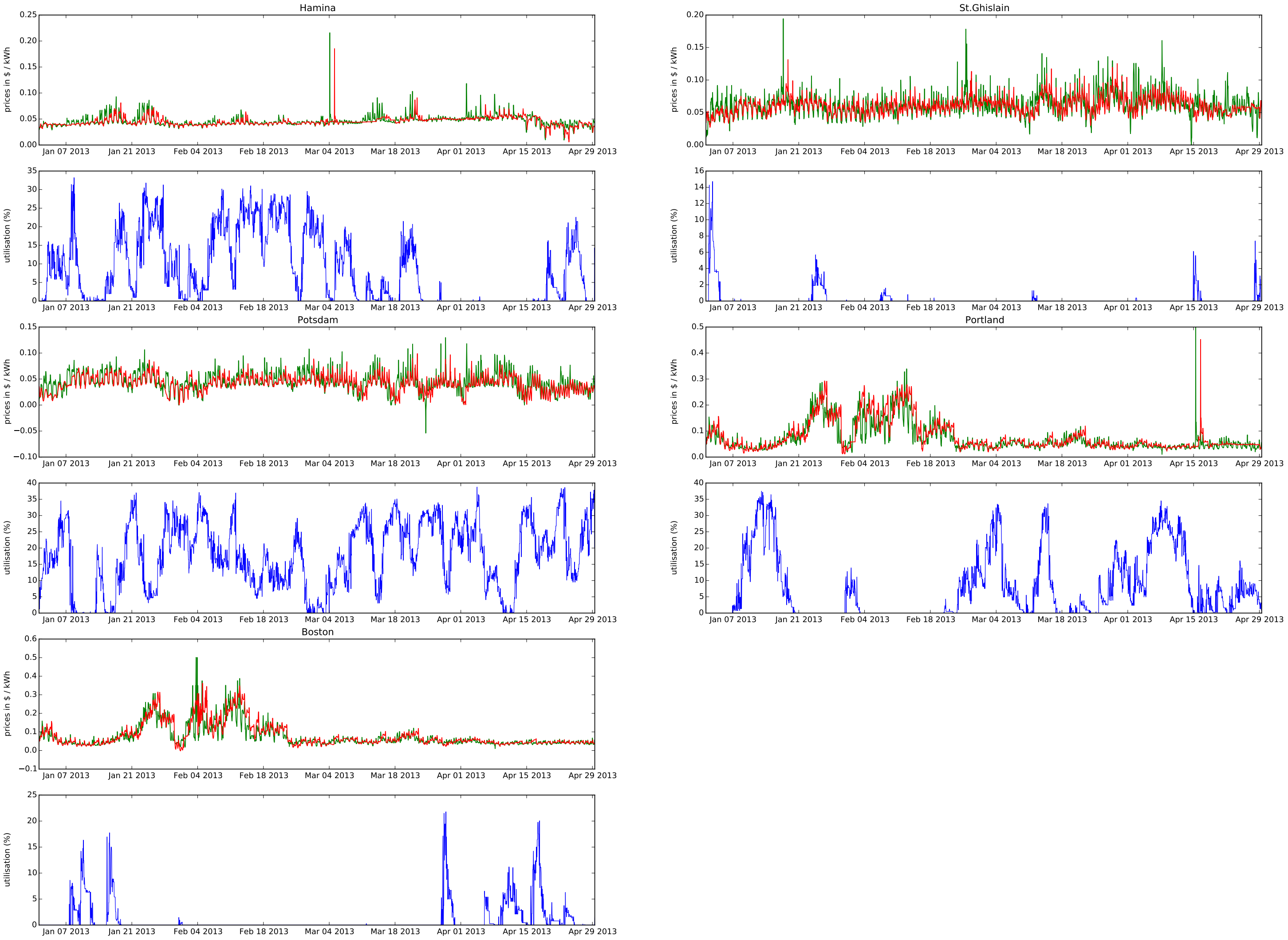


Figure B.14: Energy prices, energy price forecasts and utilization levels per location for simulation “DA Spring” and scheduler BCU_MIF

Simulation results for RT Summer

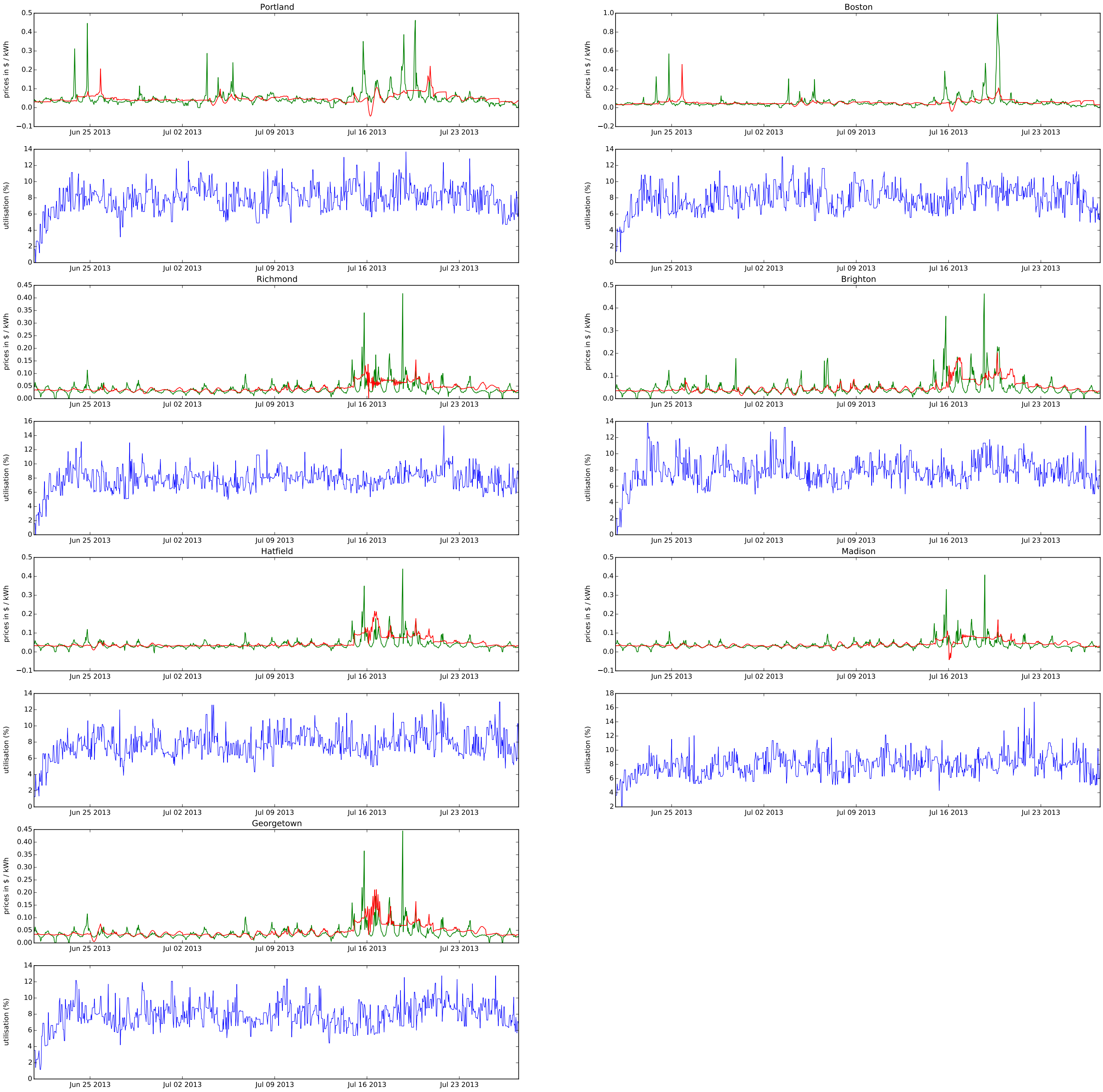


Figure B.15: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BFD

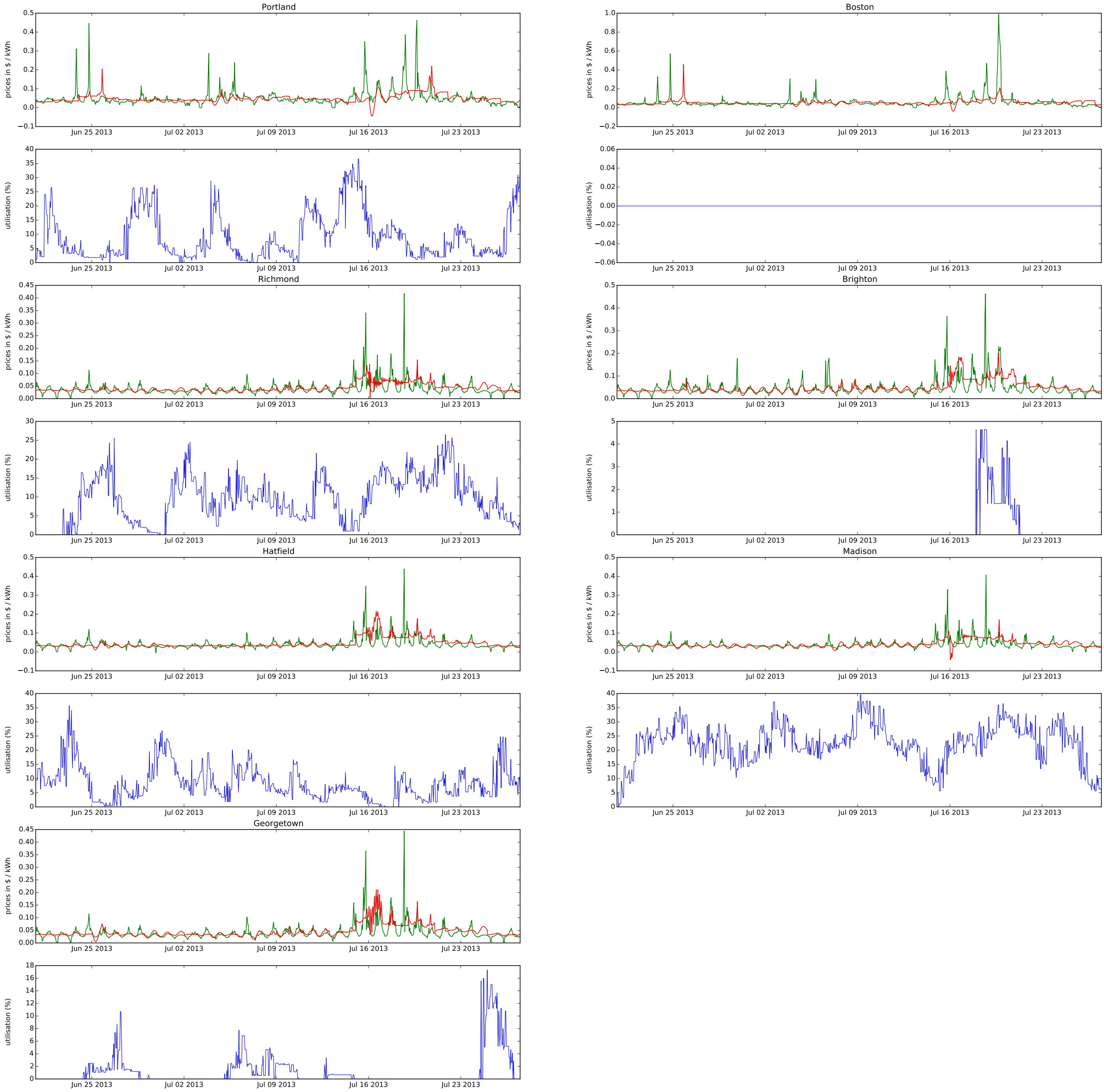


Figure B.16: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF

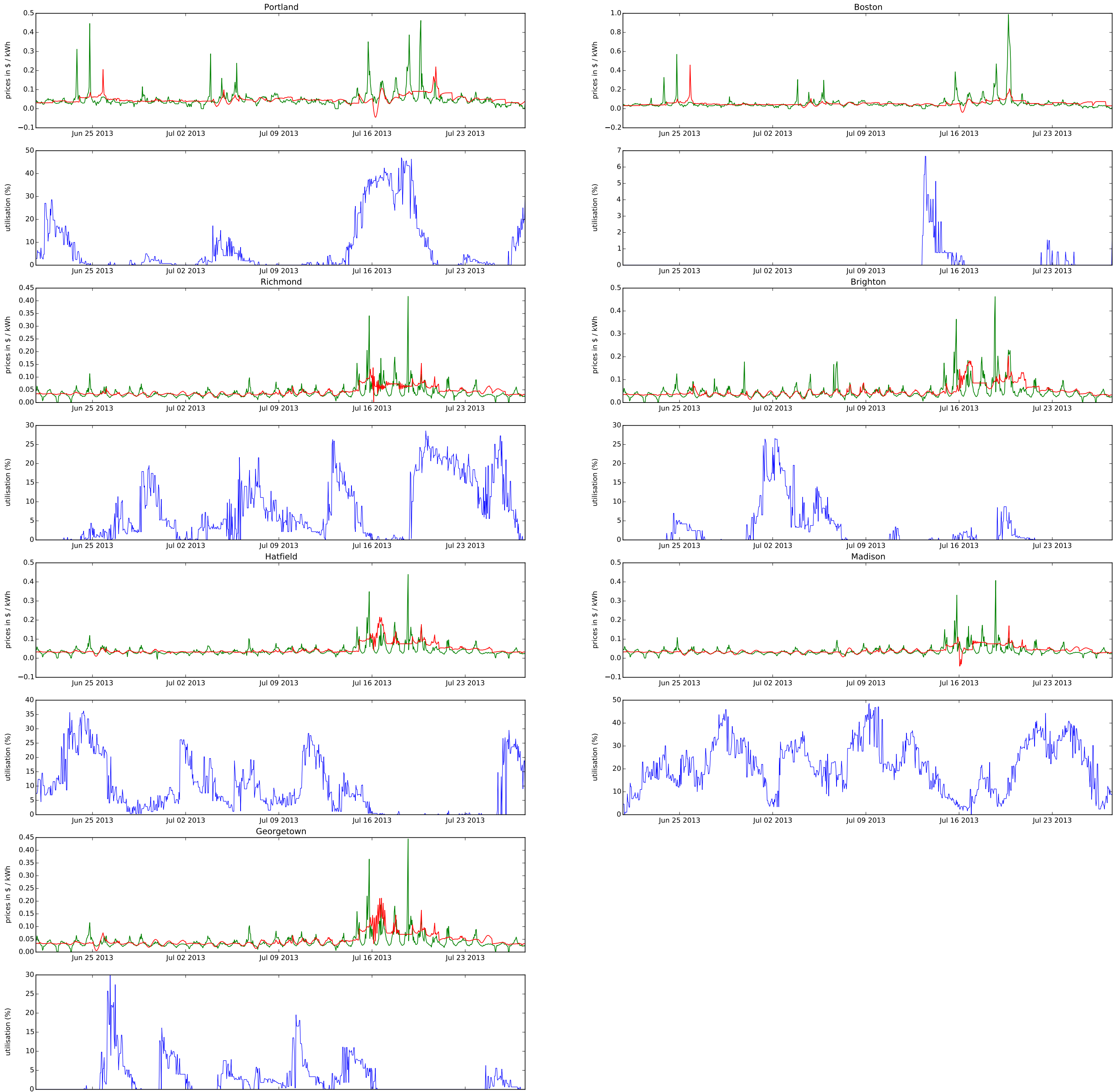


Figure B.17: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF_F

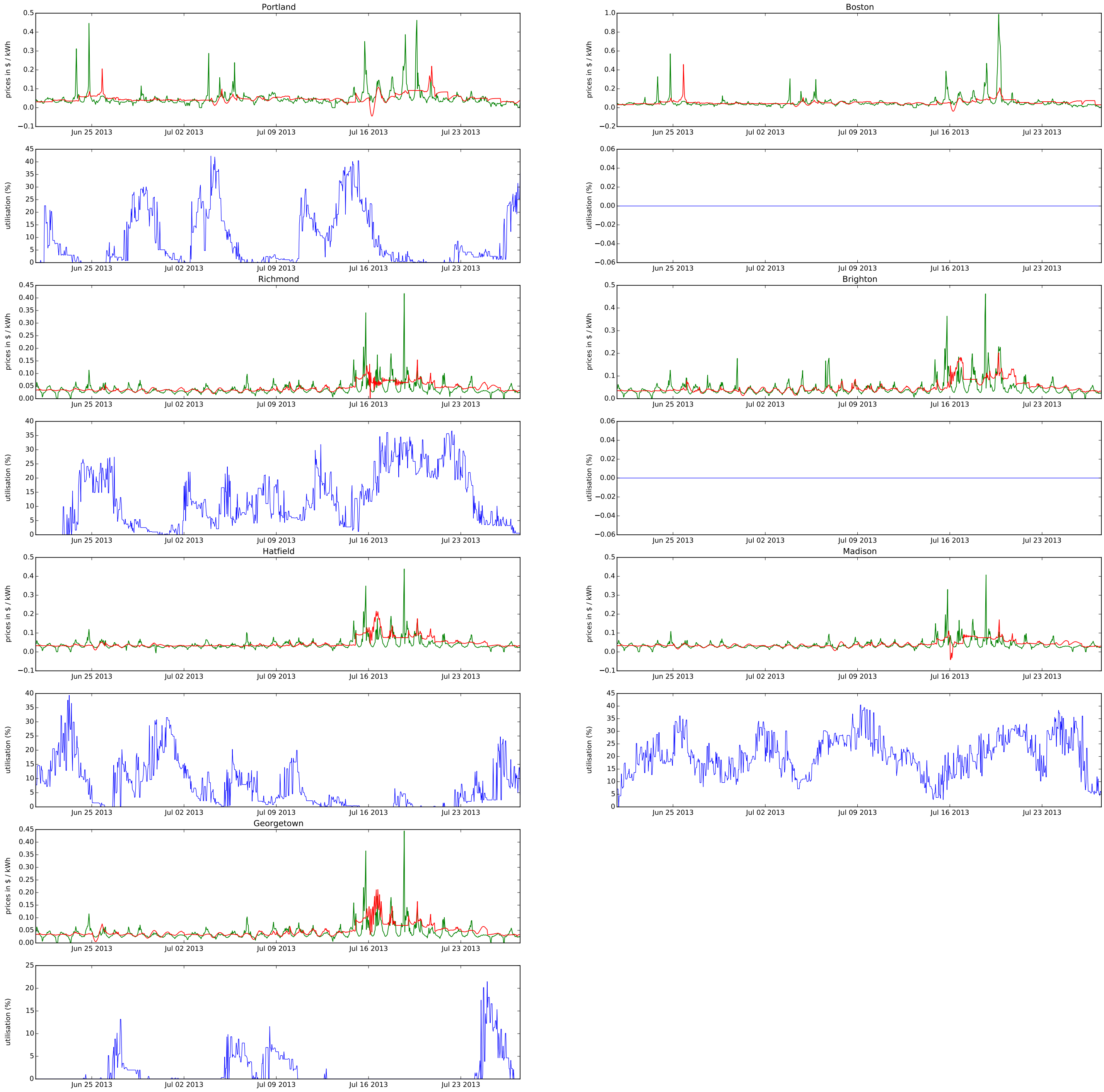


Figure B.18: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCF_IF

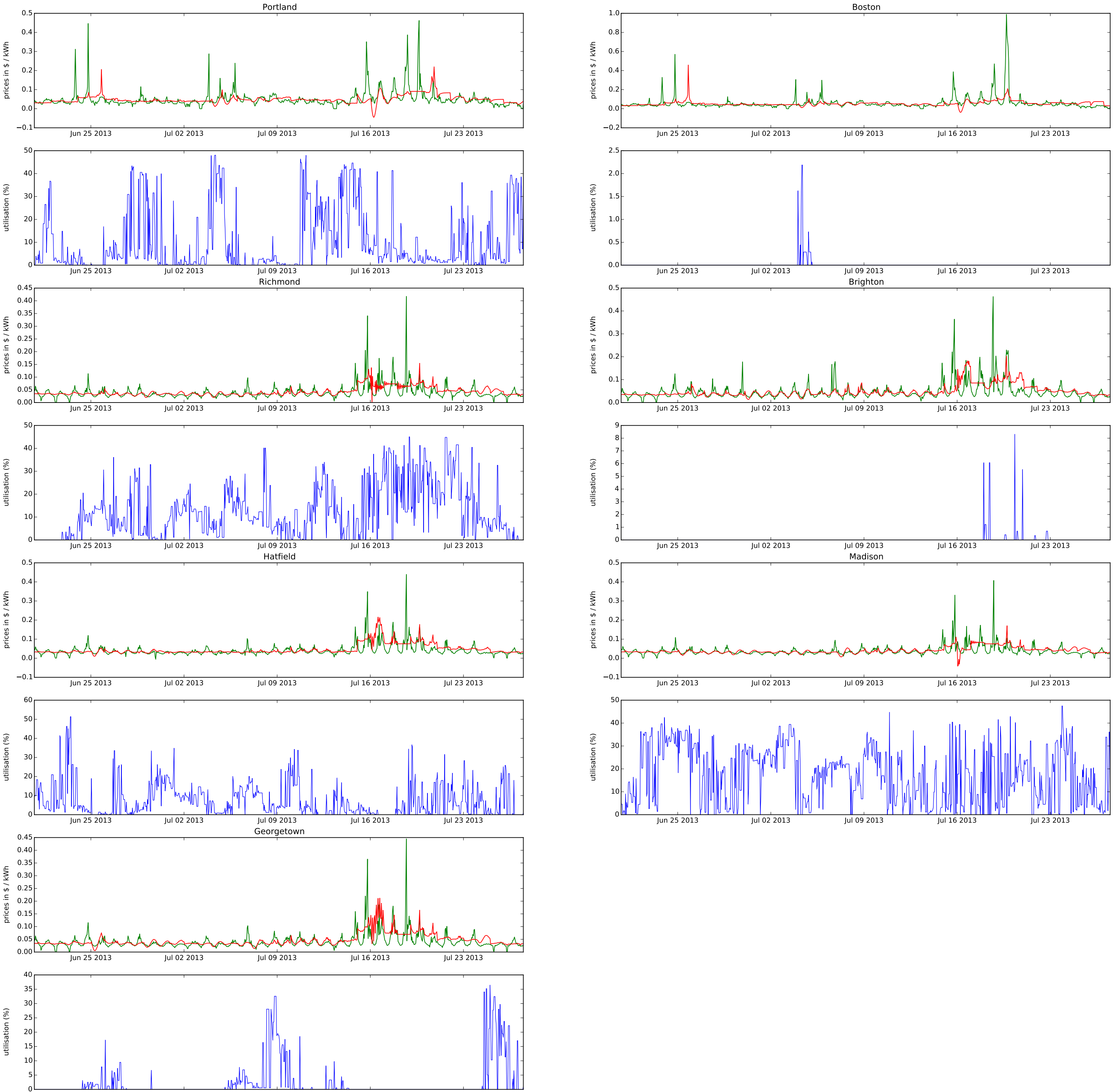


Figure B.19: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_M

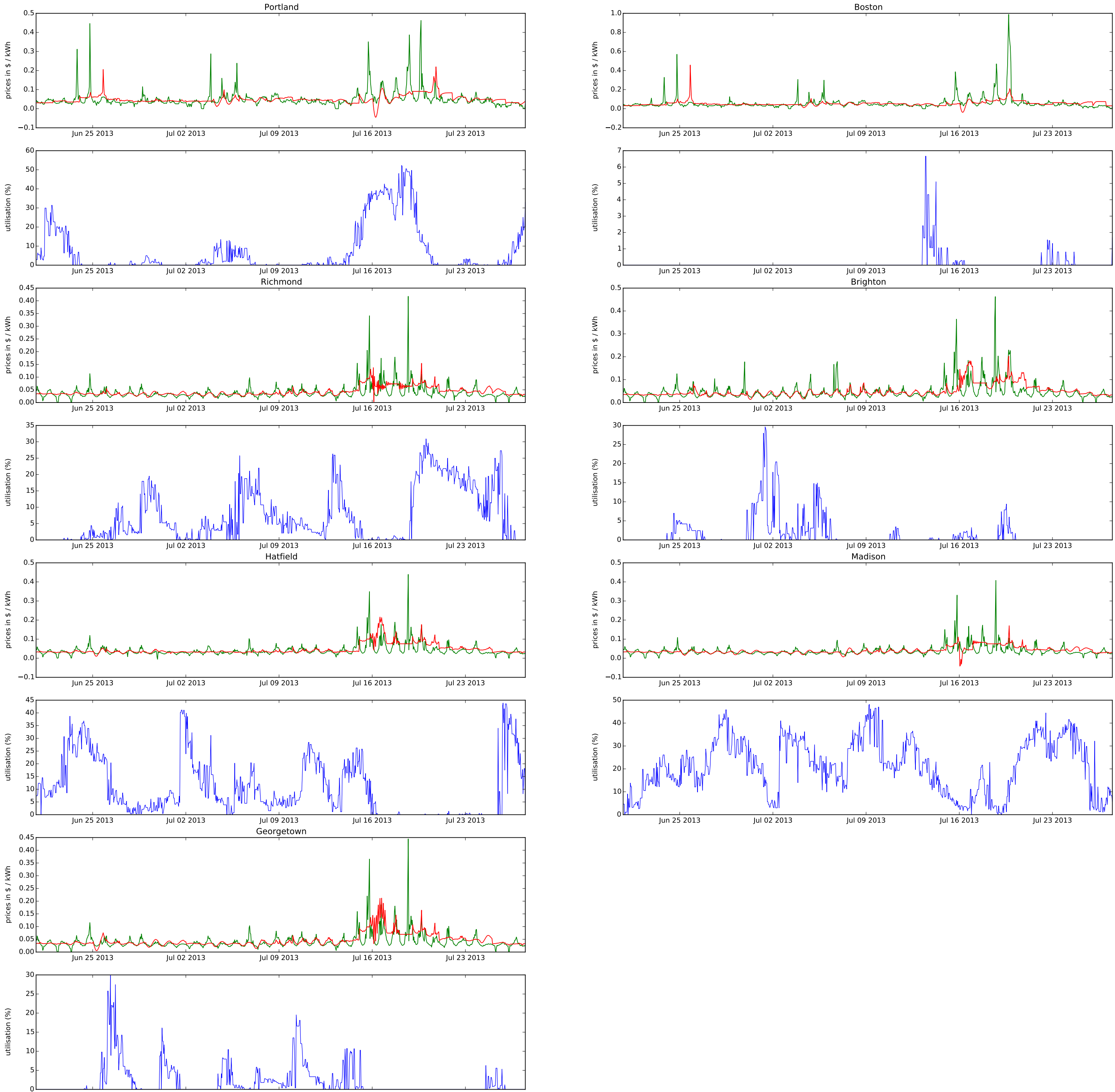


Figure B.20: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_MF

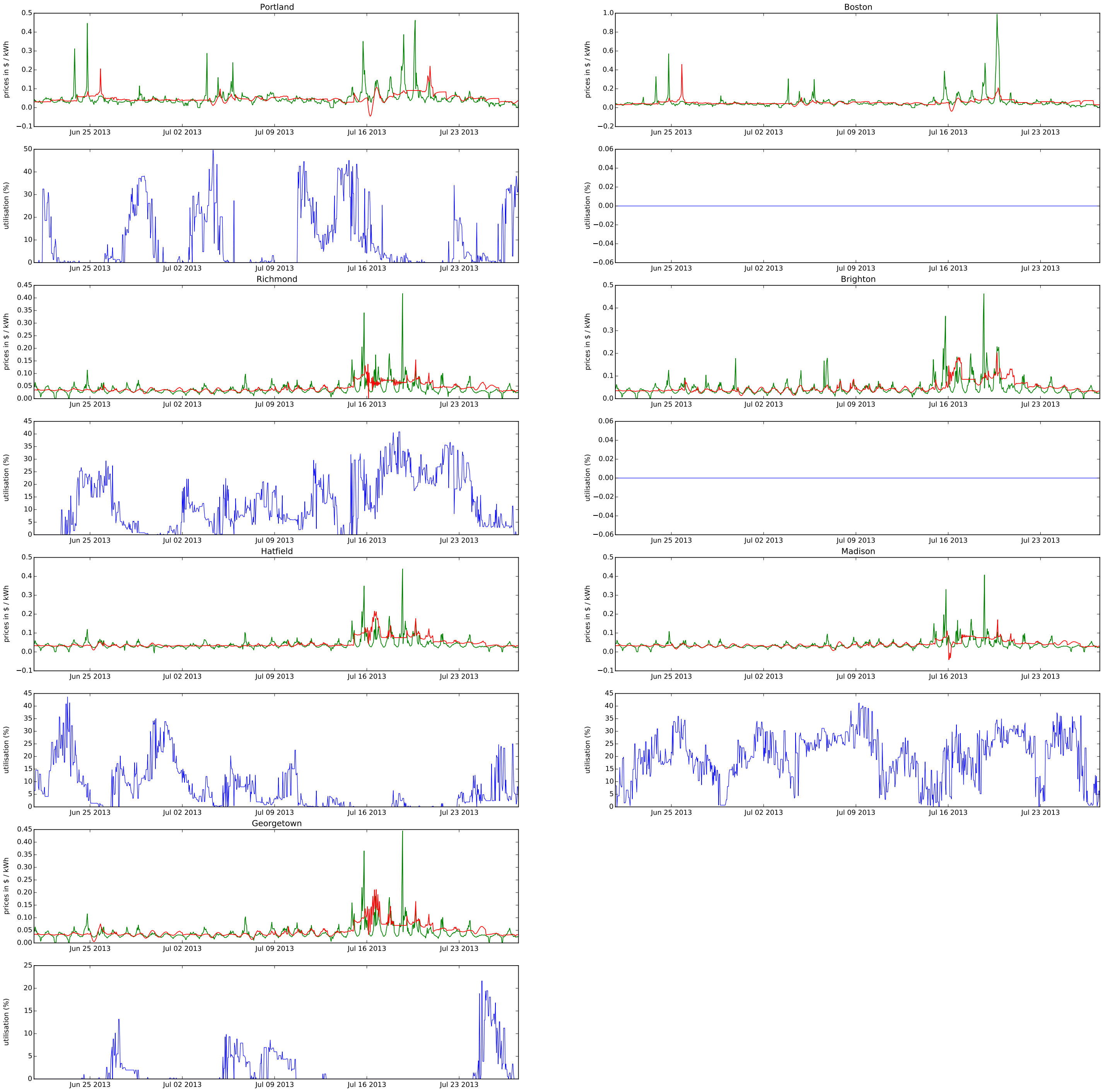


Figure B.21: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Summer” and scheduler BCU_MIF

Simulation results for RT Spring

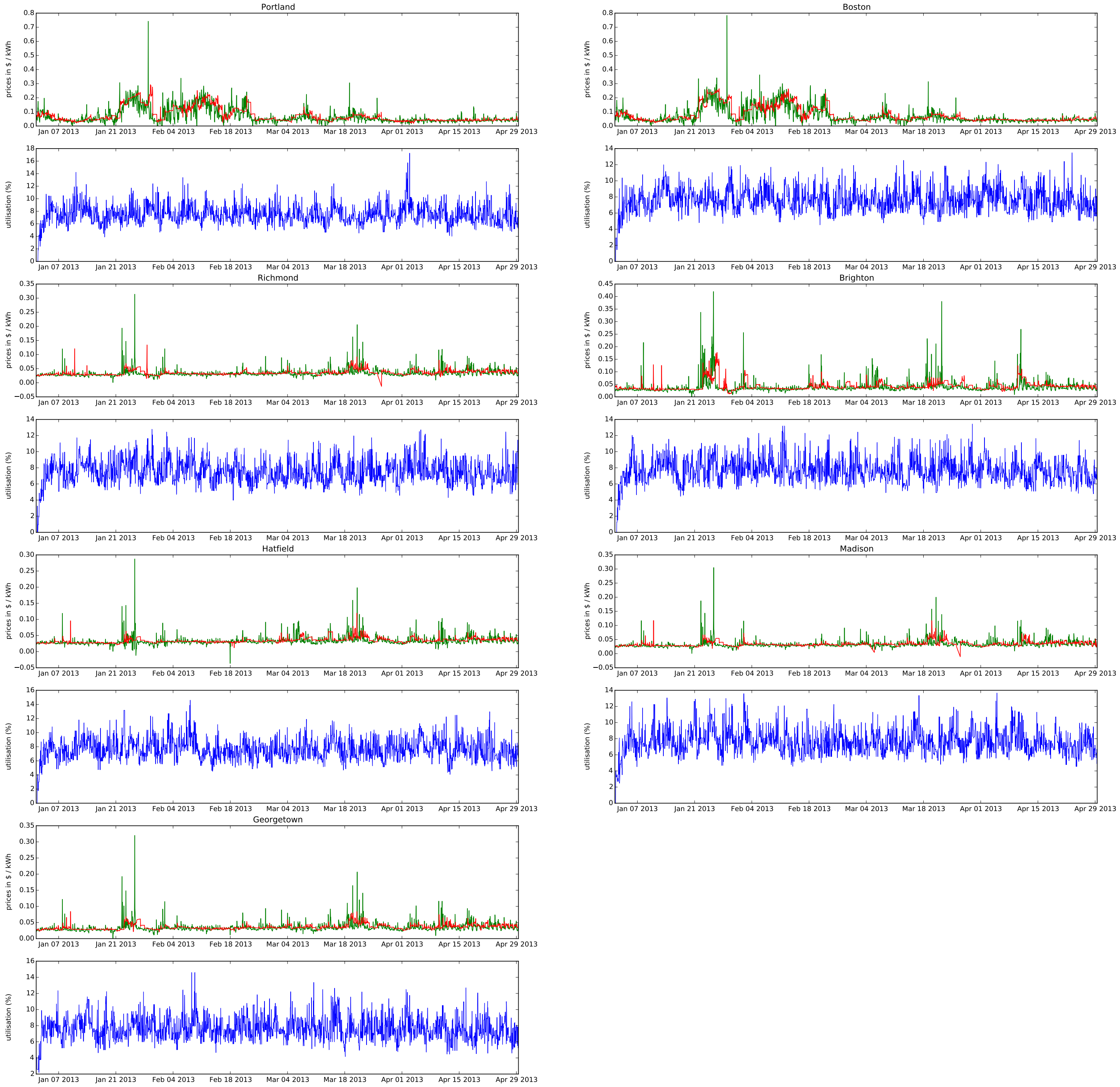


Figure B.22: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BFD

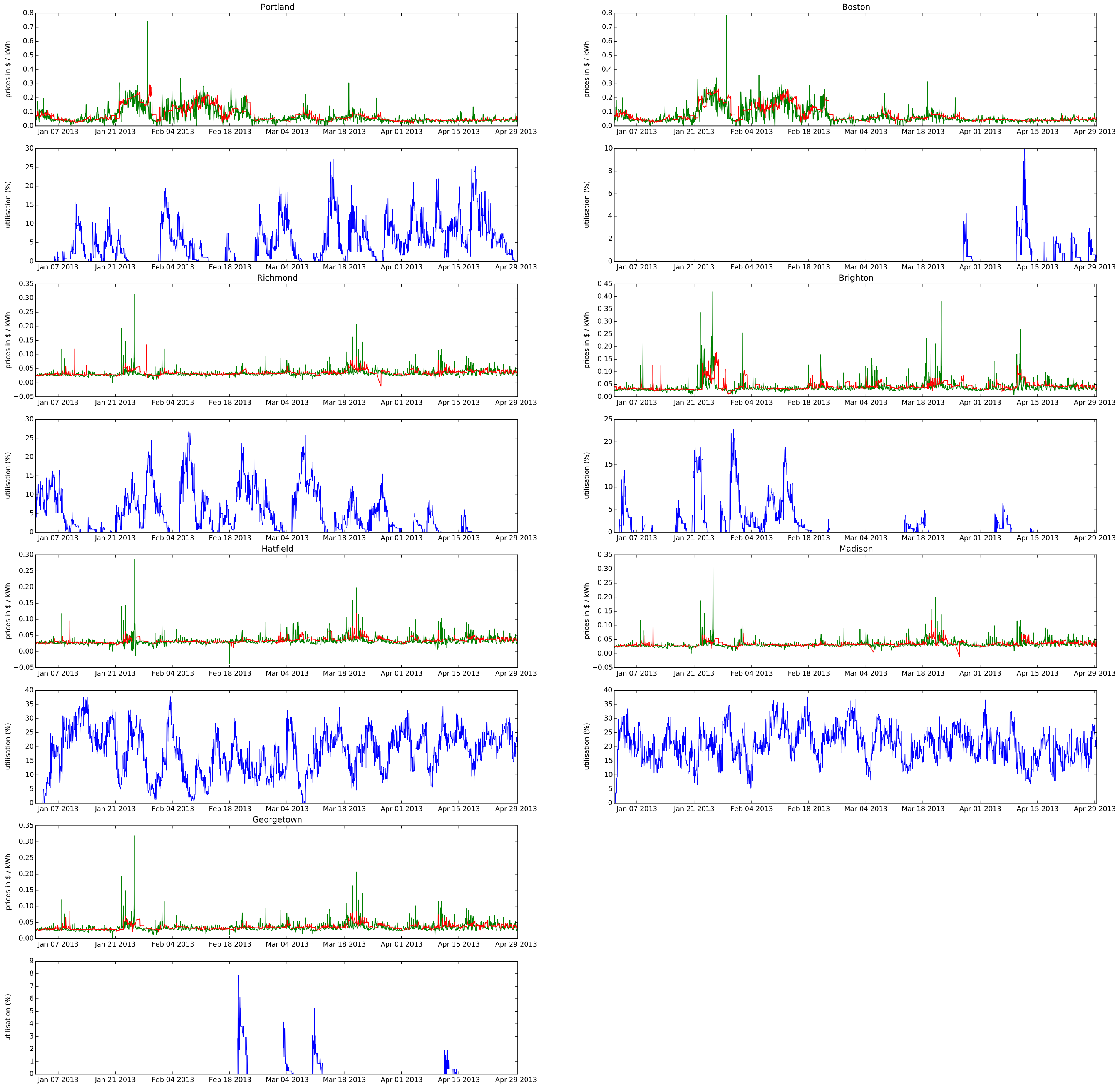


Figure B.23: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF

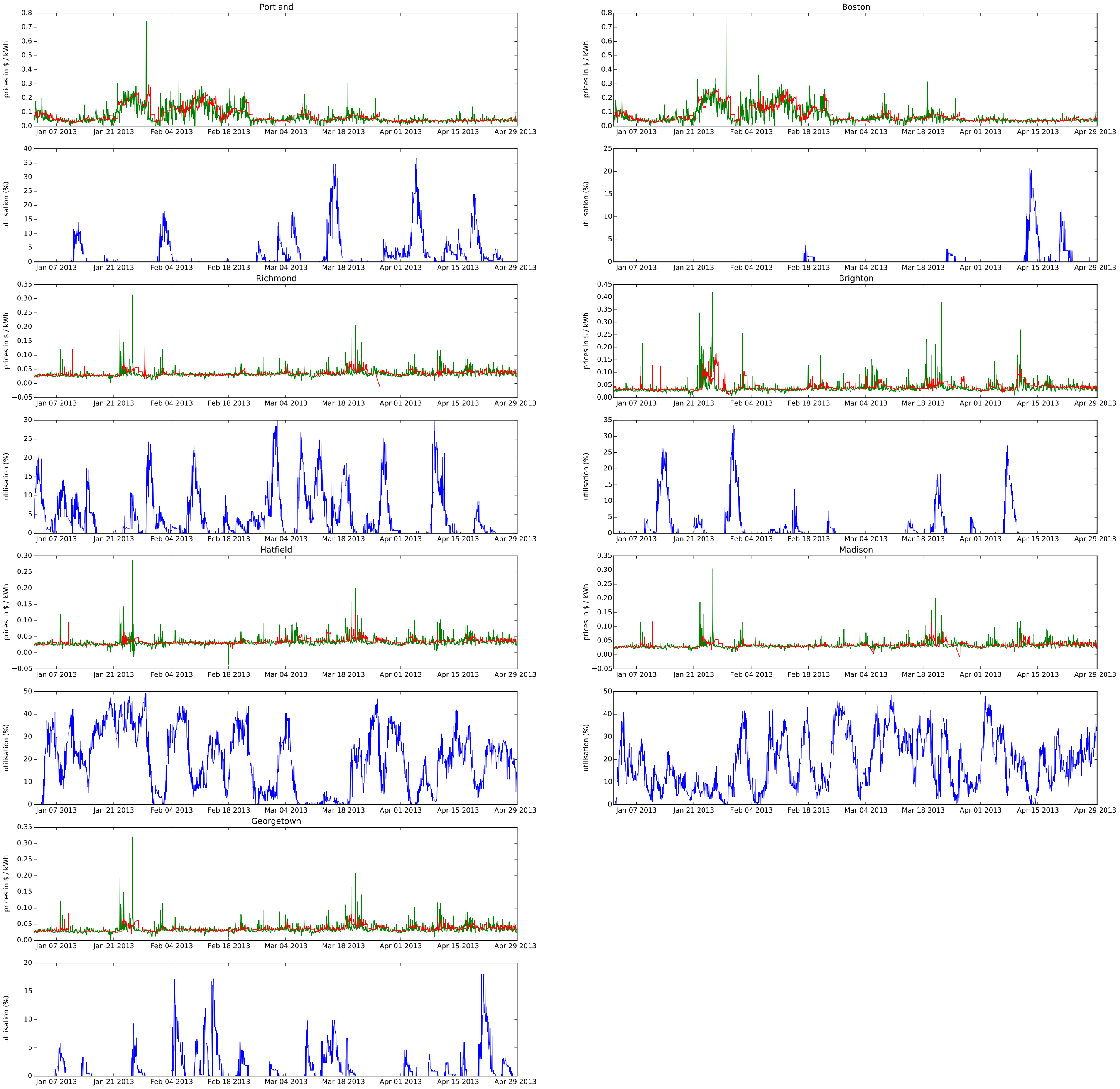


Figure B.24: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF_F

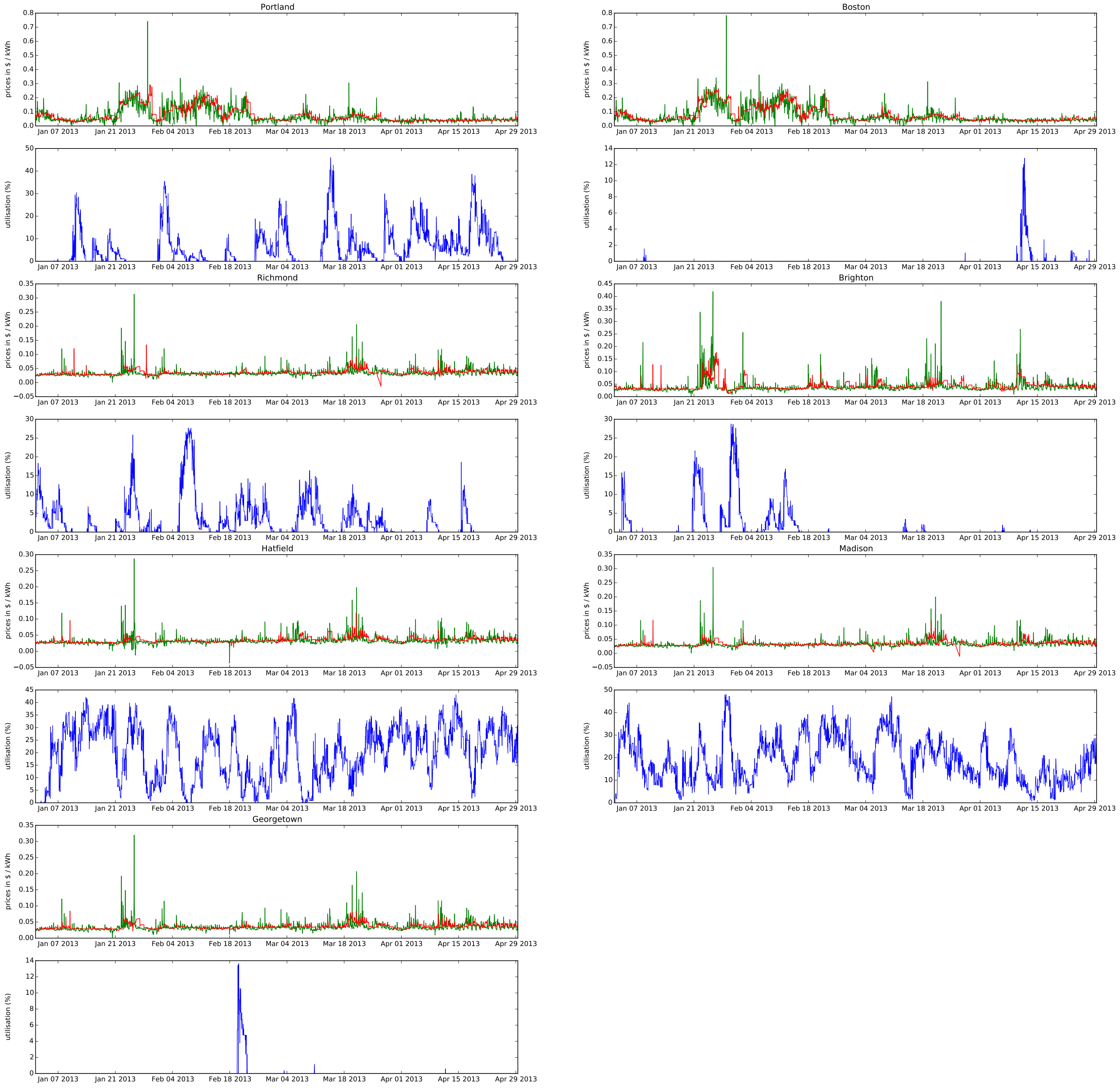


Figure B.25: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCF_IF

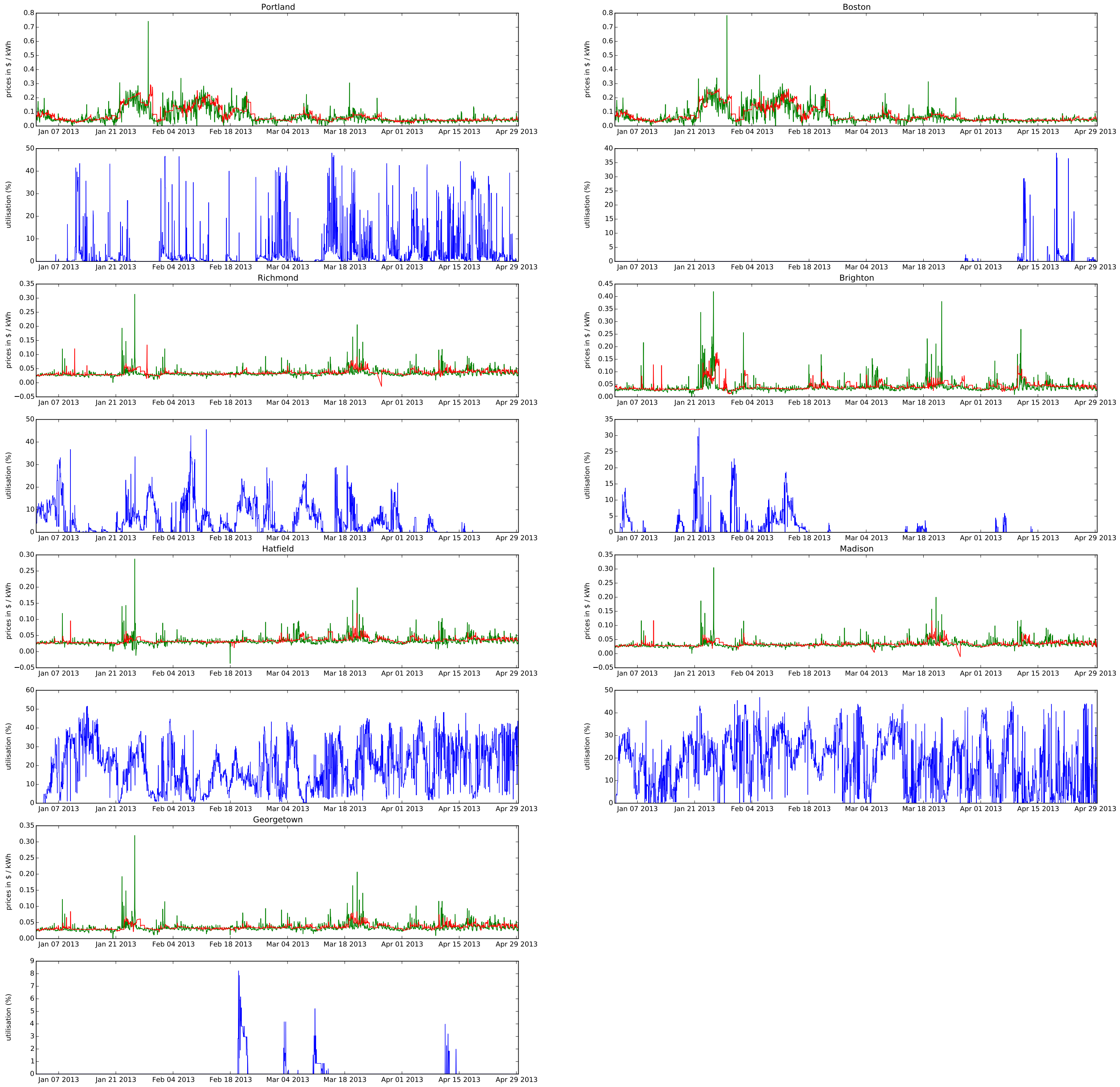


Figure B.26: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCU_M

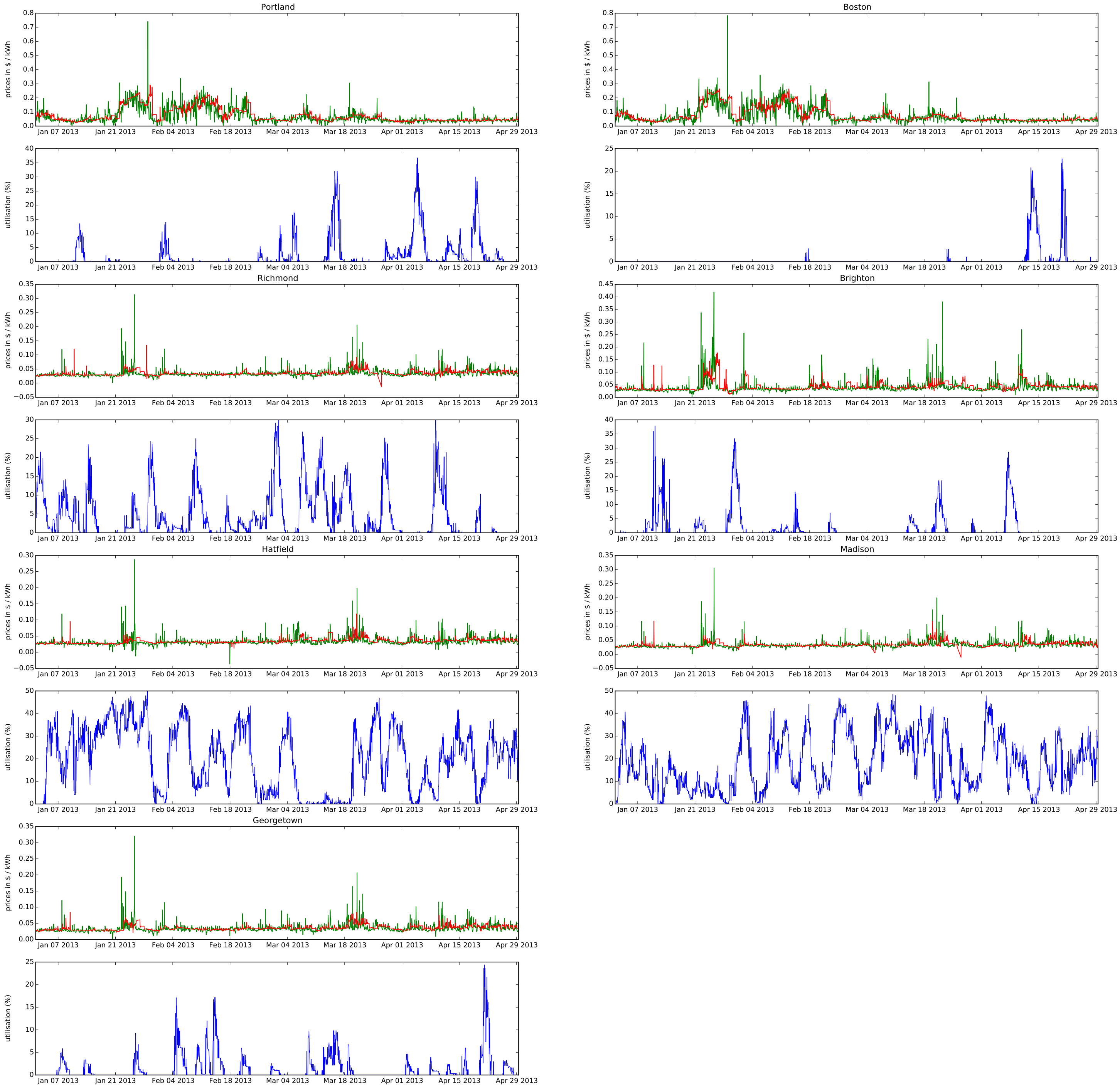


Figure B.27: Energy prices, energy price forecasts and utilization levels per location for simulation “RT Spring” and scheduler BCU_MF

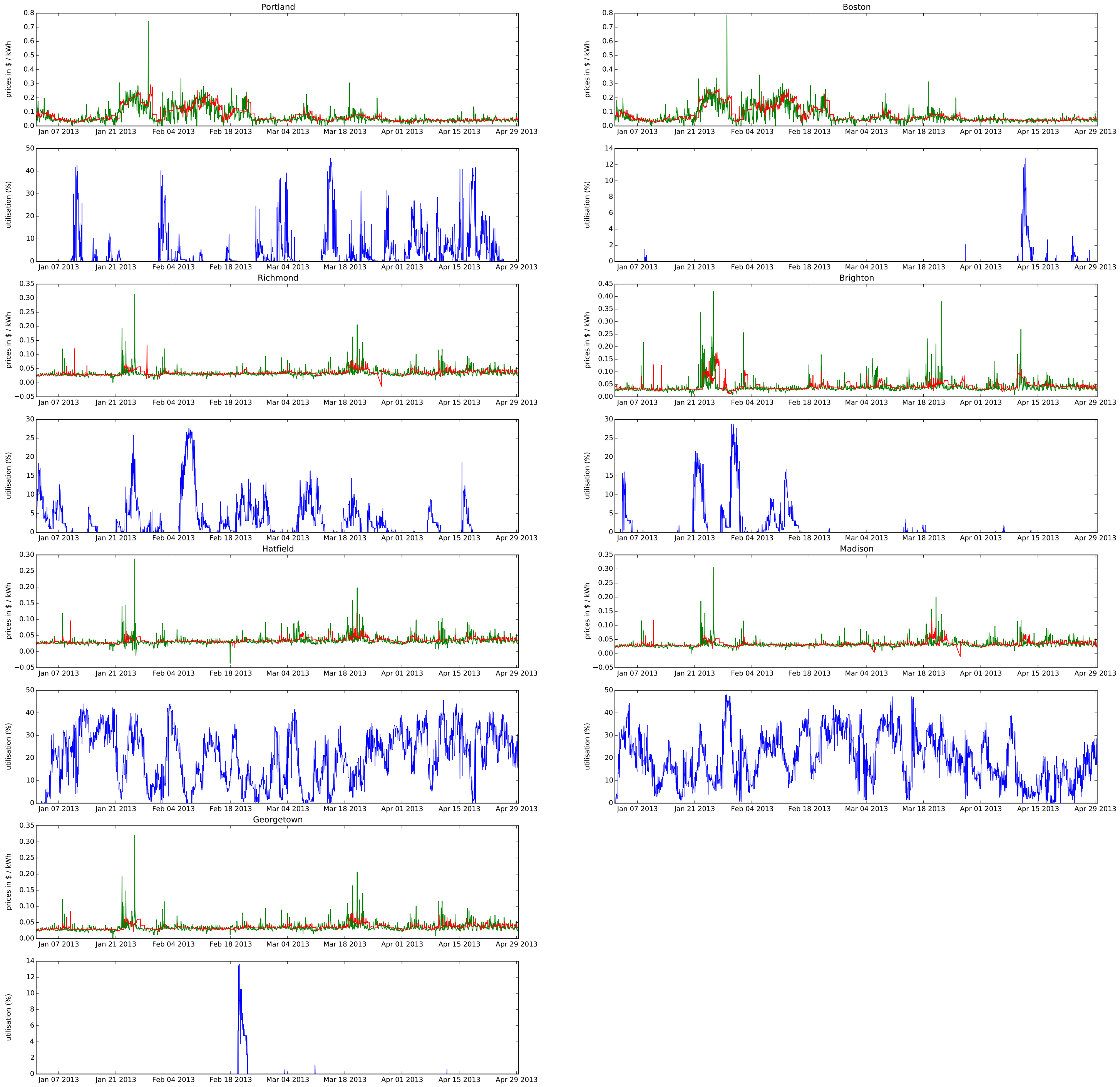


Figure B.28: Energy prices, energy price forecasts and utilization levels per location for simulation "RT Spring" and scheduler BCU_MIF

APPENDIX C

Source Repository

C.1 Projects

To provide insights to the different projects the interested reader might refer to the corresponding sources listed in this section.

Repositories containing the specific implementation for the projects are split into three parts:

1. The energy price management application (EPMA)
2. The statistical calculations and algorithms (R)
3. The cloud simulator and scheduler (Philharmonic)

The energy price management application comprises the R / Java simulation framework described in Section 4.5. It contains logic for managing energy prices from different energy markets (import, consolidation, retrieval) and provides methods for generating and retrieving forecasts for specific time periods.

The statistical calculations and algorithms provided in R form the basis for generating forecasting models and aggregating forecast accuracy measurements. It consists of a collection of functions for building models and processing and plotting energy price data.

Philharmonic denotes the cloud simulator and scheduler[1] written in Python which has been extended to meet the needs of a cloud scheduler comprising different scenarios to achieve maximum cost reductions (described in Section 5.1).

Table C.1 provides a list of URLs to link to the aforementioned projects.

All projects may be accessed via the homepage of the work (last entry in the table). For detailed information consult the corresponding README files of the projects.

---

[1] philharmonic.github.io

Project	Web location
EPMA	https://github.com/epma16/epma
R Stats	https://github.com/epma16/rstats
Philharmonic	https://github.com/epma16/philharmonic
Thesis homepage	http://epma16.github.io/

Table C.1: List of URLs linking to the projects

Bibliography

[1] Sidney N Afriat. The construction of utility functions from expenditure data. *International economic review*, 8(1):67–77, 1967.

[2] Sanjeev Kumar Aggarwal, Lalit Mohan Saini, and Ashwani Kumar. Electricity price forecasting in deregulated markets: A review and evaluation. *International Journal of Electrical Power & Energy Systems*, 31(1):13–22, 2009.

[3] Sherif Akoush, Ripduman Sohan, Andrew Rice, Andrew W Moore, and Andy Hopper. Predicting the performance of virtual machine migration. In *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*, pages 37–46. IEEE, 2010.

[4] Nima Amjady. Day-ahead price forecasting of electricity markets by a new fuzzy neural network. *Power Systems, IEEE Transactions on*, 21(2):887–896, 2006.

[5] Silvia Angilella, Salvatore Greco, Fabio Lamantia, and Benedetto Matarazzo. Assessing non-additive utility for multicriteria decision aid. *European Journal of Operational Research*, 158(3):734–744, 2004.

[6] Patrick Armstrong, Ashok Agarwal, A Bishop, Andre Charbonneau, R Desmarais, K Fransham, N Hill, Ian Gable, S Gaudet, S Goliath, et al. Cloud scheduler: a resource manager for distributed compute clouds. *arXiv preprint arXiv:1007.0050*, 2010.

[7] NORD POOL SPOT AS. Nord pool spot power market. http://www.nordpoolspot.com/, 2004. Online; accessed 10-April-2016.

[8] Victor Avelar, Dan Azevedo, Alan French, and Emerson Network Power. Pue^{TM}: A comprehensive examination of the metric. *White paper*, 49, 2012.

[9] Luiz Augusto Barroso, Teófilo H Cavalcanti, Paul Giesbertz, and Konrad Purchala. Classification of electricity market models worldwide. In *CIGRE/IEEE PES, 2005. International Symposium*, pages 9–16. IEEE, 2005.

[10] Anton Beloglazov, Jemal Abawajy, and Rajkumar Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future generation computer systems*, 28(5):755–768, 2012.

[11] Belpex. Belpex power market. https://www.belpex.be/, 2008. Online; accessed 10-April-2016.

[12] Andreas Berl, Erol Gelenbe, Marco Di Girolamo, Giovanni Giuliani, Hermann De Meer, Minh Quan Dang, and Kostas Pentikousis. Energy-efficient cloud computing. *The computer journal*, 53(7):1045–1051, 2010.

[13] bitcom. Rechenzentren in deutschland: Wirtschaftliche bedeutung und wettbewerbssituation. https://www.bitkom.org/Bitkom/Publikationen/Rechenzentren-in-Deutschland-Wirtschaftliche-Bedeutung-und-Wettbewerbssituation.html, 2014. Online; accessed 28-February-2016.

[14] George EP Box and David R Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 211–252, 1964.

[15] Niv Buchbinder, Navendu Jain, and Ishai Menache. Online job-migration for reducing the electricity bill in the cloud. In *NETWORKING 2011*, pages 172–185. Springer, 2011.

[16] Derek W Bunn. Forecasting loads and prices in competitive power markets: The technology of power system competition. *Proceedings of the IEEE*, 88(2):163–169, 2000.

[17] Derek W Bunn and Nektaria Karakatsani. Forecasting electricity prices. *London Business School*, 1, 2003.

[18] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. Energy-efficient management of data center resources for cloud computing: A vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010.

[19] Rajkumar Buyya, Chee Shin Yeo, Srikumar Venugopal, James Broberg, and Ivona Brandic. Cloud computing and emerging it platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems*, 25(6):599–616, 2009.

[20] JPSa Catalão, SJPS Mariano, VMF Mendes, and LAFM Ferreira. Short-term electricity prices forecasting in a competitive market: a neural network approach. *Electric Power Systems Research*, 77(10):1297–1304, 2007.

[21] Antonio Celesti, Francesco Tusa, Massimo Villari, and Antonio Puliafito. Improving virtual machine migration in federated cloud environments. In *Evolving Internet (INTERNET), 2010 Second International Conference on*, pages 61–67. IEEE, 2010.

[22] Hung-po Chao and Robert Wilson. Design of wholesale electricity markets. *Book Manuscript. Available at http://faculty*, 1999.

[23] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, and Andrew Warfield. Live migration of virtual machines. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*, pages 273–286. USENIX Association, 2005.

[24] Robert B Cleveland, William S Cleveland, Jean E McRae, and Irma Terpenning. Stl: A seasonal-trend decomposition procedure based on loess. *Journal of Official Statistics*, 6(1):3–73, 1990.

[25] Antonio J Conejo, Miguel A Plazas, Rosa Espinola, and Ana B Molina. Day-ahead electricity price forecasting using the wavelet transform and arima models. *Power Systems, IEEE Transactions on*, 20(2):1035–1042, 2005.

[26] PrimeEnergyIT Project Consortium. Energy efficient it and infrastructure for data centers and server rooms. http://en.energyagency.at/fileadmin/dam_en/pdf/publikationen/brochure/Energy-efficient-IT.pdf, 2011. Online; accessed 28-March-2016.

[27] Javier Contreras, Rosario Espinola, Francisco J Nogales, and Antonio J Conejo. Arima models to predict next-day electricity prices. *Power Systems, IEEE Transactions on*, 18(3):1014–1020, 2003.

[28] Jesús Crespo Cuaresma, Jaroslava Hlouskova, Stephan Kossmeier, and Michael Obersteiner. Forecasting electricity spot-prices using linear univariate time-series models. *Applied Energy*, 77(1):87–106, 2004.

[29] Milan De Cauwer and Barry O'Sullivan. A study of electricity price features on distributed internet data centers. In *Economics of Grids, Clouds, Systems, and Services*, pages 60–73. Springer, 2013.

[30] Alysha M De Livera, Rob J Hyndman, and Ralph D Snyder. Forecasting time series with complex seasonal patterns using exponential smoothing. *Journal of the American Statistical Association*, 106(496):1513–1527, 2011.

[31] Jinrui Duan. Electricity price forecasting by autoregressive model and artificial neutral network.

[32] Riccardo Dulmin and Valeria Mininno. Supplier selection using a multi-criteria decision aid method. *Journal of Purchasing and Supply Management*, 9(4):177–187, 2003.

[33] D Dutta and RC Joshi. A genetic: algorithm approach to cost-based multi-qos job scheduling in cloud computing environment. In *Proceedings of the International Conference & Workshop on Emerging Trends in Technology*, pages 422–427. ACM, 2011.

[34] Yiqiu Fang, Fei Wang, and Junwei Ge. A task scheduling algorithm based on load balancing in cloud computing. In *Web Information Systems and Mining*, pages 271–277. Springer, 2010.

[35] J Fontán, T Vázquez, L Gonzalez, Ruben S Montero, and IM Llorente. Opennebula: The open source virtual machine manager for cluster computing. In *Open Source Grid and Cluster Software Conference, San Francisco, CA, USA*, 2008.

[36] The R Foundation. The r project for statistical computing. https://www.r-project.org/, 1999. Online; accessed 25-March-2016.

[37] Reinaldo C Garcia, Javier Contreras, Marco Van Akkeren, and João Batista C Garcia. A garch forecasting model to predict day-ahead electricity prices. *Power Systems, IEEE Transactions on*, 20(2):867–874, 2005.

[38] Raquel Gareta, Luis M Romeo, and Antonia Gil. Forecasting of electricity prices with neural networks. *Energy Conversion and Management*, 47(13):1770–1778, 2006.

[39] Yujia Ge and Guiyi Wei. Ga-based task scheduler for the cloud computing systems. In *Web Information Systems and Mining (WISM), 2010 International Conference on*, volume 2, pages 181–186. IEEE, 2010.

[40] Alicia Mateo González, Antonio Muñoz San Roque, and Javier García-González. Modeling and forecasting electricity prices with input/output hidden markov models. *Power Systems, IEEE Transactions on*, 20(1):13–24, 2005.

[41] Google. Google data center efficiency. http://www.google.com/about/datacenters/efficiency/internal/. Online; accessed 28-February-2016.

[42] Meg Gottstein and Lisa Schwartz. The role of forward capacity markets in increasing demand-side and other low-carbon resources: experience and prospects. *Montpelier, Vt.: Regulatory Assistance Project*, 2010.

[43] Phillip G Gould, Anne B Koehler, J Keith Ord, Ralph D Snyder, Rob J Hyndman, and Farshid Vahid-Araghi. Forecasting time series with multiple seasonal patterns. *European Journal of Operational Research*, 191(1):207–222, 2008.

[44] Huseyin Guler, B Barla Cambazoglu, and Oznur Ozkasap. Cutting down the energy cost of geographically distributed cloud data centers. In *Energy Efficiency in Large Scale Distributed Systems*, pages 279–286. Springer, 2013.

[45] Michael Hibon and Spyros Makridakis. Arma models and the box–jenkins methodology. 1997.

[46] William Hogan. Reshaping the electricity industry. *JFK School of Government, Harvard University, http://ksghome. harvard. edu/˜. whogan. cbg. ksg/hiid797a. pdf*, 1997.

[47] William W Hogan. A competitive electricity market model. *Harvard Electricity Policy Group*, 1993.

[48] Ronald Huisman, Christian Huurman, and Ronald Mahieu. Hourly electricity prices in day-ahead markets. *Energy Economics*, 29(2):240–248, 2007.

[49] Rob J Hyndman. A blog by rob j hyndman. http://robjhyndman.com/hyndsight, 2014. Online; accessed 24-March-2016.

[50] ISO New England Inc. Iso new england power market. http://isone.com/, 1995. Online; accessed 10-April-2016.

[51] Red Hat Inc. Wildfly application server. http://wildfly.org/, 2009. Online; accessed 25-March-2016.

[52] PJM Interconnection. Pjm power market. http://www.pjm.com/, 1994. Online; accessed 10-April-2016.

[53] Aman Kansal, Feng Zhao, Jie Liu, Nupur Kothari, and Arka A Bhattacharya. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 39–50. ACM, 2010.

[54] Jonathan Koomey. Growth in data center electricity use 2005 to 2010. *A report by Analytical Press, completed at the request of The New York Times*, page 9, 2011.

[55] Jonathan G Koomey. Worldwide electricity used in data centers. *Environmental Research Letters*, 3(3):034008, 2008.

[56] Kien Le, Ricardo Bianchini, Margaret Martonosi, and Thu D Nguyen. Cost-and energy-aware load distribution across data centers. *Proceedings of HotPower*, pages 1–5, 2009.

[57] Young Choon Lee and Albert Y Zomaya. Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280, 2012.

[58] Haikun Liu, Hai Jin, Cheng-Zhong Xu, and Xiaofei Liao. Performance and energy modeling for live migration of virtual machines. *Cluster computing*, 16(2):249–264, 2013.

[59] Liang Liu, Hao Wang, Xue Liu, Xing Jin, Wen Bo He, Qing Bo Wang, and Ying Chen. Greencloud: a new architecture for green data center. In *Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session*, pages 29–38. ACM, 2009.

[60] Zhenhua Liu, Adam Wierman, Yuan Chen, Benjamin Razon, and Niangjun Chen. Data center demand response: Avoiding the coincident peak via workload shifting and local generation. *Performance Evaluation*, 70(10):770–791, 2013.

[61] Alicia T Lora, Jesús M Riquelme Santos, Antonio G Exposito, José L Martínez Ramos, and José C Riquelme Santos. Electricity market price forecasting based on weighted nearest neighbors techniques. *Power Systems, IEEE Transactions on*, 22(3):1294–1301, 2007.

[62] Dražen Lučanin. Philharmonic - a geo-distributed cloud simulator. https://philharmonic.github.io/, 2013. Online; accessed 26-March-2016.

[63] Dražen Lucanin and Ivona Brandic. Take a break: cloud scheduling optimized for real-time electricity pricing. In *Cloud and Green Computing (CGC), 2013 Third International Conference on*, pages 113–120. IEEE, 2013.

[64] Dražen Lučanin, Foued Jrad, Ivona Brandic, and Achim Streit. Energy-aware cloud management through progressive sla specification. In *Economics of Grids, Clouds, Systems, and Services*, pages 83–98. Springer, 2014.

[65] David Meisner, Brian T Gold, and Thomas F Wenisch. Powernap: eliminating server idle power. In *ACM Sigplan Notices*, volume 44, pages 205–216. ACM, 2009.

[66] Mayank Mishra, Anwesha Das, Purushottam Kulkarni, and Anirudha Sahoo. Dynamic resource management using virtual machine migrations. *Communications Magazine, IEEE*, 50(9):34–40, 2012.

[67] Christian Mugele, Svetlozar Rachev, and Stefan Trück. Stable modeling of different european power markets. *Investment Management & Financial Innovations*, 2(3), 2005.

[68] Harold L Nelson and CWJ Granger. Experience with using the box-cox transformation when forecasting economic time series. *Journal of Econometrics*, 10(1):57–69, 1979.

[69] Michael Nelson. Virtual machine migration, January 27 2009. US Patent 7,484,208.

[70] NIST/SEMATECH. e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/index.htm, 2012. Online; accessed 24-March-2016.

[71] Francisco J Nogales, Javier Contreras, Antonio J Conejo, and Rosario Espínola. Forecasting next-day electricity prices by time series models. *Power Systems, IEEE Transactions on*, 17(2):342–348, 2002.

[72] Nikos K Nomikos and Orestes A Soldatos. Modelling short and long-term risks in power markets: Empirical evidence from nord pool. *Energy Policy*, 38(10):5671–5683, 2010.

[73] EGHEOSA Onaiwu. How does bilateral trading differ from electricity pooling. *Universisty of Dundee*, 2009.

[74] Oracle. Java enterprise edition. http://www.oracle.com/technetwork/java/javaee/overview/index.html. Online; accessed 25-March-2016.

[75] Oracle. Oracle database 11g express edition. http://www.oracle.com/technetwork/database/database-technologies/express-edition/overview/index.html. Online; accessed 25-March-2016.

[76] Hsiao-Tien Pao. Forecasting electricity market pricing using artificial neural networks. *Energy Conversion and Management*, 48(3):907–912, 2007.

[77] Michael K Patterson. The effect of data center temperature on energy efficiency. In *Thermal and Thermomechanical Phenomena in Electronic Systems, 2008. ITHERM 2008. 11th Intersociety Conference on*, pages 1167–1174. IEEE, 2008.

[78] Chandrashekhar S Pawar and Rajnikant B Wagh. Priority based dynamic resource allocation in cloud computing. In *Cloud and Services Computing (ISCOS), 2012 International Symposium on*, pages 1–6. IEEE, 2012.

[79] Google Cloud Platform. Google cloud storage, google prediction api, and google bigquery sla. https://cloud.google.com/storage/sla. Online; accessed 27-March-2016.

[80] Google Cloud Platform. Google compute engine service level agreement (sla). https://cloud.google.com/compute/sla. Online; accessed 27-March-2016.

[81] Asfandyar Qureshi, Rick Weber, Hari Balakrishnan, John Guttag, and Bruce Maggs. Cutting the electric bill for internet-scale systems. In *ACM SIGCOMM Computer Communication Review*, volume 39, pages 123–134. ACM, 2009.

[82] Lei Rao, Xue Liu, Le Xie, and Wenyu Liu. Minimizing electricity cost: optimization of distributed internet data centers in a multi-electricity-market environment. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. IEEE, 2010.

[83] Christian Redl, Reinhard Haas, Claus Huber, and Bernhard Böhm. Price formation in electricity forward markets and the relevance of systematic forecast errors. *Energy Economics*, 31(3):356–364, 2009.

[84] RForge. rjava - low-level r to java interface. https://rforge.net/rJava/, 2007. Online; accessed 26-March-2016.

[85] George Athanasopoulos Rob J Hyndman. Forecasting: principles and practice. https://www.otexts.org/fpp, 2012. Online; accessed 21-March-2016.

[86] Duke University Robert Nau, Fuqua School of Business. Statistical forecasting: notes on regression and time series analysis. http://people.duke.edu/~rnau/411home.htm, 2016. Online; accessed 24-March-2016.

[87] EPEX Spot SE. Epexspot power market. https://www.epexspot.com/, 2008. Online; accessed 10-April-2016.

[88] Amazon Web Services. Amazon ec2 service level agreement. https://aws.amazon.com/de/ec2/sla/, 2013. Online; accessed 27-March-2016.

[89] Jose Luis Lucas Simarro, Rafael Moreno-Vozmediano, Ruben S Montero, and Ignacio Martín Llorente. Dynamic placement of virtual machines for cost optimization in multi-cloud environments. In *High Performance Computing and Simulation (HPCS), 2011 International Conference on*, pages 1–7. IEEE, 2011.

[90] Deepak Singhal and KS Swarup. Electricity price forecasting using artificial neural networks. *International Journal of Electrical Power & Energy Systems*, 33(3):550–555, 2011.

[91] Zhongfu Tan, Jinliang Zhang, Jianhui Wang, and Jun Xu. Day-ahead electricity price forecasting using wavelet transform combined with arima and garch models. *Applied energy*, 87(11):3606–3610, 2010.

[92] The R Core Team. R: A language and environment for statistical computing. https://cran.r-project.org/doc/manuals/r-release/fullrefman.pdf, 2016. Online; accessed 22-March-2016.

[93] Susan F Tierney, Todd Schatzki, and Rana Mukerji. Uniform-pricing versus pay-as-bid in wholesale electricity markets: Does it make a difference? *New York ISO*, 2008.

[94] Johan Tordsson, Rubén S Montero, Rafael Moreno-Vozmediano, and Ignacio M Llorente. Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers. *Future Generation Computer Systems*, 28(2):358–367, 2012.

[95] V Vahidinasab, S Jadid, and A Kazemi. Day-ahead price forecasting in restructured power systems using artificial neural networks. *Electric Power Systems Research*, 78(8):1332–1342, 2008.

[96] Panos Vassiliadis, Alkis Simitsis, and Spiros Skiadopoulos. Conceptual modeling for etl processes. In *Proceedings of the 5th ACM international workshop on Data Warehousing and OLAP*, pages 14–21. ACM, 2002.

[97] Rafal Weron. *Modeling and forecasting electricity loads and prices: a statistical approach*, volume 403. John Wiley & Sons, 2007.

[98] Rafal Weron. Forecasting wholesale electricity prices: A review of time series models. *FINANCIAL MARKETS: PRINCIPLES OF MODELLING, FORECASTING AND DECISION-MAKING, W. Milo, P. Wdowinski, eds., FindEcon Monograph Series, WUL, Lodz*, 2008.

[99] Rafał Weron. Market price of risk implied by asian-style electricity options and futures. *Energy Economics*, 30(3):1098–1115, 2008.

[100] Rafał Weron, Michael Bierbrauer, and Stefan Trück. Modeling electricity prices: jump diffusion and regime switching. *Physica A: Statistical Mechanics and its Applications*, 336(1):39–48, 2004.

[101] Rafal Weron and Adam Misiorek. Forecasting spot electricity prices with time series models. In *Proceedings of the European Electricity Market EEM-05 Conference*, pages 133–141, 2005.

[102] Rafał Weron and Adam Misiorek. Forecasting spot electricity prices: A comparison of parametric and semiparametric time series models. *International Journal of Forecasting*, 24(4):744–763, 2008.

[103] Rafał Weron, Ingve Simonsen, and Piotr Wilman. Modeling highly volatile and seasonal markets: evidence from the nord pool electricity market. In *The application of econophysics*, pages 182–191. Springer, 2004.

[104] Data Center World. Data center trends: Optimizing power markets. http://www.datacenterworld.com/fall2013/account/Uploader/uploader_files/show/335/, 2013. Online; accessed 28-February-2016.

[105] Zhen Xiao, Weijia Song, and Qi Chen. Dynamic resource allocation using virtual machines for cloud computing environment. *Parallel and Distributed Systems, IEEE Transactions on*, 24(6):1107–1117, 2013.

[106] Andrew J Younge, Gregor Von Laszewski, Lizhe Wang, Sonia Lopez-Alarcon, and Warren Carithers. Efficient resource management for cloud computing environments. In *Green Computing Conference, 2010 International*, pages 357–364. IEEE, 2010.

[107] Eric Zivot and Jiahui Wang. Vector autoregressive models for multivariate time series. In *Modeling Financial Time Series with S-Plus®*, pages 369–413. Springer, 2003.