A FRAMEWORK AND ALGORITHMS FOR MULTIVARIATE TIME SERIES ANALYTICS (MTSA): LEARNING, MONITORING, AND RECOMMENDATION

By

Chun-Kit Ngan
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

___________________________________	Dr. Alex Brodsky, Dissertation Director
___________________________________	Dr. Jessica Lin, Dissertation Director
___________________________________	Dr. Larry Kerschberg, Committee Member
___________________________________	Dr. Karla L. Hoffman, Committee Member
___________________________________	Dr. Stephen Nash, Senior Associate Dean
___________________________________	Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering
Date:______________________________	Summer Semester 2013 George Mason University Fairfax, VA

A Framework and Algorithms for Multivariate Time Series Analytics (MTSA): Learning, Monitoring, and Recommendation

A Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Information Technology at George Mason University

By

Chun-Kit Ngan
MBA in Management Information Systems
California State University, Chico, 2006

Directors: Dr. Alex Brodsky and Dr. Jessica Lin, Associate Professor
Department of Computer Science

Summer Semester 2013
George Mason University
Fairfax, VA

DEDICATION

I dedicate my dissertation work to my family. I express my special feeling of gratitude to my loving parents, Chee-Wah Ngan and Chun-Fun Wan, whose words of encouragement and push for tenacity ring in my ears. My brothers, Pui-Kit and Hon-Kit, have never left my side and are very special.

ACKNOWLEDGEMENTS

I would like to thank my committee members who were more than generous with their expertise and precious time. A special thanks to Prof. Alexander Brodsky, my Dissertation Director, for his countless hours of reflecting, reading, encouraging, supervising, and most of all patience throughout the entire process. Without sincere mentoring from Prof. Brodsky, my research would not exist. He believed in me and led me along the road to reach my goal. I feel deeply grateful. Thank you so much a lot, Prof. Brodsky!

I also want to show my appreciation to my committee members, Dr. Jessica Lin (Co-Director), Dr. Larry Kerschberg, and Dr. Karla L. Hoffman, for agreeing to serve on my committee and to give me their time and advice every time I needed them.

Finally I would like to acknowledge and thank my school division, Department of Computer Science at the Volgenau School of Engineering, for allowing me to conduct my research and providing any assistance requested.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

A FRAMEWORK AND ALGORITHMS FOR MULTIVARIATE TIME SERIES ANALYTICS (MTSA): LEARNING, MONITORING, AND RECOMMENDATION

Chun-Kit Ngan, Ph.D.

George Mason University, 2013

Dissertation Directors: Dr. Alex Brodsky and Dr. Jessica Lin

Making decisions over multivariate time series is an important topic which has gained significant interest in the past decade. A time series is a sequence of data points which are measured and ordered over uniform time intervals. A multivariate time series is a set of multiple, related time series in a particular domain in which domain experts utilize multivariate time series to make a vital decision. Through studying multivariate time series, specialists are able to understand problems of events from different perspectives within particular domains. Identification and detection of those significant events over multivariate time series can lead to a better decision-making and actionable recommendations.

Existing approaches to identifying and detecting significant events and delivering recommendations can be roughly divided into two categories: domain-knowledge-based and formal-learning-based. The former relies on solely domain experts' knowledge.

Based on their knowledge and experience, domain experts can determine the conditions and the designed parameters to detect the events of interest and then deliver appropriate actions. However, those parameters are not always accurate. In addition, the parameters are static, but the problem that we deal with is often dynamic in nature. Thus the domain-knowledge-based approach lacks a formal mathematical foundation to dynamically learn parameters to meet the need of the changing environment. The latter approach, formal-learning-based, is to utilize formal learning methods such as non-linear logistic regression models. The logistic regression models are used to predict the occurrence of an event by learning parametric coefficients of the logistic distribution function of the explanatory variables. The challenge using formal learning methods to support decision-making is that they do not always produce satisfactory results, as they are not incorporated with domain experts' knowledge into their learning processes. Clearly, both approaches, domain-knowledge-based and formal-learning-based, do not take advantage of each other to learn optimal decision parameters, which are used to detect the events and then make better actionable recommendations.

To support such an event-based decision-making and recommendation, I proposed a Web-Mashup Application Service Framework for Multivariate Time Series Analytics (MTSA). It is an integrated framework to support the MTSA service development and implementation, including parametric model definitions, query formulation, parameter learning, data monitoring, decision recommendations, and model evaluations. Domain experts can use the framework to develop and implement their web-based decision-making applications on the Internet. More specifically, the technical contributions of my

dissertation include (1) the MTSA data model and query language to support the development and implementation of those MTSA services; (2) the hybrid-based mathematical models and computational algorithms that combine the strengths of both domain-knowledge-based and formal-learning-based approaches to learn decision parameters over multivariate time series; and (3) the experimental case studies to solve the real-world problems in two different domains, i.e., the stock markets and the electric power microgrids, to evaluate the models and algorithms, which are designed for the learning, monitoring, and recommendation services.

CHAPTER 1. INTRODUCTION

1.1 Motivation and Background

A time series is a sequence of data points which are measured and ordered over uniform time intervals. Some examples of time series include wind speed, employment figures, body temperatures, etc. A multivariate time series is a set of multiple, related time series data [1, 2], e.g., heartbeats and pulses, in a particular domain in which domain experts utilize multivariate time series to make a vital decision. Thus making decisions over multivariate time series is an important topic which has gained significant interest in the past decade. For instance, financial analysts and researchers monitor daily stock prices, weekly interest rates, and monthly price indexes to analyze the state of stock markets and make trading decisions. Energy planners use the supply and demand of electricity, heating, and cooling to determine the load shedding of electric power and to procure the best energy investment option. Physicians and medical scientists measure diastolic and systolic blood pressure over time and electrocardiogram tracings to evaluate cardiovascular patients. Sociologists and demographers study annual birth rates, mortality rates, accident rates, and crime rates to discover hidden social problems within a community. The purpose of these studies over multivariate time series is to assist specialists in understanding problems of events from different perspectives within particular domains. Identification and detection of those significant events over multivariate time series can lead to a better decision-making and actionable recommendations.

Consider an example of a timely event detection of certain conditions in a stock market, i.e., a bear market bottom, which can provide investors a valuable insight into the best investment opportunity. Such identification and detection of the event over a range of financial indexes and economic indicators can aid in the task of decision-making and the determination of action plans. For instance, financial analysts suggest that a bear market bottom [3] has occurred if some changes of the financial indexes, such as the S&P 500 falls at least 20% and the Consumer Confidence Index declines at least 30 points, are detected. Note that those decision parameters, e.g., 20% and 30 points, may be given by the domain experts or learned from the formal mathematical computations. This financial decision-making detection can be developed by using web services [4], which enable domain experts to convert such a utility program into a web-based application. The decision-making function provided by this web-based application can be published and available to be used worldwide. To support such decision-making and determination over multivariate time series around the globe, it is desirable to offer a range of common web services on the Internet. For example, financial analysts can use the developed web services with the given monitoring conditions and decision parameters, e.g., the S&P 500 falls at least 20% and the Consumer Confidence Index declines at least 30 points, to monitor the bear market bottom and to recommend an appropriate action when the bottom is detected.

More specifically, the six web services for decision-making over multivariate time series are expected and shown in Figure 1. The *Model Definition Service* allows domain experts to define different types of parametric model templates, which determine the occurrence of an event, identified by domain experts. In the financial example that predicts the event, i.e., the bear market bottom, the model template may consist of the indexes such as S&P 500 percentage decline (SPD) and Consumer Confidence Index drop (CCD), where the values of the parameters may be specified. Given such a parametric model template in a given domain and of specific parametric values, the *Monitoring and Recommendation Service* continuously screens the incoming data stream of the indexes for detecting the event. When the event of interest, e.g., the bear market bottom, has occurred, the service recommends an action, e.g., buying stock. If the parameters are not known a priori, the *Parameter Learning Service* parameterizes the template and supports learning of the parameters from historical and projected time series. The accuracy of the decision parameters are ensured through the *Model Accuracy and Quality Evaluation Service*, which validates the prediction, i.e., the bear market bottom, with the observed real data, and updates the model if necessary. The *Querying Service* allows domain experts to express and implement the complex information services, mentioned above in a high-level abstraction, over multivariate time series. The *Web Portal Service* enables domain experts to develop a point-of-access service as a main entrance (1) to integrate all their implemented MTSA services together, (2) to provide a consistent style and format among all those services, and (3) to centralize users' access control and procedures on those services.

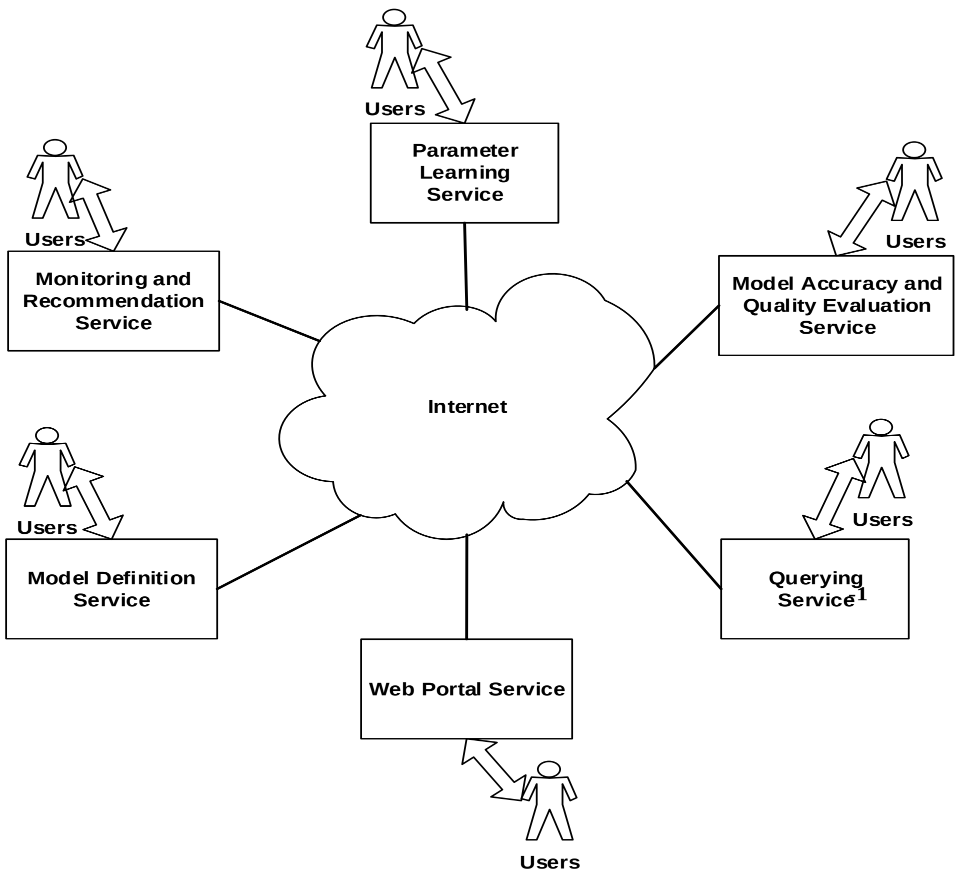

Users
Parameter Learning Service
Users
Monitoring and Recommendation Service
Users
Model Accuracy and Quality Evaluation Service
Internet
Users
Model Definition Service
Users
Querying Service1
Web Portal Service
Users

Figure 1: Web Services for Multivariate Time Series Analytics (MTSA) on the Internet

1.2 Research Challenges

For expressing and implementing the services shown in Figure 1, Structured Query Language (SQL) [5] is a language tool for those web services as it is an easy-to-learn and easy-to-use language. SQL has been utilized in many querying services, such as creating database tables and retrieving data from databases. However, the SQL has significant limitations. The SQL itself is not designed for the *continuous* monitoring and recommendation services over multivariate time series as it is a *one-time, passive* query.

It means that the queries are initiated by users, and the DBMS [6, 7] executes the queries for them. Thus the same SQL statement cannot be run repeatedly and automatically to monitor a specific event and then recommend an action.

To monitor events of interest, an extension of SQL, called SQL triggers [8], has been proposed. By defining the *Event-Condition-Action* rule, the SQL trigger automatically executes a particular action, e.g., creating a new table, when a pre-defined event, e.g., inserting tuples into a current table, has taken place and the specific condition, i.e., a true or false statement, has been satisfied. However, the SQL triggers are not designed for handling the events on data which are time sequences, i.e., temporal, and are time-sensitive-oriented [9, 10], e.g., the percentage changes of a financial indicator at an instant of time. To address the problems, several approaches have been suggested. One approach is to use a pure procedural language, Aurora Query [11, 12]. Using the provided GUI, the users can connect the system- and user-defined operators written in the procedural language, and the arrows together to form a continuous, graphical query that processes the streams in the real time for monitoring and recommendation. The second approach is to extend the conventional SQL as the relational-based language, AQuery [13]. One of the major extensions of this query is that a new clause, ASSUMING ORDER, sorts the sequence of the relations based on the timestamp, after the FROM clause is computed, for continuous monitoring and recommendation. In this dissertation, I have applied some of these proposed temporal extensions of SQL for the services of monitoring and recommendation in the proposed framework.

The two approaches, Aurora and AQuery, assume that the queries for monitoring conditions and recommending actions are given by users. However, an important question is how to automatically create those monitoring conditions and recommended actions to achieve optimal outcomes. Currently, existing approaches to support monitoring of interesting events and delivering recommendations can be roughly divided into two categories: domain-knowledge-based and formal-learning-based. The former relies on solely domain experts' knowledge. Based on their knowledge and experience, domain experts can determine the conditions to detect the events of interest and then deliver appropriate actions. In the financial example, I assume that the domain experts have identified a set of financial indexes that can be used to determine the bear market bottom or the "best buy" opportunity. The indexes include the S&P 500 percentage decline (SPD), Coppock Guide (CG), Consumer Confidence Index drop (CCD), ISM Manufacturing Survey (ISM), and Negative Leadership Composite "Distribution" (NLCD) [3]. If these indexes satisfy the pre-defined, parameterized conditions, e.g., SPD < -20%, CG < 0, etc., it signals that the best period for the investors to buy the stocks is approaching. Often these parameters may reflect some realities since they are set by the domain experts based on their past experiences, observations, intuition, and domain knowledge. However, they are not always accurate. In addition, the parameters are static, but the problem that we deal with is often dynamic in nature. The market is constantly impacted by many unknown and uncontrollable factors. Thus the domain-knowledge-based approach lacks a formal mathematical foundation to dynamically learn parameters to meet the need of the changing environment.

The latter approach, formal-learning-based, is to utilize formal learning methods such as non-linear logistic regression models [14, 15, 16, 17, 18]. The logistic regression models are used to predict the occurrence of an event (0 or 1) by learning parametric coefficients of the logistic distribution function of the explanatory variables. This is done based upon the historical data by applying nonlinear regression models and the Maximum Likelihood Estimation (MLE) [19]. The main challenge concerning using the formal learning methods to support decision-making is that they do not always produce satisfactory results, as they do not consider incorporating domain knowledge into their formal learning approaches.

Because of this reason, some existing mathematical models, e.g., the Durland and McCurdy duration-dependent Markov-switching (DDMS) models, such as DDMS-ARCH and DDMS-DD [20], integrated domain knowledge, i.e., duration dependence, into their forecasting criteria. Both models, DDMS-ARCH and DDMS-DD, are extended from the Markov-switching model [21] that is incorporated with the duration dependence to affect a transition probability that is parameterized using the logistic distribution function. The transition probability is the probability of being in a particular state at a specific point in time. The value and the trend of this probability over time demonstrate the current state of the stock market. For instance, when the economy is in the bull market, the probability of staying in the bull market increases with the duration and is always greater than 0.5 over time. The probability of staying in the bear market also increases with the duration, but the probability of staying in the bear market is less than 0.5 until after some consecutive occurrences of this state occur. The effect of the duration to the transition probability of being in a state also influences the conditional mean and the conditional variance of the stock returns. In the bull market, the conditional mean of the return is high, but the variance of the return volatility is low. In the bear market, the conditional mean of the return is low, but the variance of the return volatility is high. Due to the different properties of the conditional mean and the variance of the stock returns in a particular state, the trend and the volatility of the return in a state over time is also different. For example, the conditional return in the bull market declines with the duration, but the return volatility in the bear market increases over time. Having observed the behaviors of the probability and the return of the stock market over time, the market state can be determined. However, all of these models only consider a single element, i.e., duration, to integrate into the model to determine a market state. Thus this approach is not flexible and not complete as there are many other external, unknown factors that may affect the state of the stock market in the current environment.

Moreover, those DDMS models also involve parameters that need to be learned by formal mathematical methods. Without wide-ranging domain experts' knowledge, those formal learning methods become computationally intensive and time consuming. The whole model building is an iterative and interactive process, including model formulations, parameter estimations, and model evaluations. Despite enormous improvements in computer software in recent years, fitting such a nonlinear quantitative decision model [22] is not a trivial task, especially when the parameter learning process involves multiple explanatory variables, i.e., high dimensionality. Working with high-dimensional data creates difficult challenges, a phenomenon known as the "curse of dimensionality" [23, 24]. Specifically, the amount of observations required in order to obtain good estimates increases exponentially with the increase of dimensionality. In addition, many learning algorithms do not scale well on high dimensional data due to the high computational cost. The parameter computations by formal-learning-based approaches, e.g., the logistic regression model, are complicated and costly, and they lack the consideration of integrating sufficient domain experts' knowledge into the learning process – a step that can potentially reduce the dimensionality. Clearly, both approaches, domain-knowledge-based and formal-learning-based, do not take advantage of each other to learn optimal decision parameters, which are used to monitor the events and then deliver better actionable recommendations.

Since it is convenient to express parametric model templates, which are given by domain experts, using a SQL-like language, and the parameter learning can be reduced to a decision optimization problem, i.e., finding decision parameters that optimizes an objective function over historical and projected data, the Decision Guidance Query Language (DGQL) [25, 26, 27, 28, 29, 30, 31, 32] has been proposed and developed. The DGQL is a query language that combines the strengths of both approaches, domain-knowledge-based and formal-learning-based, to solve decision optimization problems. Using the DGQL, users can learn optimal decision parameters to maximize an objective function, e.g., the earning in the financial example, within given constraints. And then users can use the learned parameters to monitor the events and then deliver the better actionable recommendations. Currently, the DGQL uses the mixed integer linear programming (MILP) solver [33, 34, 35], i.e., the IBM ILOG CPLEX optimizer [36], from which the learning over multivariate time series is reduced to the DGQL construct and then to a MILP formulation.

To be formulated a MILP problem, its objective function and the related constraints are required to be linear expressions of decision variables, which the data types can be real, integer, or binary. The MILP problem is then sent to the CPLEX solver to optimally instantiate the decision variables that satisfy all the given constraints and then maximize the objective function. However, to solve a MILP problem that involves multiple decision parameters for detecting the occurrence of events over multivariate time series requires the solver to generate a large number of binary variables. Each binary variable, which corresponds to a time point of input parametric time series data, stores 0 or 1 to indicate whether or not a particular event occurs at a particular point in time. More precisely, the total number of binary variables being generated is determined from the size of input parametric time series. This large number of binary variables being generated significantly increases the searching space that adversely overloads the computational performance of the CPLEX solver to learn the optimal decision parameters to detect the events. Because the time complexity that the CPLEX optimizer requires to solve such problems is incredibly high, i.e., $O(k2^N)$, where k is the number of input parametric time series of an event, and N is the size of the training data set of time series data, it takes a much longer time to solve the problem. Thus this approach is not scalable for parameter learning even though the number of the time series involved in the problem is very limited. Another problem is that the DGQL is NOT a *continuous, active* query to support the monitoring and recommendation services.

1.3 Problem Statement

This dissertation addresses the approaches to close the research gaps identified in Section 1.2. More specifically, I address the following research questions:

(1) How to create an event-based service framework from which domain experts can use to develop and implement their web-based decision-making applications on the Internet.

(2) How to extend database models and SQL to enable service providers to express and deliver complex information services, including querying, learning, monitoring, and recommendation, over multivariate time series in an intuitive, high-level abstraction.

(3) How to develop "time-component-based" mathematical models and learning algorithms that combine the strengths of both domain-knowledge-based and formal-learning-based approaches to learn optimal decision parameters over multivariate time series in an efficient manner.

(4) How to design and implement experimental case studies that unify decision optimization models, computational algorithms, database models, and query language to verify and evaluate the above services.

1.4 Thesis Statement and Summary of Research Contributions

In this dissertation, I focus on a framework, models, algorithms, and experimental case studies to bridge the gaps to solve the above problems.

Thesis Statement: *It is possible to develop a Multivariate Time Series Analytics (MTSA) framework that provides database models and query language to support the services, including (1) Model Template Definition, (2) Querying, (3) Data Monitoring* *and Decision Recommendation, (4) Parameter Learning, and (5) Model Evaluation. Furthermore, it is possible to develop formal mathematical foundations and efficient algorithms for parameter learning problems over multivariate time series in the MTSA service framework that can be implemented in a practical system.*

More specifically, the technical contributions of this dissertation are as follows:

(1) Web-Mashup Application Service Framework, Data Models, and Query Language for Multivariate Time Series Analytics (MTSA).

I propose a web-mashup application service framework for MTSA [37]. This framework is an integrated development framework that provides a medium to domain experts and supports their quick implementations of web services shown in Figure 1. Using the Web-Mashup function offered by the Web 2.0 technology [38] on the framework, domain experts can collect and unify global information and data from different channels and media, such as web sites, data sources, and organization information, to generate a concentric view of collected time series data from which the parameter learning service determines optimal decision parameters. Using optimal decision parameters, domain experts can employ the monitoring service to detect events and the recommendation service to suggest actions. All these services can be integrated into the web portal service.

To support the development of those MTSA services, I develop an extension of the relational database model and SQL called MTSA Data Model and Query Language [37, 39, 40]. With the high-level MTSA constructs, the MTSA data model and query language can support those service implementations on the framework to provide the services shown in Figure 1. Using the MTSA database model and query language, users are able to construct a multivariate-time-series, binary-event, and decision-parameter database, manipulate the corresponding data, and implement the services as well. For example, the *MTSA monitoring and recommendation service* enables users to form a query. The query continuously monitors the data streams that satisfy the parameterized conditions, in which the parameters have been instantiated by the *MTSA parameter learning service*. The main reason behind why the query language, i.e., SQL, was chosen to be extended is because the query language SQL is much closer to the natural language, English. Users who even have little or no programming skills can still use the MTSA query language to write simple decision optimization problems.

(2) Models and Algorithms for Classes of MTSA Problems.

2a. Expert Query Parametric Estimation (EQPE) Model and Checkpoint Algorithm.

To formulate decision optimization problems initiated by the MTSA query language, I develop a mathematical hybrid-based model of learning over multivariate time series, named Expert Query Parametric Estimation (EQPE) Model [39, 40, 41], used for the *MTSA parameter learning service*. The EQPE model is a well-defined model which captures domain experts' knowledge in expression of multivariate time series, decision parameters, parametric constraints, time utility, and objective functions. The objective functions are dependent on the time points from which the parameters are learned. More specifically, the *MTSA model template definition service* takes the template of conditions identified by domain experts. Such template consists of inequalities of values in the time sequences. Then the learning service “parameterizes” the template, e.g., $SPD < p_1$ and $CCD < p_2$, where SPD is the S&P 500 percentage decline and CCD is the Consumer Confidence Index drop, and supports learning of the parameters, p_1 and p_2, from the historical time series. In this model, conjunctions of inequality constraints are considered, and the objective function is maximized when the parameters, which are determined from the optimal time points of the time utility function, are optimized.

To determine the optimal time points from which the optimal decision parameters are learned, I develop a computational algorithm, called Checkpoint [39, 40, 41]. The algorithm also combines the strengths of both domain experts’ knowledge in terms of parametric constraints and formal learning methodology by using the regression approach. The goal of the learning algorithm is to efficiently learn the decision parameters that maximize the objective function, e.g., earnings in the financial example, over multivariate time series data with a considerably low time complexity. This new algorithm guarantees a true optimal time point and has the complexity of $O(kNlogN)$, where N is the size of the learning data set, and k is the number of input parametric time series. To demonstrate the effectiveness and the efficiency of the algorithm, I compare my method with the domain-knowledge-based approach and the formal logistic regression model in the financial case study.

2b. Multi-Event Expert Query Parametric Estimation (ME-EQPE) Model and Multidimensional *M*-Checkpoint Algorithm.

However, the proposed EQPE model and the Checkpoint algorithm are only able to learn one set of decision parameters for one particular event at a single time point, whereas there are many real-world scenarios that the parameter learning is at multiple time points in sequence. For instance, consider the above financial example again, in which the investors would like to decide on both when the S&P 500 fund is purchased at $t_{purchased}$ and when the fund is sold at t_{sold} rather than either $t_{purchased}$ or t_{sold} only. Using the EQPE model and the Checkpoint algorithm, the investors would not be able to obtain the optimal decision parameters simultaneously at the two interrelated decision time points and then to gain the maximal earning of the S&P 500 index fund. To address the shortcomings of the one-dimensional-time-point model and algorithm, I develop an extended model, Multi-Event Expert Query Parametric Estimation (ME-EQPE), and an algorithm, Multidimensional *M*-Checkpoint [42, 43], to solve the problems. The new extended model and algorithm not only maintain the advantages of one-dimensional-time-point model, EQPE, and algorithm, Checkpoint, but also learn decision parameters at multiple, inter-related time points optimally for multi-events. More specifically, the *M*-Checkpoint algorithm is able to:

- Combine the strengths of both domain-knowledge-based and formal-learning-based approaches to solve the decision optimization problems that involve multiple decision time points to maximize the utility.
- Maintain an optimality of the learned multiple sets of decision parameters in their respective events during the computations.
- Guarantee a satisfactory forecasting result by using the learned multiple sets of decision parameters.

2c. Relaxed *R*-Checkpoint Algorithm.

Although the proposed *M*-Checkpoint algorithm is able to learn multiple sets of decision parameters optimally and guarantee a satisfactory forecasting result, the computational complexity of this algorithm is considerably high, i.e., $O(N^m)$, where N is the size of the learning data set and m is the number of the events. To solve the issue of the high complexity, I develop a relaxed algorithm, *R*-Checkpoint [44, 45]. This algorithm is able to learn multiple sets of decision parameters that are fairly close to the optimal parameters learned from the *M*-Checkpoint algorithm, produce reasonably forecasting results, and maintain a satisfactorily low time complexity, i.e., $O(QkNlogN)$, as compared with $O(N^m)$ of the *M*-Checkpoint algorithm, where Q is the total number of m-event, time-point combinations which yields the top-Q time utility, and k is the number of input parametric time series for each event. To demonstrate the performance of the new algorithm, I conduct the experiment in the financial case study. Specifically, I compare the forecasting results that are detected by the decision parameters learned from the *R*-Checkpoint with the results that are determined by the optimal parameters obtained from the *M*-Checkpoint, as well as the parametric coefficients of the logistic regression model. I show that the forecasting results by the *R*-Checkpoint are slightly lower than those of the optimal *M*-Checkpoint algorithm and are considerably higher than those of the logistic regression methodology.

2d. Hybrid-Based Multivariate Time Series Analytics - Parameter Estimation (MTSA-PE) Model.

However, the discussed hybrid-based models, EQPE and ME-EQPE, and algorithms, Checkpoint, *M*-Checkpoint, and *R*-Checkpoint, are only able to solve a specific class of problems that (1) their decision parameters of an objective function are learned from optimal time points of a time utility function, (2) the monitoring template has to be in the considered form, i.e., conjunctions of inequality constraints, and (3) the constraints being used are solely for monitoring purposes. To address the above weaknesses, I develop a general, hybrid-based model, Multivariate Time Series Analytics – Parameter Estimation (MTSA-PE) [37]. This model maintains a combination of both domain-knowledge-based and formal-learning-based approaches with possibly incorporating any global constraints that are applied to an entire problem, and monitoring constraints, which are used to detect the occurrence of events. Both types of inequality constraints, global and monitoring, are allowed in any possible combinations and forms. Using the MTSA-PE model associated with an external solver, e.g., the IBM ILOG CPLEX optimizer, domain experts can learn decision parameters that satisfy all the given constraints and then optimize the objective function, which is independent of a particular time point.

(3) Experimental Case Studies.

In order to demonstrate the capability of the service framework, I design and implement the experimental case studies to solve the real-world problems in two different domains, i.e., the stock markets and the electric power microgrids, to verify and evaluate the models, algorithms, and query language, which are designed for the learning, querying, monitoring, and recommendation services stated above. I conduct the experiments in the financial domain, i.e., the stock market in which when it is the best time for financial analysts to buy and or sell the S&P 500 index funds for their clients, to demonstrate the performance of EQPE and ME-EQPE models, as well as the Checkpoint, *M*-Checkpoint, and *R*-Checkpoint algorithms. Using the electric power microgrids on the campuses at George Mason University as an example, I illustrate how the MTSA-PE model with the external solver to solve the energy problems, such as determinations of optimal peak demand bounds and decisions of the best energy investment option.

The rest of this dissertation is organized as follows. In Chapter 2, I briefly discuss related materials on data models, query language, and existing approaches of parameter learning. In Chapter 3, I describe the proposed web-mashup application service framework and the developed data models and query language for the MTSA services, which are mainly designed for learning, monitoring, and recommendation. In Chapter 4, I explain the single-time-point EQPE model and Checkpoint algorithm to learn decision parameters for an event and demonstrate the effectiveness of the model and algorithm in the financial case study, i.e., the detection of the bear market bottom. In Chapter 5, I present the multiple-time-point ME-EQPE model and *M*-Checkpoint algorithm to learn decision parameters for multi-events, as well as utilize the detection of the bear market bottom and the bull market top in sequence as an example to show the efficiency of the model and algorithm. In Chapter 6, I detail the *R*-Checkpoint algorithm for multi-events and conduct the same financial case study as in Chapter 5 to illustrate the performance among the logistic regression model, the *M*-Checkpoint algorithm, and the *R*-Checkpoint algorithm. In Chapter 7, I explain the MTSA-PE model to learn parameters with possibly incorporating any types of different constraints regardless of particular time points. In Chapter 8, I conduct two real case studies, i.e., the Decision-Guided Load Shedding (DGLS) system for optimal load shedding in electric power microgrids and the Decision-Guided Energy Investment (DGEI) framework for optimal power, heating, and cooling capacity investment, to demonstrate the efficiency and expressiveness of the MTSA-PE model. In Chapter 9, I conclude the dissertation and discuss the future work.

CHAPTER 2. RELATED WORK

In this chapter, I briefly discuss related materials on data models and query language, i.e., SQL Triggers, Aurora Query, AQuery, and DGQL, as well as the existing approaches of parameter learning, i.e., Logistic Regression Model and Mixed Integer Linear Programming.

2.1 Logistic Regression Model

In business and finance, economic researchers develop mathematical models and apply statistical methods to analyze economic data and their relationships to help users make a better decision. This study is called econometrics, which focuses on modeling the data relationship between explanatory (independent) and response (dependent) variables. In the real world, not all the response variables are numeric and continuous. In many cases, the responses may only take one of two possible answers, e.g., yes or no, success or failure, buy or sell, live or die, etc. Each outcome of the responses is assigned to a value 1 if the probability of the event happening is above 0.5 and 0 or otherwise. I assume that the dependent variable Y_i only takes the binary response in the i^{th} observation. A statistical model explains the binary variable Y_i with Prob[Y_i = 1] = p_i and Prob[Y_i = 0] = 1 – Prob[Y_i = 1] = 1 – p_i, where $0 \leq p_i \leq 1$. This mathematical model is called Binary Response Model (BRM) described by Dougherty, Heij, and others [14, 15, 16, 17, 18].

In this section, I focus on reviewing the logistic regression model, which forces its predicted probability bound between 0 and 1 because of its logistic curve. This model is non-linear, and its parameters can be computed by the Maximum Likelihood Estimation (MLE) [19]. Assume that there are N observations in the random sample of the binary response Y_i. If the probability of an event happening, p_i, is the same for all the observations, the probability distribution of the i^{th} observation in the sample can be written as ${p_i}^{Y_i} \cdot (1 - p_i)^{1-Y_i}$, where N is the total number of observations.

For example, if $Y_i = 1$, the probability p_i can be expressed as ${p_i}^1 \cdot (1 - p_i)^0$. This formula is also valid for the observation if $Y_i = 0$. Suppose that the observations are mutually independent. The corresponding likelihood function (LF) of the model is $\prod_{i=1}^{N} {p_i}^{Y_i} \cdot (1 - p_i)^{1-Y_i}$.

In the logistic regression model, the probability p_i is $\frac{e^{Z_i}}{1+e^{Z_i}}$, where $Z_i = \sum_{j=0}^{k} \beta_j X_{ij}$, β_j is a parametric coefficient of its X_{ij}, $X_{i0} = 1$, $1 \le i \le N$, $0 \le j \le k$, and k is the total number of independent variables.

The likelihood function LF($\boldsymbol{\beta}$)

$= \prod_{i=1}^{N} {p_i}^{Y_i} \cdot (1 - p_i)^{1-Y_i}$, where $\boldsymbol{\beta}$ is a vector of the parametric coefficient β_j.

$= \prod_{i=1}^{N} \frac{(e^{\sum_{j=0}^{k} \beta_j X_{il}})^{Y_i}}{1+e^{\sum_{j=0}^{k} \beta_j X_{ij}}}$.

By taking the natural logarithm of LF($\boldsymbol{\beta}$), the log-likelihood is given by ln(LF($\boldsymbol{\beta}$))

$= \sum_{i=1}^{N} \left(Y_i \cdot \sum_{j=0}^{k} \beta_j X_{ij}\right) - \sum_{i=1}^{N} ln(1 + e^{\sum_{j=0}^{k} \beta_j X_{ij}})$.

The goal of the logistic regression model is to find the optimal $\boldsymbol{\beta}$ such that the ln(LF($\boldsymbol{\beta}$)) can be maximized. If the ln(LF($\boldsymbol{\beta}$)) can be maximized, the LF($\boldsymbol{\beta}$) can be maximized as well. This can be achieved by the MLE in two steps. First, by taking the first-order partial derivative of each parametric coefficient β_j on ln(LF($\boldsymbol{\beta}$)) and setting each first derivative equal to 0, I have the k + 1 nonlinear equations that I can solve to find out the values of the k + 1 unknown β_j. However, solving the k + 1 nonlinear equations is not trivial at all. The values of unknown β_j have to be computed by an iterative approach. One of the most popular methods used is the Newton-Raphson method [46] that I do not discuss here. After solving the k + 1 equations, I can get a set of possible solutions that may be the critical points which either maximize or minimize LF($\boldsymbol{\beta}$). And then, I take the second-order partial derivative and evaluate the Hessian matrix [47] of it. If each value on the matrix diagonal is negative, I can conclude that the optimal $\boldsymbol{\beta}$ is found to maximize LF($\boldsymbol{\beta}$). After that, I can apply the model with the learned $\boldsymbol{\beta}$ on the test dataset to predict whether an event happens or not for decision-making. Unfortunately, the whole learning process takes the quadratic time complexity, $O(k^2N)$, to complete the computation to determine the optimal $\boldsymbol{\beta}$ and also lacks the consideration of integrating domain experts' knowledge into the learning process.

2.2 Mixed Integer Linear Programming

Proposed by Nemhauser and Wolsey [48], a Mixed Integer Linear Programming (MILP) model contains a set of decision variables, an objective function, and a set of constraints. Each nonnegative decision variable is an unknown quantity that is instantiated to optimize the objective function, that is, either maximized or minimized, and to satisfy all the given constraints at the same time. Some of the variables are integers, and some of them are real values. The objective function and each constraint are required to be a linear function of those decision variables. Suppose $x_1, x_2, \dots, x_n$ are a set of decision variables. The form of a MILP model is shown as follows:

Minimize or maximize $\qquad z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$

Subject to $\qquad a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n(\leq, =, \geq)b_1$

$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n(\leq, =, \geq)b_2$$

$$\cdots$$

$$a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n(\leq, =, \geq)b_m$$

$$x_j \geq 0, \forall j = 1,2, \dots n.$$

The values c_j, where $j = 1,2, \dots n$, are referred to as objective coefficients. Each constraint is either equal to, not less than, or not more than, a scalar value, i.e., b_i, where $i = 1,2, \dots m$. The values $b_1, b_2, \dots, b_m$ are the right-hand-side values of the constraints, and often represent amounts of available resources (especially for ≤ constraints) or requirements (especially for ≥ constraints). The a_{ij}-values denote how much of resource/requirement i is consumed/satisfied by decision j. Unfortunately, because the time complexity that the CPLEX optimizer requires to solve a MILP problem is incredibly high, i.e., $O(n \cdot 2^N)$, where n is the number of decision variables, and N is the size of the training data set, it takes a much longer time to solve the problem. Thus this approach is not scalable.

2.3 Procedural- and Relational- Based Extension: SQL Triggers

Proposed by Cochrane, Pirahesh, and Mattos [8], an SQL trigger is a small piece of codes which is composed of both procedural and relational statements. Each SQL-trigger creation statement has the three main components: the Event, the Condition, and the Action. I call this an *ECA*-rule. The event may be one of insert, delete, or update statements on a table. The optional condition specifies the trigger context which contains an SQL condition that must be satisfied before the trigger action can be executed. The trigger action is a specified PL/SQL [49] block that is fired automatically when the event has taken place and the optional condition is met. For example, in the financial scenario, the event is an insertion of tuples. Each newly inserted tuple is a set of index parameters, such as the percentage drop of the S&P 500 and the point decline of the Consumer Confidence Index, at a particular instant of time. The condition is the parametric constraint, for example, the percentage drop of the S&P 500 more than 20% (SPD) and the point decline of the Consumer Confidence Index more than 30 points (CCD), which the index parameters have to be satisfied. The outputs of the trigger action may be “Market Bottom Is Detected”, and “Buy Stock Is Recommended.” when the event has taken place and the condition has been satisfied. However, the semantics of the events of such a query responds to the events, such as insertions, deletions, and updates of tuples on tables, rather than the events on data which are time sequences, i.e., temporal, and are time-sensitive-oriented, e.g., the percentage changes of a financial indicator at an instant of time. For instance, the definition of the CREATE TRIGGER statement for the financial example is given in the Box 1.

Box 1.

```
CREATE TRIGGER marketBottomTrigger
       AFTER INSERT ON markeBottom
       REFERENCING NEW AS newRow
       FOR EACH ROW
       WHEN   (newRow.SPD  <  -20  AND  newRow.CCD  <  -30  AND  newRow.CG  <  0  AND
              newRow.ISM < 45 AND newRow.NCLD > 180)
       BEGIN
              DBMS_OUTPUT.PUT_LINE  (‘Market  Bottom  Is  Detected.  Buy  Stock  Is
              Recommended.’);
       END;
```

In addition to that, SQL triggers are not designed for supporting the parameter learning and cannot be formed without the assistance of the domain-knowledge-based and formal-learning-based approaches.

2.4 Procedural-Based Extension: Aurora Query

Proposed by Carney, Cetintemel, and others [11, 12], Aurora is a data-flow system that basically processes incoming streams. Its query model uses boxes to represent primitive operators and arrows to represent tuple-flows. There are some system-defined operators, including Filter, Drop, Map, Join, etc., for expressing the stream processing. Aurora also enables users to write their procedural programs to define their own functions. As the Aurora query is temporal-based, each incoming tuple is also timestamped. Using the provided GUI, users can connect the system- and user-defined boxes and the arrows together to form a continuous, monitoring query, which is a graph of a workflow diagram, to process the streams in the real time. The outputs of the graphical query are then sent to the external application, which provides recommended actions according to the outputs of the monitoring query. Although Aurora query can resolve the temporal incapability of SQL triggers for monitoring and recommendation, forming the query is still difficult without the assistance of the domain-knowledge-based and formal-learning-based approaches. Like the SQL triggers, Aurora query is not designed for the parameter learning as well.

2.5 Relational-Based Extension: AQuery

AQuery was proposed by Lerner and Shasha [13]. The data on which the query manipulates is arrables rather than tables. Each column of a conventional table is treated as an array, and the size of the arrays are the same. The table is called an arrable, standing for array-tables, which consists of a collection of named arrays. AQuery is an SQL-based language. One major extension of the query is a new clause called ASSUMING ORDER which defines the order of the arrables identified in the FROM clause. The clause ASSUMING ORDER is translated to a sort after the FROM clause is computed. Suppose that an arrable “marketBottom” has six arrays, including timestamp, SPD, CCD, CG, ISM, and NLCD. The below AQuery is continuously processing the incoming tuples of the “marketBottom” arrable and is monitoring the conditions suggested by the financial expert. And then the query informs the investors that the bear market bottom is detected and recommends the proper action if the conditions are satisfied. For example shown in the Box 2,

Box 2.

```
SELECT CASE
         WHEN SPD < -20 AND CCD < -30 AND CG < 0 AND ISM < 45 AND NLCD > 180
         THEN ‘Market Bottom Is Detected. Buy Stock Is Recommended.’
       END
FROM   marketBottom
ASSUMING ORDER timestamp;
```

Like the SQL trigger and Aurora query, AQuery cannot be formed without the assistance of the domain-knowledge-based and formal-learning-based approaches and also cannot support the parameter learning.

2.6 Decision-Guidance Query Language

Decision-Guidance Query Language (DGQL), developed by Brodsky and others [25, 26, 27, 28, 29, 30, 31, 32], combines the decision optimization capability of MILP in Operation Research (OR) and the data manipulation capability of traditional Structured Query Languages (SQL) in DBMS. It is a development and implementation tool that enables users to access and define all the different views, e.g., prediction views, decision optimization views, etc. It also provides a means for a system to derive learned knowledge and for users to issue queries, i.e., DGQL that incorporates the domain experts' knowledge and learned knowledge in its queries. The core syntax of a DGQL for expressing a decision-guidance problem is shown in the below steps:

STEP 1: Adds columns to a relation with decision variables by using the AUGMENT statement shown in the Box 3.

Box 3.

```
CREATE VIEW   <View_Name1> AS
       AUGMENT       SELECT <Attribute_Lists>
                     FROM <Table_Lists>
                     WHERE <Conditions>
       WITH   {REAL|INTEGER} <Augmented_Attribute> ≥ 0
       SUCH THAT Constraints
       FROM <Table_Lists>
       WHERE <Conditions>;
```

STEP 2: Define objective functions over the augmented table by using regular query language constructs shown in the Box 4.

Box 4.

```
CREATE VIEW <View_Name2> AS
       SELECT <Attribute_Lists>
       FROM <View_Name1>, <Other_Table_Lists>
       WHERE <Conditions>;
```

STEP 3: Use an INSTANTIATE statement to instantiate the variables inserted by the AUGMENT statements with the objective value optimized and with the specific constraints satisfied shown in the Box 5.

Box 5.

```
INSTANTIATE <View_Name1>
TO {MINIMIZE|MAXIMIZE} <Objective_Value>
FROM <View_Name2>;
```

Although DGQL has the capability of learning, prediction, and optimization, the scalability of solving optimization problems over multivariate time series by DGQL is not desirable. Unlike Aurora, AQuery, and SQL triggers, DGQL cannot be used for monitoring and recommendation as well.

2.7 Chapter Summary

SQL is a query language, which is designed for managing data on DBMS. However, SQL is a one-time, passive query that is not capable for continuously monitoring the events and recommending the actions. Even if some researchers have proposed several extensions, such as SQL triggers, Aurora Query, and AQuery to achieve this purpose, these extended query languages cannot be constructed without the assistance of domain-knowledge-based and formal-learning-based approaches, e.g., logistic regression models and mixed integer linear programming. Although these two approaches, domain-knowledge-based and formal-learning-based, can help users form the monitoring and recommendation query, they do not take advantage of each other to learn optimal decision parameters; thus the monitoring results and recommendation actions are always not desirable. Even though the proposed DGQL combines the strengths of both learning approaches to solve decision optimization problems, solving the problems over multivariate time series are not efficient and scalable at all because of a large number of binary variables being generated. In addition, DGQL is not designed for monitoring and recommendations, and SQL triggers, Aurora Query, and AQuery are not developed for parameter learning. Therefore, the web-mashup application service framework is proposed. The framework provides the mathematical models and learning algorithms, which unify both approaches, domain-knowledge-based and formal-learning-based, to learn decision parameters, and database models and query language, which support the quick implementations of the learning, monitoring, and recommendation services.

CHAPTER 3. WEB-MASHUP APPLICATION SERVICE FRAMEWORK, DATA MODEL, AND QUERY LANGUAGE FOR MULTIVARIATE TIME SERIES ANALYTICS

In this chapter, I describe the proposed web-mashup application service framework and its components, as well as the developed data models and query language for the MTSA services, i.e., querying, learning, monitoring, and recommendation.

3.1 Web-Mashup Application Service Framework

The proposed web-mashup application service framework is illustrated in Figure 2. It consists of six major components: Data Source Collector (DSC), Data Mashup Integrator (DMI), MTSA Data Model Definition and Query Language Interface (DMD-QLI), MTSA Compiler, MTSA Built-in Algorithms, and Web Application Designer (WAD).

Integrated with the mashup technology of Web 2.0 [50, 51], the DSC allows domain experts to directly interact with external data services and collect multivariate time series data from heterogeneous sources, including web data, XML documents, enterprise databases, excel/CSV files, WSDL-based web services information, and RSS feeds, around the globe. After multivariate time series are collected by the DSC, domain experts can operate the DMI that is a data integration processing unit to provide a concentric view and maintain a consistency of the collected data, which are then archived in the local databases of multivariate time series.

The DMD-QLI enables domain experts to use the extended relational database models with the time series and binary events, and SQL with the high-level MTSA constructs. These constructs include MTSA parametric model templates, querying, decision parameter learning, data monitoring and decision recommendations, as well as model evaluations. Using the DMD-QLI, domain experts first utilize the MTSA query language to create the extended relational database models for multivariate time series data, which are used in parametric model templates. Second, the experts can create and initiate learning events to the framework. If the events are a specific class (SC) of problems (read Chapter 3, 4, 5, and 6), the events are then sent to the MTSA built-in algorithms, such as Checkpoint, *M*-Checkpoint, and *R*-Checkpoint, from which the optimal decision parameters on the parametric model templates are learned over multivariate time series. Whereas, if the events are a general class (GC) of problems (read Chapter 7 and 8), those learning events are transformed into the IBM Optimization Programming Language (OPL) constructs [52, 53] by the MTSA compiler. The compiler then sends the OPL constructs to the external IBM optimization solver, i.e., the IBM ILOG CPLEX optimizer, to learn optimal decision parameters. Third, the experts develop and implement the data monitoring and decision recommendation services by using the traditional SQLs with the MONITOR keyword and the learned decision parameters to monitor the events. Finally, the experts develop a MTSA construct to evaluate model accuracy and quality. Note that those extended relational database models, learned decision parameters, and parametric model templates all are stored in the local database repositories for future use.

The WAD provides a user-friendly designer that offers domain experts a Java IDE environment and a JSF/JSP work space in which they can develop and implement a web portal and its associated web pages, which directly interact with all the MTSA services developed from the DMD-QLI. Domain experts can also use the web portal to centralize, administrate, and access all the developed MTSA services.

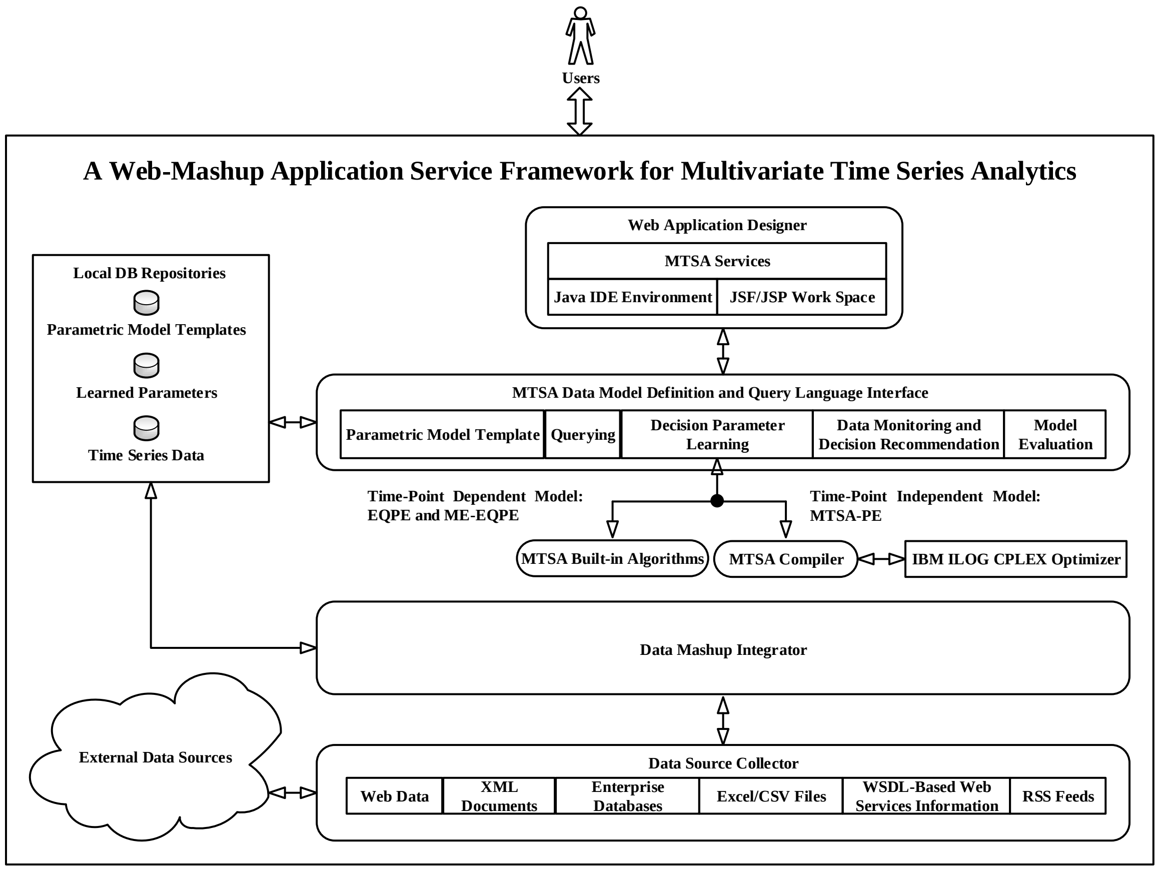

Users
A Web-Mashup Application Service Framework for Multivariate Time Series Analytics
Web Application Designer
MTSA Services
Java IDE Environment
JSF/JSP Work Space
Local DB Repositories
Parametric Model Templates
Learned Parameters
Time Series Data
MTSA Data Model Definition and Query Language Interface
Parametric Model Template
Querying
Decision Parameter Learning
Data Monitoring and Decision Recommendation
Model Evaluation
Time-Point Dependent Model: EQPE and ME-EQPE
Time-Point Independent Model: MTSA-PE
MTSA Built-in Algorithms
MTSA Compiler
IBM ILOG CPLEX Optimizer
Data Mashup Integrator
External Data Sources
Data Source Collector
Web Data
XML Documents
Enterprise Databases
Excel/CSV Files
WSDL-Based Web Services Information
RSS Feeds

Figure 2. A Web-Mashup Application Service Framework for Multivariate Time Series Analytics

3.2 Data Models

Before those MTSA services are developed from the DMD-QLI, a time-series data models for those services are needed to be created. The time-series (TS) data models are SQL-based extensions of the relational database model with specialized schemas.

- A time-series (TS) schema is of the form TSname(Tname:Ttype, Vname:Vtype). Ttype and Vtype are data types, where Ttype can be any system date/time format or integer-based time interval to show the time sequence of data, and Vtype is either Real or Integer. TSname is a schema name. Tname and Vname are the attribute names. All the names are chosen by users.
- A time-event (TE) schema is of the form TEname(Tname:Ttype, Ename:Binary), where Binary is the binary type corresponding to the domain {0,1}. TEname is a schema name, and Ename are attribute names, which are chosen by users as well.
- A decision-parameter (DP) schema is of the form DParameter(Tname: Ttype, Pname: Ptype, Vname:Vtype), where Pname is an optional attribute, and Ptype can be any system date/time format or integer-based period interval that is corresponding to a set of time intervals, Tname. DParameter is a schema name. Same as TS and TE schemas, a DP schema is named by users.
- A time series database schema is a set of relational schemas which may include TS, TE, and DP schemas.

A TS tuple over a schema TSname(Tname:Ttype, Vname:Vtype) is a relational tuple over that schema, i.e., a mapping m: $\{Tname, Vname\} \rightarrow Dom(Ttype) \times Dom(Vtype)$, such that $m(Tname) \in Dom(Ttype)$ and $m(Vname) \in Dom(Vtype)$.

A TE tuple over a schema TEname(Tname:Ttype, Ename:Binary) is a mapping m: {Tname, Ename} → Dom(Ttype) x Dom(Binary), such that m(Tname) ∈ Dom(Ttype) and m(Ename) ∈ Dom(Binary).

A DP tuple over a schema DParameter(Tname: Ttype, Pname: Ptype, Vname:Vtype) is a mapping m: {Tname, Pname, Vname} → Dom(Ttype) x Dom(Ptype) x Dom(Vtype), such that m(Tname) ∈ Dom(Ttype), m(Pname) ∈ Dom(Ptype), and m(Vname) ∈ Dom(Vtype).

Let us consider the financial example as an illustration. In the bear-market-bottom scenario, the service provider can use the querying service to create the base, derived, and related time-series tables as inputs and store them in the database. The base time-series tables are SP(T, Index), CG(T, Index), CC(T, Index), ISM(T, Index), and NLC(T, Index), where SP is the S&P 500 Index, CG is the Coppock Guide, CC is the Consumer Confidence Index, ISM is the ISM Manufacturing Survey, and NLC is the Negative Leadership Composite.

3.3 Querying Service

Using the base time series tables, the service provider can generate derived time series tables (if any) by using the traditional SQLs. In the financial case study, some derived time series tables, such as SPD(t) (Box 6) and CCD(t) (Box 7), where SPD is the S&P 500 percentage decline and CCD is the Consumer Confidence Index drop, are:

Box 6.

```
CREATE VIEW SPD AS (
       SELECT After.T, After.Average / Before.Average – 1 AS Value
       FROM   (SELECT SP1.T, AVG(SP2.Index) AS Average
              FROM SP SP1, SP SP2
              WHERE SP2.T <= SP1.T AND SP2.T >= SP1.T – 6 AND SP1.T – 6 >= 0
              GROUP BY SP1.T) After,

              (SELECT SP1.T, AVG(SP2.Index) AS Average
              FROM SP SP1, SP SP2
              WHERE SP2.T <= SP1.T - 150 AND SP2.T >= SP1.T – 156
              AND SP1.T – 156 >= 0
              GROUP BY SP1.T) Before
       WHERE After.T = Before.T);
```

Box 7.

```
CREATE VIEW CCD AS (
       SELECT After.T, (After.Average - Before.Average) AS Value
       FROM   (SELECT CC1.T, AVG(CC2.Index) AS Average
              FROM CC CC1, CC CC2
              WHERE CC2.T <= CC1.T AND CC2.T >= CC1.T – 6 AND CC1.T – 6 >= 0
              GROUP BY CC1.T) After,

              (SELECT CC1.T, AVG(CC2.Index) AS Average
              FROM CC CC1, CC CC2
              WHERE CC2.T <= CC1.T - 150 AND CC2.T >= CC1.T – 156
              AND CC1.T – 156 >= 0
              GROUP BY CC1.T) Before
       WHERE After.T = Before.T);
```

3.4 Monitoring and Recommendation Service

Using the monitoring and recommendation service over the new incoming data, the financial analyst can recommend the investors whether or not they should buy the stock. In the financial case study, the input parametric time series tables for monitoring are SPD(T, Value), CG(T, Index), CCD(T, Value), ISM(T, Index), and NLCD(T, Value), where NLCD is the Negative Leadership Composite “Distribution”. The monitoring and recommendation service can be expressed by a monitoring view and executed by the **MONITOR** command (Box 8).

Box 8.

```
CREATE VIEW MarketBottomTable AS
       (SELECT SPD.T, (CASE
                          WHEN   SPD.Value < PR1.p AND
                                 CG.Index <  PR2.p AND
                                 CCD.Value <  PR3.p AND
                                 ISM.Index <  PR4.p AND
                                 NLCD.Value >  PR5.p
                          THEN   '1'    ELSE   '0'
                   END) AS MB
       FROM SPD, CG, CCD, ISM, NLCD, Para1 PR1, Para2 PR2, Para3 PR3, Para4
       PR4, Para5 PR5
       WHERE SPD.T = CG.T AND CG.T = CCD.T AND CCD.T = ISM.T AND ISM.T = NLCD.T
       AND NLCD.T = PR1.T AND PR1.T = PR2.T AND PR2.T = PR3.T AND PR3.T = PR4.T
       AND PR4.T = PR5.T);

CREATE VIEW MB_Monitoring_Recommendation AS
       (SELECT MBT.T, (CASE
                          WHEN   MBT.MB = '1'
                          THEN   'Market  Bottom  Is  Detected.  Buy  Stock  Is
                                 Recommended.'
                     END) AS Action
       FROM MarketBottomTable MBT);

MONITOR MB_Monitoring_Recommendation;
```

, where Para1, Para2, Para3, Para4, and Para5 are a set of tables to store the decision parameters, e.g., -20, 0, -30, 45, and 180, respectively over a time horizon. If the parametric monitoring constraint in the "CASE WHEN" clauses is satisfied at the current time point t, the value of the attribute "MB" dedicates "1". The service then recommends the financial analysts to buy the index fund for the investors since the bear market bottom is detected.

3.5 Parameter Learning Service

As I discussed before, the expert's suggested parameters (-20. 0, -30, 45, 180) are not accurate enough to monitor the dynamic financial market at all time; thus, the parameter learning service should be adopted by expressing as follows:

STEP 1: Store base TS tables, e.g., SP, CG, CC, ISM, and NLC, in the database.

STEP 2: Define SQL views for derived TS tables, e.g, SPD, CCD, etc., shown in Section 3.3.

STEP 3: Create parameter tables (Box 9), which store optimal decision parameters over a time horizon.

Box 9.

```
CREATE TABLE Para1 (T INTEGER, p REAL);
CREATE TABLE Para2 (T INTEGER, p REAL);
CREATE TABLE Para3 (T INTEGER, p REAL);
CREATE TABLE Para4 (T INTEGER, p REAL);
CREATE TABLE Para5 (T INTEGER, p REAL);
```

STEP 4: Create a TS view for the time utility (Box 10).

Box 10.

```
CREATE VIEW Earning AS (
       SELECT SP1.T, ((Last.Index/SP1.Index - 1) * 100) AS Percent
       FROM SP SP1, (SELECT SP2.Index
                     FROM SP SP2
                     WHERE SP2.T >= ALL (SELECT SP3.T FROM SP500 SP3)) Last);
```

STEP 5: Create a learning event and then execute the event construct to learn the parameters (Box 11).

Box 11.

```
CREATE EVENT LearnMarketBottomParameter (
       SC_LEARN PR1, PR2, PR3, PR4, PR5
       FOR MAXIMIZE E.Percent
       WITH SPD.Value < PR1.p AND CG.Index < PR2.p AND CCD.Value < PR3.p AND
       ISM.Index < PR4.p AND NLCD.Value > PR5.p
       FROM SPD, CG, CCD, ISM, NLCD, Earning E, Para1 PR1, Para2 PR2,
       Para3 PR3, Para4 PR4, Para5 PR5
       WHERE SPD.T = CG.T AND CG.T = CCD.T AND CCD.T = ISM.T AND
       ISM.T = NLCD.T AND NLCD.T = E.T AND E.T = PR1.T AND PR1.T = PR2.T AND
       PR2.T = PR3.T AND PR3.T = PR4.T AND PR4.T = PR5.T;)

EXECUTE LearnMarketBottomParameter;
```

When the event “LearnMarketBottomParameter” is executed, the command SC_LEARN will call for the Checkpoint algorithm to solve the corresponding EQPE problem and will put its solution in the tables, i.e., Para1, Para2, Para3, Para4, and Para5, where all parameters are instantiated with optimal values.

3.6 Chapter Summary

The web-mashup application service framework for MTSA provides model definition, querying, parameter learning, model evaluation, data monitoring, decision recommendation, and web portal on events over multivariate time series. This framework is an integrated framework that includes the six components, i.e., DSC, DMI, DMD-QLI, MTSA complier, MTSA built-in algorithms, and WAD, to enable domain experts to develop those MTSA services on the Internet. More specifically, I develop the MTSA database model and query language for the services of learning, monitoring, and recommendation. Domain experts can use the proposed database models, such as time-series and time-event schemas, to store historical and projected time series and binary events from which optimal decision parameters are learned. Using the parameter learning service, domain experts can formulate a learning event and initial the event to the database, in which the built-in algorithms, such as the Checkpoint algorithm, are invoked to solve the corresponding EQPE problem and learn the optimal decision parameters, which are then stored in decision-parameter tables. Both the EQPE model and the Checkpoint algorithm are discussed in Chapter 4. After the parameters are learned and stored, domain experts can utilize the monitoring and recommendation service to monitor the incoming data streams and recommend actions when those streams satisfy the monitoring template.

CHAPTER 4. EXPERT QUERY PARAMETRIC ESTIMATION MODEL AND CHECKPOINT ALGORITHM

In this chapter, I discuss in detail the EQPE model and Checkpoint algorithm to learn decision parameters for a single event over multivariate time series and demonstrate the effectiveness of the model and algorithm in the financial case study, i.e., the detection of the bear market bottom.

4.1 Expert Query Parametric Estimation (EQPE) Model

In this section, I explain the methodologies used in the *Parameter Learning Service* in the MTSA service framework. More specifically, I review the mathematical formulations of the Expert Query Parametric Estimation (EQPE) problem and solution. I also use the financial example to explain them in detail.

The goal of an EQPE problem is to find optimal values of decision parameters that maximize an objective function over multivariate time series. For an EQPE problem being constructed, I need to define a set of mathematical notations and a model for it. I assume that the time domain **T** is represented by a set of natural numbers: **T** = **N**, and that I am also given a vector of n real-valued parameter variables (p_1, p_2, …, p_n).

Definition 1. *Time Series*: A time series S is a function S: $T \rightarrow R$, where T is the time domain, and R is the set of real numbers.

Definition 2. *Parametric Monitoring Constraint*: A parametric monitoring constraint C($S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n$) is a symbolic expression in terms of $S_1(t), S_2(t), \ldots,$

$S_k(t), p_1, p_2, \ldots, p_n$, where $S_1(t), S_2(t), \ldots, S_k(t)$ are time series, $t \in$ T is a time point, and $(p_1, p_2, \ldots, p_n)$ is a vector of parameters.

I assume a constraint C written in a language that has the truth-value interpretation I: R^k x $R^n \rightarrow$ {True, False}, i.e., I(C($S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n$)) = True if and only if the constraint C is satisfied at the time point $t \in$ T and with the parameters $(p_1, p_2, \ldots, p_n) \in R^n$. In this model, I focus on conjunctions of inequality constraints: C($S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n$) = $\wedge_i$ ($S_i(t)$ *op* p_j), where *op* $\in \{<, \leq, =, \geq, >\}$.

Definition 3. *Time Utility Function*: A time utility function U is a function U: $T \rightarrow R$.

Definition 4. *Objective Function*: Given a time utility function U: $T \rightarrow R$ and a parametric constraint C, an objective function O is a function O: $R^n \rightarrow R$, which maps a vector of n parameters on R^n to a real value R, defined as follows. For $(p_1, p_2, \ldots, p_n) \in R^n$, O($p_1, p_2, \ldots, p_n$) $\overset{\text{def}}{=}$ U(t), where U is the utility function, and $t \in$ T is the earliest time point that satisfies C, i.e.,

(1) $S_1(t)\ op_1\ p_1 \wedge S_2(t)\ op_2\ p_2 \wedge \ldots \wedge S_n(t)\ op_n\ p_n$ is satisfied, and

(2) There does not exist $0 \leq t' < t$, such that $S_1(t')\ op_1\ p_1 \wedge S_2(t')\ op_2\ p_2 \wedge \ldots \wedge S_n(t')\ op_n\ p_n$ is satisfied.

Definition 5. *Expert Query Parametric Estimation (EQPE) Problem*: An EQPE problem is a tuple <$\dot{\boldsymbol{S}}$, $\dot{\boldsymbol{P}}$, $\boldsymbol{C}$, $\boldsymbol{U}$>, where $\dot{\boldsymbol{S}} = \{S_1, S_2, \ldots, S_k\}$ is a set of k time series, $\dot{\boldsymbol{P}} = \{p_1, p_2, \ldots, p_n\}$ is a set of n real-value parameter variables, $\boldsymbol{C}$ is a parametric constraint in $\dot{\boldsymbol{S}}$ and $\dot{\boldsymbol{P}}$, and $\boldsymbol{U}$ is a time utility function.

Intuitively, a solution to an EQPE problem is an instantiation of values into the vector $\dot{\boldsymbol{P}}$ of n real-value parameters that maximizes the objective $\boldsymbol{O}$.

Definition 6. *Expert Query Parametric Estimation (EQPE) Solution*: A solution to the EQPE problem $<\dot{\boldsymbol{S}}, \dot{\boldsymbol{P}}, \boldsymbol{C}, \boldsymbol{U}>$ is argmax $\boldsymbol{O}(p_1, p_2, \ldots, p_n)$, i.e., the optimal values of parameters, $p_1, p_2, \ldots, p_n$, that maximize $\boldsymbol{O}$, where $\boldsymbol{O}$ is the objective function corresponding to $\boldsymbol{U}$.

The base time series in my financial example, i.e., detecting the bear market bottom, are shown in Table 1. I suppose that the first starting date in any time-series data set is **t = 0**.

Table 1. Base Time-Series Data

Base Time Series S	**Abbreviation**
S&P 500	SP(t)
Coppock Guide	CG(t)
Consumer Confidence	CC(t)
ISM Manufacturing Survey	ISM(t)
Negative Leadership Composite	NLC(t)

Note that some base time series are the direct inputs, whereas some are used to derive another set of time series. For instance, the derived time series in my case study are shown in Table 2.

Table 2. Derived Time-Series Data

Derived Time Series S	**Abbreviation**
Percentage decline in SP(t) at the time point t	SPD(t)
Points drop in CC(t) at the time point t	CCD(t)
Number of consecutive days in Bear Market "DISTRIBUTIOIN" of NLC(t) at and before the time point t	NLCD(t)
Time Utility Earning at the time point t, i.e., the index fund is bought at t and sold at t_s, where t_s is the last day of the learning data set	Earning(t)

The decision parameters used in the case study are defined in Table 3.

Table 3. Decision Parameters

Parameter	**Interpretation**
p_1	Test if SPD(t) is less than p_1 at ***t***.
p_2	Test if CG(t) is less than p_2 at ***t***.
p_3	Test if CCD(t) is less than p_3 at ***t***.
p_4	Test if ISM(t) is less than p_4 at ***t***.
p_5	Test if NLCD(t) is greater than p_5 at ***t***.

Let us consider the following constraint **C** as an illustration:

$$C(SPD(t), CG(t), CCD(t), ISM(t), NLCD(t), p_1, p_2, p_3, p_4, p_5)$$

$$= SPD(t) < p_1 \wedge CG(t) < p_2 \wedge CCD(t) < p_3 \wedge ISM(t) < p_4 \wedge NLCD(t) > p_5$$

It means that the parametric monitoring constraint **C** is satisfied, i.e., its interpretation is ***True***, if the above inequalities with the decision parameters are satisfied at the time point ***t***. The interpretation also indicates that the monitoring event occurs.

I assume that the investor buys the S&P 500 index fund at the decision variable time t and sell it at the given t_S, which is the last day of the given training data set. The earning function $SP(t_S)/ SP(t) – 1 \in \mathbf{R}$ is the utility, which is maximized by choosing the optimal value $t \in \mathbf{T}$, where $SP(t_S)$ and SP(t) are the sell and buy value of the S&P 500 index fund at the time t_S and t respectively.

The EQPE problem and solution for the financial example can be constructed by putting the considered time series, parameters, constraints, and functions to the definitions shown in Table 4.

Table 4. EQPE Problem and Solution Formulation for the S&P 500 Index Fund

Problem and Solution
Problem:
$<\dot{\boldsymbol{S}}, \dot{\boldsymbol{P}}, \boldsymbol{C}, \boldsymbol{U}>$, where
$\dot{\boldsymbol{S}}$ = {SPD, CG, CCD, ISM, NLCD}
$\dot{\boldsymbol{P}} = \{p_1, p_2, p_3, p_4, p_5\}$
$\boldsymbol{C} = \mathrm{SPD}(t) < p_1 \wedge \mathrm{CG}(t) < p_2 \wedge \mathrm{CCD}(t) < p_3 \wedge \mathrm{ISM}(t) < p_4 \wedge \mathrm{NLCD}(t) > p_5$
$\boldsymbol{U} = \mathrm{SP}(t_s)/\mathrm{SP}(t) - 1$
Solution:
argmax $\boldsymbol{O}(p_1, p_2, p_3, p_4, p_5) \overset{\text{def}}{=} \boldsymbol{U}(t)$

The values of the optimal decision parameters can be determined by using the learning algorithm, Checkpoint. Before explaining the Checkpoint algorithm in detail, I first review the concept of *Dominance*.

Definition 7. *Dominance* $\succ$: Given an EQPE problem $<\dot{\boldsymbol{S}}, \dot{\boldsymbol{P}}, \boldsymbol{C}, \boldsymbol{U}>$ and any two time points $t, t' \in \mathrm{T}$, I say that t' dominates t, denoted by $t' \succ t$, if the following conditions are satisfied:

(1) $0 \leq t' < t$, and

(2) $\forall (p_1, p_2, \ldots, p_n) \in R^n$, $\mathrm{C}(S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n) \rightarrow \mathrm{C}(S_1(t'), S_2(t'), \ldots, S_k(t'), p_1, p_2, \ldots, p_n)$.

Intuitively, t' dominates t if for any selection of parametric values, the query constraint satisfaction at t implies the satisfaction at t'. Clearly, the dominated time points should be discarded when the optimal time point is being determined. I formally claim that:

Claim 1 - Given the conjunctions of inequality constraints, $S_1(t)\ op_1\ p_1 \wedge S_2(t)\ op_2\ p_2 \wedge \ldots \wedge S_k(t)\ op_k\ p_k$ and the two time points t', t such that $0 \leq t' < t$, $t' \succ t$ if and only if $S_1(t')\ op_1\ S_1(t) \wedge S_2(t')\ op_2\ S_2(t) \wedge \ldots \wedge S_k(t')\ op_k\ S_k(t)$.

For example, suppose there are three time series S_1, S_2, S_3 and three decision parameters p_1, p_2, p_3. And the constraints are $C(S_1(t), S_2(t), S_3(t), p_1, p_2, p_3) = S_1(t) \geq p_1 \wedge S_2(t) \geq p_2 \wedge S_3(t) \leq p_3$. Also assume the values for S_1, S_2, *and* S_3 at the time point t_1, t_2, and t_3 respectively in Table 5.

Table 5. Values of S1, S2, S3, and U at the Time Point t_1, t_2, and t_3

Time	S_1	S_2	S_3	U
t_1	13	27	3	10
t_2	25	15	2	200
t_3	10	20	5	150
t_4	20	10	15	250

In this case, the time point t_3 is dominated because there is a time point t_1 that make the inequality, $S_1(t_1) \geq S_1(t_3) \wedge S_2(t_1) \geq S_2(t_3) \wedge S_3(t_1) \leq S_3(t_3)$, equal to true.

On the contrary, for all $t' < t$, if $S_1(t') \neg op_1\ S_1(t) \vee S_2(t') \neg op_2\ S_2(t) \vee \ldots \vee S_n(t') \neg op_n\ S_n(t)$ is satisfied, t is not dominated by t' denoted by $t' \nsucc t$. Let us consider the same example above. Because $S_1(t_1) < S_1(t_2) \vee S_3(t_1) > S_3(t_2)$, t_2 is not dominated.

4.2 Checkpoint Algorithm

Conceptually, I can search a particular set of parameters $\{p_1, p_2, \ldots, p_n\}$ which is at the earliest time point t that is not dominated by any t' such that the value of the objective function ***O*** is maximal among all the instantiations of values into parameters. However, the problem of this approach is that for every single parameter set at *t* in a learning data set, the parameter set at *t* has to be examined with all the previous sets of parameters at *t'* for checking the non-dominance before the optimal solution can be found. In fact, due to the quadratic nature, the conceptual approach is time consuming and expensive particularly if the size of the learning data set is significantly large. Instead, the Checkpoint algorithm uses the KD-tree data structure and searching algorithm [54, 55, 56] to evaluate whether a time point *t* is dominated based on the **Claim 1** for checking the non-dominance. The pseudo code of the algorithm is shown in the Box 12.

Box 12.

```
Input: <Ṡ, Ṗ, C, U>

Output: p[1…k] is an array of the optimal parameters that maximize the objective.

Data Structures:
1. N is the size of the learning data set.
2. T_kd is a KD tree that stores the parameter vectors that are not dominated so
   far.
3. MaxT is the time point that gives the maximal U so far, denoted by MaxU.

Processing:
STEP 1: T_kd := <S_1(0), S_2(0), …, S_k(0)>; MaxT := 0; MaxU := U(0);

STEP 2: FOR t := 1 TO N - 1 DO {
            Non-Dominance Test: Query the T_kd to find if there exists a point
            (p_1, p_2, …, p_k) in the T_kd, which is in the range [S_1(t), ∞) x
            [S_2(t), ∞) x … x [S_k(t), ∞).
            IF (NOT AND t is not dominated) THEN
                Add <S_1(t), S_2(t), …, S_k(t)> to T_kd;
                IF (U(t) > MaxU) THEN
                    MaxT := t;
                    MaxU := U(t);
                ENDIF
            ENDIF
        }

STEP 3: FOR i := 1 TO k DO {
            p[i] := S_i(MaxT);
        }
STEP 4: RETURN p[1…k];
```

Clearly, the first time point is not dominated because there is no time point preceding it. Therefore, $<S_1(0), S_2(0), \ldots, S_k(0)>$ can be added to T_{kd}. *0* and U(*0*) can be assigned to MaxT and MaxU respectively.

Using the Checkpoint algorithm step by step for the problem shown in Table 5, I can search through a particular set of parameters $\{p_1, p_2, p_3\}$ which is at the earliest time point *t* that is not dominated by any *t'* such that the value of the utility function ***U*** is maximal. In the **STEP 1**, the $<S_1(t_1), S_2(t_1), S_3(t_1)>$ is added to the T_{kd} since it is the first time point. Then t_1 and U(t_1) are assigned to MaxT and MaxU respectively. In the **STEP 2**, t_2 is not dominated because $S_1(t_1) < S_1(t_2) \wedge S_2(t_1) > S_2(t_2) \wedge S_3(t_1) > S_3(t_2)$ does not satisfy the **Claim 1**. However, t_3 is dominated because $S_1(t_1) > S_1(t_3) \wedge S_2(t_1) > S_2(t_3) \wedge S_3(t_1) < S_3(t_3)$ does satisfy the **Claim 1**. $<S_1(t_2), S_2(t_2), S_3(t_2)>$ is added to the T_{kd} because t_2 is not dominated and U(t_2) > U(t_1). Thus t_2 and U(t_2) are assigned to MaxT and MaxU respectively. In the **STEP 3**, ***p***[1] := S_1(*MaxT*), ***p***[2] := S_2(*MaxT*), and ***p***[3] := S_3(*MaxT*) in the for-loop statement. In the **STEP 4**, the algorithm returns 25, 15, and 2.

Theorem 1: For N parameter vectors in the data set, the Checkpoint algorithm correctly computes an EQPE solution, i.e., *argmax* $\mathbf{O}(p_1, p_2, \ldots, p_n)$, where ***O*** is the objective function of the EQPE problem, with the complexity $O(kN\log N)$.

4.3 Experimental Case Study

Using the Checkpoint algorithm, I can obtain the optimal decision parameters and the maximal earning from the training data set for the financial problem shown in Table 6. The time complexity of the MLE for the logistic regression model is $O(k^2N)$, where *k* is the number of input parametric time series, and N is the size of the learning data set. For the Checkpoint algorithm, the complexity is $O(kNlogN)$. Using the decision parameters from the financial expert (i.e., -20%, 0, -30, 45, 180 days), the logistic regression model, and the Checkpoint algorithm, the "Best Buy" opportunities in stock and their earnings are shown in Table 7. Note that the Checkpoint algorithm considerably outperforms both the financial expert's criteria and the logistic regression model.

Table 6. Optimal Decision Parameters and Maximum Earning (%) from the Learning Data Set[1]

p_1	p_2	p_3	p_4	p_5	$O(p_1,p_2,p_3,p_4,p_5)$
-29.02	-20.01	-26.61	49	70	53.37

Table 7. Investors' Earning of the S&P 500 Index Fund from the Test Data Set[2]

Decision Approach	Best Buy	S&P 500 Index	Earning%
Financial Expert's Criteria	10/09/08	909.92	1.03
Logistic Regression Model	11/26/08	887.68	3.56
Checkpoint Algorithm with Financial Expert's Template	03/10/09	719.6	27.8

4.4 Chapter Summary

The parameter learning services combine the strengths of both domain-knowledge-based and formal-learning-based approaches for maximizing utility on events over multivariate time series. This service includes a mathematical model, i.e., EQPE, and a learning algorithm, Checkpoint, to solve Expert Query Parametric Estimation problems. The EQPE model is a well-defined model that captures domain experts' knowledge in expression of multivariate time series, decision parameters, a set of

[1] The learning data set is from 06/01/1997 to 06/30/2005.
[2] The test data set is from 07/01/2005 to 06/30/2009 that is the sell date of the fund with the value of 919.32.

parametric constraints, a time utility, and an objective function. Each multivariate time series is an input parametric time series for an event, e.g., the bear market bottom or the bull market top. Each decision parameter is instantiated from its input parametric time series to learn the optimal value that satisfies the given parametric constraint and maximizes its time utility collectively. The time utility is a function of the decision time point that the objective function is dependent upon from which the parameters are learned.

The Checkpoint algorithm also combines the strengths of both domain experts' knowledge in terms of a parametric constraint and formal-learning-based methodology by using the regression approach. The goal of the Checkpoint algorithm is to learn the decision parameters that maximize the objective function, e.g., the earning of the financial example, over multivariate time series. This algorithm guarantees an optimality of the learned decision parameters in their event, i.e., a true optimal decision time point, e.g., $t_{purchased}$, of the S&P 500 Index Fund, during the computations. To demonstrate the effectiveness of the algorithm, I compare my method with the formal-learning-based approach, i.e., the logistic regression methodology, and the domain-knowledge-based approach, e.g., the financial experts' criteria, in the financial domain. Using the learned decision parameters, I show that the Checkpoint algorithm is more effective and guarantees the satisfactory forecasting results that are superior to those from the logistic regression methodology and the financial experts' criteria.

CHAPTER 5. MULTI-EVENT EXPERT QUERY PARAMETRIC ESTIMATION MODEL AND MULTIDIMENSIONAL *M*-CHECKPOINT ALGORITHM

In this chapter, I present the ME-EQPE model and the *M*-Checkpoint algorithm to learn multiple sets of optimal decision parameters over multivariate time series for multi-events, as well as utilize the detection of the bear market bottom and the bull market top in sequence as an example to show the effectiveness of the model and algorithm.

5.1 Multi-Event Expert Query Parametric Estimation (ME-EQPE) Model

As discussed in Chapter 1, the proposed EQPE model and the Checkpoint algorithm are only able to learn one set of optimal decision parameters for one particular event at a single time point, whereas there are many real-world scenarios that the parameter learning is at multiple time points in sequence. To address this problem, I develop an extended model, Multi-Event Expert Query Parametric Estimation (ME-EQPE), and an algorithm, Multidimensional *M*-Checkpoint, which are discussed in this chapter.

Consider a sequence of m events that I would like to detect. Each event *i* has a vector of real parameters $\vec{P_i}$ that will be learned from the training data set of its multivariate-parametric-time-series vector $\vec{S_i}(t)$. After the learning process, the vector of the learned parameters $\vec{P_i}$ is used to detect the occurrence of the event *i*.

Before explaining the new learning algorithm, I first extend a set of mathematical notations and a model for the ME-EQPE problem. I still assume that the time domain **T** is represented by the set of natural numbers: **T = N**. I am also given a set of time sequences ($\vec{S}_1(t), \vec{S}_2(t), \ldots, \vec{S}_m(t)$), and the learning algorithm generates a set of m vectors of real parameters ($\vec{P}_1, \vec{P}_2, \ldots, \vec{P}_m$) where $\vec{S}_i(t) = (S_1(t), S_2(t), \ldots, S_{k_i}(t)) \in R^{k_i}$ and $\vec{P}_i = (p_1, p_2, \ldots, p_{n_i}) \in R^{n_i}$ for $1 \le i \le m$, k_i , $n_i \in Z^+$, k_i is the number of input parametric time series in $\vec{S}_i(t)$ for the event i, n_i is the number of parameters in $\vec{P}_i$ for the event i, and m is the number of time points that are corresponding to their events being monitored.

Definition 8. *Parametric Monitoring Constraint*: A parametric monitoring constraint $C_i(\vec{S}_i(t),\ \vec{P}_i)$ is a symbolic expression in terms of $\vec{S}_i(t)$ and $\vec{P}_i$, where $\vec{S}_i(t)$ is the i^{th} time sequence vector at the time t, and $\vec{P}_i$ is the i^{th} decision parameter vector for $1 \le i \le m$.

I assume a constraint C_i written in a language that has the truth-value interpretation $I: R^{k_i} x R^{n_i} \rightarrow \{True, False\}$, i.e., $I(C_i(\vec{S}_i(t),\ \vec{P}_i)) = True$ if and only if the constraint C_i is satisfied at the time point $t \in$ T with the vectors, time sequence $\vec{S}_i(t) = (S_1(t), S_2(t), \ldots, S_{k_i}(t)) \in R^{k_i}$ and parameter $\vec{P}_i = (p_1, p_2, \ldots, p_{n_i}) \in R^{n_i}$. Again I focus on a combinational conjunction of inequality constraints: $\mathrm{C_i}(\vec{\mathrm{S}}_\mathrm{i}(\mathrm{t}), \vec{\mathrm{P}}_\mathrm{i}) = \mathrm{S_1(t) op\ p_1} \wedge \mathrm{S_2(t) op\ p_2} \wedge \ldots \wedge \mathrm{S_{k_i}(t) op\ p_{n_i}}$, where *op* ∈ {<, ≤, =, ≥, >}.

Definition 9. *Time Utility Function*: A time utility function U is a function U: $T^m \rightarrow R$, where m is the number of time points.

Definition 10. *Objective Function*: Given a time utility function U: $T^m \rightarrow R$ and m parametric constraints (C_1, C_2, …, C_m), an objective function O is a function

$O: R^{n_1} x R^{n_2}, \dots, R^{n_m} \rightarrow R$, which maps a set of vectors of parameters on $R^{n_1} x R^{n_2}, \dots, R^{n_m}$ to a real value *R*, defined as follows: For a set of $(\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m)$, $O(\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m) \stackrel{\text{def}}{=} U(t_1, t_2, \dots, t_m)$, where U is the utility function, and $(t_1, t_2, \dots, t_m)$ are the earliest time points that satisfies their corresponding parametric constraints, i.e.,

(1) $0 \le t_1 < t_2 < \dots < t_m$

(2) $C_1(\vec{S}_1(t_1), \vec{P}_1), C_2(\vec{S}_2(t_2), \vec{P}_2), \dots, C_m(\vec{S}_m(t_m), \vec{P}_m)$ are satisfied, and

(3) There does not exist $(0 \le t'_1 < t'_2 < \cdots < t'_m)$ such that

(a) $t'_1 \le t_1, t'_2 \le t_2, \dots, t'_m \le t_m$

(b) for some i = 1, 2, …, m, $t'_i < t_i$, and

(c) $C_1(\vec{S}_1(t'_1), \vec{P}_1), C_2(\vec{S}_2(t'_2), \vec{P}_2), \dots, C_m(\vec{S}_m(t'_m), \vec{P}_m)$ are satisfied.

Definition 11. *Multi-Event Expert Query Parametric Estimation (ME-EQPE) Problem*: A ME-EQPE problem is a tuple <$\dot{\boldsymbol{S}}$, $\dot{\boldsymbol{P}}$, ***C***, ***U***>, where $\dot{\boldsymbol{S}} = \{\vec{S}_1(t), \vec{S}_2(t), \dots, \vec{S}_m(t)\}$ is a set of m vectors of time sequences, $\dot{\boldsymbol{P}} = \{\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m\}$ is a set of m decision parameter vectors, $\boldsymbol{C} = (C_1, C_2, \dots, C_m)$ is a set of m parametric constraints in $\dot{\boldsymbol{S}}$ and $\dot{\boldsymbol{P}}$, and ***U*** is a time utility function.

Intuitively, a solution to a ME-EQPE problem is an instantiation of values into all the parameter vectors, $\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m$, that maximize the objective.

Definition 12. *Multi-Event Expert Query Parametric Estimation (ME-EQPE) Solution*: A solution to the ME-EQPE problem <$\dot{\boldsymbol{S}}$, $\dot{\boldsymbol{P}}$, ***C***, ***U***> is *argmax* $\boldsymbol{O}(\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m)$, i.e., the estimated values of all the parameter vectors, $\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m$, that maximize ***O***, where ***O*** is the objective function corresponding to ***U***.

Definition 13. *Time Length (T_L)*: T_L is the length of the time duration in terms of the number of business days that the events among them may occur.

In the financial example, i.e., the “Best Buy” and “Best Sell” opportunities in the investment, the base time series, the derived time series, and the decision parameters are shown in Table 8, Table 9, and Table 10 respectively.

Table 8. Base Time-Series Data

Base Time Series S	Abbreviation
S&P 500	SP(t)
Coppock Guide	CG(t)
Consumer Confidence	CC(t)
ISM Manufacturing Survey	ISM(t)
Negative Leadership Composite “Distribution”	NLC(t)
Negative Leadership Composite “Selling Vacuum”	NLCSV(t)

Table 9. Derived Time-Series Data

Derived Time Series S	Abbreviation
Percentage decline in SP(t) at the time point t	SPD(t)
Percentage increase in SP(t) at the time point t	SPI(t)
Points drop in CC(t) at the time point t	CCD(t)
Points increase in CC(t) at the time point t	CCI(t)
Number of consecutive days in Bear Market “DISTRIBUTIOIN” of NLC(t) at and before the time point t	NLCD(t)
Time Utility Earning at the time points t_1 and t_2, i.e., the index fund is bought at t_1 and sold at t_2 of the learning data set	Earning(t_1, t_2)

Table 10. Decision Parameters

Parameter	Interpretation
p_1	Test if SPD(t) is less than p_1 at $\boldsymbol{t_1}$.
p_2	Test if CG(t) is less than p_2 at $\boldsymbol{t_1}$.
p_3	Test if CCD(t) is less than p_3 at $\boldsymbol{t_1}$.
p_4	Test if ISM(t) is less than p_4 at $\boldsymbol{t_1}$.
p_5	Test if NLCD(t) is greater than p_5 at $\boldsymbol{t_1}$.
q_1	Test if SPI(t) is greater than or equal to q_1 at $\boldsymbol{t_2}$.
q_2	Test if CG(t) is greater than or equal q_2 at $\boldsymbol{t_2}$.
q_3	Test if CCI(t) is greater than or equal q_3 at $\boldsymbol{t_2}$.
q_4	Test if ISM(t) is greater than or equal q_4 at $\boldsymbol{t_2}$.
q_5	Test if NLCSV(t) is greater than q_5 at $\boldsymbol{t_2}$.

The constraints C_1 and C_2 are illustrated as follows:

$C_1(\text{SPD}(t_1), \text{CG}(t_1), \text{CCD}(t_1), \text{ISM}(t_1), \text{NLCD}(t_1), p_1, p_2, p_3, p_4, p_5)$

$= \text{SPD}(t_1) < p_1 \wedge \text{CG}(t_1) < p_2 \wedge \text{CCD}(t_1) < p_3 \wedge \text{ISM}(t_1) < p_4 \wedge \text{NLCD}(t_1) > p_5$

$C_2(\text{SPI}(t_2), \text{CG}(t_2), \text{CCI}(t_2), \text{ISM}(t_2), \text{NLCSV}(t_2), q_1, q_2, q_3, q_4, q_5)$

$= \text{SPI}(t_2) \geq q_1 \wedge \text{CG}(t_2) \geq q_2 \wedge \text{CCI}(t_2) \geq q_3 \wedge \text{ISM}(t_2) \geq q_4 \wedge \text{NLCSV}(t_2) > q_5$

The earning function $\text{U}(t_1, t_2) = \text{SP}(t_2)/\ \text{SP}(t_1) - 1 \in \mathbf{R}$ is the utility, which is maximized by choosing the optimal value t_1 and $t_2 \in \mathbf{T}$, where $\text{SP}(t_2)$ and $\text{SP}(t_1)$ are the sell and buy value of the S&P 500 index fund at the time t_2 and t_1 respectively. The ME-EQPE problem and solution for the financial example can be constructed by putting the considered time sequence vectors, parameter vectors, constraints, and functions to the definitions shown in Table 11.

Table 11. ME-EQPE Problem and Solution Formulation for the S&P 500 Index Fund

Problem and Solution
Problem: $<\dot{\boldsymbol{S}}, \dot{\boldsymbol{P}}, \boldsymbol{C}, \boldsymbol{U}>$, where $\dot{\boldsymbol{S}} = \{\vec{S}_1(t_1), \vec{S}_2(t_2)\}$, where $\vec{S}_1(t_1) = (\text{SPD}(t_1), \text{CG}(t_1), \text{CCD}(t_1), \text{ISM}(t_1), \text{NLCD}(t_1))$ and $\vec{S}_2(t_2) = (\text{SPI}(t_2), \text{CG}(t_2), \text{CCI}(t_2), \text{ISM}(t_2), \text{NLCSV}(t_2))$ $\dot{\boldsymbol{P}} = \{\vec{P}_1, \vec{P}_2\}$, where $\vec{P}_1 = (p_1, p_2, p_3, p_4, p_5)$ and $\vec{P}_2 = (q_1, q_2, q_3, q_4, q_5)$ $\boldsymbol{C} = \{C_1, C_2\}$, where $C_1 = \text{SPD}(t_1) < p_1 \wedge \text{CG}(t_1) < p_2 \wedge \text{CCD}(t_1) < p_3 \wedge \text{ISM}(t_1) < p_4 \wedge \text{NLCD}(t_1) > p_5$ and $C_2 =$ $\text{SPI}(t_2) \geq q_1 \wedge \text{CG}(t_2) \geq q_2 \wedge \text{CCI}(t_2) \geq q_3 \wedge \text{ISM}(t_2) \geq q_4 \wedge \text{NLCSV}(t_2) > q_5$ $\boldsymbol{U} = \text{SP}(t_2)/\text{SP}(t_1) - 1$ *Solution*: $argmax\ \boldsymbol{O}(\vec{P}_1, \vec{P}_2) \stackrel{\text{def}}{=} \boldsymbol{U}(t_1, t_2)$

The values of the optimal decision parameters can be determined by using the learning algorithm, *M*-Checkpoint.

5.2 Multidimensional *M*-Checkpoint Algorithm

The extended version, *M*-Checkpoint algorithm, keeps using the KD-tree data structure and searching techniques to evaluate whether or not the time point t_1 of the first event is dominated. The pseudo code of the *M*-Checkpoint algorithm is shown in the Box 13.

Box 13.

```
Input: <$\dot{S}$, $\dot{P}$, $C$, $U$>

Output: $p$[i][j] is a two-dimensional array of the decision parameter vectors
that maximize the objective, where 1 ≤ i ≤ $m$, 1 ≤ j ≤ $n_i$, and $m$, $n_i$ ∈ $Z^+$.

Data Structures:
1. N is the size of the learning data set.
2. $T_{kd}$ is a KD tree that stores the parameter vectors that are not dominated so
   far for the 1st event.
3. isDominated[$t_1$] is a boolean array to signal that a time point $t_1$ of the 1st
   event is dominated, i.e., isDominated[$t_1$] := true, by at least one previous
   time point, or else isDominated[$t_1$] := false.
4. MaxT[i] is the array of time points that gives the maximal $U$ so far,
   denoted by MaxU, where 1 ≤ i ≤ $m$ and MaxT[1] < MaxT[2] < …< MaxT[$m$].

Initialization: FOR i := 1 TO $m$ DO {
                      $t_i$ := MaxT[i] := 0;
                }

                MaxU := U(0, 0, …, 0) := 0;
                $T_L$ := 365; // Assume that the event occurrences among them are
                within 365 business days.

Processing:
STEP 1: $T_{kd}$ := $\vec{S}_1(0)$; isDominated[0] = false;

STEP 2: FOR $t_1$ := 1 TO N - 1 DO {
                Non-Dominance Test: Query the $T_{kd}$ to find if there exists a point
                ($p_1$, $p_2$, …, $p_{k_1}$) in the $T_{kd}$, which is in the range $\prod_{j=1}^{k_1}[S_{j_1}(t_1),\infty)$.

                IF (NOT AND $t_1$ is not dominated) THEN
                   Add $\vec{S}_1(t_1)$ to $T_{kd}$;
                   isDominated[$t_1$] = false;
                ELSE
                   isDominated[$t_1$] = true;
                ENDIF
        }
```

```
STEP 3: FOR (t := N - 1; $t_m$ ≥ 0; $t_m$ := $t_m$ - 1) DO {
            FOR ($t_{m-1}$ := $t_m$ - 1; $t_{m-1}$ ≥ 0 AND $t_{m-1}$ ≥ $t_m$ - $T_L$ AND $t_{m-1}$ $\nsucc$ $t_m$; $t_{m-1}$ := $t_{m-1}$ - 1) DO {
                FOR ($t_{m-2}$ := $t_{m-1}$ - 1; $t_{m-2}$ ≥ 0 AND $t_{m-2}$ ≥ $t_{m-1}$ - $T_L$ AND
                $t_{m-2}$ $\nsucc$ $t_{m-1}$; $t_{m-2}$ := $t_{m-2}$ - 1) DO {
                    :
                    :
                    FOR ($t_1$ := $t_2$ - 1; $t_1$ ≥ 0 AND $t_1$ ≥ $t_2$ - $T_L$ AND
                    $t_1$ $\nsucc$ $t_2$; $t_1$ := $t_1$ - 1) DO {

                         IF ((NOT isDominated[$t_1$]) AND
                         U($t_1$, $t_2$, …, $t_m$) > MaxU) THEN
                            MaxU := U($t_1$, $t_2$, …, $t_m$);
                            MaxT[1] := $t_1$, MaxT[2] := $t_2$, …, MaxT[$m$] := $t_m$;
                         ENDIF
                    }
                    :
                    :
                }
            }
        }

STEP 4: FOR i := 1 TO $m$ DO {
           FOR j := 1 TO $n_i$ DO {
                $\boldsymbol{p}$[i][j] := $S_{j_i}$(MaxT[i]);
           }
        }

STEP 5: RETURN $\boldsymbol{p}$[i][j], where 1 ≤ i ≤ $m$ and 1 ≤ j ≤ $n_i$;
```

For example, suppose there are two inter-related events, E_1 and E_2, that occur in the time sequence, i.e., $0 \leq t_1 < t_2$, respectively to maximize the utility $\boldsymbol{U}$. The time series vector $\vec{S}_i(t_i)$, the parameter vector $\vec{P}_i$, and the parametric monitoring constraint $C_i(\vec{S}_i(t),\ \vec{P}_i)$ of each event E_i, where i = 1 and 2, are shown in Table 12. Also assume that the values of $\vec{S}_1(t_1)$, $\vec{S}_2(t_2)$, and their corresponding $\boldsymbol{U}$ are shown in Table 13, Table 14, and Table 15 respectively, and the value of T_L is 2.

Table 12. Time Series Vectors, Parameter Vectors, and Parametric Monitoring Constraints for E_1 and E_2

Event	Time Series Vector	Parameter Vector	Parametric Monitoring Constraint
E₁	$\vec{S}_1(t_1) = (S_{11}(t_1), S_{21}(t_1), S_{31}(t_1))$	$\vec{P}_1 = (p_{11}, p_{21}, p_{31})$	$C_1(\vec{S}_1(t_1), \vec{P}_1) = S_{11}(t_1) \geq p_{11} \wedge S_{21}(t_1) \geq p_{21} \wedge S_{31}(t_1) \leq p_{31}$
E₂	$\vec{S}_2(t_2) = (S_{12}(t_2), S_{22}(t_2))$	$\vec{P}_2 = (p_{12}, p_{22})$	$C_2(\vec{S}_2(t_2), \vec{P}_2) = S_{12}(t_2) \geq p_{12} \wedge S_{22}(t_2) \leq p_{22}$

Table 13. Values of $\vec{S}_1(t_1)$ for E_1

t_1	$S_{11}(t_1)$	$S_{21}(t_1)$	$S_{31}(t_1)$
0	13	27	3
1	12	10	7
2	13	13	5
3	25	15	2
4	12	9	15
5	10	20	5

Table 14. Values of $\vec{S}_2(t_2)$ for E_2

t_2	$S_{12}(t_2)$	$S_{22}(t_2)$
0	19	24
1	12	29
2	14	21
3	17	20
4	15	25
5	18	27

Table 15. Utility U(t_1, t_2)

t_1\\t_2	**0**	**1**	**2**	**3**	**4**	**5**
0		170	250	180	240	190
1			160	170	100	90
2				180	160	240
3					210	200
4						190
5						

Using the *M*-Checkpoint algorithm step by step for the problem, I can search through a particular set of parameter vectors $\{\vec{P}_1, \vec{P}_2\}$, which is at the earliest time point pairs t_1 and t_2 that are not dominated by *any* t_1' and t_2' respectively, where $0 \leq t_1' < t_1 \leq t_2' < t_2$, such that the value of the utility function ***U*** is maximal. In the **STEP 1** for E_1, the

$\vec{S}_1(0)$ is added to the T_{kd} and isDominated[0] is set to false since $t_1 = 0$ is the first time point that is not dominated by any previous time point. After running the for-loop statement, the **STEP 2** only adds the $\vec{S}_1(3)$ to the T_{kd} and sets isDominated[3] to false because $t_1 = 3$ is not dominated by any previous time point, i.e., $t_1 = 0$, 1, and 2. In the **STEP 3**, as started, t_2 is 5 in the outer loop. In the inner loop, $t_1 = 4 \geq (t_2 - T_L)$ and $t_1 \nsucc t_2$ for E_2 because $S_{12}(4) \geq S_{12}(5) \wedge S_{22}(4) \leq S_{22}(5)$ does not satisfy the **Claim 1**. However, referred back to the **STEP 1**, isDominated[4] has been set to true that means $t_1 = 4$ for E_1 is dominated, so the boolean condition in the if-statement inside the inner loop is false. Thus the MaxU, MaxT[1], and MaxT[2] are still equal to 0. When $t_1 = 3$ and $t_2 =5$, $t_1 \geq (t_2 - T_L)$ and $t_1 \nsucc t_2$ for E_2 because $S_{12}(3) \geq S_{12}(5) \wedge S_{22}(3) \leq S_{22}(5)$ does not satisfy the **Claim 1**. Since (NOT isDominated[3]) = true and U(3, 5) > MaxU, MaxU := 200, MaxT[1] := 3, and MaxT[2] := 5. This nested loop then keeps running without further assigning any new value to MaxU, MaxT[1], and MaxT[2] respectively until $t_1 = 0$ and $t_2 = 2$. Since $t_1 \nsucc t_2$ for E_2 when $t_1 = 0$ and 1, (NOT isDominated[0]) = true, and U(0, 2) > MaxU, MaxU := 250, MaxT[1] := 0, and MaxT[2] := 2. In the **STEP 4**, $\boldsymbol{p}$[1][1] := S_{11}(MaxT[1]), $\boldsymbol{p}$[1][2] := S_{21}(MaxT[1]), $\boldsymbol{p}$[1][3] := S_{31}(MaxT[1]), $\boldsymbol{p}$[2][1] := S_{12}(MaxT[2]), and $\boldsymbol{p}$[2][2] := S_{22}(MaxT[2]) in the nested loops. In the **STEP 5**, the algorithm returns (13, 27, 3) and (14, 21).

Theorem 2: For m sets of N parameter vectors in the data set, the M-Checkpoint algorithm correctly computes a ME-EQPE solution, i.e., *argmax* $\boldsymbol{O}(\vec{P}_1, \vec{P}_2, \dots, \vec{P}_m)$, where $\boldsymbol{O}$ is the objective function of the ME-EQPE problem, with the complexity $O(N^m)$.

5.3 Experimental Case Study

The time complexity of the MLE for the logistic regression model is $O(mk^2N)$, where m is the number of time points (events), k is the number of input parametric time series, and N is the size of the learning data set. For the *M*-Checkpoint algorithm, the complexity is $O(N^m)$. Although the time complexity of the MLE is more efficient than that of the *M*-Checkpoint algorithm, the *M*-Checkpoint algorithm generates the vectors of optimal learned parameters that maximize the earning from the training data set for this financial problem shown in Table 16. Using the *M*-Checkpoint algorithm, I can use the optimal decision parameters for detecting both investment events, i.e., the “Best Buy” and “Best Sell” opportunities on the S&P 500 Index Fund. More specifically, given the domain expert’s parametric model templates, $\boldsymbol{C} = \{C_1, C_2\}$, the *M*-Checkpoint algorithm parameterizes the template of each investment event, i.e., the “Best Buy” and “Best Sell” opportunities, and generates the optimal decision parameter vectors, $\boldsymbol{P} = \{\vec{P}_1, \vec{P}_2\}$, which are the best decision parametric values of the template of each event that guarantee the optimality, i.e., $\boldsymbol{O}(\vec{P}_1, \vec{P}_2) \stackrel{\text{def}}{=} \boldsymbol{U}(t_1, t_2)$.

Using the logistic regression model and the *M*-Checkpoint algorithm, the “Best Buy” and “Best Sell” opportunities in this investment and their earnings are shown in Table 17. The results show that the earning obtained from the logistic regression methodology is 0.52%, but the profit gained from my *M*-Checkpoint algorithm is 12.71% that is almost 25 times higher than the earning obtained from the logistic regression approach. Thus I can see that the *M*-Checkpoint algorithm considerably outperforms the logistic regression model.

Table 16. Optimal Parameters and Maximum Earning (%) from the Learning Data Set[3]

p_1	p_2	p_3	p_4	p_5	q_1	q_2	q_3	q_4	q_5	$O(\vec{P}_1, \vec{P}_2)$
-15.34	19.70	-16.89	48.70	79	0.09	16.24	-4.60	54.9	0	55.70%

Table 17. Investors' Earning of the S&P 500 Index Fund from the Test Data Set[4]

Decision Approach	Best Buy	S&P 500 Index	Best Sell	S&P 500 Index	Earning%
Logistic Regression Model	10/14/08	998.01	09/03/09	1003.24	0.52%
M-Checkpoint Algorithm with Financial Expert's Template	10/31/08	968.75	09/07/10	1091.84	12.71%

* Note that the financial experts do not provide the decision parameters that can be used to determine the "Best Buy" and "Best Sell" opportunity in the sequence of occurrence.

5.4 Chapter Summary

The Multi-Event Expert Query Parametric Estimation (ME-EQPE) model is a well-defined model that captures domain experts' knowledge in expression of multivariate time series vectors, decision parameter vectors, a set of parametric constraints, a time utility, and an objective function. Each multivariate time series vector is a set of input parametric time series for each event, e.g., the bear market bottom or the bull market top. Each vector of decision parameters is instantiated from its input parametric time series vector to learn the optimal values that satisfy the given parametric constraints and maximize their time utility collectively. The time utility is a function of the multiple decision time points that the objective function is dependent upon from which the parameter vectors are learned.

[3] The learning data set is from 06/01/1997 to 01/31/2004.
[4] The test data set is from 02/01/2004 to 03/31/2011.

The Multidimensional *M*-Checkpoint algorithm also combines the strenghts of both domain experts' knowledge in terms of a set of parametric constraints and formal-learning-based methodology by the regression appoarch. The goal of the *M*-Checkpoint algorithm is to learn the multiple decision parameter vectors that maximize the objective function, e.g., the earning of the financial example, over multivariate time series vectors. This new algorithm guarantees an optimality of the learned multiple sets of decision parameters in their respective events, i.e., multiple, true optimal decision time points, e.g., $t_{purchased}$ and t_{sold} of the S&P 500 Index Fund, during the computations. To demonstrate the effectiveness of the algorithm, I compare it with the formal-learning-based approach, i.e., logistic regression methodology in the financial domain. Using the learned multiple sets of decision parameters, I show that the *M*-Checkpoint algorithm is more effective and guarantees the satisfactory forecasting results that are superior to those from the logistic regression methodology.

CHAPTER 6. RELAXED *R*-CHECKPOINT ALGORITHM FOR MULTI-EVENT DECISION MAKING

In this chapter, I detail the *R*-Checkpoint algorithm for multi-events and conduct the same financial case study as in Chapter 5 to illustrate the performance among the logistic regression model, the *M*-Checkpoint algorithm, and the *R*-Checkpoint algorithm.

6.1 Relaxed *R*-Checkpoint Algorithm

As discussed in Chapter 1, although the proposed *M*-Checkpoint algorithm is able to learn multiple sets of optimal decision parameters and guarantee a satisfactory forecasting result, the computational complexity of this algorithm is considerably high, i.e., $O(N^m)$, where N is the size of the learning data set and m is the number of the events. To solve the issue of the high complexity, I develop a relaxed algorithm, *R*-Checkpoint, which is discussed in this chapter.

Let us reconsider the financial example again, which is formulated by the ME-EQPE model, shown in Table 18. The problem can be solved by the *M*-Checkpoint algorithm, which learns the optimal time points to maximize the earning function $U(t_1, t_2) = \frac{SP(t_2)}{SP(t_1)} - 1$, which can be expressed as $U^*(f_1(t_1), f_2(t_2)) \in \mathbf{R}$. This earning function is the time utility, which is maximized by choosing the optimal value t_1 and $t_2 \in \mathbf{T}$, where $f_2(t_2) = SP(t_2)$ and $f_1(t_1) = \frac{1}{SP(t_1)}$, for $SP(t_2)$ is the sell value and $SP(t_1)$ is the buy value of the S&P 500 index fund. Since the selected t_1 and t_2 maximize $f_1(t_1)$ and $f_2(t_2)$, the U(t_1, t_2) is also maximized.

However, the problem of the *M*-Checkpoint algorithm is that for every single parameter vector $\vec{P}_i$ in a learning data set, the parameter set at t_i has to be examined with a range of previous sets of parameters at $t_i' < t_i$ for checking the dominance by using the KD-tree data structure and searching algorithm before the optimal solution can be found. More specifically, the *M*-Checkpoint algorithm is brute-force that **STRICTLY** checks all the non-dominated time points of each single event *i*, where $1 \leq i \leq m$, and then determines which m-event, non-dominated time-point combination yields the highest earning. In this approach, the *M*-Checkpoint algorithm can guarantee the optimality but with a higher time complexity, $O(N^m)$. Due to the polynomial time complexity, i.e., $O(N^m)$, the *M*-Checkpoint algorithm is very time consuming and expensive particularly if the size of the learning data set is significantly large.

Table 18. ME-EQPE Problem and Solution Formulation for the S&P 500 Index Fund

Problem and Solution
Problem:
$<\dot{\boldsymbol{S}}, \dot{\boldsymbol{P}}, \boldsymbol{C}, \boldsymbol{U}>$, where
$\dot{\boldsymbol{S}} = \{\vec{S}_1(t_1), \vec{S}_2(t_2)\}$,
where $\vec{S}_1(t_1)$ = (SPD(t_1), CG(t_1), CCD(t_1), ISM(t_1), NLCD(t_1))
and $\vec{S}_2(t_2)$ = (SPI(t_2), CG(t_2), CCI(t_2), ISM(t_2), NLCSV(t_2))
$\dot{\boldsymbol{P}} = \{\vec{P}_1, \vec{P}_2\}$,
where $\vec{P}_1 = (p_1, p_2, p_3, p_4, p_5)$ and $\vec{P}_2 = (q_1, q_2, q_3, q_4, q_5)$
$\boldsymbol{C} = \{C_1, C_2\}$,
where C_1 = SPD(t_1) $< p_1 \wedge$ CG(t_1) $< p_2 \wedge$ CCD(t_1) $< p_3 \wedge$ ISM(t_1) $< p_4 \wedge$ NLCD(t_1) $> p_5$ and C_2 = SPI(t_2) $\geq q_1 \wedge$ CG(t_2) $\geq q_2 \wedge$ CCI(t_2) $\geq q_3 \wedge$ ISM(t_2) $\geq q_4 \wedge$ NLCSV(t_2) $> q_5$
$\boldsymbol{U} = f_2(t_2) * f_1(t_1) - 1$, where $f_2(t_2) = SP(t_2)$ and $f_1(t_1) = \frac{1}{SP(t_1)}$
Solution:
argmax $\boldsymbol{O}(\vec{P}_1, \vec{P}_2) \overset{\text{def}}{=} \boldsymbol{U}(t_1, t_2)$

Instead of finding all the non-dominated time points of each event i initially, I develop the R-Checkpoint algorithm. This algorithm first locates the m-event, time-point combinations, where each $f_i(t_i)$ function of an event is a local maximum at the time point t_i if $f_i(t_i) \geq f_i(t_i \pm \lambda w_L)$, where $1 \leq i \leq m$, λ is a pre-defined parameter, i.e., 6, and w_L = 30 days, which is explained in [57]. The values of the m local maxima $f_i(t_i)$ combinations yield the top-Q time utility $\boldsymbol{U}$, in which their time points $(t_1, t_2, \ldots, t_m)$ with the time utility U$(t_1, t_2, \ldots, t_m)$ are stored in the sorted set $\boldsymbol{S}$ in the descending order according to the values of their utility. Afterward, the algorithm **ONLY** evaluates the dominance of those time points by using the KD-tree data structure and searching technique. In this methodology, the R-Checkpoint algorithm is more efficient to learn the parameters than the M-checkpoint algorithm and has a lower time complexity. The pseudo code of the algorithm is shown in Table 19.

Table 19. *R*-Checkpoint Algorithm

Input: <***S***, ***P***, ***C***, ***U***>

Output: ***p***[i][j] is a two-dimensional array of the parameter vectors that stores the decision parameters for each event i, where $1 \leq i \leq m$, $1 \leq j \leq n_i$, and m, $n_i \in Z^+$.

Data Structures:

1. N is the size of the learning data set.
2. MaxT[i] is an integer array of time points that gives the maximal ***U***, denoted by MaxU, where $1 \leq i \leq m$, and MaxT[1] < MaxT[2] < …< MaxT[m].
3. k_i is an integer variable to store the total number of local maxima of $f_i(t_i)$.
4. $u_i[j_i]$ is a double array to store the value $f_i(t_i)$ when $f_i(t_i)$ is a local maximum at t_i.
5. $t_i[j_i]$ is an integer array to store the time point t_i when $f_i(t_i)$ is a local maximum at t_i.
6. ***S*** is a set of tuples that are sorted in the descending order according to the value $U(t_1[j_1], \ldots, t_m[j_m])$ of each tuple. Each tuple has $m + 1$ attributes, i.e., $\{t_1[j_1], \ldots, t_m[j_m], U(t_1[j_1], \ldots, t_m[j_m])\}$, and each $t_i[j_i]$ is the time point of its local maximum $f_i(t_i)$ of the event i.
7. Q is the total number of m-event, time-point combinations which yields the top-Q time utility ***U*** in the set ***S***.
8. $T_{kd}[i]$ is a KD-tree array that stores the parameter vectors that are not dominated so far for the event i.
9. isDominated[i][t_i] is a boolean array to indicate whether or not a time point t_i of an event i is dominated, i.e., isDominated[i][t_i] := true, by at least one previous time point of the event i, or else isDominated[i][t_i] := false.
10. Δ is a delta time window frame that defines a range of neighbours of $t_i[j_i]$.

Initialization: Initialize MaxT[i] to 0 and assign an empty set to ***S***, where $1 \leq i \leq m$.

Processing:

STEP 1: Search all the m local maxima $f_i(t_i)$s and their respective local optimal time points t_is that correspond to the sequence of the m events, where $t_1 < t_2 < \ldots < t_m$.

```
FOR i := 1 TO $m$ DO {
  Test $f_i(t_i)$ to find if there exists a time point, which is the range of $f_i(t_i \pm 6w_L) > f_i(t_i)$.
  IF (NOT AND $t_i + 6w_L \leq N$ AND $t_i - 6w_L \geq 0$) THEN
      $t_i[j_i] := t_i$;
      $u_i[j_i] := f_i(t_i)$;
  ENDIF
}
```

STEP 2: Compute the value of the objective function for each time-point sequence, i.e., $\boldsymbol{O}(t_1, t_2, \ldots, t_m) \overset{\text{def}}{=} \boldsymbol{U}(t_1, t_2, \ldots, t_m)$, where each optimal time point t_i is located in the **STEP 1**, and then find the m-event, time-point combinations, which yields the top-Q time utility ***U*** and are stored in the set ***S***.

```
FOR ($j_m := 1$; $j_m \leq k_m$; $j_m := j_m + 1$) DO {
  FOR ($j_{m-1} := 1$; $j_{m-1} \leq k_{m-1}$ AND $t_m[j_m] > t_{m-1}[j_{m-1}]$; $j_{m-1} := j_{m-1} + 1$) DO {
          :
          :
      FOR ($j_1 := 1$; $j_1 \leq k_1$ AND $t_2[j_2] > t_1[j_1]$; $j_1 := j_1 + 1$) DO {
          IF ($|S| < Q$) THEN
             $S := S \cup \{t_1[j_1], \ldots, t_m[j_m], U(t_1[j_1], \ldots, t_m[j_m])\}$;
          ELSE IF (Min{$U(t_1, \ldots, t_m)$} < $U(t_1[j_1], \ldots, t_m[j_m])$) AND $t_1 \neq t_1[j_1]$ AND … AND $t_m \neq t_m[j_m]$) THEN
                  $S := S - \{t_1, \ldots, t_m, U(t_1, \ldots, t_m)\}$;
                  $S := S \cup \{t_1[j_1], \ldots, t_m[j_m], U(t_1[j_1], \ldots, t_m[j_m])\}$;
              ENDIF
          ENDIF
      }
      :
      :
  }
}
```

STEP 3: Sort the time-point sequences in the set $\boldsymbol{S}$ in the descending order according to the values of their objective functions, and then use the KD-tree data structure and searching techniques to evaluate whether or not the first time-point sequence (t_1, t_2, …, t_m) in the sorted set $\boldsymbol{S}$ is dominated. If all the time points are not dominated, those time points are the solutions. However, if there are some time points, (t_1, t_2, …, t_m), in the first sequence that are dominated, the algorithm searches the closest neighbors (t_i') of those time points, where $t_i' < t_i$ for $i = 1, 2, …, m$, to evaluate whether or not they are dominated. If the closest neighbors are not dominated, the solution is obained. However, if all the neighbors are dominated, the algorithm applies the same procedure to the subsequent time-point sequences one after the other according to the order in the sorted set $\boldsymbol{S}$. This process keeps repeating until all of the chosen time points are not domainted in a sequence. Then the solution is found.

```
SET i := 1, q := 1, $t_i$ := 1, $T_{kd}$[i] := $\overrightarrow{S_i}(0)$, isDominated[i][0] := false, upperBound := $t_i[q] + \Delta$, lowerBound := $t_i[q] - \Delta$;
    DO
        DO
            SET found := false;
            FOR $t_i'$ := $t_i$ TO upperBound DO {
                Non-Dominance Test: Query $T_{kd}$[i] to find if there exists a point ($p_1$, $p_2$, …, $p_{n_i}$) in the $T_{kd}$[i], which is in the range $\prod_{j=1}^{n_i}[S_{j_i}(t_i'), \infty)$.
                IF (NOT AND $t_i'$ is not dominated) THEN
                    Add $\vec{S}_i(t_i')$ to $T_{kd}$[i];
                    isDominated[i][$t_i'$] := false;
                ELSE
                    isDominated[i][$t_i'$] := true;
                ENDIF}
            IF (NOT isDominated[i][ $t_i$[q]]) THEN
                MaxT[i] := $t_i$[q];
                found := true;
            ELSE
                Search the closest, non-dominated neighbor $\bar{t}_i$ of $t_i$[q], where lowerBound ≤ $\bar{t}_i$ ≤ upperBound, and $\bar{t}_i \neq t_i$[q]. If there are two non-dominated neighors, which are the same distance from the $t_i$[q], select the $\bar{t}_i$, which has a higher value of $f_i(\bar{t}_i)$.
            ENDIF
            IF (found AND i + 1 ≤ $m$) THEN
                $t_{i+1}$ := $t_i$[q] + 1, i := i + 1, $T_{kd}$[i] := $\vec{S}_i(t_i)$;
                isDominated[i][$t_i$] := false, $t_i$ := $t_i$ + 1;
                upperBound := $t_i[q] + \Delta$, lowerBound := $t_i[q] - \Delta$;
            ELSE IF (NOT found AND q + 1 ≤ Q) THEN
                    FOR i := 1 TO $m$ DO {
                      Clear $T_{kd}$[i];
                     }
                    i := 1, q := q + 1, $t_i$ := 1;
                    $T_{kd}$[i] := $\vec{S}_i(0)$, isDominated[i][0] := false;
                    upperBound := $t_i[q] + \Delta$, lowerBound := $t_i[q] - \Delta$;
                 ENDIF
            ENDIF
        LOOP WHILE (i ≤ $m$ AND found)
    LOOP WHILE (NOT found AND q ≤ Q AND i < $m$)
```

STEP 4: Assign the parametric values S_{j_i}(MaxT[i]) to $\boldsymbol{p}$[i][j] for each event i, where $1 \leq i \leq m$, and $1 \leq j \leq n_i$, after the m-event, non-dominated, time-point combinations have been found in the **STEP 3**.

```
    FOR i := 1 TO $m$ DO {
        FOR j := 1 TO $n_i$ DO {
            $\boldsymbol{p}$[i][j] := $S_{j_i}(MaxT[i])$;
         }
     }
```

```
STEP 5: Return the vector of decision parameters p[i][j], i.e., $\vec{P_i}$, for each event $i$, where $1 \le i \le m$, and $1 \le j \le n_i$,
according to the time-point sequence obtained in the STPE 4.
    RETURN p[i][j], where $1 \le i \le m$, and $1 \le j \le n_i$;
```

For example, suppose there are two inter-related events, E_1 and E_2, that occur in the time sequence, i.e., $0 \le t_1 < t_2$, respectively to maximize the utility ***U***. The time series vector $\vec{S}_i(t_i)$, the parameter vector $\vec{P_i}$, and the parametric monitoring constraint $C_i(\vec{S}_i(t),\ \vec{P_i})$ of each event E_i, where i = 1 and 2, are shown in Table 20. I also assume that the values of $f_1(t_1)$ and $f_2(t_2)$ are shown in Table 21, Q is 2, and Δ is 1.

Table 20. Time Series Vectors, Parameter Vectors, and Parametric Monitoring Constraints for E_1 and E_2

Event	Time Series Vector	Parameter Vector	Parametric Monitoring Constraint
E_1	$\vec{S}_1(t_1) = (S_{11}(t_1), S_{21}(t_1), S_{31}(t_1))$	$\vec{P}_1 = (p_{11}, p_{21}, p_{31})$	$C_1(\vec{S}_1(t_1), \vec{P}_1) = S_{11}(t_1) \ge p_{11} \wedge S_{21}(t_1) \ge p_{21} \wedge S_{31}(t_1) \le p_{31}$
E_2	$\vec{S}_2(t_2) = (S_{12}(t_2), S_{22}(t_2))$	$\vec{P}_2 = (p_{12}, p_{22})$	$C_2(\vec{S}_2(t_2), \vec{P}_2) = S_{12}(t_2) \ge p_{12} \wedge S_{22}(t_2) \le p_{22}$

Table 21. Values of $f_1(t_1)$ and $f_2(t_2)$ for E_1 and E_2

t_i	$f_1(t_1)$	$f_2(t_2)$
180	80	70
181	90	50
182	70	80
183	50	20
184	100 (Non-Dominated)	10
185	40	40
186	10	60
187	5	30
188	10	15
189	15	80 (Non-Dominated)
190	50	90 (Dominated)
191	20	85 (Non-Dominated)

Using the *R*-Checkpoint algorithm for the problem, I can find the highest-ranked tuple $\{t_1, t_2, \mathrm{U}(t_1, t_2)\}$ in the set ***S***, in which the two time points, t_1 and t_2, are not dominated by any previous t'_1 and t'_2, where $0 \leq t'_1 < t_1 \leq t'_2 < t_2$, such that the value of the utility function $\boldsymbol{U}$ is maximal. In the **STEP 1**, I assume that the values $f_1(181)$ and $f_1(184)$ of E_1 and $f_2(182)$ and $f_2(190)$ of E_2 are the local maxima. After the assignments in this step, {t_1[1] = 181, u_1[1] = 90}, {t_1[2] = 184, u_1[2] = 100}, and $k_1 = 2$ for E_1. For E_2, {t_2[1] = 182, u_2[1] = 80}, {t_2[2] = 190, u_2[2] = 90}, and $k_2 = 2$. Inside the for-loop statements in the **STEP 2**, the tuples, {181, 182, 7200} and {181, 190, 8100}, are first inserted into the set $\boldsymbol{S}$ because the size $|\boldsymbol{S}|$ is less than Q initially. After that, the tuple {181, 182, 7200} is removed from the set $\boldsymbol{S}$, and the tuple {184, 190, 9000} is added to the set $\boldsymbol{S}$ because Min{7200, 8100} is 7200 that is less than 9000. Thus the resultant tuples stored in the set $\boldsymbol{S}$ are {{184, 190, 9000}, {181, 190, 8100}}. In the **STEP 3** (see Figure 3), the $\vec{S}_1(0)$ is first added to the $\mathrm{T_{kd}}$[1], and isDominated[1][0] is set to false since the time point 0 is not dominated by any previous time point. Using the KD data structure $\mathrm{T_{kd}}$[1], the algorithm continues querying the subsequent time points up to the upperbound, i.e., 185, and adds those non-dominated time points to $\mathrm{T_{kd}}$[1]. As the time point t_1[1] = 184 is not dominated, the t_1[1] = 184 is assigned to MaxT[1], and the boolean variable found is set to true. After the non-dominated time point of E_1 has been found, the first time point of E_2 would be equal to t_1[1] plus 1, i.e., 185. Because the time point 185 is the first time point of E_2, the $\vec{S}_2(185)$ is added to $\mathrm{T_{kd}}$[2], and isDominated[2][185] is set to false. Using the KD tree data structure $\mathrm{T_{kd}}$[2], the algorithm starts querying the subsequent time points up to the upper bound, i.e., 191, and adds those non-dominated time points to $\mathrm{T_{kd}}$[2]. Unfortunately, the time point 190 is dominated, so the algorithm checks whether or not both neighboring time points 189 and

191 are dominated. Because both time points are not dominated, the algorithm evaualates which values, $f_2(189)$ and $f_2(191)$, are higher. According to Table 21, the $f_2(191)$ is higher, so MaxT[2] := 191. As both MaxT[1] and MaxT[2] have been found, the algorithm does not repeat the same procedures for the next tuple in the set $\boldsymbol{S}$. In the **STEP 4**, $\boldsymbol{p}$[1][1] := S_{11}(MaxT[1]), $\boldsymbol{p}$[1][2] := S_{21}(MaxT[1]), $\boldsymbol{p}$[1][3] := S_{31}(MaxT[1]), $\boldsymbol{p}$[2][1] := S_{12}(MaxT[2]), and $\boldsymbol{p}$[2][2] := S_{22}(MaxT[2]) in the nested loops. In the **STEP 5**, the algorithm returns ($\boldsymbol{p}$[1][1], $\boldsymbol{p}$[1][2], $\boldsymbol{p}$[1][3]) and ($\boldsymbol{p}$[2][1], $\boldsymbol{p}$[2][2]).

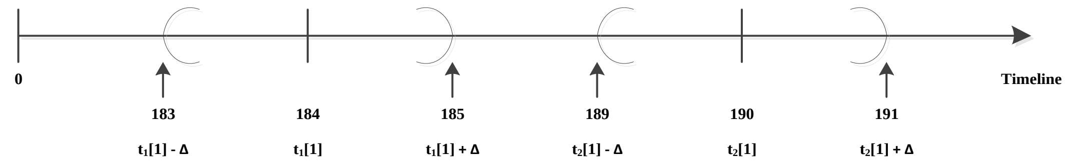


Figure 3. Timeline for E_1 and E_2

6.2 Experimental Case Study

Consider the financial example again, which determines the occurrence of the events, i.e., the bear market bottom and the bull market top, as an illustration. Instead of finding the non-dominated time points in the first place, the algorithm first locates all the pairs of local minimal (t_{min}), i.e., “Best Buy”, and local maximal (t_{max}), i.e., “Best Sell”, time points that correspond to the sequence of the events and then evaluates the values of their objective function, i.e., $\boldsymbol{O}(t_{min}, t_{max}) \overset{\text{def}}{=} \boldsymbol{U}(t_{min}, t_{max})$ = SP(t_{max})/ SP(t_{min}) – 1. Based on their values of the objective function, the algorithm sorts those time-point pairs in the descending order and then uses the KD-tree data structure and searching techniques to learn the non-dominated time point of each event for the first time-point pair in the sorted list. If both time points, i.e., t_{min} and $t_{max,}$, in the first pair are not dominated, the solution is found. However, if t_{min} and/or $t_{max,}$ of the first time-point pair are dominated, the algorithm evaluates their nearby time points, $t'_{min} < t_{min}$ and/or $t'_{max} < t_{max,}$, whether or not they are dominated. If their nearby time points are not dominated, I conclude that the solution is obtained or otherwise. Intuitively, those non-dominated time points correspond to their vectors of parameters $(\vec{P}_1, \vec{P}_2)$ that can reasonably maximize the objective function and guarantee a satisfactory forecasting result. However, if the local minimal, the local maximal, and their nearby time points in the first time-point pair are all dominated, the algorithm continues to search the non-dominated time points in the second pair, and so on and so forth until the solution is found. To the end, the algorithm returns two parameter vectors, one for the best buy, i.e., $\vec{P}_1$ = (SPD(t_{buy}), CG(t_{buy}), CCD(t_{buy}), ISM(t_{buy}), NLCD(t_{buy})) and one for the best sell, i.e., $\vec{P}_2$ = (SPI(t_{sell}), CG(t_{sell}), CCI(t_{sell}), ISM(t_{sell}), NLCSV(t_{sell})), where $t_{buy} \leq t_{min}$ and $t_{sell} \leq t_{max}$, which the investors can use to detect the market bottom and the market top.

The time complexity of the MLE for the logistic regression model is $O(mk^2N)$, where m is the number of events, k is the number of parametric coefficients, and N is the size of the learning data set. For the R-Checkpoint algorithm, the complexity is $O(QkNlogN)$. Although the algorithm [58] used for the MLE is more efficient than that of the R-Checkpoint algorihtm, the R-Checkpoint algorithm can generate the vectors of learned decision parameters that yield a better earning from the training data set for this financial problem shown in Table 22. Using the logistic regression model and the *R*-Checkpoint algorithm, the “Best Buy” and “Best Sell” opportunities in this investment and their earnings are shown in Table 23. The results show that the earning obtained from the logistic regression methodology is 0.52% and from the *R*-Checkpoint is 8.68% that is almost 17 times higher than the earning obtained from the logistic regression approach. On the contrary, even if the *M*-Checkpoint algorithm can guarantee the optimality and yield a better earning, i.e., 12.71%, from the testing data set, but the efficiency of the *R*-Checkpoint algorithm, i.e., $O(QkNlogN)$, is much better than that of the *M*-Checkpoint algorithm, i.e., $O(N^m)$.

Table 22. Decision Parameters and Earning (%) Generated by the *M*- and *R*-Checkpoint Algorithm from the Learning Data Set[5]

Alg	p_1	p_2	p_3	p_4	p_5	q_1	q_2	q_3	q_4	q_5	$O(\vec{P}_1, \vec{P}_2)$
***M*-**	-15.34	19.70	-16.89	48.70	79	0.09	16.24	-4.60	54.9	0	55.70%
***R*-**	-0.68	20.01	-2.01	49	0	0.21	17.6	0	45.8	0	48.74%

Table 23. Investors’ Earning of the S&P 500 Index Fund from the Test Data Set[6]

Decision Approach	Best Buy	S&P 500 Index	Best Sell	S&P 500 Index	Earning%
Logistic Regression Model	10/14/08	998.01	09/03/09	1003.24	0.52%
M-Checkpoint Algorithm with Financial Expert’s Template	10/31/08	968.75	09/07/10	1091.84	12.71%
R-Checkpoint Algorithm with Financial Expert’s Template	10/06/08	1056.89	09/24/10	1148.67	8.68%

* The financial experts do not provide the decision parameters that can be used to determine the “Best Buy” and “Best Sell” opportunity in the sequence of occurrence.

[5] The learning data set is from 06/01/1997 to 01/31/2004.
[6] The test data set is from 02/01/2004 to 03/31/2011.

6.3 Chapter Summary

The ME-EQPE is an original model for the multi-event, decision-making problems that can be solved by the *M*-Checkpoint algorithm. Due to the high complexity $O(N^m)$ of the *M*-Checkpoint algorithm, I develop the *R*-Checkpoint to solve the ME-EQPE problems over multivariate time series. This algorithm combines the strengths of both domain-knowledge-based and formal-learning-based approaches to learn multiple sets of decision parameters that are fairly close to the optimal parameters learned from the *M*-Checkpoint algorithm, produce reasonably forecasting results, and maintain a satisfactorily low time complexity, i.e., $O(QkNlogN)$, as compared with $O(N^m)$ of the *M*-Checkpoint algorithm, where Q is the total number of m-event, time-point combinations which yields the top-Q time utility, and k is the number of input parametric time series for each event. To demonstrate the performance of the new algorithm, I conduct an experiment in the financial domain. Specifically, I compare the forecasting results that are detected by the decision parameters learned from the *R*-Checkpoint with the results that are determined by the optimal parameters obtained from the *M*-Checkpoint algorithm, as well as the parametric coefficients of the logistic regression model. The experimental study shows that the forecasting results by the heuristic *R*-Checkpoint algorithm are slightly lower than those of the optimal *M*-Checkpoint algorithm and are considerably higher than those of the logistic regression methodology.

CHAPTER 7. HYBRID-BASED MULTIVARIATE TIME SERIES ANALYTICS - PARAMETER ESTIMATION (MTSA-PE) MODEL

In this chapter, I explain the MTSA-PE model, which incorporates any types of different constraints, to learn decision parameters over multivariate time series regardless of particular time points.

7.1 Introduction

The hybrid-based models, EQPE and ME-EQPE, and algorithms, Checkpoint, *M*-Checkpoint, and *R*-Checkpoint, are only able to solve a specific class of problems that (1) their decision parameters of an objective function are learned from optimal time points of a time utility function, (2) the parametric template has to be in the considered form, i.e., conjunctions of inequality constraints, and (3) the constraints being used are solely for monitoring purposes. To address the above weaknesses, in this chapter, I develop a general, hybrid-based model, Multivariate Time Series Analytics – Parameter Estimation (MTSA-PE). This model maintains a combination of both domain-knowledge-based and formal-learning-based approaches with possibly incorporating any global constraints that are applied to an entire problem, and monitoring constraints, which are used to detect the occurrence of events. Both types of inequality constraints, global and monitoring, are allowed in any possible combinations and forms. Using the MTSA-PE model associated with an external solver, e.g., the IBM ILOG CPLEX optimizer, domain experts can learn decision parameters that satisfy all the given constraints and that optimize the objective function, which is independent of a particular time point. To demonstrate the MTSA-PE model, I conduct a real case study on the electric power microgrid at the GMU campuses. I utilize the MTSA-PE model to illustrate the GMU problem and the extended MTSA-query constructs (read Chapter 3) to express the model and related MTSA services, such as querying, parameter learning, data monitoring, and decision recommendation, to solve the problem.

7.2 Problem Description of a Real Case Study

Let us consider the real case study at the George Mason University (GMU) campuses, where there are more than 33,000 students, and the total size of all the campuses is more than 800 acres. The electric power demand across those expanding campuses is expected to increase. The increase in electric power consumption results in a higher electricity cost, which is composed of the two main components: (1) a total kilowatt-hour (kWh) charge, i.e., the charge for the total electricity consumption, and (2) an Electricity Supply (ES) service charge, i.e., the charge for the peak demand usage in any 30-minute interval over the past 12 months. The first total kWh charge is priced particularly high during the business office hours between 09:00 a.m. and 06:00 p.m. from Monday to Friday. The second ES service charge (*monthlyEServiceCharge*) is a proxy for the cost of capital investment for power generation capacity, since the power company, such as Virginia Electric and Power Company, needs to build generation, transmission, and distribution facilities that are capable of supporting the peak demand usage, even though the average power demand could be considerably lower. This ES service charge, i.e., the peak demand charge, amounts to approximately 30% of the electric bill in each monthly pay period (*payPeriod*) and is determined based upon the electricity supply demand (*payPeriodSupplyDemand*). This electricity supply demand is decided on the highest of either (*C1*) or (*C2*) according to the electric utility contract:

C1: The highest average kilowatt measured in any 30-minute interval of the current billing month during the on-peak hours of either:

- Between 10 a.m. and 10 p.m. from Monday to Friday for the billing months of June through September, or
- Between 7 a.m. and 10 p.m. from Monday to Friday for all other billing months.

C2: 90% of the highest kilowatt of demand at the same location as determined under (*C1*) above during the billing months of June through September of the preceding eleven billing months.

Thus it is possible that a high peak demand usage just for one minute of electricity consumption over the past year would result in a very significant increase in the total charge of the electric bill of the next monthly pay period. Therefore, controlling the peak demand usage is crucial for decreasing the electricity cost. To mitigate the peak demand problem, my key idea is to learn an optimal peak demand bound over historical and projected electric power demands for each future pay period. This optimal peak demand bound is then used to monitor the prospective demand usage in any time interval of that future pay period. Once the demand usage exceeds the bound, some electricity loads are shed to shut down some electric account units so that the peak demand charge can be controlled. However, to determine an optimal peak demand bound is challenging, as if the bound is set too high, although power services are not interrupted, customers will be charged a significant electricity expense. If the bound is set too low, a low electricity charge is billed, but more power interruptions to customers occur. In order to make an optimal balance of this trade-off, the web-mashup application service framework for multivariate time series analytics has been proposed in Chapter 3. This framework is designed for domain experts to solve this class of dilemmatic situations as well.

7.3 MTSA Querying, Monitoring, Recommendation, and Learning Service

Let us consider the GMU case study again. Using the extended MTSA data model in Chapter 3, the energy planners can create the time-series tables as the inputs and stores them with the data in the database. For example, $ElectricPowerDemand(time, value)$ (Box 14) is the input time-series table, and $PeakDemandBound(time, period, value)$ (Box 15) is the decision-parameter table. Both tables are created as follows by using the Querying Service.

Querying Service

Box 14.

```
CREATE TABLE ElectricPowerDemand (
       time HOURLY_INTERVAL,
       value REAL);
```

Box 15.

```
CREATE TABLE PeakDemandBound (
       time HOURLY_INTERVAL,
       period MONTHLY_INTERVAL,
       value REAL,
       UNIQUE_MAP(time, period));
```

`HOURLY_INTERVAL`, `DAILY_INTERVAL`, `MONTHLY_INTERVAL`, `QUARTERLY_INTERVAL`, and `YEARLY_INTERVAL` are the new integer-based data types to show the sequence of the data. `UNIQUE_MAP()` is the new function that ensures each hourly interval is mapped to one monthly interval, for example. Note that I use the negative and zero integers, e.g., time ≤ 0, period ≤ 0, etc., indicate the historical time series, and the positive integers, e.g., time ≥ 1, period ≥ 1, etc., denote the projected time series.

Monitoring and Recommendation Service

Using the monitoring and recommendation service, the energy planners can determine when they should execute the electric load shedding. In the GMU case study, one of the input time series tables is $ElectricPowerDemand(time, value)$ (Box 14), which is created to store the new incoming electric power demand for monitoring. The monitoring and recommendation service can be expressed by a monitoring-event view and executed by the **MONITOR** command (Box 16 and Box 17).

Box 16.

```
CREATE VIEW ElectricLoadShedding AS (
       SELECT EPD.time, (CASE WHEN EPD.value > PDB.value
                              THEN '1' ELSE '0'
                         END) AS Indicator
       FROM   ElectricPowerDemand EPD, PeakDemandBound PDB
       WHERE  EPD.time = PDB.time);
```

Box 17.

```
CREATE VIEW ELS_Monitoring_Recommendation AS (
       SELECT ELS.time, (CASE WHEN ELS.Indicator = '1'
                              THEN 'The Electric Power Demand Greater Than The
                              Peak Demand Bound. The Electric Load Shedding Is
                              Recommended.'
                         END) AS Action
       FROM ElectricLoadShedding ELS);

MONITOR ELS_Monitoring_Recommendation;
```

`PeakDemandBound` stores the given decision parameter, e.g., 17200 kWh for all the hourly time intervals within the same monthly pay period, e.g., July 2012. If the monitoring constraint in the "`CASE WHEN`" clause of the `ElectricLoadShedding` view (Box 16) is satisfied at the current time interval, the value of the attribute "`Indicator`" indicates "1". The service then recommends the energy planners to execute the electric load shedding since the electric power demand is greater than the peak demand bound (Box 17).

Parameter Learning Service

As I discussed in Chapter 3, domain experts' suggested parameters are not accurate enough to monitor the dynamics of the rapidly changing conditions, such as electric power consumptions at different periods of time, e.g., hourly, daily, monthly, quarterly, and yearly; thus, the parameter learning service should be adopted to learn the optimal decision parameters, e.g., `PeakDemandBound`, and this service can be expressed as follows:

STEP 1: Store the input time series tables, e.g., `ElectricPowerDemand(time, value)`, `PayPeriod(time, period)`, `WeekDay(time, d)`, `Hour(time, h)`, `Month(time, m)`, etc., in the database.

STEP 2: Create the parameter tables, e.g., `PeakDemandBound(time, period, value)`, `PayPeriodSupplyDemand(time, period, value)`, `KW(time, period, value)`, etc., to store the optimal decision parameters over a time and period horizon, where `KW` is a table to store the electric power demand for each hourly time interval.

STEP 3: Create a time series view for the monthly ES service charge for each pay period (Box 18). I assume that the future pay periods are two years, i.e., 24 pay periods as there are 24 months.

Box 18.

```
CREATE VIEW MonthlyEServiceCharge AS (
       SELECT PPSD.time, PPSD.period, 8.124 * PPSD.value AS Charge
       FROM PayPeriodSupplyDemand PPSD);
```

,where $8.124 is the peak demand charge per kilowatt according to the contract of the GMU electric bill.

STEP 4: Create the global constraints, e.g., the condition *C1* of the contractual terms of the GMU electric bill (Box 19), which I described in Section 7.2.

Box 19.

```
CREATE VIEW CurrentBillingMonth AS (
       SELECT PayPeriod.time, PayPeriod.period,
              PayPeriodSupplyDemand.value AS payPeriodSupplyDemand,
              KW.value AS kw,
              (CASE WHEN (WeekDay.d >= 1 AND WeekDay.d <= 5)
                     AND ((Hour.h >= 10 AND Hour.h <= 22
                     AND Month.m >= 6 AND Month.m <= 9)
                     OR ((Hour.h >= 7 AND Hour.h <= 22)
                     AND (Month.m <= 5 OR Month.m >= 10)))
                     AND PayPeriod.time = WeekDay.time
                     AND WeekDay.time = Hour.time
                     AND Hour.time = Month.time
                     AND Month.time = PayPeriodSupplyDemand.time
                     AND PayPeriodSupplyDemand.time = KW.time
              THEN '1' ELSE '0' END) AS Indicator
       FROM PayPeriod, WeekDay, Hour, Month, PayPeriodSupplyDemand, KW);
```

STEP 5: Create the monitoring constraints (Box 20), e.g., electricPowerDemand > peakDemandBound.

Box 20.

```
CREATE VIEW ElectricPowerPeakDemandBound AS (
       SELECT PayPeriod.time, PayPeriod.period,
              PeakDemandBound.value AS peakDemandBound, KW.value AS kw,
              (CASE WHEN ElectricPowerDemand.value > PeakDemandBound.value
                     AND PayPeriod.time >= 1
                     AND PayPeriod.time = ElectricPowerDemand.time
                     AND ElectricPowerDemand.time = PeakDemandBound.time
                     AND PeakDemandBound.time = KW.time
              THEN '1' ELSE '0' END) AS Indicator
       FROM PayPeriod, ElectricPowerDemand, PeakDemandBound, KW);
```

STEP 6: Create the parameter learning event and then execute the event construct to learn the parameters (Box 21), which are stored in their tables respectively.

Box 21.

```
CREATE EVENT LearnPeakDemandBoundParameter (
       GC_LEARN PeakDemandBound, PayPeriodSupplyDemand, KW
       FOR MINIMIZE SUM(MESC.Charge) AS TotalCharge
       WITH CBM.Indicator = '1' THEN CBM.payPeriodSupplyDemand >= CBM.kw
       AND PBM.Indicator = '1' THEN PBM.payPeriodSupplyDemand >= 0.9 * PBM.kw
       AND PDB.value <= PPSD.value
       AND PDB.value >= 0
       AND EPGPDB.Indicator = '1' THEN EPGPDB.kw = EPGPDB.peakDemandBound
       AND EPLEPDB.Indicator = '1' THEN  EPLEPDB.kw =
       EPLEPDB.electricPowerDemand
       FROM CurrentBillingMonth CBM, PrecedingBillingMonth PBM, PeakDemandBound
       PDB, PayPeriodSupplyDemand PPSD, KW, ElectricPowerGreaterPeakDemandBound
       EPGPDB, ElectricPowerLessEqualPeakDemandBound EPLEPDB,
       MonthlyEServiceCharge MESC
       WHERE CBM.time = PBM.time
       AND PBM.time = PDB.time
       AND PDB.time = PPSD.time
       AND PPSD.time = KW.time
       AND KW.time = EPGPDB.time
       AND EPGPDB.time = EPLEPDB.time
       AND EPLEPDB.time = MESC.time);

EXECUTE LearnPeakDemandBoundParameter;
```

This learning query (Box 21) is to learn `PeakDemandBound`, `PayPeriodSupplyDemand`, and `KW` that minimize the `TotalCharge` of the 24 future pay periods and satisfy all the six constraints in the `WITH…THEN` clause. For example, the first two constraints denote the *C1* and *C2* of the contractual terms of the

GMU electric bill. The last two constraints express the monitoring templates. When the event `LearnPeakDemandBoundParameter` is executed, the command `GC_LEARN` sends the SQL-learning event to the MTSA compiler, where GC stands for “General Class”. The compiler transforms this event to the IBM OPL construct, which is then sent to the IBM ILOG CPLEX optimizer to learn the parameters, e.g., peakDemandBound. The learned parameters are stored in their corresponding tables, e.g., `PeakDemandBound`. Note that all the parameters are instantiated with the optimal values.

7.4 MTSA Query Semantics: Parameter Estimation Model

In this section, I formalize the Multivariate Time Series Analytics - Parameter Estimation (MTSA-PE) problem and solution that I propose for the *Parameter Learning Service*. The goal of a MTSA-PE problem is to learn optimal decision parameters that maximize or minimize an objective function over historical and projected multivariate time series data.

I assume that time is split into base time intervals of a fixed duration, e.g., hourly, for simplicity, and each time interval is indexed by an integer, and that I am also given a *m*-sets of decision parameters $\{\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_m\}$. The mathematical components of the MTSA-PE problem and solution and its formulations are shown as follows:

- *Time Horizon*: A time horizon TH is defined as $Z_k = \{t | t \in Z \land t \geq k\}$, where Z is a set of integers, t is a time interval in Z_k, and $k \in Z$.

More specifically, I use negative or zero integers $t \leq 0$ represented for the past and positive integers $t > 0$ represented for the future time intervals. For example, Z_{-5} is the set of all integers that are greater than or equal to -5. It means that the time horizon starts from the past hourly time interval -5 to the future infinite time interval.

- *Past Time Horizon*: A past time horizon is defined as $PastTH = \{t | t \in Z_k \wedge t \leq 0\}$.
- *Future Time Horizon*: A future time horizon is defined as $FutureTH = \{t | t \in Z_k \wedge t > 0\}$.
- *Period Horizon*: A period horizon PH is defined as $Z_l = \{p | p \in Z \wedge p \geq l\}$, where Z is a set of integers, p is a period in Z_l, and $l \in Z$.

I also assume that a sequence of time intervals t_s in Z_k is grouped into periods, e.g., daily, weekly, or monthly periods. Each period p contains consecutive time intervals and is also indexed by an integer, where a positive integer $p > 0$ corresponds to the future period, and a negative or zero integer $p \leq 0$ corresponds to the past period.

- *Past Period Horizon*: A past period horizon is defined as $PastPH = \{p | p \in Z_l \wedge p \leq 0\}$.
- *Future Period Horizon*: A future period horizon is defined as $FuturePH = \{p | p \in Z_l \wedge p > 0\}$.

The mapping between a time interval in TH and a period in PH is a function: $TH \rightarrow PH \text{ such that } (\forall t_1, t_2 \in TH): (t_1 < t_2 \rightarrow Period(t_1) \leq Period(t_2))$. Now suppose I have a sequence of time intervals in TH and of their corresponding periods in PH that are shown in Figure 4. For instance, both the hourly time intervals, 2 and 3, are mapped to the same period, i.e., $Period(2)$ = $Period(3)$ = 1, as 2 is less than

3 so that 2 and 3 are grouped into the same period 1. Some other examples are $Period(0)$ = 0, $Period(8)$ = 3, and $Period(-6)$ = -2.

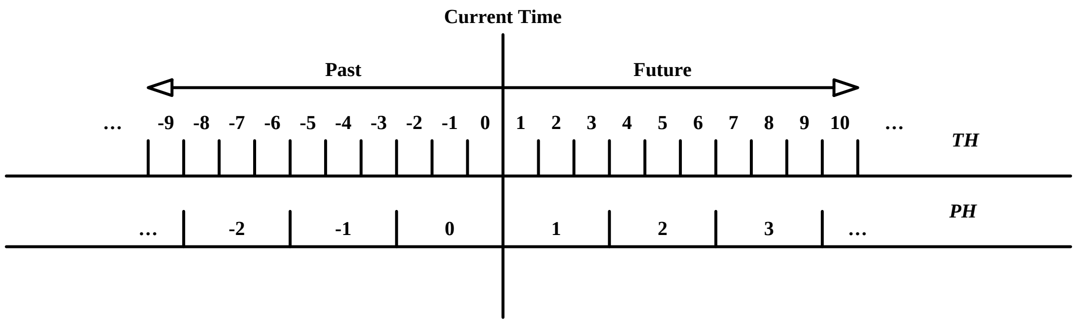


Figure 4. Time Intervals in TH and Corresponding Periods in PH

- *Time Series*: A time series *S* is a function *S*: *TH* → *D*, where *D* is a numerical domain, e.g., *D* = *R* or *D* = *Z*.
- *Parametric Estimation Constraint*: A parametric estimation constraint $C(S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n)$ is a symbolic expression in terms of $S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n$, where $S_1(t), S_2(t), \ldots, S_k(t)$ are the *k* input time series, and $(p_1, p_2, \ldots, p_n)$ is a vector of *n* parameters that have been given by domain experts or are instantiated at a particular time interval over TH or a period over PH.

I suppose that a parametric estimation constraint is written in a language that has the truth-value interpretation *I*: $R^k x R^n$ → {TRUE, FALSE}, i.e., $I(C(S_1(t), S_2(t), \ldots, S_k(t), p_1, p_2, \ldots, p_n))$ = TRUE if and only if the constraint *C* is satisfied at *t* with the parameters $(p_1, p_2, \ldots, p_n) \in R^n$. In this model, I focus on any possible combinations of inequality constraints of the general form: $(S_1(t)\ eop\ p_1)\ op\ (S_2(t)\ eop\ p_2)\ op\ \ldots\ op\ (S_k(t)$

eop p_n) *op* (p_1 *eop* p_2) *op* (p_1 *eop* p_3) *op* ... *op* (p_i *eop* p_j), where *eop* ∈ {<, ≤, ==, ≥, >}, *op* ∈ {∧, ∨}, and $i \neq j$.

- *Parametric Implication Constraint*: A parametric implication constraint ($C_\ell \rightarrow C_j$) is a logical constraint, i.e., "C_ℓ implies C_j" or "if C_ℓ then C_j" is TRUE if both C_ℓ and C_j are TRUE, where C_ℓ and C_j are a parametric estimation constraint C.

Some of the Cs and ($C_\ell \rightarrow C_j$)s are parametric global constraints $\boldsymbol{C_P}$ that are the general constraints, e.g., the contractual terms of the GMU electric bill, such as *C1*, to be applied to an entire problem. Some of them are parametric monitoring constraints $\boldsymbol{C_M}$ that are used to detect the occurrence of an event of a problem, e.g., *electricPowerDemand* > *peakDemandBound*.

- *Objective Function*: An objective function O is a function O: $R^{w_1} \text{x} R^{w_2} \text{x} \ldots \text{x} R^{w_m} \rightarrow R$, where m is the total number of sets of parameters, w_i is the total number of parameters in a set i, and R is the set of real numbers, for i = 1, 2, …, m.

MTSA Parametric Estimation (*MTSA-PE*) *Problem*: A *MTSA-PE* problem is a tuple <$\boldsymbol{S}$, $\boldsymbol{P}$, $\boldsymbol{C_P}$, $\boldsymbol{C_M}$, $\boldsymbol{O}$>, where $\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$ is a set of the k input time series, $\boldsymbol{P} = \{\overline{w}_1, \overline{w}_2, \ldots, \overline{w}_m\}$ is a m-sets of parameters, $\boldsymbol{C_P}$ is a set of parametric global constraints in $\boldsymbol{S}$ and $\boldsymbol{P}$, $\boldsymbol{C_M}$ is a set of parametric monitoring constraints in $\boldsymbol{S}$ and $\boldsymbol{P}$, and $\boldsymbol{O}$ is an objective function.

MTSA Parametric Estimation (*MTSA-PE*) *Solution*: A solution to the *MTSA-PE* problem <$\boldsymbol{S}$, $\boldsymbol{P}$, $\boldsymbol{C_P}$, $\boldsymbol{C_M}$, $\boldsymbol{O}$> is *argmin* $\boldsymbol{O}(\boldsymbol{P})$, i.e., the optimal values of a m-sets of parameters that minimize $\boldsymbol{O}$.

Let us reconsider the GMU case study as an explanation for the above mathematical formulations. First, Table 24 shows the input multivariate time series $\boldsymbol{S}$, and the time interval t is an integer hourly time interval. All the input time series are stored in the tables that I discussed in the **STEP 1** of the parameter learning service in Section 7.3.

Table 24. Multivariate Time Series Data S

Time Series S	**Abbreviation**	**Table**
Electric Power Demand	$electricPowerDemand(t)$	ElectricPowerDemand
Monthly Pay Period	$payPeriod(t)$	PayPeriod
Annual Year	$year(t)$	Year
Month of a Year	$month(t)$	Month
Day of a Month	$day(t)$	Day
Day of a Week	$weekDay(t)$	WeekDay
Hour of a Day	$hour(t)$	Hour

The decision parameter sets $\boldsymbol{P}$ used in the case study are defined and explained in Table 25, and p is a monthly pay period. All the tables of the parameter sets are created in the **STEP 2** of the parameter learning service in Section 7.3.

Table 25. Decision Parameter Sets

Parameter Set	Usage Intrepretation	Table
$peakDemandBound[p]$ is an array to store the peak demand bound for each future monthly pay period.	▪ Test if the $electricPowerDemand(t)$ exceeds the $peakDemandBound[p]$ when $p == payPeriod(t) \wedge t \geq 1$ for $\forall t \in FutureTH$ and $p \in FuturePH$. ▪ Test if the $electricPowerDemand(t)$ less than or equal to $peakDemandBound[p]$ when $p == payPeriod(t)$ or $t \leq 0$ for $\forall t \in TH$ and $p \in PH$.	PeakDemandBound
$kW[t]$ is an array to store the electric power demand for each hourly time interval.	▪ Instantiate the values into kWover the historical and projected electric power demand for $\forall t \in TH, p \in PH$, i.e., $kW[t]$ stores the electric power demand when the electric power demand is less than or equal to the peak demand bound, or $kW[t]$ stores the peak demand bound when the electric power demand is greater than the peak demand bound.	KW
$payPeriodSupplyDemand[p]$ is an array to store the electricity supply demand for each future monthly pay period.	▪ Instantiate the values into $payPeriodSupplyDemand$ over kW depended on which contractual condition *C1* or *C2* is satisfied such that $payPeriodSupplyDemand$ minimizes the objective function ***O*** for $\forall t \in TH, p \in FuturePH$.	PayPeriodSupplyDemand

$\boldsymbol{C_P}$ and $\boldsymbol{C_M}$ are illustrated as follows. Both types of the constraints are created in the **STEP 4** and **STEP 5** of the parameter learning service in Section 7.3.

▪ Parametric global constraint $\boldsymbol{C_P}$

C_{ℓ}: $C(payPeriod(t), weekDay(t), hour(t), month(t), p, 1,5, 6, 7,9, 10,22)$ = $(payPeriod(t) == p \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge ((hour(t) \geq 10 \wedge hour(t) \leq 22 \wedge month(t) \geq 6 \wedge month(t) \leq 9) \vee ((hour(t) \geq 7 \wedge hour(t) \leq 22) \wedge (month(t) \leq 5 \vee month(t) \geq 10))))$

$C_{\not\ell}$: $C(payPeriodSupplyDemand[p], kW[t])$ = $(payPeriodSupplyDemand[p] \geq kW[t])$

$\boldsymbol{C_1}$**:** $(\forall t \in TH, p \in FuturePH)$: $C_{\ell} \rightarrow C_{\not\ell}$ denotes the contractual condition *C1* in Section 7.2. This condition *C1* is also constructed by the `CBM.Indicator = '1'` THEN

`CBM.payPeriodSupplyDemand >= CBM.kw` that is evaluated by the parameter learning event, i.e., LearnPeakDemandBoundParameter, in the **STEP 6** of Section 7.3.

C_{ℓ}: $C(payPeriod(t), weekDay(t), hour(t), month(t), p-11, p, 1{,}5, 6{,}9, 10{,}22)$ = $((payPeriod(t) \geq p-11 \wedge payPeriod(t) < p) \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge (month(t) \geq 6 \wedge month(t) \leq 9) \wedge (hour(t) \geq 10 \wedge hour(t) \leq 22))$

C_{j}: $C(payPeriodSupplyDemand[p], kW[t], 0.9)$ = $(payPeriodSupplyDemand[p] \geq 0.9 * kW[t])$

***C*₂**: $(\forall t \in TH, p \in FuturePH)$: $C_{\ell} \rightarrow C_{j}$ represents the contractual condition *C2* in Section 7.2, which is constructed by the `PBM.Indicator = '1'` THEN `PBM.payPeriodSupplyDemand >= 0.9 * PBM.kw` in the **STEP 6** of Section 7.3.

C_3: $(\forall p \in FuturePH)$: $(peakDemandBound[p] \leq payPeriodSupplyDemand[p])$ restricts the peak demand bound not greater than the electricity supply demand. This constraint is constructed by the `PDB.value <= PPSD.value` in the **STEP 6** of Section 7.3.

C_4: $(\forall p \in FuturePH)$: $(peakDemandBound[p] \geq 0)$ ensures that the peak demand bound must be non-negative values. This constraint is constructed by the `PDB.value >= 0` in the **STEP 6** of Section 7.3.

- Parametric Monitoring Constraint C_M

C_ℓ: $C(payPeriod(t), electricPowerDemand(t), peakDemandBound[p], t, 1, p)$ = $(t \geq 1 \wedge p == payPeriod(t) \wedge electricPowerDemand(t) > peakDemandBound[p])$

$C_\mathscr{f}$: $C(kW[t], peakDemandBound[p])$ = $(kW[t] == peakDemandBound[p])$

C

C_5: $(\forall t \in FutureTH, p \in FuturePH)$: $C_\ell \rightarrow C_\mathscr{f}$ monitors whether the electric power demand exceeds the peak demand bound when the hourly time interval t is positive. If this monitoring constraint, C_ℓ, is triggered, the peak demand bound is stored in the kW. This monitoring constraint is constructed by the `EPGPDB.Indicator = '1'` THEN `EPGPDB.kw = EPGPDB.peakDemandBound` that is evaluated by the parameter learning event, i.e., LearnPeakDemandBoundParameter, in the **STEP 6** of Section 7.3.

C_ℓ: $C(payPeriod(t), electricPowerDemand(t), peakDemandBound[p], p, t, 0)$ = $((electricPowerDemand(t) \leq peakDemandBound[p] \wedge p == payPeriod(t)) \vee (t \leq 0))$

$C_\mathscr{f}$: $C(kW[t], peakDemandBound[p])$ = $(kW[t] == electricPowerDemand(t))$

C_6: $(\forall t \in TH, p \in PH)$: $C_\ell \rightarrow C_\mathscr{f}$ monitors whether the electric power demand is less than or equal to the peak demand bound or the hourly time interval t is non-positive. If this monitoring constraint, C_ℓ, is triggered, the electric power demand is stored in the kW. This constraint is constructed by the `EPLEPDB.Indicator = '1'` THEN `EPLEPDB.kw = EPLEPDB.electricPowerDemand` in the **STEP 6** of Section 7.3.

Regarding the objective function $\boldsymbol{O}$, I assume that the GMU energy planners evaluate the total peak demand charge of the ES service for the projected 24 pay periods in the future two years, i.e., $\sum_{p=1}^{24}(8.124 * payPeriodSupplyDemand[p])$, where $8.124 * payPeriodSupplyDemand[p]$ is the monthly ES service charge, which is created in the **STEP 3** of the parameter learning service in Section 7.3. This total peak demand charge is the objective function $\boldsymbol{O}$, which is minimized by optimally determining the $payPeriodSupplyDemand[p]$ that satisfies all the given constraints, where $8.124 is the peak demand charge per kilowatt according to the contractual terms of the GMU electric bill, and $1 \leq p \leq 24$.

Shown in Table 26, the MTSA-PE problem and solution of the case study can be constructed by putting all the considered time series $\boldsymbol{S}$, the parameter sets $\boldsymbol{P}$, the constraints $\boldsymbol{C_P}$ and $\boldsymbol{C_M}$, and the objective function $\boldsymbol{O}$ to the formulations of the *MTSA-PE* problem and solution. More specifically, a *MTSA-PE* problem and solution is:

$$\underset{\boldsymbol{P}}{argmin}\, \boldsymbol{O}(\boldsymbol{P})$$
$$subject\ to\ \boldsymbol{C_P}(\boldsymbol{P}) \wedge \boldsymbol{C_M}(\boldsymbol{P})$$

This MTSA-PE problem and solution is constructed by the learning event LearnPeakDemandBoundParameter, which learns the parameter sets, $peakDemandBound$, $payPeriodSupplyDemand$, and kW.

Table 26. Formulation of the MTSA-PE Problem and Solution for the GMU Peak Electric Power demand

Problem and Solution
Problem: $\langle \boldsymbol{S}, \boldsymbol{P}, \boldsymbol{C_P}, \boldsymbol{C_M}, \boldsymbol{O} \rangle$ $\boldsymbol{S} = \{electricPowerDemand, payPeriod, year, month, day, weekDay, hour\}$,   where $electricPowerDemand(t) \geq 0, -8759 \leq t \leq 17520, -11 \leq payPeriod(t) \leq 24, 2011 \leq year(t) \leq 2013, 1 \leq month(t) \leq 12, 1 \leq day(t) \leq 31, 0 \leq weekDay(t) \leq 6, 0 \leq hour(t) \leq 23$ $\boldsymbol{P} = \{peakDemandBound, payPeriodSupplyDemand, kW\}$,   where $peakDemandBound[p] \geq 0, payPeriodSupplyDemand[p] \geq 0, kW[t] \geq 0, -8759 \leq t \leq 17520, 1 \leq p \leq 24$ $\boldsymbol{C_P} = \{C_1, C_2, C_3, C_4\}$, where $C_1 = (\forall t \in TH, p \in FuturePH): ((payPeriod(t) == p \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge ((hour(t) \geq 10 \wedge hour(t) \leq 22 \wedge month(t) \geq 6 \wedge month(t) \leq 9) \vee ((hour(t) \geq 7 \wedge hour(t) \leq 22) \wedge (month(t) \leq 5 \vee month(t) \geq 10)))) \rightarrow (payPeriodSupplyDemand[p] \geq kW[t]))$, $C_2 = (\forall t \in TH, p \in FuturePH): (((payPeriod(t) \geq p - 11 \wedge payPeriod(t) < p) \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge (month(t) \geq 6 \wedge month(t) \leq 9) \wedge (hour(t) \geq 10 \wedge hour(t) \leq 22)) \rightarrow (payPeriodSupplyDemand[p] \geq 0.9 * kW[t]))$, $C_3 = (\forall p \in FuturePH): (peakDemandBound[p] \leq payPeriodSupplyDemand[p])$, $C_4 = (\forall p \in FuturePH): (peakDemandBound[p] \geq 0)$ $\boldsymbol{C_M} = \{C_5, C_6\}$, where $C_5 = (\forall t \in FutureTH, p \in FuturePH): ((t \geq 1 \wedge p == payPeriod(t) \wedge electricPowerDemand(t) > peakDemandBound[p]) \rightarrow (kW[t] == peakDemandBound[p]))$ $C_6 = (\forall t \in TH, p \in PH): (((electricPowerDemand(t) \leq peakDemandBound[p] \wedge p == payPeriod(t)) \vee (t \leq 0)) \rightarrow (kW[t] == electricPowerDemand(t)))$, $\boldsymbol{O} = \sum_{p=1}^{24}(8.124 * payPeriodSupplyDemand[p])$ *Solution:* $\underset{\boldsymbol{P}}{argmin}\, \boldsymbol{O}(\boldsymbol{P})$ $subject\ to\ \boldsymbol{C_P}(\boldsymbol{P}) \wedge \boldsymbol{C_M}(\boldsymbol{P}))$

This MTSA-PE problem is then expressed by the MTSA SQL according to the **STEP 6** of the parameter learning service in Section 7.3. Once this MTSA-SQL construct of this problem is initiated, the optimal values of the decision parameter sets $\boldsymbol{P}$ are determined by sending this MTSA-SQL construct to the MTSA compiler. This MTSA compiler transforms the MTSA-SQL construct to the OPL format that is then sent to the external optimization solver, i.e., IBM ILOG CPLEX optimizer, to learn the parameter sets $\boldsymbol{P}$. After the optimal decision parameters, e.g., $peakDemandBound[1]$ of the monthly pay period 1, is learned from the optimizer, I can apply the parametric monitoring constraints, e.g., $electricPowerDemand(t) > peakDemandBound[1]$, to the new incoming electricity consumption of the monthly pay period 1 from the GMU campuses and perform the event monitoring in an hourly basis, where $1 \leq t \leq 720$, for the entire monthly pay period 1. Once the monitoring constraints, e.g., C_5, are triggered, the recommended action, e.g., the electric load shedding, is delivered to the service providers, that is, the GMU energy planners, to execute the electric load shedding to shut down some electric account units according to the prioritization scheme from the energy manager.

7.5 Implementation of a High-level Architecture for Parameter Learning Process

Figure 5 illustrates the parameter learning process for the optimal decision parameters. As this figure shows, domain experts use the parameter learning service to construct the query for the MTSA learning event, e.g., `LearnPeakDemandBoundParameter`. Once this learning event is initiated, the MTSA compiler calls the query translator to transform the learning event into the IBM OPL construct, which is shown in Figure 6. Note that this OPL construct is manually created as the compiler is still being developed. This IBM OPL construct is then sent to the external optimization solver, i.e., the IBM ILOG CPLEX optimizer, to learn optimal decision parameters, e.g., $peakDemandBound$. These decision parameters are then processed by the output formatter associated with the query translator to return the answer back to the parameter learning service, which presents the results to the experts.

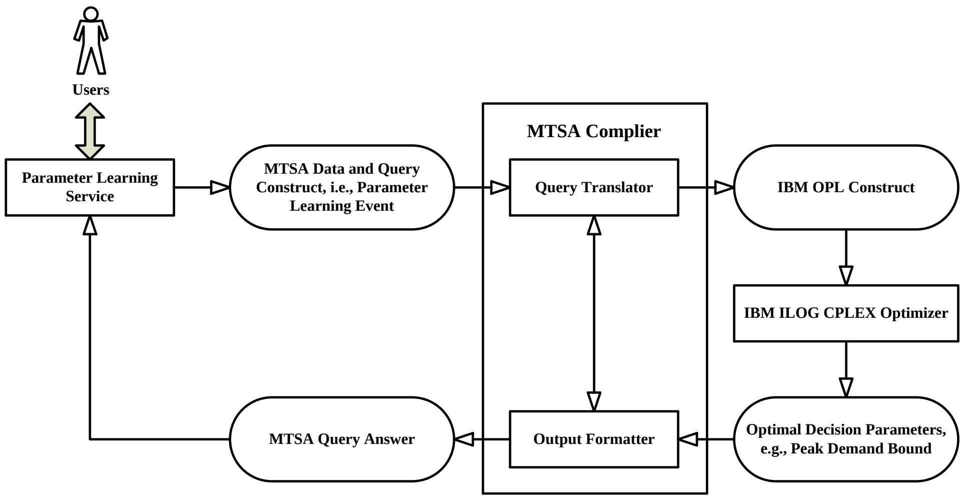
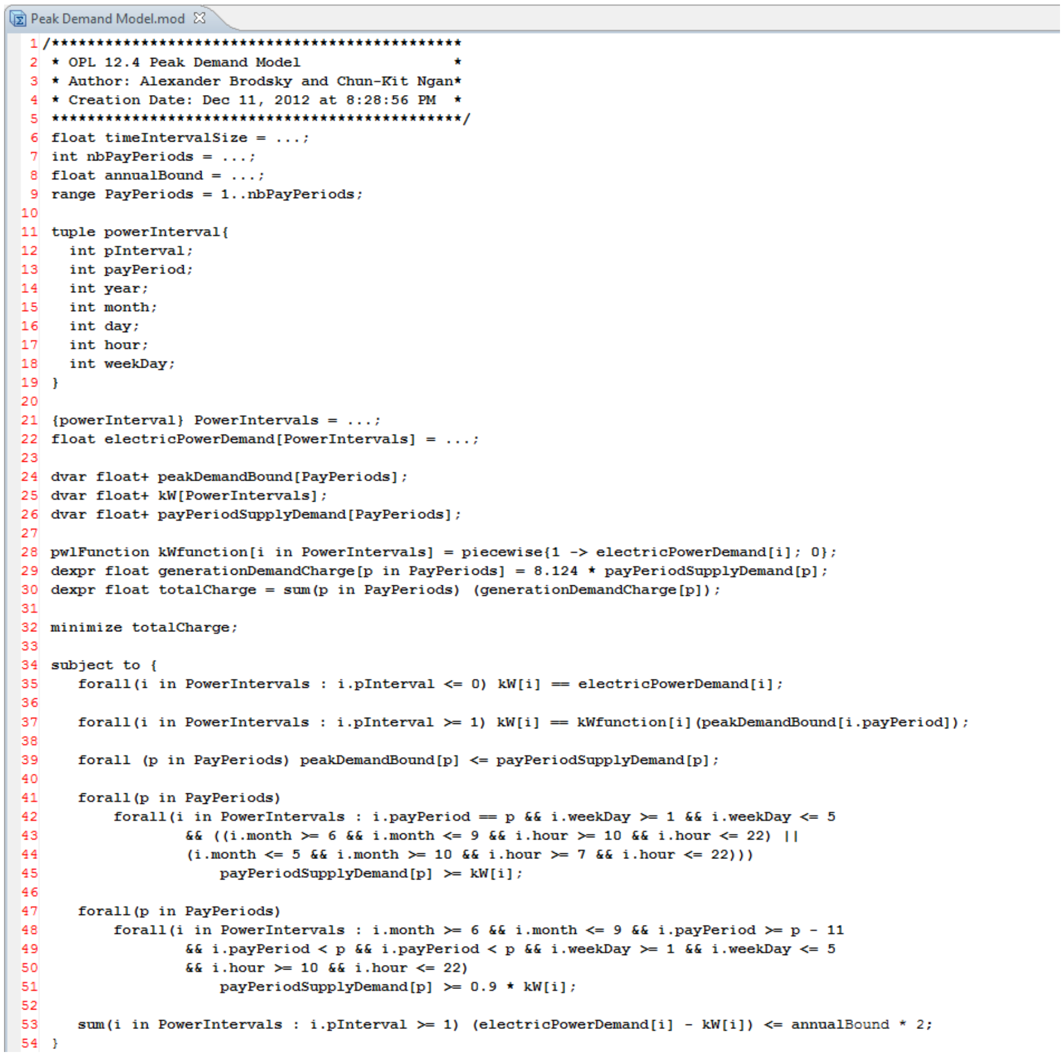

Users
Parameter Learning Service
MTSA Data and Query Construct, i.e., Parameter Learning Event
MTSA Complier
Query Translator
IBM OPL Construct
IBM ILOG CPLEX Optimizer
Optional Decision Parameters, e.g., Peak Demand Bound
Output Formatter
MTSA Query Answer

Figure 5. Parameter Learning Architecture for Optimal Decision Parameters

Figure 6 shows the OPL model transformed from the MTSA-SQL construct. First, the value 24, i.e., the total 24 months from 2012 to 2013, is assigned to the variable *nbPayPeriods* in the line number 7. The value 0 is assigned to the variable *annualBound*, that is, the annual maximal power interruption allowed in the line number 8. From the line number 11 to 19, I declare a tuple of a power interval that has the attributes, including *pInterval*, *payPeriod*, *year*, *month*, *day*, *hour*, and *weekDay*. The line number 21 declares the *PowerIntervals* that include both the past and the future power intervals. The line number 22 declares the *electricPowerDemand[PowerIntervals]* array. The line number 24, 25, and 26 declares the parameter sets, including the *peakDemandBound[PayPeriods]*, *kW[PowerIntervals]*, and *payPeriodSupplyDemand[PayPeriods]*. The monthly ES service charge, i.e., the peak demand charge, is declared on the line number 29, and the total peak demand charge, which is declared on the line number 30, is minimized on the line number 32. All the constraints, *C1* – *C6*, are declared from the line number 35 to 54.



```
 1 /*********************************************
 2  * OPL 12.4 Peak Demand Model                 *
 3  * Author: Alexander Brodsky and Chun-Kit Ngan*
 4  * Creation Date: Dec 11, 2012 at 8:28:56 PM  *
 5  *********************************************/
 6  float timeIntervalSize = ...;
 7  int nbPayPeriods = ...;
 8  float annualBound = ...;
 9  range PayPeriods = 1..nbPayPeriods;
10
11  tuple powerInterval{
12    int pInterval;
13    int payPeriod;
14    int year;
15    int month;
16    int day;
17    int hour;
18    int weekDay;
19  }
20
21  {powerInterval} PowerIntervals = ...;
22  float electricPowerDemand[PowerIntervals] = ...;
23
24  dvar float+ peakDemandBound[PayPeriods];
25  dvar float+ kW[PowerIntervals];
26  dvar float+ payPeriodSupplyDemand[PayPeriods];
27
28  pwlFunction kWfunction[i in PowerIntervals] = piecewise{1 -> electricPowerDemand[i]; 0};
29  dexpr float generationDemandCharge[p in PayPeriods] = 8.124 * payPeriodSupplyDemand[p];
30  dexpr float totalCharge = sum(p in PayPeriods) (generationDemandCharge[p]);
31
32  minimize totalCharge;
33
34  subject to {
35     forall(i in PowerIntervals : i.pInterval <= 0) kW[i] == electricPowerDemand[i];
36
37     forall(i in PowerIntervals : i.pInterval >= 1) kW[i] == kWfunction[i](peakDemandBound[i.payPeriod]);
38
39     forall (p in PayPeriods) peakDemandBound[p] <= payPeriodSupplyDemand[p];
40
41     forall(p in PayPeriods)
42         forall(i in PowerIntervals : i.payPeriod == p && i.weekDay >= 1 && i.weekDay <= 5
43                 && ((i.month >= 6 && i.month <= 9 && i.hour >= 10 && i.hour <= 22) ||
44                 (i.month <= 5 && i.month >= 10 && i.hour >= 7 && i.hour <= 22)))
45                     payPeriodSupplyDemand[p] >= kW[i];
46
47     forall(p in PayPeriods)
48         forall(i in PowerIntervals : i.month >= 6 && i.month <= 9 && i.payPeriod >= p - 11
49                 && i.payPeriod < p && i.payPeriod < p && i.weekDay >= 1 && i.weekDay <= 5
50                 && i.hour >= 10 && i.hour <= 22)
51                     payPeriodSupplyDemand[p] >= 0.9 * kW[i];
52
53     sum(i in PowerIntervals : i.pInterval >= 1) (electricPowerDemand[i] - kW[i]) <= annualBound * 2;
54  }
```

Figure 6. The OPL Constructs for the MTSA-SQL Parameter Learning Service

7.6 Experimental Case Study

Using the historical electric power consumption in the past years, e.g., 2011, the projected electricity demand over a future time horizon, e.g., 2012 and 2013, the maximal annual power interruption allowed, e.g., 0, and the parametric model templates, i.e., global and monitoring constraints, which are identified by the GMU energy planners and are required by the utility contracts that are supplied from the *Model Definition Service*, I input all of these data, constraints, and requirements to the *Parameter Learning Service*.

Using the *Parameter Learning Service*, I formulate the MTSA query construct, i.e., the parameter learning event, which has been demonstrated in Section 7.3. Based on the parameter learning event, I manually formulate the corresponding OPL construct, shown in Figure 6, and run the construct on the IBM ILOG CPLEX Optimization Studio to obtain the learned optimal peak demand bounds for all the future pay periods shown in Table 27.

Table 27. The GMU Peak Electric Power Demands over Pay Periods

Pay Period	Peak Demand Bound kW	Pay Period	Peak Demand Bound kW
January 2012	12189	January 2013	12953
February 2012	12654	February 2013	13447
March 2012	12268	March 2013	13037
April 2012	15410	April 2013	16376
May 2012	14729	May 2013	15653
June 2012	14921	June 2013	15856
July 2012	17211	July 2013	18291
August 2012	14575	August 2013	15490
September 2012	15998	September 2013	17001
October 2012	15020	October 2013	15962
November 2012	12856	November 2013	13662
December 2012	12654	December 2013	13447

The energy planners use the above optimal peak demand bounds to perform the optimal event monitoring over the actual incoming electricity demand in each monthly pay period through the *Monitoring and Recommendation Service* described in Section 7.3.

Using the results from the *Monitoring and Recommendation Service*, i.e., when to perform the load shedding, the actual QoS, i.e., the power interruption, the actual cost saving, i.e., the monthly electricity charge, and the corresponding optimal parameters and values as inputs, I evaluate the parametric model templates through the *Model Accuracy and Quality Evaluation Service*. Based upon the differences in terms of the load shedding, the QoS, and the electricity charge, this evaluation module generates a MTSA query construct to update the model templates accordingly. After that, the energy planners can use the updated templates with the input time series, QoS, and requirements to repeat the same process to learn a new set of optimal peak demand bounds for monitoring in the future pay period.

More details and considerations about this case study are presented in Chapter 8.

7.7 Chapter Summary

Using the experimental case study on the campus microgrids at George Mason University, I demonstrate the MTSA query language to express and deliver the services, including querying, monitoring, recommendation, and learning, to domain users. I also identify a hybrid-based model, Multivariate Time Series Analytics – Parameter Estimation, to solve a general class of problems in which the objective function is maximized or minimized from the optimal decision parameters regardless of particular time points. This model allows domain experts to include multiple types of constraints, e.g., global constraints and monitoring constraints. At the end, I develop a parameter learning architecture from which the parameter learning event is transformed into the IBM OPL construct by the MTSA compiler. This OPL construct is then sent to the CPLEX solver to learn the optimal decision parameters that are returned to the learning event. To further prove the capability of the MTSA-PE model, I conduct two real case students at GMU, which are described in Chapter 8. Using the electric power microgrids at GMU as examples, I illustrate how the MTSA-PE model with the external solver to solve the energy problems, including the determination of optimal peak demand bounds and the decision on the best energy investment options.

CHAPTER 8. CASE STUDIES: DECISION-GUIDED LOAD SHEDDING (DGLS) SYSTEM AND ENERGY INVESTMENT (DGEI) FRAMEWORK

In this chapter, I conduct two real case studies, i.e., the Decision-Guided Load Shedding (DGLS) system for optimal load shedding in electric power microgrids and the Decision-Guided Energy Investment (DGEI) framework for optimal power, heating, and cooling capacity investment, to demonstrate the efficiency and expressiveness of the MTSA-PE model.

8.1 DGLS System for Optimal Load Shedding in Electric Power Microgrids

8.1.1 Introduction

Increasing electricity demand has been widely recognized as a global trend in every business and industry. Population growth and economic development are among the key factors that lead to a higher total electricity consumption and a peak demand usage which, in turn, result in a rising energy cost to consumers. In this case study, I focus on the management of peak power demand within microgrids of commercial and industrial customers in order to minimize energy costs and maximize customers' savings while preserving the desired quality of service (QoS) in terms of power interruption.

Typically, it is considerably more expensive to generate electric power for the peak demand. In addition to a higher electricity cost, the peak demand of electric power also results in unpredictable, demand-side power fluctuations, as well as the possible misbalance between the power supply and the customers' demand, which cause power system outages. Existing approaches to solve the power system outage by electric power companies can be roughly divided into two categories: diverse power-load shedding schemes [59, 60, 61, 62, 63] and various time-differentiated pricing models [64, 65, 66, 67, 68, 69, 70, 71, 72]. The former approach uses the frequency magnitude, the frequency decline rate, or both of them of the power system to determine when the power load should be shed so that a complete balance between the system supply and the customers' demand can be made. Specifically, if the frequency magnitude drops below a certain threshold, the frequency decline rate reaches a certain limit, or a combination of both, a certain amount of power load is shed in order to rebalance the supply and demand.

The latter approach is to use various time-differentiated pricing models rather than a common average-pricing scheme. The average-pricing scheme charges the customers the average price over a certain period of time of power consumption. However, this pricing scheme does not incentivize the customers to shift power usage to lower demand periods, and thus reduce the total peak demand. To address this issue, various time-differentiated pricing models, such as real-time pricing (RTP), day-ahead pricing (DAP), and time-of-use pricing (TOUP) have been proposed. All of these pricing models reflect the fluctuating prices to the end customers so that they pay what the electricity is worth at different periods of a day. Specifically, these time-differentiated pricing models encourage the customers to shift operations and appliances to the off-peak hours so that their electricity costs can be reduced and the occurrence of the power system outage can be prevented.

However, an important question is how the commercial and industrial customers - for example, at the George Mason University (GMU) campuses, an unusually high peak demand usage of just a few minutes may significantly increases the cost of the electric bill for the following year - should respond to those time-differentiated pricing approaches. Answering this question is exactly the focus of this study. To mitigate the peak demand problem, the key idea is to learn an optimal peak demand bound over historical and projected electric power demands for each future pay period. This optimal peak demand bound is then used to monitor the prospective demand usage in any time interval of that future pay period. Once the demand usage exceeds the bound, some electricity loads are shed by shutting down some electric account units so that the peak demand charge can be controlled.

To address this problem, in this study, I propose and develop a decision guidance system, called DGLS, for load shedding of electric power in microgrids in order to minimize energy costs and maximize customers' savings while preserving the desired quality of service (QoS) in terms of power interruption. More specifically, the DGLS system is designed to support energy managers to forecast electric power demand over a time horizon, use the predicted peak demand usage to optimize the peak demand bound for every monthly pay period, continuously monitor the hourly electricity demand, and shed load when the demand exceeds the optimal peak demand bound using a service prioritization scheme. More specifically, the technical contributions of this chapter include (1) the design of the DGLS system, (2) the development of a Multivariate Time Series Analytics – Parameter Estimation (MTSA-PE) model for the peak demand optimization, which is both accurate and efficient, and its implementation using the IBM Optimization Programming Language (OPL), and (3) an experimental case study for the GMU university campus microgrid, which utilizes the MTSA-PE model and a proposed graphical methodology for making a trade-off between cost savings and power interruptions.

The rest of the study is organized as follows. In Section 8.1.2, I provide a descriptive overview on the DGLS system. Using the GMU energy cost problem as an example, I describe and demonstrate the MTSA-PE model and its OPL implementation in Section 8.1.3 and 8.1.4 respectively. In Section 8.1.5, I show the trade-off graph between the annual power saving and the annual power interruption, as well as explain the graph in detail on the GMU energy cost problem. In Section 8.1.6, I conclude this case study.

8.1.2 Decision-Guided Load-Shedding (DGLS) System

To better understand the load shedding problem, I consider the real case study at the George Mason University (GMU) campuses, where there are more than 33,000 students, and the total size of all the campuses is more than 800 acres, in which the electric power demand across those expanding campuses is expected to increase. The increase in electric power consumption results in a higher electricity cost, which is composed of the two main components: (1) a total kilowatt-hour (kWh) charge, i.e., the charge for the total electricity consumption, and (2) an Electricity Supply (ES) service charge, i.e., the charge for the peak demand usage in any 30-minute interval over the past 12 months. The first total kWh charge is priced particularly high during the business office hours between 09:00 a.m. and 06:00 p.m. from Monday to Friday. The second ES service charge is a proxy for the cost of capital investment for power generation capacity, since the power company, such as Virginia Electric and Power Company, needs to build generation, transmission, and distribution facilities that are capable of supporting the peak demand usage, even though the average power demand could be considerably lower. This ES service charge, i.e., the peak demand charge, amounts to approximately 30% of the electric bill in each monthly pay period and is determined based upon the electricity supply demand. This electricity supply demand is decided on the highest of either (*C1*) or (*C2*) according to the electric utility contract:

C1: The highest average kilowatt measured in any 30-minute interval of the current billing month during the on-peak hours of either:

- Between 10 a.m. and 10 p.m. from Monday to Friday for the billing months of June through September, or
- Between 7 a.m. and 10 p.m. from Monday to Friday for all other billing months.

C2: 90% of the highest kilowatt of demand at the same location as determined under (*C1*) above during the billing months of June through September of the preceding eleven billing months.

Thus it is possible that a high peak demand usage just for one minute of electricity consumption over the past year would result in a very significant increase in the total charge of the electric bill of the next monthly pay period. Therefore, controlling the peak demand usage is crucial for decreasing the electricity cost.

To address this peak demand problem, I propose the Decision-Guided Load-Shedding (DGLS) System, shown in Figure 7. This system has four main components:

Energy Management System (EMS), Load Shedding Priority Controller (LSPC), DGLS Optimizer, and Demand Prediction Learner (DPL). The EMS is a system of computer-aided tools to monitor, control, and optimize the performance of the microgrid. One operation of the EMS is to monitor and receive the meter/sensor data from the electric power microgrid and then sends those historical power-demand data to the DPL, which uses the received data and the Energy Manager's Facility Expansion Plan, e.g., the planned size of the future area increased at the GMU Fairfax campus in the next academic year, to generate the predicted electric power demand over a time horizon, e.g., two years. Another operation of the EMS is to issue the control command to the microgrid to shut down some electric account units to prevent the electricity demand from exceeding the peak demand bound. That control command generated and initiated by the EMS to the microgrid is based on the load-shedding command inputted from the LSPC, which receives the input data and information, i.e., the load shedding prioritization scheme from the Energy Manager and the optimal peak demand bound from the DGLS optimizer, to generate the control command. The DGLS optimizer utilizes the input information, including the QoS requirements, e.g., the annual maximal power interruption in kWh, and the electric utility contractual terms, e.g., the ES service charge deterination (*C1* and *C2*), from the Energy Manager, as well as the predicted electricity demand over a time horizon, e.g., two years from 2012 to 2013, from the DPL, to generate a decision optimization model described in Section 8.1.3. This model is used to learn the optimal peak demand bound, which is described in Section 8.1.4, as an input to the LSPC. Note that the DGLS optimizer learns the peak demand bound monthly in advance for the next monthly pay period, and the EMS monitors the real-time demand on an hourly basis for preventing the demand from exceeding the learned, optimal peak demand bound.

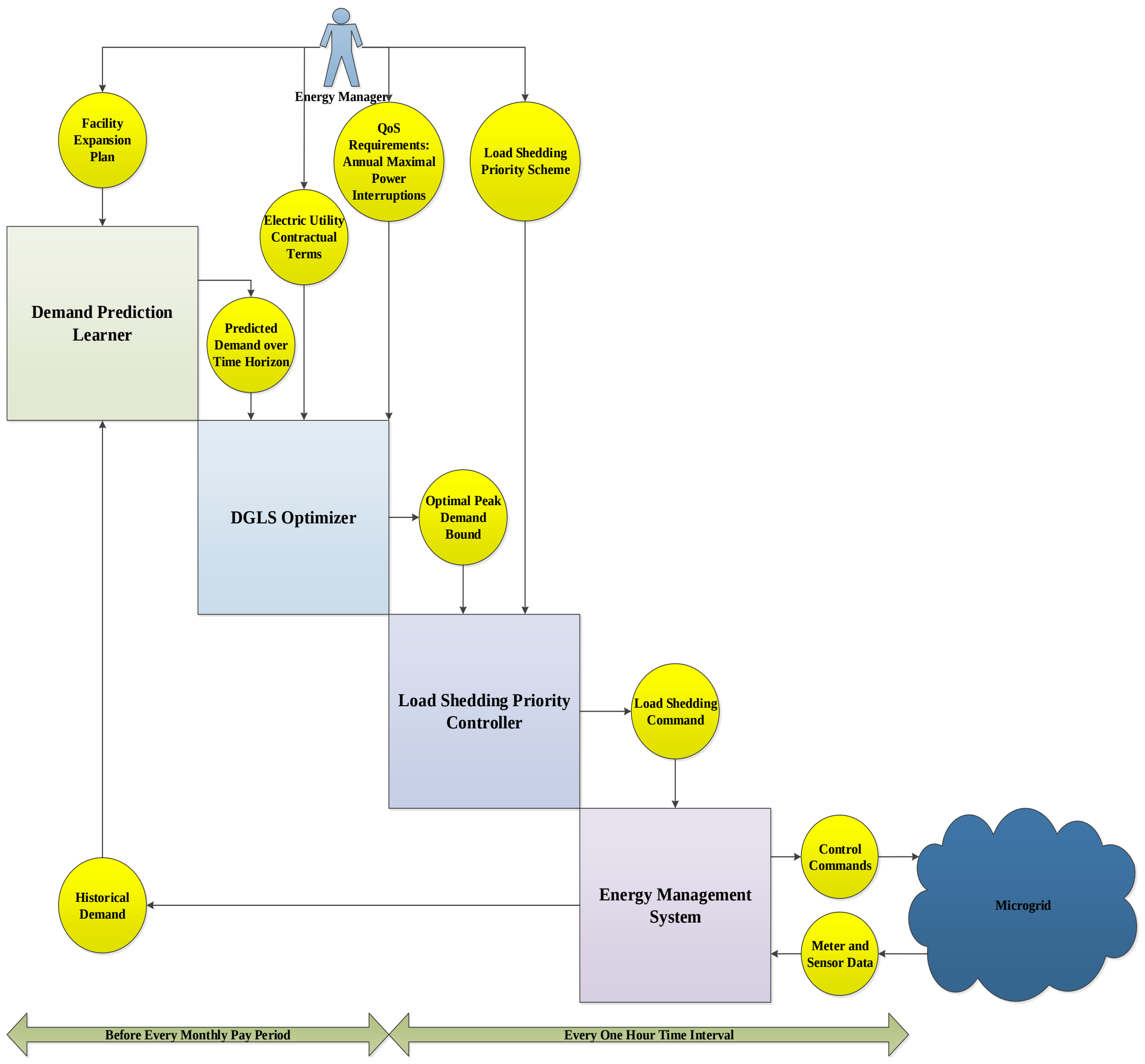

Energy Manager
Facility Expansion Plan
QoS Requirements: Annual Maximal Power Interruptions
Load Shedding Priority Scheme
Electric Utility Contractual Terms
Demand Prediction Learner
Predicted Demand over Time Horizon
DGLS Optimizer
Optimal Peak Demand Bound
Load Shedding Priority Controller
Load Shedding Command
Energy Management System
Control Commands
Meter and Sensor Data
Microgrid
Historical Demand
Before Every Monthly Pay Period
Every One Hour Time Interval

Figure 7. Decision-Guided Load-Shedding (DGLS) System

8.1.3 The MTSA-PE Optimization Model

To formulate a MTSA-PE optimization model to learn an optimal peak demand bound for every monthly pay period requires three input data sets: the historical and predicted electricity demand over a time horizon, the electric utility contractual terms of each pay period payment, and the maximal power interruptions allowed per year. Using the GMU energy cost problem as an example, I explain the terminology and the optimization problem formulation used in this case study, which are shown in Table 28, Table 29, and Table 30 respectively. In addition to those, Table 31 includes the descriptions for all the constant values, from the electric utility contract, used in the model for the GMU energy cost problem.

Table 28. Input Data Set from the Demand Prediction Learner

Input Data Item	Description
Electric Power Demand	demandKw(t) ≥ 0, where $-8759 \leq t \leq 17520$.
Pay Period	$-11 \leq$ payPeriod(t) ≤ 24.
Year	$2011 \leq$ year(t) ≤ 2013.
Month	$1 \leq$ month(t) ≤ 12.
Day	$1 \leq$ day(t) ≤ 31.
Hour	$0 \leq$ hour(t) ≤ 23.
WeekDay	$0 \leq$ weekDay(t) ≤ 6.

t is a hourly time interval denoted by an integer value, which indicates a historical data if t is less than or equal to zero, e.g., $-8759 \leq t \leq 0$ for the year 2011, and a future forecasted data if t is greater than zero, e.g., $1 \leq t \leq 17520$ for the year 2012 and 2013.

payPeriod is a monthly pay period denoted by an integer value, which indicates a historical data if payPeriod is less than or equal to zero, e.g., $-11 \leq$ payPeriod ≤ 0 for the year 2011, and a future forecasted data if payPeriod is greater than zero, e.g., $1 \leq$

$\text{payPeriod} \leq 24$ for the year 2012 and 2013. Each payPeriod corresponds to a range of ts; for example, the payPeriod = 1 consists of ts from 1 to 730.

Table 29. Input Data Set from the Energy Manager

Input Data Item	Description
Annual Maximal Power Demand Interruption	annualBound $\in Z^*$ is the annual maximal power demand interruption.

Actual Power Consumption Per Time Interval	kW[t] is the actual power consumption for each time interval that satisfies the below constraints:
	
	if ($t \leq 0 \lor$ demandKw(t) $\leq$ peakDemandBound[payPeriod(t)])
	    kW[t] = demandKw(t);
	
	else if (demandKw(t) > peakDemandBound[payPeriod(t)])
	    kW[t] = peakDemandBound[payPeriod(t)];,
	
	where $-8759 \leq t \leq 17520$, demandKw(t) ≥ 0, $-11 \leq$ payPeriod(t) ≤ 24, and peakDemandBound[payPeriod(t)] is the decision parameter for every pay period.
	
	It means that if the time interval is historical or the actual demand demandKw(t) is less than or equal to the peak demand bound, the actual power consumption kW[t] has to be bound by demandKw(t). However, if the time interval is positive and the actual demand demandKw(t) is greater than the peak demand bound, kW[t] has to be bound by the peak demand bound instead.

Peak Demand Usage Per Pay Period	payPeriodSupplyDemand[p] is the peak demand usage per pay period, which is defined in the below constraints according to the electric utility contract, i.e., *C1* and *C2*:
	
	if (payPeriod(t) == p $\land$ weekday(t) $\geq 1 \land$ weekday(t) $\leq 5 \land$ ((month(t) $\geq 6 \land$ month(t) $\leq 9 \land$ hour(t) $\geq 10 \land$ hour(t) ≤ 22) $\lor$ (month(t) $\leq 5 \land$ month(t) $\geq 10 \land$ hour(t) $\geq 7 \land$ hour(t) ≤ 22)))
	    payPeriodSupplyDemand[p] $\geq$ kW[t];
	
	else if (month(t) $\geq 6 \land$ month(t) $\leq 9 \land$ payPeriod(t) $\geq p - 11 \land$ payPeriod(t) < p $\land$ weekday(t) $\geq 1 \land$ weekday(t) $\leq 5 \land$ hour(t) $\geq 10 \land$ hour(t) ≤ 22)
	    payPeriodSupplyDemand[p]) ≥ 0.9 * kW[t];,
	
	where $-8759 \leq t \leq 17520$, $-11 \leq p \leq 24$, $1 \leq$ payPeriod(t) ≤ 24, $0 \leq$ weekDay(t) ≤ 6, $1 \leq$ month(t) ≤ 12, and $0 \leq$ hour(t) ≤ 23.

Generation Demand Charge	generationDemandCharge[p] = 8.124 * payPeriodSupplyDemand[p] is the ES service charge, i.e., the peak demand charge, where $1 \leq p \leq 24$.

Aggregated Annual Maximal Power Interruption over the future time intervals from 2012 to 2013	∑(demandKw(t) – kW[t]) ≤ 2 * annualBound is the aggregated annual maximal power interruptions for two years, where 1 ≤ t ≤ 17520.
Peak Demand Bound	peakDemandBound[p] cannot exceed payPeriodSupplyDemand[p], i.e., peakDemandBound[p] ≤ payPeriodSupplyDemand[p], where 1 ≤ p ≤ 24.
Total Power Consumption Per Pay Period	payPeriodKwh[p] is the total power consumption in each pay period, i.e., payPeriodKwh[p] = ∑kW[t] * timeIntervalSize**,** where -8759 ≤ t ≤ 17520, 1 ≤ p ≤ 24, payPeriod(t) = p, and timeIntervalSize = 1.

Total kWh Charge Per Pay Period	payPeriodKwhCharge[p] is total kWh charge per pay period, i.e., payPeriodKwhCharge[p] ≥ 0, which satisfies the below constraints according to the electric utility contract:
	
	if (payPeriodKwh[p] ≤ 24000)
	    payPeriodKwhCharge[p] = 0.01174 * payPeriodKwh[p]
	
	else if (payPeriodKwh[p] ≤ 210000 + extraKwhBound[p])
	    payPeriodKwhCharge[p] = 0.01174 * 24000 + 0.00606 * (payPeriodKwh[p] – 24000)
	
	else
	    payPeriodKwhCharge[p] = 0.01174 * 24000 + 0.00606 * (186000 + extraKwhBound[p]) + 0.00244 * (payPeriodKwh[p] – (210000 + extraKwhBound[p])),
	
	where extraKwhBound[p] = 210 * (payPeriodSupplyDemand[p] – 1000) and 1 ≤ p ≤ 24.

Total Cost Per Pay Period	The total cost per pay period is the sum of payPeriodKwhCharge[p] and generationDemandCharge[p], i.e., (payPeriodKwhCharge[p] + generationDemandCharge[p]), where 1 ≤ p ≤ 24.
Total Cost of all the PayPeriods	The total cost of all the PayPeriods is the aggregations of all the total costs for each pay period, i.e., totalCost = ∑(payPeriodKwhCharge[p] + generationDemandCharge[p]), where 1 ≤ p ≤ 24.

Table 30. The MTSA-PE Optimization Model for the GMU Energy Cost Problem

Problem and Solution
Problem: <$\boldsymbol{S}$, $\boldsymbol{P}$, $\boldsymbol{C_P}$, $\boldsymbol{C_M}$, $\boldsymbol{O}$> $\boldsymbol{S} = \{demandKw, payPeriod, year, month, day, weekDay, hour\}$, where $demandKw(t) \geq 0, -8759 \leq t \leq 17520, -11 \leq payPeriod(t) \leq 24, 2011 \leq year(t) \leq 2013, 1 \leq month(t) \leq 12, 1 \leq day(t) \leq 31, 0 \leq weekDay(t) \leq 6, 0 \leq hour(t) \leq 23$

$\boldsymbol{P}$ = $\{peakDemandBound, payPeriodSupplyDemand, kW, payPeriodKwhCharge, payPeriodKwh, generationDemandCharge, extraKwhBound\}$,

where $peakDemandBound[p] \geq 0, payPeriodSupplyDemand[p] \geq 0, kW[t] \geq 0, payPeriodKwhCharge[p] \geq 0, payPeriodKwh[p] \geq 0, generationDemandCharge[p] \geq 0, extraKwhBound[p] \geq 0, -8759 \leq t \leq 17520, 1 \leq p \leq 24$

$\boldsymbol{C_P}$ = $\{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9\}$, where

$C_1 = (\forall t \in TH, p \in FuturePH): (payPeriodKwh[p] == \sum_{p=payPeriod(t)} kW[t])$,

$C_2 = (\forall p \in FuturePH): (generationDemandCharge[p] == 8.124 * payPeriodSupplyDemand[p])$,

$C_3 = (\forall p \in FuturePH): (extraKwhBound[p] == 210 * (payPeriodSupplyDemand[p] - 1000))$,

$C_4 = (\forall t \in TH, p \in FuturePH): ((payPeriod(t) == p \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge ((hour(t) \geq 10 \wedge hour(t) \leq 22 \wedge month(t) \geq 6 \wedge month(t) \leq 9) \vee ((hour(t) \geq 7 \wedge hour(t) \leq 22) \wedge (month(t) \leq 5 \vee month(t) \geq 10)))) \rightarrow (payPeriodSupplyDemand[p] \geq kW[t]))$,

$C_5 = (\forall t \in TH, p \in FuturePH): (((payPeriod(t) \geq p - 11 \wedge payPeriod(t) < p) \wedge (weekDay(t) \geq 1 \wedge weekDay(t) \leq 5) \wedge (month(t) \geq 6 \wedge month(t) \leq 9) \wedge (hour(t) \geq 10 \wedge hour(t) \leq 22)) \rightarrow (payPeriodSupplyDemand[p] \geq 0.9 * kW[t]))$,

$C_6 = (\forall p \in FuturePH): (peakDemandBound[p] \leq payPeriodSupplyDemand[p])$,

$C_7 = (\forall p \in FuturePH): (peakDemandBound[p] \geq 0)$

$C_8 = (\forall p \in FuturePH): ((payPeriodKwh[p] \leq 24000 \rightarrow payPeriodKwhCharge[p] == 0.01174 * payPeriodKwh[p]) \wedge (payPeriodKwh[p] \geq 24000 \wedge payPeriodKwh[p] \leq 210000 + extraKwhBound[p] \rightarrow payPeriodKwhCharge[p] == 0.01174 * 24000 + 0.00606 * (payPeriodKwh[p] - 24000)) \wedge (payPeriodKwh[p] \geq 210000 + extraKwhBound[p] \rightarrow payPeriodKwhCharge[p] == 0.01174 * 24000 + 0.00606 * (186000 + extraKwhBound[p]) + 0.00244 * (payPeriodKwh[p] - (24000 + 186000 + extraKwhBound[p]))))$

$C_9 = (\forall t \in TH): (\sum_{t=1}^{17520}(demandKw(t) - kw[t]) \leq annualBound * 2)$

$\boldsymbol{C_M}$ = $\{C_{10}, C_{11}\}$, where

$C_{10} = (\forall t \in TH, p \in PH): ((t \geq 1 \wedge p == payPeriod(t) \wedge demandKw(t) > peakDemandBound[p]) \rightarrow (kW[t] == peakDemandBound[p]))$

$C_{11} = (\forall t \in TH, p \in PH): (((demandKw(t) \leq peakDemandBound[p] \wedge p == payPeriod(t)) \vee (t \leq 0)) \rightarrow (kW[t] == demandKw(t))$,

$\boldsymbol{O} = \sum_{p=1}^{24}(payPeriodKwhCharge[p] + generationDemandCharge[p])$

Solution:

$\underset{\boldsymbol{P}}{argmin}\, \boldsymbol{O}(\boldsymbol{P})$

$subject\ to\ \boldsymbol{C_P}(\boldsymbol{P}) \wedge \boldsymbol{C_M}(\boldsymbol{P}))$

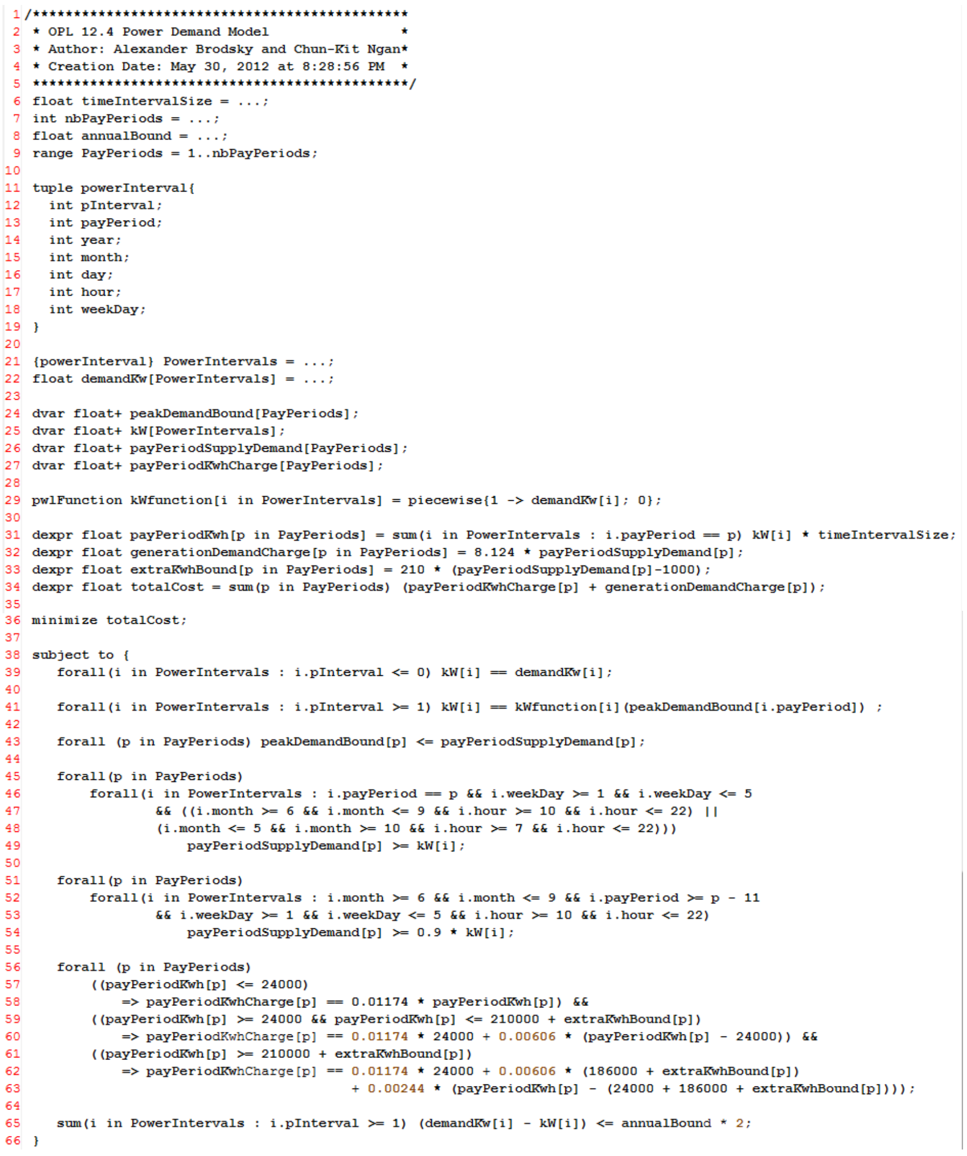
From the MTSA-PE model shown in Table 30, I can see that *demandKw(t)* is assigned to *kW[t]* when the time interval *t* is in the past or *demandKw(t)* is less than or equal to *peakDemandBound[p]*, where *payPeriod(t) = p*. However, when the time interval *t* is in the future and *demandKw(t)* is greater than *peakDemandBound[p]*, *peakDemandBound[p]* is assigned to *kW[t]*. *peakDemandBound[p]* is also bound by *payPeriodSupplyDemand[p]*, which is restricted by the electric contractual constraints, i.e., *C1* and *C2*. *payPeriodKwhCharge[p]* is determined based upon the electric utility contract of the electric bill, which is described in Table 29. The total allowed power interruption over the future power intervals cannot be more than the twice of *annualBound*, where *annualBound* is the annual maximal power demand interruption over the future time intervals from 2012 and 2013. Lastly, all the decision control variables, including *peakDemandBound[p]*, *payPeriodSupplyDemand[p]*, *payPeriodKwh[p]*, *extraKwhBound[p]*, *payPeriodKwhCharge[p]*, and *kW[t]* must be restricted in sign, i.e., non-negative real values, where $1 \leq p \leq 24$ and $-8759 \leq t \leq 17520$.

Table 31. Descriptions for the Constant Values in the MTSA-PE Optimization Model for the GMU Energy Cost Problem

Constant	Description
0.9	Percentage of the highest kW of demand during the billing months of June through September of the preceding 11 billing months.
8.124	Amount ($) of Electricity Supply (ES) demand charged per kW.
24000	First ES kWh
0.01174	Amount ($) of the first 24000 ES kWh charged per kWh
186000	Next ES kWh
0.00606	Amount ($) of the next 186000 ES kWh charged per kWh
210000	Sum of the first ES kWh and the next ES kWh
0.00244	Amount ($) of the additional ES kWh charged per kWh
1000	kW of ES demand
210	kWh for each ES kW of demand over 1000 kW

8.1.4 The OPL Implementation for the MTSA-PE Optimization Model

The MTSA-PE optimization model has been implemented by using the IBM OPL language. Using the GMU historical data of power usage in the past years, e.g., 2011, and its predicted electricity demand over a future time horizon, e.g., 2012 and 2013, I use the OPL language to implement and demonstrate the MTSA-PE optimization model for the GMU energy cost problem, which is shown in Figure 8, as an example. First, the value 24, i.e., the total 24 months from 2012 to 2013, is assigned to the variable *nbPayPeriods* in the line number 7. The value starting from 0 to 1500000000 kWh is assigned to the variable *annualBound*, that is, the maximal annual power interruption allowed in the line number 8, one by one to demonstrate the tradeoff in Section 8.1.5. From the line number 11 to 19, I declare a tuple of a power interval that has the attributes, including *pInterval*, *payPeriod*, *year*, *month*, *day*, *hour*, and *weekDay*. The line number 21 declares the *PowerIntervals* that include both the past and the future power intervals. The line number 22 declares the *demandKw[PowerIntervals]* array. The line number 24, 25, 26, and 27 declare the decision parameter sets, including the *peakDemandBound[PayPeriods]*, *kW[PowerIntervals]*, *payPeriodSupplyDemand[PayPeriods]*, and *payPeriodKwhCharge[PayPeriods]*. The line number 31, 32, 33, and 34 declare decision parameter expressions, i.e., *payPeriodKwh[PayPeriods]*, *generationDemandChange[PayPeriods]*, and *extraKwhBound[PayPeriods]*. The total cost, which is declared on the line number 34, is minimized on the line number 36. All the constraints, *C1 – C9*, are declared from the line number 31, 32, 33, and 43 to 66. The constraints *C10* and *C11* are declared from the line number 29, 39, and 41.



```
/***********************************************
 * OPL 12.4 Power Demand Model                 *
 * Author: Alexander Brodsky and Chun-Kit Ngan*
 * Creation Date: May 30, 2012 at 8:28:56 PM  *
 **********************************************/
 float timeIntervalSize = ...;
 int nbPayPeriods = ...;
 float annualBound = ...;
 range PayPeriods = 1..nbPayPeriods;

 tuple powerInterval{
   int pInterval;
   int payPeriod;
   int year;
   int month;
   int day;
   int hour;
   int weekDay;
 }

 {powerInterval} PowerIntervals = ...;
 float demandKw[PowerIntervals] = ...;

 dvar float+ peakDemandBound[PayPeriods];
 dvar float+ kW[PowerIntervals];
 dvar float+ payPeriodSupplyDemand[PayPeriods];
 dvar float+ payPeriodKwhCharge[PayPeriods];

 pwlFunction kWfunction[i in PowerIntervals] = piecewise{1 -> demandKw[i]; 0};

 dexpr float payPeriodKwh[p in PayPeriods] = sum(i in PowerIntervals : i.payPeriod == p) kW[i] * timeIntervalSize;
 dexpr float generationDemandCharge[p in PayPeriods] = 8.124 * payPeriodSupplyDemand[p];
 dexpr float extraKwhBound[p in PayPeriods] = 210 * (payPeriodSupplyDemand[p]-1000);
 dexpr float totalCost = sum(p in PayPeriods) (payPeriodKwhCharge[p] + generationDemandCharge[p]);

 minimize totalCost;

 subject to {
    forall(i in PowerIntervals : i.pInterval <= 0) kW[i] == demandKw[i];

    forall(i in PowerIntervals : i.pInterval >= 1) kW[i] == kWfunction[i](peakDemandBound[i.payPeriod]) ;

    forall (p in PayPeriods) peakDemandBound[p] <= payPeriodSupplyDemand[p];

    forall(p in PayPeriods)
        forall(i in PowerIntervals : i.payPeriod == p && i.weekDay >= 1 && i.weekDay <= 5
                && ((i.month >= 6 && i.month <= 9 && i.hour >= 10 && i.hour <= 22) ||
                (i.month <= 5 && i.month >= 10 && i.hour >= 7 && i.hour <= 22)))
                    payPeriodSupplyDemand[p] >= kW[i];

    forall(p in PayPeriods)
        forall(i in PowerIntervals : i.month >= 6 && i.month <= 9 && i.payPeriod >= p - 11
                && i.weekDay >= 1 && i.weekDay <= 5 && i.hour >= 10 && i.hour <= 22)
                    payPeriodSupplyDemand[p] >= 0.9 * kW[i];

    forall (p in PayPeriods)
        ((payPeriodKwh[p] <= 24000)
            => payPeriodKwhCharge[p] == 0.01174 * payPeriodKwh[p]) &&
        ((payPeriodKwh[p] >= 24000 && payPeriodKwh[p] <= 210000 + extraKwhBound[p])
            => payPeriodKwhCharge[p] == 0.01174 * 24000 + 0.00606 * (payPeriodKwh[p] - 24000)) &&
        ((payPeriodKwh[p] >= 210000 + extraKwhBound[p])
            => payPeriodKwhCharge[p] == 0.01174 * 24000 + 0.00606 * (186000 + extraKwhBound[p])
                                        + 0.00244 * (payPeriodKwh[p] - (24000 + 186000 + extraKwhBound[p])));

    sum(i in PowerIntervals : i.pInterval >= 1) (demandKw[i] - kW[i]) <= annualBound * 2;
 }
```

Figure 8. The OPL Implementation for the GMU Energy Cost Problem

8.1.5 Trade-off Graph for the Annual Power Interruption (kWh) vs. the Projected Annual Saving (USD) for the GMU Energy Cost Problem

Using the OPL language to formulate and solve the GMU energy cost problem as an example, I plot the trade-off graph between the annual power interruption (kWh) and the projected annual saving (USD) that is shown in Figure 9. From the graph, I see that there are three clear regions. Between 0 and 10000 kWh, the curve of the graph is increasing very slowly. It means that an annual power interruption in this region does not bring a significant amount of annual saving to the university although the power supply service to the university is kind of minimal. When the power interruption starts after the 10000 kWh, the curve of the graph starts getting steep and keeps increasing that leads to a higher amount of annual saving for the university, but the power interruption begins bringing a significant impact on the GMU campus. In the last portion of the graph, I find that even though the power interruption reaches 95000000 kWh or more, the annual saving still remains the same, i.e., $1551246.81. It means that increasing the power interruption infinitely does not result in an increasing annual saving limitlessly. Such infinite power interruption only worsens the power supply service to the entire Fairfax campus at GMU. Using this given trade-off graph, the GMU Energy Manager can determine which level of desired quality of service (QoS) in terms of power interruption on the campus can be preserved while earning a desirable annual saving.

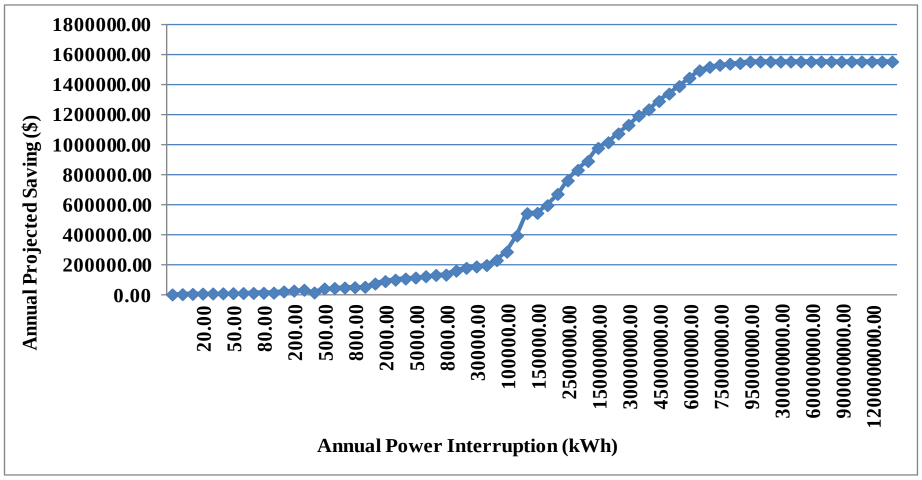


Figure 9. Trade-off Graph between the Annual Power Interruption and the Projected Annual Saving at George Mason University

8.1.6. Case Summary

In this case study, I propose and report on the development of DGLS, a Decision-Guidance System for Load Shedding of electric power in microgrids in order to minimize energy costs and maximize customers' savings while preserving the desired quality of service (QoS) in terms of power interruption. The DGLS system is designed to support energy managers to forecast electric power demand over a time horizon, use the predicted peak demand usage to optimize the peak demand bound for every monthly pay period, continuously monitor the hourly electricity demand, and shed load when the demand exceeds the optimal peak demand bound using a service prioritization scheme. The core technical challenge is the development of the MTSA-PE model for the peak demand optimization that is very accurate in terms of the electric contractual terms and engineering constraints, and yet efficient and scalable, which is done by the careful modeling of mainly continuous decision variables and using constructs that avoid introduction of combinatorics, e.g., explicit or implicit binary variables, into the model. The model has also been implemented and demonstrated by the IBM OPL language for the GMU energy cost problem.

8.2 DGEI Framework for Optimal Power, Heating, and Cooling Capacity Investment

8.2.1 Introduction

Sustainable enterprise development has been considered a significant and competitive strategy of corporate growth in manufacturing and service organizations. A significant part of sustainable development involves new technologies for local electricity, heating, and cooling generation. Making optimal decisions on planning and investment of these technologies to support commercial and industrial facilities is an involved problem because of complex operational dependencies of these technologies. This is exactly the focus of this study.

Currently, the existing approaches to support the optimization of energy plants can be divided into two categories: (1) optimal operation of an energy system and (2) a better plant-process design [73]. The former category is related to the optimized scheduling of an electric power plant. Some researchers, such as Bojic and Stojanovic [74], proposed an optimization procedure based on a MILP solver [75] to provide an operation diagram which allows users to find an optimum composition of energy consumption that minimizes the operating expenses of an energy system. The latter approach includes the analysis of simulations carried out to determine the most suitable matching between a plant and its loads that could increase the plant power output. Some researchers, e.g., Savola et al., [76] did extensive research to propose an off-design simulation and mathematical modelling of the operation at part loads and a Mixed-Integer Non-Linear Programming (MINLP) optimization model for increasing power production [77, 78].

However, neither of the above approaches considers optimizing the complex interactions between the existing components and the newly added energy equipment that would result in a higher operating cost, such as the charges on electricity and gas consumptions, as well as significant environmental impacts, i.e., greenhouse gas (GHG) emissions, e.g., carbon dioxide (CO_2) and mono-nitrogen oxide (NO_x). Without considering such interactions for every time interval over an investment time horizon, it would be impossible to make optimal recommendations on energy planning and investment.

Thus this study focuses on addressing the above shortcomings. More specifically, the contributions of this study are as follows. First, I propose a Decision-Guided Energy Investment (DGEI) Framework shown in Figure 10. Given electricity, heating, and cooling generation processes, utility contracts, historical and projected demand, facility expansions, and Quality of Service (QoS) requirements, the DGEI framework is designed to recommend optimal settings of decision control variables. These decision control variables include the amount of electricity, heating, and cooling that is generated by the supply of water and gas, which is inputted to each deployed component in every time interval. The goal of the DGEI framework is to learn optimal values of those decision control variables in order to minimize the total operating cost within the required quality of service and within the bound for GHG emissions, as well as to take into account all components' interactions. Second, to support the DGEI framework, I develop a Multivariate Time Series Analytics – Parameter Estimation (MTSA-PE) model to solve the adjusted cost minimization problem. Furthermore, I implement the MTSA-PE model by using the IBM Optimization Programming Language (OPL). Third, I propose an analytical and graphical methodology to determine the best available investment option based upon the evaluation parameters shown in Figure 10. The parameters include investment costs, maintenance expenditures, replacement charges, operating expenses, cost savings, return on investment (ROI), and GHG emissions. Finally, I use the methodology and the DGEI framework to conduct an experimental case study on the microgrid at the Fairfax campus of George Mason University (GMU). This study has been conducted and used by the GMU Facilities Management Department (FMD) to make actual investment decisions.

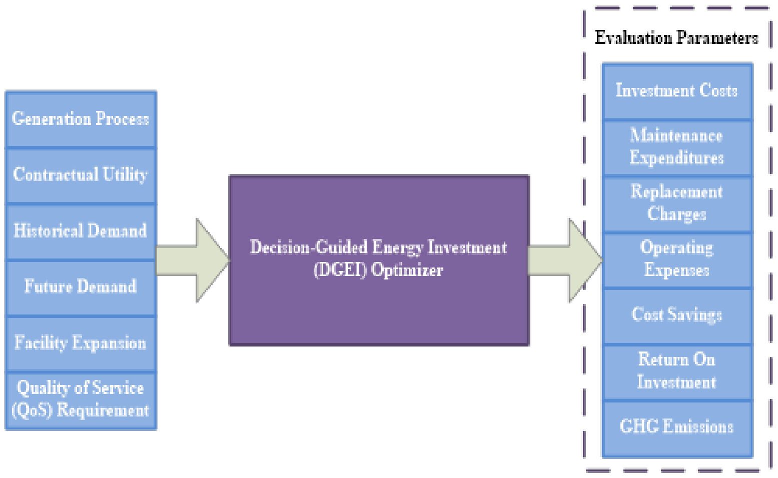


Figure 10. Decision-Guided Energy Investment (DGEI) Framework

The rest of the study is organized as follows. Using the GMU Fairfax campus microgrid as an example, I describe its energy investment problem in Section 8.2.2. I explain my DGEI framework and optimization model in Section 8.2.3 and demonstrate the OPL implementation in Section 8.2.4. In Section 8.2.5, I present the analytical and graphical methodology to determine an optimal investment option. In Section 8.2.6, I conduct the experimental analysis on the GMU energy investment case and illustrate the relationships among the investment costs, ROI, and GHG emissions of the various options in tabular and graphical formats. I also explain and draw the conclusion for the investment options from the graphs and tables in detail on the GMU energy investment problem. In Section 8.2.7, I conclude the study.

8.2.2 Problem Description of Real Case Study

Consider the real case study at GMU, in which the GMU Facilities Management Department (FMD) is planning to extend and or expand the existing energy equipment in order to meet the current and future demand of electricity, heating, and cooling across the expanding Fairfax campus in Virginia. Presently, the GMU existing energy facilities at the Fairfax campus operate a centralized heating and cooling plant (CHCP) system and utilize the electricity purchased from the Dominion Virginia Power Company (DVPC) to satisfy all the energy demand. Over the past 10 years, the campus has experienced a significant growth on a square-foot basis in terms of land use. Since the campus continues its expansion at a rapid rate, the existing CHCP system and the electricity consumption have reached a saturated point where the current capacity and facilities will not be able to satisfy the future energy demand, i.e., electricity, heating, and cooling. For these reasons, a study has been conducted to determine the best available investment option, e.g., a new cogeneration (CoGen) plant, with regards to a possible methodology to meet the current and future electricity, heating, and cooling demand, while also addressing the optimal operations of the newly added facility with the existing energy equipment.

The diagram in Figure 11 depicts the GMU energy generation process which supplies heating, cooling, and electricity to the entire Fairfax campus. The GMU energy facilities have a CHCP system to supply the hot and cold water (see the red and blue resources) which are distributed across the facilities to the campus buildings to meet the heating and cooling demand (see the upper two sub-processes on the right), i.e., heating and air-conditioning to the buildings. To supply the heating and cooling to the campus buildings, the CHCP system needs the inputs, i.e., natural gas (see the yellow resource on the left), water (see the light blue resource on the left), and electric power (see the green resource on the left). These resources come from the gas supply, i.e., Washington Gas Light Company (WGLC), the water supply, i.e., Fairfax County Water Authority (FCWA), and the electricity supply, i.e., Dominion Virginia Power Company (DVPC), correspondingly. In addition, the facilities also need to satisfy the electricity demand across the entire campus, where the electricity demand is beyond the demand from the CHCP consumption. Any excessive electric power supply can also be resold to the DVPC (see the electricity resell on the right). Furthermore, the facilities also commit a curtailment demand (see the curtailment demand on the right) to the energy curtailment program through EnergyConnect (EC), Inc. Both the electricity resell and the curtailment commitment can bring certain revenues and savings to offset the overall operational costs on a monthly basis and the capital expenditures in the long run. The facilities also generate greenhouse gas (GHG) emissions, such as carbon dioxide (CO_2) (see the black resource at the bottom right).

Given the expansion of the GMU Fairfax campus, in addition to the increasing electricity demand, the heating and cooling demand is also expected to increase. The CHCP system will not have enough capacity to meet the future need. The GMU plan is to employ a procurement strategy, i.e., the deployment of the best available investment option, which will satisfy projected demand and minimize investment costs, maintenance expenditures, replacement charges, operating expenses, and GHG emissions, as well as maximize cost savings and return on investment (ROI) at the same time. The FMD managers are now considering some viable options. One of the considerable options is to integrate a new cogeneration (CoGen) plant (see the lower sub-process in the middle), i.e., the Combined Heating and Power (CHP) Plant [73, 79] into the existing facilities shown in Figure 11. The new CoGen plant has turbines to generate electricity to complement the electricity demand, uses the generated heat as a by-product to complement the heating demand, and collaborates with the ammonia process technology [80] to supply the cooling demand. Now, the challenging question is how to analytically determine the best investment option that satisfies all the energy demand, i.e., electricity, heating, and cooling, at the lowest operating costs.

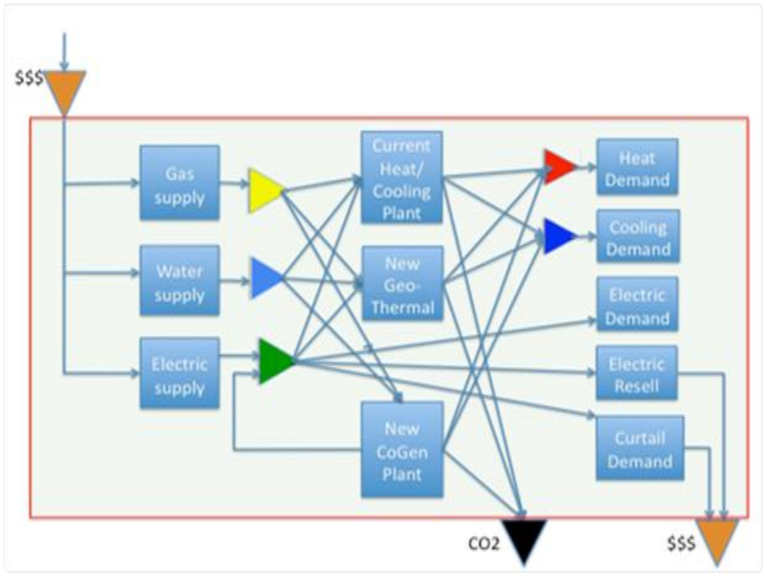


Figure 11. Prospective Heating, Cooling, and Electric Power Facilities at the GMU Fairfax Campus

8.2.3 Decision-Guided Energy Investment Framework and MTSA-PE Model

To answer the above question, I propose the DGEI framework depicted in Figure 10. This framework is composed of six energy-investment libraries, i.e., Energy Generation Process (EGP), Energy Contractual Utility (ECU), Energy Historical Demand (EHD), Energy Future Demand (EFD), Energy Facility Expansion (EFE), Quality of Service (QoS) requirements, and a DGEI optimizer. The EGP is an extensible library that enables domain experts to construct an energy generation process to supply electricity, heating, and cooling. The ECU is a library that contains energy contractual terms for calculating bill utilities, e.g., an electricity bill, a water bill, and a gas bill. The EHD and EFD are the libraries that store historical and projected energy demand respectively. The EFE library archives the facility expansion of an organization in terms of square-footage increase. The QoS library stores the QoS requirements that the energy facilities of an organization need to meet, e.g., the maximal power interruptions allowed per monthly pay period in an organization. The DGEI optimizer supports energy managers to utilize all the libraries, i.e., EGP, ECU, EHD, EFD, EFE, and QoS, as inputs to the decision optimization process, which minimizes operating expenses and maximize cost savings. This decision optimization process not only optimizes the interactions between the existing and the considerable energy facility options but also minimizes the environmental impacts on the surroundings, i.e., minimizing the GHG emissions. In addition to the GHG emissions, energy managers also utilize (1) return on investment (ROI), i.e., the gain return efficiency among different investments, (2) the investment costs, i.e., an amount spent to acquire a long-term asset, and (3) equipment expenses, i.e., maintenance expenditures plus replacement charges, to evaluate all the available investments and then to determine the best option.

To solve an energy investment optimization problem in terms of minimizing the operating cost and the GHG emissions is to formulate a MTSA-PE model. This model optimally learns decision control variables, which require several input data sets, i.e., the historical and projected electricity, heating, and cooling demand over a time horizon, the electric and gas contractual utility, the operational parameters and capacity constraints of the existing and the new electric power plants, as well as the energy aggregation of the supply and demand, e.g., electricity, gas, heating, and cooling, to minimize the entire operating expenses. Using the GMU energy investment optimization problem over the 10-year time horizon as an example, I explain the above terminologies used in this case study in the following subsections.

8.2.3.1 Electricity, Heating, and Cooling Demand over a Time Horizon

The electricity, heating, and cooling demand over a time horizon is the input, including the usage of the historical and projected quantities, which are provided from the GMU Facilities Management Department, to the MTSA-PE model that requires the domain users to define all (i.e., past plus future), past, and future power intervals over the 10-year time horizon.

- AllPowerIntervals is a set of all powerIntervals, where each powerInterval is a tuple which includes several attributes, i.e., pInterval, payPeriod, year, month, day, hour, and weekDay. I use negative and zero integers to represent the past time horizon and positive integers to denote the future time horizon. For example, pInterval is an hourly time interval of the energy demand, where $-8759 \leq$ pInterval ≤ 78840. payPeriod is a monthly pay period of the energy demand, where $-11 \leq$ payPeriod ≤ 108. Other attributes' intervals include $2011 \leq$ year ≤ 2020, $1 \leq$ month ≤ 12, $1 \leq$ day ≤ 31, $0 \leq$ hour ≤ 23, and $0 \leq$ weekDay ≤ 6.
- PastPowerIntervals is a set of past powerIntervals of tuples, where $-8759 \leq$ pInterval ≤ 0, $-11 \leq$ payPeriod ≤ 0, year = 2011, $1 \leq$ month ≤ 12, $1 \leq$ day ≤ 31, $0 \leq$ hour ≤ 23, and $0 \leq$ weekDay ≤ 6.
- FuturePowerIntervals is a set of future powerIntervals of tuples, where $1 \leq$ pInterval ≤ 78840, $1 \leq$ payPeriod ≤ 108, $2012 \leq$ year ≤ 2020, $1 \leq$ month ≤ 12, $1 \leq$ day ≤ 31, $0 \leq$ hour ≤ 23, and $0 \leq$ weekDay ≤ 6.

After declaring the power intervals, the quantities of electricity, heating, and cooling demand can be stored in their arrays over their power intervals. These three quantities of demand are provided by the GMU Facilities Management Department.

- demandKw[AllPowerIntervals] ≥ 0 is an array of electricity demand over the AllPowerIntervals. This array stores both the historical and the projected demand over the PastPowerIntervals and the FuturePowerIntervals respectively.
- demandHeat[FuturePowerIntervals] ≥ 0 is an array of projected heating demand over the FuturePowerIntervals.
- demandCool[FuturePowerIntervals] ≥ 0 is an array of projected cooling demand over the FuturePowerIntervals.

8.2.3.2 Electric and Gas Contractual Utility

To determine the total operating cost, I need to compute the consumption expenses of electricity and gas supply according to their utility contracts.

The consumption expenses of electricity include both the peak demand charge and the total power consumption charge that are explained in detail as follows.

Peak Demand Charge

For the electricity supply, utilityKw[AllPowerIntervals] ≥ 0 is an array of electricity supplied from the DVPC over the AllPowerIntervals.

historicUtilityKw[i] is an array of past electricity demand from the GMU, i.e., `historicUtilityKw[i] = demandKw[i]`, which satisfies the constraint, i.e., `utilityKw[i] == historicUtilityKw[i]`, where i ∈ PastPowerIntervals. This constraint is to assure that the electricity consumed by the GMU in the past year, i.e., 2011, is equivalent to the supply from the DVPC.

payPeriodSupplyDemand[p] is the peak demand usage per future pay period (p). This peak demand usage meets the below contractual constraints (*C1* and *C2*) and is determined based upon the highest of either (*C1*) or (*C2*):

C1: The highest average kilowatt measured in any hourly time interval of the current billing month during the on-peak hours of either between 10 a.m. and 10 p.m. from Monday to Friday for the billing months of June through September or between 7 a.m. and 10 p.m. from Monday to Friday for all other billing months.

C2: 90% of the highest kilowatt of demand at the same location as determined under (*C1*) above during the billing months of June through September of the preceding eleven billing months.

The logic constraints of both *C1* and *C2* can be expressed as follows:

```
if (i.payPeriod == p ∧ i.weekDay ≥ 1 ∧ i.weekDay ≤ 5 ∧
((i.month ≥ 6 ∧ i.month ≤ 9 ∧ i.hour ≥ 10 ∧ i.hour ≤ 22) ∨
(i.month ≤ 5 ∧ i.month ≥ 10 ∧ i.hour ≥ 7 ∧ i.hour ≤ 22)))
  payPeriodSupplyDemand[p] ≥ utilitykW[i]
else if (i.month ≥ 6 ∧ i.month ≤ 9 ∧ i.payPeriod ≥ p – 11 ∧
i.payPeriod ≤ p ∧ i.weekPay ≥ 1∧ i.weekDay ≤ 5 ∧ i.hour ≥ 10
∧ i.hour ≤ 22)
  payPeriodSupplyDemand[p] ≥ 0.9 * utilitykW[i];, where i ∈
```

AllPowerIntervals, p ∈ FuturePayPeriods, and 1 ≤ FuturePayPeriods ≤ 108. Using these logic constraints, I can determine the optimal peak demand usage per future pay period, which consumes more than the expected electricity supply per powerInterval from the DVPC.

generationDemandCharge[p], i.e., `generationDemandCharge[p] = 8.124 * payPeriodSupplyDemand[p];`, is the Electricity Supply (ES) service charge, i.e., the peak demand charge, where p ∈ FuturePayPeriods, and 8.124 is the dollar charge per kW.

Total Power Consumption Charge

payPeriodKwh[p] is the total power consumption per future pay period, i.e., `payPeriodKwh[p] = ∑utilitykW[i];`, where i ∈ AllPowerIntervals, p ∈ FuturePayPeriods, and i.payPeriod = p.

payPeriodKwhCharge[p] is the total kWh charge per future pay period, i.e., payPeriodKwhCharge[p] ≥ 0, which satisfies the below contractual constraints:

```
if (payPeriodKwh[p] ≤ 24000)
  payPeriodKwhCharge[p] = 0.01174 * payPeriodKwh[p]
else if (payPeriodKwh[p] ≤ 210000)
  payPeriodKwhCharge[p] = 0.01174 * 24000 + 0.00606 *
  (payPeriodKwh[p] - 24000)
else
  payPeriodKwhCharge[p] = 0.01174 * 24000 + 0.00606 *
  186000 + 0.00244 * (payPeriodKwh[p] - 210000);
```

, where p ∈ FuturePayPeriods,

0.01174 is the dollar charge of the first 24000 kWh consumed, 0.00606 is the dollar charge of the next 186000 kWh consumed, and 0.00244 is the dollar charge of the additional kWh consumed. Note that if payPeriodSupplyDemand[p] is 1000 kW or more, 210 kWh for each peak demand usage over 1000 kW is added to the total power consumption to calculate payPeriodKwhCharge[p].

Total Electricity Cost

The total electricity cost per future pay period is the sum of payPeriodKwhCharge[p] and generationDemandCharge[p], i.e., `electricCostPerFuturePayPeriod = (payPeriodKwhCharge[p] + generationDemandCharge[p]);`, where p ∈ FuturePayPeriods.

The total electricity cost of all the FuturePayPeriods is the aggregations of all the total electricity costs per future pay period, i.e., `electricCost = ∑(payPeriodKwhCharge[p] + generationDemandCharge[p]);`, where p ∈ FuturePayPeriods.

Table 32 summarizes the descriptions of all the constant values from the electric utility contract used in the MTSA-PE model for the GMU energy investment problem.

Table 32. Descriptions for the Constant Values in the MTSA-PE Optimization Model of the GMU Energy Investment Problem

Constant	Description
0.9	Percentage of the highest kW of demand during the billing months of June through September of the preceding 11 billing months
8.124	Amount ($) of Electricity Supply (ES) demand charged per kW
24000	First ES kWh
0.01174	Amount ($) of the first 24000 ES kWh charged per kWh
186000	Next ES kWh
0.00606	Amount ($) of the next 186000 ES kWh charged per kWh
210000	Sum of the first ES kWh and the next ES kWh
0.00244	Amount ($) of the additional ES kWh charged per kWh
210	kWh for each ES kW of demand over 1000 kW

Total Gas Consumption Charge

Regarding the gas supply, utilityGas[FuturePowerIntervals] ≥ 0 is an array of gas supplied from the WGLC over the FuturePowerIntervals. The total gas cost of all the FuturePowerIntervals is the aggregations of all the total gas utility per future power interval, i.e., `gasCost = (∑(utilityGas[i]/btuPerDth)) * gasPricePerDth;`, where i ∈ FuturePowerIntervals, btuPerDth = 1000000 BTU, which is the amount of energy per decatherm, and gasPricePerDth = $6.5, which is the gas charge per decatherm.

Total Operating Cost

The total operating cost is the sum of the total electricity cost of all the future pay periods and the total gas cost of all the future power intervals, i.e., `totalCost = electricCost + gasCost;`.

8.2.3.3 Operational Parameters and Capacity Constraints of the CHCP and the CoGen Plant

In addition to the supply and demand of gas and electricity, the operational parameters and the capacity constraints of the CHCP and the CoGen plant are also considered.

The CHCP Plant

For the CHCP plant, gasIntoCHCP[FuturePowerIntervals] ≥ 0 is an array of natural gas input to the CHCP over the FuturePowerIntervals to generate the heat supply. kwIntoCHCP[FuturePowerIntervals] ≥ 0 is an array of power input to the CHCP over the FuturePowerIntervals to generate the cool supply. heatOutCHCP[FuturePowerIntervals] ≥ 0 is an array of heat output from the CHCP over the FuturePowerIntervals to satisfy the partial heating demand. coolOutCHCP[FuturePowerIntervals] ≥ 0 is an array of cool output from the CHCP over the FuturePowerIntervals to satisfy the partial cooling demand. The CHCP constraints include:

- `heatOutCHCP[i] * gasPerHeatUnit ≤ gasIntoCHCP[i];`, i.e., the amount of gas consumed to generate the heat cannot be more than that of the gas input;
- `coolOutCHCP[i] * kwhPerCoolUnit ≤ kwIntoCHCP[i];`, i.e., the amount of electric power consumed to generate the cool cannot be more than that of the power input;
- `heatOutCHCP[i] ≤ chcpMaxHeatPerHr;`, i.e., the amount of heat generated cannot be more than the maximal heat output of the CHCP; and

- `coolOutCHCP[i] ≤ chcpMaxCoolPerHr;`, i.e., the amount of cool generated cannot be more than the maximal cool output of the CHCP, where i ∈ FuturePowerIntervals, gasPerHeatUnit = (1 / 0.78), and kwhPerCoolUnit = (1 / 0.94).

The CoGen Plant

For the CoGen plant, gasIntoCogen[FuturePowerIntervals] ≥ 0 is an array of gas input to the CoGen plant over the FuturePowerIntervals to generate the power supply. kwOutCogen[FuturePowerIntervals] ≥ 0 is an array of power output from the CoGen plant over the FuturePowerIntervals to satisfy the partial electricity demand. heatOutCogen[FuturePowerIntervals] ≥ 0 is an array of heat output from the CoGen plant over the FuturePowerIntervals to satisfy the partial heating demand. coolOutCogen[FuturePowerIntervals] ≥ 0 is an array of cool output from the CoGen plant over the FuturePowerIntervals to satisfy the partial cooling demand. The constraints of the CoGen plant include:

- `kwOutCogen[i] * cogenGasPerKwh ≤ gasIntoCogen[i];`, i.e., the amount of gas consumed to generate the power cannot be more than that of the gas input;
- `kwOutCogen[i] ≤ cogenMaxKw;`, i.e., the amount of power generated cannot be more than the maximal electricity output of the CoGen plant;
- `heatOutCogen[i] ≤ cogenHeatPerKwh * kwOutCogen[i];`, i.e., the amount of heat generated cannot be more than the maximal heat supply that is restricted by the power output of the CoGen plant;

- `heatOutCogen[i] ≤ cogenMaxHeatPerHr * (kwOutCogen[i]/cogenMaxKw);`, i.e., the amount of heat generated cannot be more than the maximal heat output of the CoGen plant;
- `coolOutCogen[i] ≤ (cogenMaxHeatPerHr * (kwOutCogen[i]/cogenMaxKw) - heatOutCogen[i]) * cogenHeatToCoolRatio;`, i.e., the amount of cool generated cannot be more than the maximal cool supply that is restricted by the power and heat output of the CoGen plant; and
- `coolOutCogen[i] ≤ cogenMaxCoolPerHr;`, i.e., the amount of cool generated cannot be more than the maximal cool output of the CoGen plant,

  where i ∈ FuturePowerIntervals, cogenMaxKw = 7200 kW is the maximal power output, cogenHeatPerKwh = 10300 kWh is the amount of heat generated per kWh, cogenHeatToCoolRatio = cogenMaxCoolPerHr/cogenMaxHeatPerHr is the ratio of converting heat to cool supply, cogenMaxHeatPerHr = 40000000 BTU is the maximal heat supply of the CoGen plant per hour, cogenMaxCoolPerHr = 2400 Tons is the maximal cool supply of the CoGen plant per hour, cogenGasPerKwh = gasBTUPerGallon/kWhPerGallon/cogenGasToKwhEfficiency is the amount of natural gas consumed per kWh, for gasBTUPerGallon = 114000 BTU is the amount of energy generated per gallon of gas, kwhPerGallon = 33.41 is the amount of kWh generated per gallon of gas, and cogenGasToKwhEfficiency = 0.33 is the efficiency of the CoGen plant to generate power from natural gas.

8.2.3.4 Energy Aggregations of Supply and Demand

The aggregations of energy supply and demand within the entire energy system include:

- `kwIntoCHCP[i] + demandKw[i] ≤ utilityKw[i] + kwOutCogen[i];`, i.e., the amount of power input to the CHCP and the power demand from the GMU cannot exceed the amount of power supply provided from the DVPC and the power output generated from the CoGen plant, where i ∈ FuturePowerIntervals.
- `demandReduction[i] ≤ (utilityKw[i] + kwOutCogen[i]) - (kwIntoCHCP[i] + demandKw[i]);`, i.e., the power supply reduction cannot exceed the difference between the total power supply (utilityKw[i] + kwOutCogen[i]) and the total power demand (kwIntoCHCP[i] + demandKw[i]), where demandReduction[FuturePowerIntervals] ≥ 0 is an array of extra power supply that can be cut from the power inputs over the FuturePowerIntervals, and i ∈ FuturePowerIntervals.
- `∑demandReduction[i] ≤ maxKwReductionPerPayPeriod;`, i.e., the total power reductions over the future power intervals cannot exceed the allowable maximal power interruptions per future pay period, where i ∈ FuturePowerIntervals, p ∈ FuturePayPeriods, and i.payPeriod = p.
- `utilityGas[i] ≥ gasIntoCogen[i] + gasIntoCHCP[i];`, i.e., the gas input to the CoGen plant and to the CHCP cannot exceed the gas supply provided from the WGLC, where i ∈ FuturePowerIntervals.

- `heatOutCogen[i] + heatOutCHCP[i] ≥ demandHeat[i];`, i.e., the heat demand from GMU cannot exceed the heat supply generated from the CoGen plant and the CHCP, where i ∈ FuturePowerIntervals.
- `coolOutCogen[i] + coolOutCHCP[i] ≥ demandCool[i];`, i.e., the cool demand from GMU cannot exceed the cool supply generated from the CoGen plant and the CHCP, where i ∈ FuturePowerIntervals.

After declaring all the input data sets and the above constraints, which the input data sets need to satisfy, the MTSA-PE model for the GMU energy investment problem can be formulated as follows in Table 33.

Table 33. The MTSA-PE Model for the GMU Energy Investment Problem

Problem and Solution
Problem: $<\boldsymbol{S}, \boldsymbol{P}, \boldsymbol{C_P}, \boldsymbol{C_M}, \boldsymbol{O}>$ $\boldsymbol{S} = \{demandKw, demandHeat, demandCool, payPeriod, year, month, day, weekDay, hour\}$, where $demandKw(t) \geq 0, demandHeat(t) \geq 0, demandCool(t) \geq 0, -8759 \leq t \leq 78840, -11 \leq payPeriod(t) \leq 108, 2011 \leq year(t) \leq 2020, 1 \leq month(t) \leq 12, 1 \leq day(t) \leq 31, 0 \leq weekDay(t) \leq 6, 0 \leq hour(t) \leq 23$ $\boldsymbol{P} = \{utilityKw, payPeriodSupplyDemand, demandReduction, payPeriodKwh, payPeriodKwhCharge, generationDemandCharge, electricCost, utilityGas, gasCost, gasIntoCHCP, kwIntoCHCP, heatOutCHCP, coolOutCHCP, gasIntoCogen, heatOutCogen, coolOutCogen, kwOutCogen\}$, where $utilityKw[t] \geq 0, payPeriodSupplyDemand[p] \geq 0, demandReduction[t] \geq 0, payPeriodKwhCharge[p] \geq 0, payPeriodKwh[p] \geq 0, generationDemandCharge[p] \geq 0, utilityGas[t] \geq 0, electricCost \geq 0, gasCost \geq 0, gasIntoCHCP[t] \geq 0, kwIntoCHCP[t] \geq 0, heatOutCHCP[t] \geq 0, coolOutCHCP[t] \geq 0, gasIntoCogen[t] \geq 0, heatOutCogen[t] \geq 0, coolOutCogen[t] \geq 0, kwOutCogen[t] \geq 0, 1 \leq t \leq 78840, and\ 1 \leq p \leq 108$ $\boldsymbol{C_P} = \{C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8, C_9, C_{10}, C_{11}, C_{12}, C_{13}, C_{14}, C_{15}, C_{16}, C_{17}\}$, where $C_1 = (\forall t \in PastTH)$: $(historicUtilityKw(t) == demandKw(t))$, $C_2 = (\forall t \in TH, p \in FuturePH)$: $(payPeriodKwh[p] == \sum_{p=payPeriod(t)} utilityKw[t])$, $C_3 = (kwhCost == piecewise(0.01174 \to 24000; 0.00606 \to 210000; 0.00244))$, $C_4 = (\forall p \in FuturePH)$: $(payPeriodKwhCharge[p] == kwhCost(payeriodKwh[p])$,

$C_5 = (\forall p \in FuturePH)$: $(generationDemandCharge[p] == 8.124 * payPeriodSupplyDemand[p])$,

$C_6 = (\forall p \in FuturePH)$: $(electricCost == \sum(payPeriodKwhCharge[p] + generationDemandCharge[p]))$,

$C_7 = (\forall t \in FutureTH)$: $(gasCost == (\sum utilityGas[t]/btuPerDth) * gasPricePerDth)$,

$C_8 = (\forall t \in FutureTH)$: $((heatOutCHCP[t] * gasPerHeatUnit \leq gasIntoCHCP[t]) \wedge (coolOutCHCP[t] * kwhPerCoolUnit \leq kwIntoCHCP[t]) \wedge (heatOutCHCP[t] \leq chcpMaxHeatPerHr) \wedge (coolOutCHCP[t] \leq chcpMaxCoolPerHr))$,

$C_9 = (\forall t \in FutureTH)$: $((kwOutCogen[t] * cogenGasPerKwh \leq gasIntoCogen[t]) \wedge (kwOutCogen[t] \leq cogenMaxKw) \wedge (heatOutCogen[t] \leq cogenHeatPerKwh * kwOutCogen[t]) \wedge (heatOutCogen[t] \leq cogenMaxHeatPerHr * (kwOutCogen[t]/cogenMaxKw)) \wedge (coolOutCogen[t] \leq (cogenMaxHeatPerHr * (kwOutCogen[t]/cogenMaxKw) - heatOutCogen[t]) * cogenHeatToCoolRation) \wedge (coolOutCogen[t] \leq cogenMaxCoolPerHr))$,

$C_{10} = (\forall t \in PastTH)$: $(utilityKw[t] == historicUtilityKw[t])$,

$C_{11} = (\forall p \in FuturePH, t \in TH)$: $(payPeriod(t) == p \wedge weekDay(t) \geq 1 \wedge weekDay \leq 5 \wedge ((month(t) \geq 6 \wedge month(t) \leq 9 \wedge hour(t) \geq 10 \wedge hour(t) \leq 22) \vee (month(t) \leq 5 \wedge month(t) \geq 10 \wedge hour(t) \geq 7 \wedge hour(t) \leq 22))) \rightarrow payPeriodSupplyDemand[p] \geq utilityKw[t]$,

$C_{12} = (\forall p \in FuturePH, t \in TH)$:$(month(t) \geq 6 \wedge month(t) \leq 9 \wedge payPeriod(t) \geq p - 11 \wedge payPeriod(t) \leq p \wedge weekDay(t) \geq 1 \wedge weekDay(t) \leq 5 \wedge hour(t) \geq 10 \wedge hour(t) \leq 22) \rightarrow payPeriodSupplyDemand[p] \geq 0.9 * utilityKw[t]$,

$C_{13} = (\forall t \in FutureTH)$: $((kwIntoCHCP[t] + demandKw[t] \leq utilityKw[t] + kwOutCogen[t]) \wedge (demandReduction[t] \leq (utilityKw[t] + kwOutCogen[t]) - (kwIntoCHCP[t] + demandKw[t])))$,

$C_{14} = (\forall p \in FuturePH, t \in FutureTH)$: $(payPeriod(t) == p \wedge demandReduction[t] \leq maxKwReductionPerPayPeriod)$

$C_{15} = (\forall t \in FutureTH)$: $(utilityGas[t] \geq gasIntoCogen[t] + gasIntoCHCP[t])$,

$C_{16} = (\forall t \in FutureTH)$: $(heatOutCogen[t] + heatOutCHCP[t] \geq demandHeat[t])$,

$C_{17} = (\forall t \in FutureTH)$: $(coolOutCogen[t] + coolOutCHCP[t] \geq demandCool[t])$,

$\boldsymbol{C_M} = \{\}$

$\boldsymbol{O} = electricCost + gasCost$
Solution:
$\underset{\boldsymbol{P}}{argmin}\, \boldsymbol{O}(\boldsymbol{P})$
subject to $\boldsymbol{C_P}(\boldsymbol{P}) \wedge \boldsymbol{C_M}(\boldsymbol{P}))$

Table 34 summarizes the descriptions of all the constant values from the electric utility contract used in the MTSA-PE model for the GMU energy investment problem.

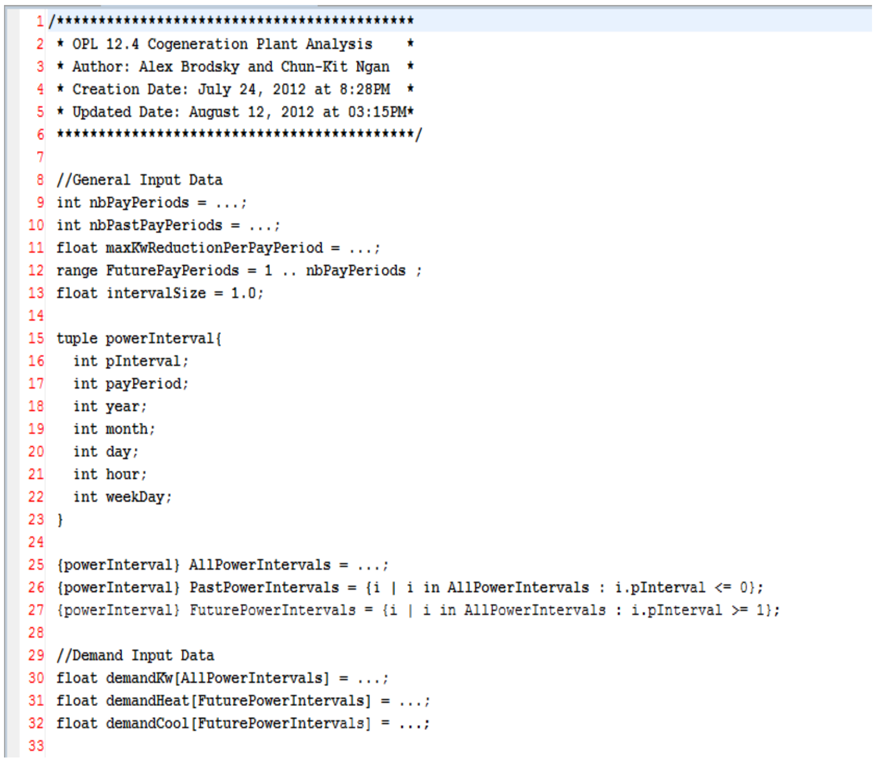
Table 34. Descriptions for the Constant Values in the MTSA-PE Optimization Model for the GMU Energy Investment Problem

Constant	Description
0.9	Percentage of the highest kW of demand during the billing months of June through September of the preceding 11 billing months
8.124	Amount ($) of Electricity Supply (ES) demand charged per kW
24000	First ES kWh
0.01174	Amount ($) of the first 24000 ES kWh charged per kWh
186000	Next ES kWh
0.00606	Amount ($) of the next 186000 ES kWh charged per kWh
210000	Sum of the first ES kWh and the next ES kWh
0.00244	Amount ($) of the additional ES kWh charged per kWh
210	kWh for each ES kW of demand over 1000 kW
6.5	gasPricePerDth
1000000	btuPerDth
(1/0.78)	gasPerHeatUnit
(1/0.94)	kwhPerCoolUnit
108000000	chcpMaxHeatPerHr
11880	chcpMaxCoolPerHr
40000000	cogenMaxHeatPerHr
2400	cogenMaxCoolPerHr
7200	cogenMaxKw
0.33	cogenGasToKwhEfficiency
114000	gasBTUPerGallon
33.41	kWhPerGallon
gasBTUPerGallon/kWhPerGallon/cogenGasToKwhEfficiency	cogenGasPerKwh
10300	cogenHeatPerKwh
cogenMaxCoolPerHr/cogenMaxHeatPerHr	cogenHeatToCoolRatio

8.2.4 OPL Implementation for the MTSA-PE Model

The MTSA-PE model has been implemented by using the OPL language. Using the GMU historical data of power usage in the past year, i.e., 2011, and its projected electricity, cooling, and heating demand over a future time horizon from 2012 to 2020, I use the OPL language to implement and demonstrate the MTSA-PE model to solve the GMU energy investment problem and minimize the operating cost, which is shown from Figure 12.1 to Figure 12.9 as an illustration.

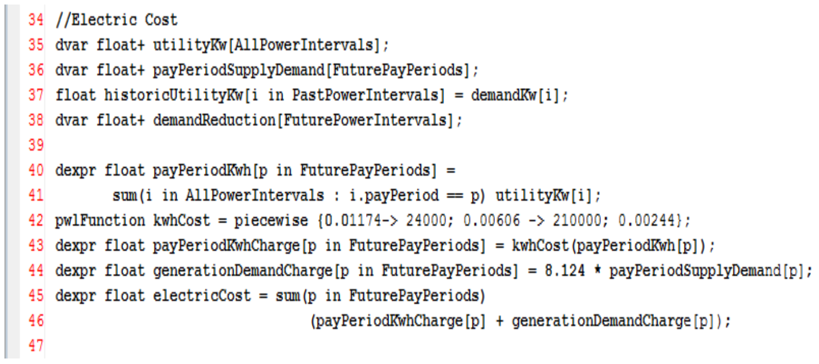
In Figure 12.1, from the line number 9 to 12, the value 12, i.e., the total 12 months of 2011, is assigned to the variable *nbPastPayPeriods*, the value 108, i.e., the total 108 months from 2012 to 2020, is assigned to the variable *nbPayPeriods*, and the value 0 is assigned to the maximal power interruptions, i.e., *maxKwReductionPerPeriod*. The *FuturePayPeriods* is ranged from 1 to 108. From the line number 15 to 23, I declare a tuple of a power interval that has the attributes, including *pInterval*, *payPeriod*, *year*, *month*, *day*, *hour*, and *weekDay*. The line number 25 to 27 declares and initializes *AllPowerIntervals* that include both *PastPowerIntervals* and *FuturePowerIntervals*. The line number 30 to 32 declares and initializes the *demandKw[AllPowerIntervals]*, the *demandHeat[FuturePowerIntervals]*, and the *demandCool[FuturePowerIntervals]* arrays.



```
 1 /********************************************
 2  * OPL 12.4 Cogeneration Plant Analysis    *
 3  * Author: Alex Brodsky and Chun-Kit Ngan  *
 4  * Creation Date: July 24, 2012 at 8:28PM  *
 5  * Updated Date: August 12, 2012 at 03:15PM*
 6  ********************************************/
 7 
 8  //General Input Data
 9  int nbPayPeriods = ...;
10  int nbPastPayPeriods = ...;
11  float maxKwReductionPerPayPeriod = ...;
12  range FuturePayPeriods = 1 .. nbPayPeriods ;
13  float intervalSize = 1.0;
14 
15  tuple powerInterval{
16    int pInterval;
17    int payPeriod;
18    int year;
19    int month;
20    int day;
21    int hour;
22    int weekDay;
23  }
24 
25  {powerInterval} AllPowerIntervals = ...;
26  {powerInterval} PastPowerIntervals = {i | i in AllPowerIntervals : i.pInterval <= 0};
27  {powerInterval} FuturePowerIntervals = {i | i in AllPowerIntervals : i.pInterval >= 1};
28 
29  //Demand Input Data
30  float demandKw[AllPowerIntervals] = ...;
31  float demandHeat[FuturePowerIntervals] = ...;
32  float demandCool[FuturePowerIntervals] = ...;
33 
```

Figure 12.1. General and Demand Input Data

Figure 12.2 declares the decision control variables, i.e., *utilityKw[AllPowerIntervals]*, the *payPeriodSupplyDemand[FuturePayPeriods]*, and the *payPeriodKwh[FuturePayPeriods]*, to compute the *payPeriodKwhCharge[FuturePayPeriods]* and *generationDemandCharge[FuturePayPeriods]* that are summed together to determine the total electricity cost over all the future pay periods while satisfying the electric contractual constraints.

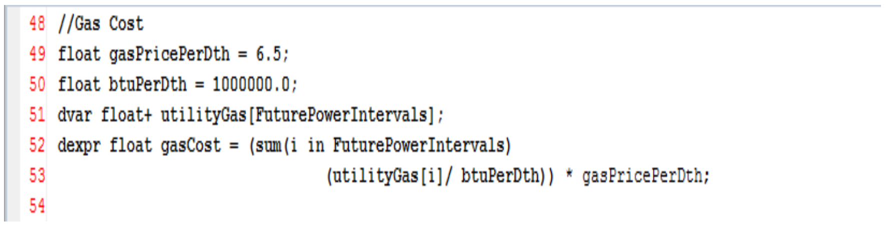


```
//Electric Cost
dvar float+ utilityKw[AllPowerIntervals];
dvar float+ payPeriodSupplyDemand[FuturePayPeriods];
float historicUtilityKw[i in PastPowerIntervals] = demandKw[i];
dvar float+ demandReduction[FuturePowerIntervals];

dexpr float payPeriodKwh[p in FuturePayPeriods] =
       sum(i in AllPowerIntervals : i.payPeriod == p) utilityKw[i];
pwlFunction kwhCost = piecewise {0.01174-> 24000; 0.00606 -> 210000; 0.00244};
dexpr float payPeriodKwhCharge[p in FuturePayPeriods] = kwhCost(payPeriodKwh[p]);
dexpr float generationDemandCharge[p in FuturePayPeriods] = 8.124 * payPeriodSupplyDemand[p];
dexpr float electricCost = sum(p in FuturePayPeriods)
                               (payPeriodKwhCharge[p] + generationDemandCharge[p]);
```

Figure 12.2. Total Electricity Cost

Figure 12.3 declares the constants, i.e., *gasPricePerDth* and *btuPerDth*, and the *utilityGas[FuturePowerIntervals]* to calculate the total gas cost over all the future power intervals.

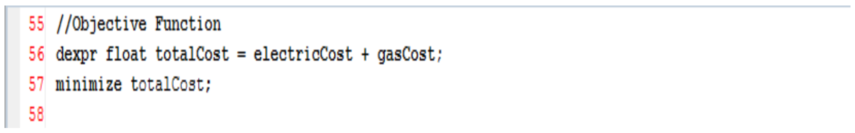


```
//Gas Cost
float gasPricePerDth = 6.5;
float btuPerDth = 1000000.0;
dvar float+ utilityGas[FuturePowerIntervals];
dexpr float gasCost = (sum(i in FuturePowerIntervals)
                               (utilityGas[i]/ btuPerDth)) * gasPricePerDth;

```

Figure 12.3. Total Gas Cost

Figure 12.4 declares the objective function to minimize the total operating cost, i.e., the total electricity cost plus the total gas cost.

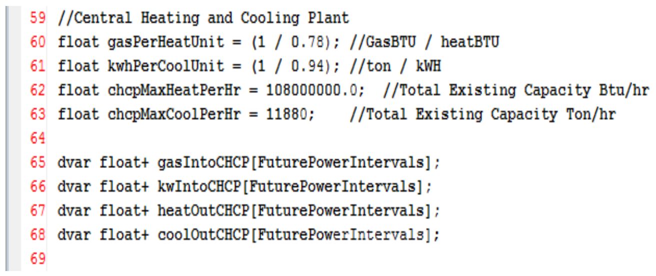


```
//Objective Function
dexpr float totalCost = electricCost + gasCost;
minimize totalCost;

```

Figure 12.4. Total Operating Cost

Figure 12.5 declares the constants, i.e., *gasPerHeatUnit*, *kwhPerCoolUnit*, *chcpMaxHeatPerHr*, and *chcpMaxCoolPerHr*, and the arrays, i.e., *gasIntoCHCP[FuturePowerIntervals]*, *kwIntoCHCP[FuturePowerIntervals]*, *heatOutCHCP[FuturePowerIntervals]*, and *coolOutCHCP[FuturePowerIntervals]*, used in the CHCP capacity constraints.



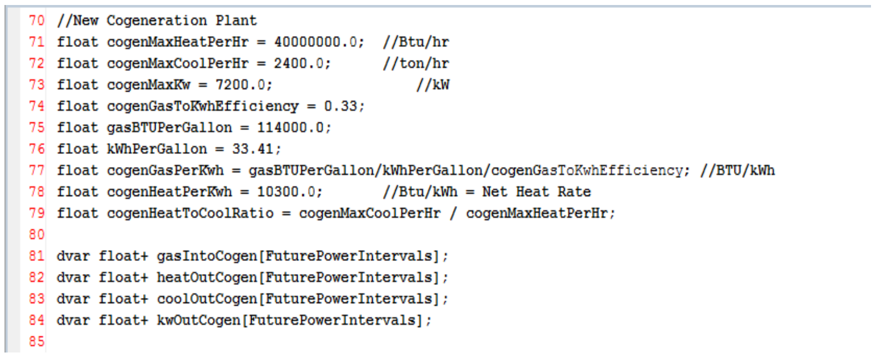
```
//Central Heating and Cooling Plant
float gasPerHeatUnit = (1 / 0.78); //GasBTU / heatBTU
float kwhPerCoolUnit = (1 / 0.94); //ton / kWH
float chcpMaxHeatPerHr = 108000000.0;  //Total Existing Capacity Btu/hr
float chcpMaxCoolPerHr = 11880;    //Total Existing Capacity Ton/hr

dvar float+ gasIntoCHCP[FuturePowerIntervals];
dvar float+ kwIntoCHCP[FuturePowerIntervals];
dvar float+ heatOutCHCP[FuturePowerIntervals];
dvar float+ coolOutCHCP[FuturePowerIntervals];

```

Figure 12.5. Operational Parameters and Data Structures of the CHCP

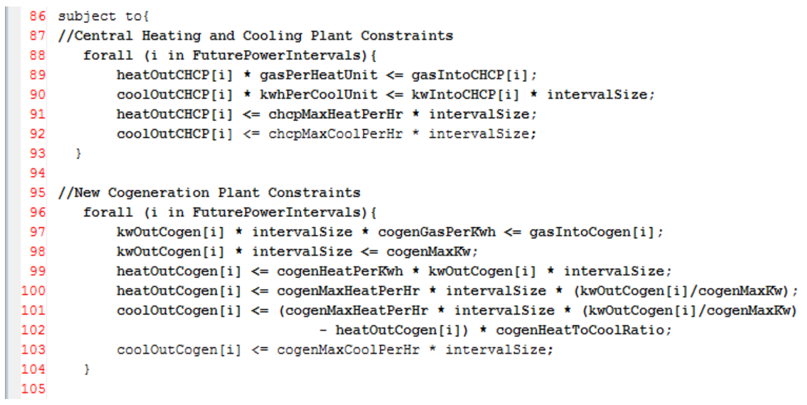
Figure 12.6 declares the constants from the line number 71 to 79, and the arrays, i.e., *gasIntoCogen[FuturePowerIntervals]*, *heatOutCogen[FuturePowerIntervals]*, *coolOutCHCP[FuturePowerIntervals]*, and *kwOutCHCP[FuturePowerIntervals]*, which are used in the capacity constraints of the CoGen plant.



```
70 //New Cogeneration Plant
71 float cogenMaxHeatPerHr = 40000000.0;  //Btu/hr
72 float cogenMaxCoolPerHr = 2400.0;      //ton/hr
73 float cogenMaxKw = 7200.0;                 //kW
74 float cogenGasToKwhEfficiency = 0.33;
75 float gasBTUPerGallon = 114000.0;
76 float kWhPerGallon = 33.41;
77 float cogenGasPerKwh = gasBTUPerGallon/kWhPerGallon/cogenGasToKwhEfficiency; //BTU/kWh
78 float cogenHeatPerKwh = 10300.0;       //Btu/kWh = Net Heat Rate
79 float cogenHeatToCoolRatio = cogenMaxCoolPerHr / cogenMaxHeatPerHr;
80
81 dvar float+ gasIntoCogen[FuturePowerIntervals];
82 dvar float+ heatOutCogen[FuturePowerIntervals];
83 dvar float+ coolOutCogen[FuturePowerIntervals];
84 dvar float+ kwOutCogen[FuturePowerIntervals];
85
```

Figure 12.6. Operational Parameters and Data Structures of the CoGen Plant

Figure 12.7 defines all the capacity constraints for the CHCP and the CoGen plant.

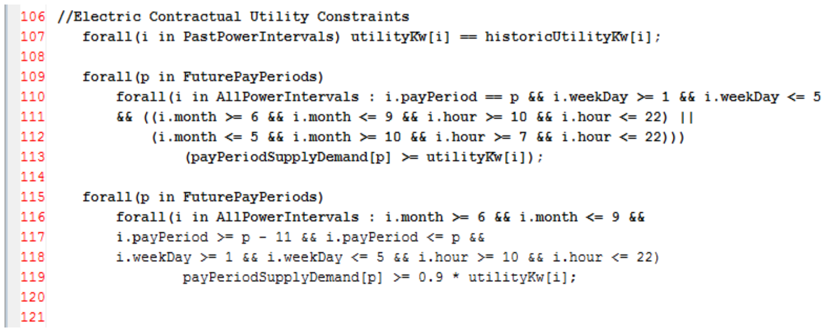


```
subject to{
//Central Heating and Cooling Plant Constraints
   forall (i in FuturePowerIntervals){
       heatOutCHCP[i] * gasPerHeatUnit <= gasIntoCHCP[i];
       coolOutCHCP[i] * kwhPerCoolUnit <= kwIntoCHCP[i] * intervalSize;
       heatOutCHCP[i] <= chcpMaxHeatPerHr * intervalSize;
       coolOutCHCP[i] <= chcpMaxCoolPerHr * intervalSize;
  }

//New Cogeneration Plant Constraints
   forall (i in FuturePowerIntervals){
       kwOutCogen[i] * intervalSize * cogenGasPerKwh <= gasIntoCogen[i];
       kwOutCogen[i] * intervalSize <= cogenMaxKw;
       heatOutCogen[i] <= cogenHeatPerKwh * kwOutCogen[i] * intervalSize;
       heatOutCogen[i] <= cogenMaxHeatPerHr * intervalSize * (kwOutCogen[i]/cogenMaxKw);
       coolOutCogen[i] <= (cogenMaxHeatPerHr * intervalSize * (kwOutCogen[i]/cogenMaxKw)
                               - heatOutCogen[i]) * cogenHeatToCoolRatio;
       coolOutCogen[i] <= cogenMaxCoolPerHr * intervalSize;
   }
```

Figure 12.7. Capacity Constraints of the CHCP and the CoGen Plant

Figure 12.8 defines the contractual constraints for the electricity bill.



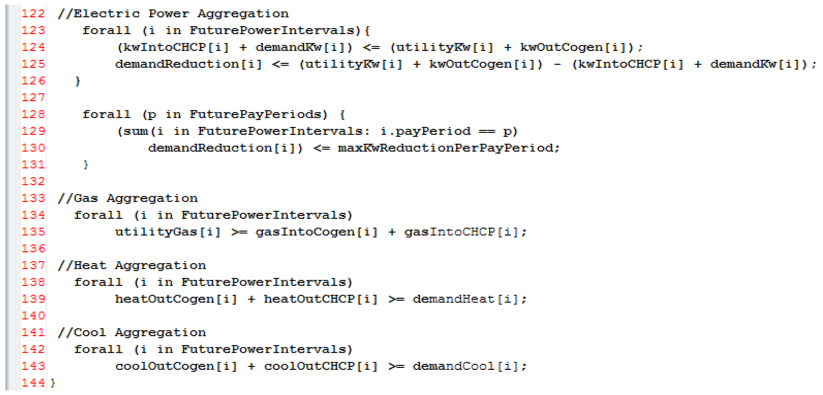
```
//Electric Contractual Utility Constraints
   forall(i in PastPowerIntervals) utilityKw[i] == historicUtilityKw[i];

   forall(p in FuturePayPeriods)
       forall(i in AllPowerIntervals : i.payPeriod == p && i.weekDay >= 1 && i.weekDay <= 5
       && ((i.month >= 6 && i.month <= 9 && i.hour >= 10 && i.hour <= 22) ||
           (i.month <= 5 && i.month >= 10 && i.hour >= 7 && i.hour <= 22)))
               (payPeriodSupplyDemand[p] >= utilityKw[i]);

   forall(p in FuturePayPeriods)
       forall(i in AllPowerIntervals : i.month >= 6 && i.month <= 9 &&
       i.payPeriod >= p - 11 && i.payPeriod <= p &&
       i.weekDay >= 1 && i.weekDay <= 5 && i.hour >= 10 && i.hour <= 22)
               payPeriodSupplyDemand[p] >= 0.9 * utilityKw[i];
```

Figure 12.8. Contractual Electricity Utility Constraints

Figure 12.9 defines the constraints for the energy aggregations of electric power, gas, heat, and cool.



```
//Electric Power Aggregation
   forall (i in FuturePowerIntervals){
       (kwIntoCHCP[i] + demandKw[i]) <= (utilityKw[i] + kwOutCogen[i]);
       demandReduction[i] <= (utilityKw[i] + kwOutCogen[i]) - (kwIntoCHCP[i] + demandKw[i]);
  }

   forall (p in FuturePayPeriods) {
       (sum(i in FuturePowerIntervals: i.payPeriod == p)
           demandReduction[i]) <= maxKwReductionPerPayPeriod;
   }

//Gas Aggregation
  forall (i in FuturePowerIntervals)
       utilityGas[i] >= gasIntoCogen[i] + gasIntoCHCP[i];

//Heat Aggregation
  forall (i in FuturePowerIntervals)
       heatOutCogen[i] + heatOutCHCP[i] >= demandHeat[i];

//Cool Aggregation
  forall (i in FuturePowerIntervals)
       coolOutCogen[i] + coolOutCHCP[i] >= demandCool[i];
}
```

Figure 12.9. Energy Aggregations of Supply and Demand

8.3.5 Analytical Methodology on Evaluation among Energy Investment Options

For domain experts being able to formulate and implement the above MTSA-PE model to determine the best investment option, I propose an analytical methodology that guides the domain experts to achieve this goal. The methodology includes six steps.

STEP 1: Collect historical energy demand, such as electricity, heating, and cooling, from each building unit, and forecast those demands in terms of growth on a square-foot basis over the future time horizon.

STEP 2: Identify all the possible energy investment options, such as the expansion of current facilities and the procurement of cogeneration plants.

STEP 3: Formulate, implement, and execute the MTSA-PE model that integrates historical and projected energy demand, electric and gas contractual utility, operational parameters and capacity constraints of energy equipment, as well as energy aggregations of supply and demand in each considered option under the assumption of optimal interactions among available resources.

STEP 4: Compute the annualized evaluation parameters for each option based upon the results from the optimization process in the STEP 3.

The parameters include the investment cost (I_i), equipment cost (E_i), i.e., maintenance expenditure (M_i) plus replacement charge (R_i), operating expense (C_i), i.e., the charges on electricity and gas consumptions, cost saving (S_i), i.e., $C_0 – C_i$, where $i \geq 0$ denotes an investment option and C_0 is the operating cost of a base investment option that the other available options compare with, and return on investment (ROI_i), i.e., $S_i / (I_i – I_0)$, as well as the GHG emissions ($MTCDE_i$), i.e., G_i * 0.053 MTCDE/Million-Btu + P_i * 0.513 MTCDE/Million-Wh, shown in Table 35, against the various investment options, where 0.053 and 0.513 are the factors, which are calculated from the historical data.

Note that the base investment option is the option that the current capacity of the existing facilities is expanded without procuring any new energy equipment.

Using the ROI and GHG emissions, domain users and experts can plot the analytical graphs to illustrate the relationships among the ROI, GHG emissions, and investment expenses, which enable the domain experts to determine the best investment option among all of the options being considered.

Table 35. Evaluation Parameters of ROI and GHG Emissions for Determining the Best Investment Option

Parameter	Symbol
Investment Cost	I_i
Maintenance Expenditure	M_i
Replacement Charge	R_i
Equipment Cost	E_i
Operating Expense	C_i
Cost Saving	S_i
Return on Investment	ROI_i
Average Annual Gas Consumption MBTU	G_i
Average Annual Electric Power Consumption MWh	P_i
GHG Emission	$MTCDE_i$

STEP 5: Remove any option that is dominated by the other options in terms of the evaluation parameters.

STEP 6: Construct a trade-off graph to evaluate the options that are not dominated among others and then make a final decision.

Note that although the STEP 1, 2, 4, 5, and 6 are typical processes of evaluations, the STEP 3 is not typical at all as the problem that I solve is a non-trivial optimization problem.

8.2.6 Analytical Methodology on Experimental Case Study

After the process from the **STEP 1** to **STEP 3** in the experimental case study at GMU, the four investment options, including ① the expansion of the existing CHCP only, ② the addition of a CoGen plant to the existing CHCP, ③ the half capacity of the Option ① with the half planned capacity of the CoGen plant, and ④ the full capacity of the Option ① with the full planned capacity of the CoGen plant, have been chosen to be evaluated to meet the electricity, heating, and cooling demand of the Fairfax campus over the next 9 years from 2012 to 2020.

In the **STEP 4**, using the evaluation parameters, i.e., ROI and GHG emissions, discussed in Section 8.2.5 and the OPL to solve the GMU energy investment problem in Section 8.2.4, I obtained Table 36, Table 37, and Figure 13 that can be used to determine the best investment option.

Table 36. Evaluation Parameters of ROI for Determining the GMU Energy Investment Options

Investment Option	Investment Cost ($M)	Annual Maintenance Cost ($)	Annualized Replacement Cost ($M)	Annualized Equipment Cost ($M)	Annualized Average Operational Cost ($M)	Annualized Saving over the Expanded CHCP ($M)	ROI (%)
1 Expanded CHCP	$34.293	$343,200	$3.429	$3.772	$6.244	$0.000	0.000%
1 CoGen Plant + 1 Current CHCP	$65.328	$655,600	$3.850	$4.506	$5.494	$0.016	0.052%
½ CoGen Plant + ½ Expanded CHCP	$46.995	$499,400	$4.699	$5.199	$5.557	-$0.740	-5.827%
1 CoGen Plant + 1 Expanded CHCP	$99.621	$998,800	$7.279	$8.278	$5.492	-$3.754	-5.747%

Table 37. Evaluation Parameters of GHG Emissions for Determining the GMU Energy Investment Options

Investment Option	Investment Cost ($M)	Average Annual Gas Consumption (MBTU)	Average Annual Electric Power Consumption (MWh)	GHG Emission (MTCDE)
1 Expanded CHCP	$34.293	510,500.00	141,433.33	99611.799
1 CoGen Plant + 1 Current CHCP	$65.328	523,622.22	141,333.33	100255.977
½ CoGen Plant + ½ Expanded CHCP	$46.995	520,888.89	141,344.44	100116.811
1 CoGen Plant + 1 Expanded CHCP	$99.621	523,600.00	141,333.33	100254.799

In the **STEP 5**, the Option ③ and ④ are the dominated cases that can be removed from the consideration list because of the negative ROI.

In the **STEP 6**, according to Table 35, Table 36, and Figure 13, I can conclude that the Option ① should be chosen because of the three observations. First, the GHG emissions and the equipment cost of the Option ① are the lowest. Second, even though the ROI of the Option ②, i.e., 0.052%, is marginally better than that of the Option ①, the GHG emissions of the Option ② is the highest among all the options being considered. Third, it is not economical at all for GMU to invest $31 million dollars, i.e., the Option ② investment cost minus the Option ① investment cost, more to earn only 0.052% ROI in the next 9-year timeframe. Thus, the Option 1 is the best long-term option for GMU.

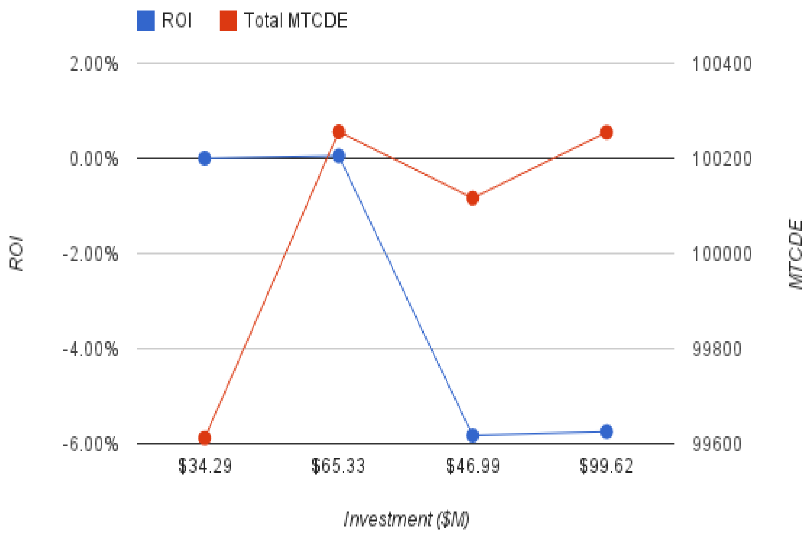


Figure 13. ROI (%) and GHG Emissions (MTCDE) vs. Investment Cost ($M) across the Four Investment Options

8.2.7 Case Summary

In this study, I propose a Decision-Guided Energy Investment (DGEI) Framework to optimize power, heating, and cooling capacity. The DGEI framework is designed to support energy managers to (1) use the analytical and graphical methodology to determine the best investment option that satisfies the designed evaluation parameters, such as ROI and GHG emissions; (2) develop a MTSA-PE model to solve energy investment problems that the operating expenses are minimal in each considered investment option; (3) implement the MTSA-PE model using the IBM OPL language with historical and projected energy demand data, i.e., electricity, heating, and cooling, to solve energy investment optimization problems; and (4) conduct an experimental case study on the Fairfax campus microgrid at George Mason University (GMU) and utilize the MTSA-PE model and its OPL implementations, as well as the graphical and analytical methodology to make the investment decision and trade-offs among the cost savings, investment costs, maintenance expenditures, replacement charges, operating expenses, GHG emissions, and return on investment (ROI) for all the considered options.

CHAPTER 9. CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

This research introduces the web-mashup application service framework for Multivariate Time Series Analytics (MTSA), which provides model definition, querying, parameter learning, model evaluation, data monitoring, decision recommendation, and web portal on events over multivariate time series. This integrated framework enables domain experts to develop the MTSA services on the Internet. More specifically, I develop the MTSA database model and query language for the services of learning, monitoring, and recommendation. Domain experts can use the proposed database models, such as time-series and time-event schemas, to store historical and projected time series, as well as binary events from which optimal decision parameters are learned. Using the parameter learning service, domain experts can formulate a learning event and initial the event to the database, in which the built-in algorithms, such as the Checkpoint algorithm, or the external solvers, e.g., the IBM ILOG CPLEX optimizer, are invoked to solve the corresponding classes of the MTSA problems and then learn the optimal decision parameters. The classes of the MTSA problems, which I solve, include the learning of decision parameters of an objective function (1) specifically dependent on their optimal time points of a time utility and (2) generally independent of a particular time point. After the parameters are learned and stored, domain experts can utilize the monitoring and recommendation service to monitor the incoming data streams and recommend actions when those streams satisfy the monitoring template.

To learn decision parameters of an objective function that is dependent on their optimal time points of a time utility, I develop a mathematical model, i.e., EQPE, and a learning algorithm, Checkpoint, to solve Expert Query Parametric Estimation problems. These model and algorithm combine the strengths of both domain-knowledge-based and formal-learning-based approaches to maximize utility on events over multivariate time series. The EQPE model is a well-defined model that captures domain-expert knowledge in expression of multivariate time series, decision parameters, a set of parametric constraints, a time utility, and an objective function. Each multivariate time series is an input parametric time series for an event. Each decision parameter is instantiated from its input parametric time series to learn the optimal value that satisfies the given parametric constraint and maximizes its time utility collectively. The time utility is a function of the decision time point that the objective function is dependent upon from which the parameters are learned.

The Checkpoint algorithm also combines the strengths of both domain experts' knowledge in terms of a parametric constraint and formal-learning-based methodology by using the regression approach. The goal of the Checkpoint algorithm is to learn the decision parameters that maximize the objective function over multivariate time series. This algorithm guarantees an optimality of the learned decision parameters in their event, i.e., a true optimal decision time point, during the computations. To demonstrate the effectiveness of the algorithm, I compare my method with the formal-learning-based approach, i.e., logistic regression methodology, and domain-knowledge-based approach in the financial domain. Using the learned decision parameters, I show that the Checkpoint algorithm is more effective and guarantees the satisfactory forecasting results that are superior to those from the logistic regression methodology and the financial experts' criteria.

However, the proposed EQPE and the Checkpoint algorithm are only able to learn one set of decision parameters for one particular event at a single time point, whereas there are many real-world problems that the parameter learning is at multiple time points in sequence. To address this shortcoming, I develop the Multi-Event Expert Query Parametric Estimation (ME-EQPE) model. This model is a well-defined model that captures domain-expert knowledge in expression of multivariate time series vectors, decision parameter vectors, a set of parametric constraints, a time utility, and an objective function. Each multivariate time series vector is a set of input parametric time series for each event. Each vector of decision parameters is instantiated from its input parametric time series vector to learn the optimal values that satisfy the given parametric constraints and maximize their time utility collectively. The time utility is a function of the multiple decision time points that the objective function is dependent upon from which the parameter vectors are learned.

The Multidimensional *M*-Checkpoint algorithm also combines the strenghts of both domain experts' knowledge in terms of a set of parametric constraints and formal-learning-based methodology by using the regression appoarch. The goal of the *M*-Checkpoint algorithm is to learn the multiple decision parameter vectors that maximize the objective function over multivariate time series vectors. This new algorithm guarantees an optimality of the learned multiple sets of decision parameters in their respective events, i.e., multiple, true optimal decision time points during the computations. To demonstrate the effectiveness of the algorithm, I compare it with the formal-learning-based approach, i.e., logistic regression methodology in the financial domain. Using the learned multiple sets of decision parameters, I show that the *M*-Checkpoint algorithm is more effective and guarantees the satisfactory forecasting results that are superior to those from the logistic regression methodology.

Due to the high complexity $O(N^{m})$ of the *M*-Checkpoint algorithm, I develop a *R*-Checkpoint to solve the ME-EQPE problems over multivariate time series. This algorithm combines the strengths of both domain-knowledge-based and formal-learning-based approaches to learn multiple sets of decision parameters that are fairly close to the optimal parameters learned from the *M*-Checkpoint algorithm, produce reasonably forecasting results, and maintain a satisfactorily low time complexity, i.e., $O(QkNlogN)$, as compared with $O(N^{m})$ of the *M*-Checkpoint algorithm, where Q is the total number of m-event, time-point combinations which yields the top-Q time utility, and k is the number of input parametric time series for each event. To demonstrate the performance of the new algorithm, I conduct an experiment in the financial domain. Specifically, I compare the forecasting results that are detected by the decision parameters learned from the *R*-Checkpoint with the results that are determined by the optimal parameters obtained from the *M*-Checkpoint algorithm, as well as the parametric coefficients of the logistic regression model. The experimental study shows that the forecasting results by the heuristic *R*-Checkpoint algorithm are slightly lower than those of the optimal *M*-Checkpoint algorithm and are considerably higher than those of the logistic regression methodology.

However, the discussed models, EQPE and ME-EQPE, and algorithms, Checkpoint, *M*-Checkpoint, and *R*-Checkpoint, are only able to solve a specific class of problems that (1) their decision parameters of an objective function are learned from optimal time points of a time utility function, (2) the monitoring template has to be in the considered form, i.e., conjunctions of inequality constraints, and (3) the constraints being used are solely for monitoring purposes. To address the weaknesses, I develop a hybrid-based model, Multivariate Time Series Analytics – Parameter Estimation, to solve a general class of problems in which the objective function is maximized or minimized from the optimal decision parameters regardless of particular time points. This model allows domain experts to include multiple types of constraints, e.g., global constraints and monitoring constraints. At the end, I develop a parameter learning architecture from which the parameter learning event is transformed into the IBM OPL construct by the MTSA compiler. This OPL construct is then sent to the IBM ILOG CPLEX optimizer to learn the optimal decision parameters that are returned to the learning event. To further prove the capability of the MTSA-PE model, I conduct two real case students at GMU. Using the electric power microgrids at GMU as examples, I illustrate how the MTSA-PE model with the external solver to solve the energy problems, including (1) the determination of optimal peak demand bounds and (2) the decision on the best energy investment options.

In the first case study, I propose and report on the development of DGLS, a Decision-Guidance System for Load Shedding of electric power in microgrids in order to minimize energy costs and maximize customers' savings while preserving the desired quality of service (QoS) in terms of power interruption. The DGLS system is designed to support energy managers to forecast electric power demand over a time horizon, use the predicted peak demand usage to optimize the peak demand bound for every monthly pay period, continuously monitor the hourly electricity demand, and shed load when the demand exceeds the optimal peak demand bound using a service prioritization scheme. The core technical challenge is the development of the MTSA-PE model for the peak demand optimization that is very accurate in terms of the electric contractual terms and engineering constraints, and yet efficient and scalable, which is done by the careful modeling of mainly continuous decision variables and using constructs that avoid introduction of combinatorics, e.g., explicit or implicit binary variables, into the model. The model has also been implemented and demonstrated by the IBM OPL language for the GMU energy cost problem.

In the second case study, I propose a Decision-Guided Energy Investment (DGEI) Framework to optimize power, heating, and cooling capacity. The DGEI framework is designed to support energy managers to (1) use the analytical and graphical methodology to determine the best investment option that satisfies the designed evaluation parameters, such as ROI and GHG emissions; (2) develop a MTSA-PE model to solve energy investment problems that the operating expenses are minimal in each considered investment option; (3) implement the MTSA-PE model using the IBM OPL language with historical and projected energy demand data, i.e., electricity, heating, and cooling, to solve energy investment optimization problems; and (4) conduct an experimental case study on the Fairfax campus microgrid at George Mason University (GMU) and utilize the MTSA-PE model and its OPL implementations, as well as the graphical and analytical methodology to make the investment decision and trade-offs among the cost savings, investment costs, maintenance expenditures, replacement charges, operating expenses, GHG emissions, and return on investment (ROI) for all the considered options.

9.2 Future Work

In addition to the contributions made by this research, there are still numerous interesting topics for further exploration. They mainly include the issues regarding a more efficient algorithm for the ME-EQPE problems, the multi-event MTSA-PE model, the effective algorithms for parameter learning on one- and multi-event MTSA-PE problems, the query language for parameter learning on multi-sequential events, and the development of the Model Accuracy and Quality Evaluation module.

First, to solve the ME-EQPE problems, I develop the *M*-Checkpoint algorithm to learn multiple sets of optimal decision parameters for multi-events. However, the time complexity of this algorithm to solve the ME-EQPE problems is polynomial, i.e., $O(N^m)$. To mitigate this computational cost, I develop the relaxed *R*-Checkpoint algorithm to solve the ME-EQPE problems. The core contribution of this algorithm is to learn multiple sets of decision parameters that are close to the optimal ones but at a lower time complexity, i.e., $O(QkNlogN)$. Nevertheless, the *R*-Checkpoint algorithm does not guarantee optimality. In addition, if Q and k are sufficiently large, they lead the time complexity to become polynomial again. Thus the future research will be how to further develop a new algorithm that combines the optimality of M-Checkpoint and the low time complexity of R-Checkpoint algorithm to learn parameter sets of multi-events.

Second, the MTSA-PE model is designed to solve a class of problems that involve a single event, e.g., the electric load shedding is executed when the electricity demand exceeds the optimal peak demand bound. However, there are many real-world cases that multiple related events occur in sequence. For instance, consider the above load-shedding example again, in which the energy managers would like to determine when the electric account units should be turned off and when those accounts should be turned on in order. Using the MTSA-PE model, the energy managers would not be able to properly decide on when to shed or unshed the load at the two interrelated events and then to gain the maximal cost savings and achieve the minimal power interruptions. To address the shortcomings of this issue, the future research will focus on developing a multi-event MTSA-PE model. This new model will maintain the advantages of the single-event MTSA-PE model and also support the parameter learning on multiple events in sequence.

Third, the IBM ILOG CPLEX optimizer that I use to solve the class of MTSA-PE problems is the branch-and-bound-based algorithm, which the time complexity is exponential, i.e., $O(k2^N)$ if the problem is a single event, and $O(k2^{m \cdot N})$ if the problem is a sequence of multiple events. Thus the furture research will focus on developing a new algorithm that will be able to solve the class of single- and multi-event MTSA-PE problems at a lower computational cost.

Fourth, the proposed MTSA query language is only able to formulate the single-event parameter learning on both EQPE and MTSA-PE problems. The future research will be how to extend the proposed MTSA query language to support the formulation of the ME-EQPE and the multi-event MTSA-PE problems.

Finally, in order to improve the accuracy of the parametric model templates and the quality of the monitoring and recommendation service in the future, I will develop an evaluation model and algorithm to identify the performance gaps in terms of QoS, event utility, decision-making actions, etc. I will also develop a MTSA query construct to update the model template based on those performance gaps.

APPENDIX: LIST OF PUBLICATIONS RELATED TO THE DISSERTATION

Journal Articles

J1. Ngan, C.K. & Brodsky, A. (2013). *Optimal Event Monitoring through Internet Mash-up of Multivariate Time Series*. Special Issue on Collaborative Networking Environments and the Internet Technology. International Journal of Decision Support Systems Technology.

J2. Ngan, C.K., Brodsky, A., & Lin, J. (2012). *R-Checkpoint Algorithm for Multi-Event Decision Making over Multivariate Time Series*. Fusing Decision Support Systems into the Fabric of the Context. IOS Press.

J3. Ngan, C.K., Brodsky, A., & Lin, J. (2013). *Multi-Event Decision Making over Multivariate Time Series*. Special Issue on Collaborative Decision Support Systems. International Journal of Information and Decision Sciences.

J4 Ngan, C.K., Brodsky, A., & Lin, J. (2012). *An Event-Based Service Framework for Querying, Monitoring, and Learning Multivariate Time Series*. Lecture Notes in Business Information Processing, Enterprise Information Systems. Springer-Verlag.

Conference Papers

C1. Ngan, C.K., Brodsky, A., Egge, N., & Backus, E. (2013). *A Decision-Guided Energy Framework for Optimal Power, Heating, and Cooling Capacity Investment*. The 15th International Conference on Enterprise Information Systems. Angers, France.

C2. Ngan, C.K. & Brodsky, A. (2013). *DGLS System: Decision Guidance for Optimal Load Shedding in Electric Power Microgrids*. The 2013 International Conference on Artificial Intelligence. Las Vegas, USA.

C3. Ngan, C.K., Brodsky, A., & Lin, J. (2012). *R-Checkpoint Algorithm for Multi-Event Decision Making over Multivariate Time Series*. The 16th IFIP WG8.3 International Conference on Decision Support Systems. Anávissos, Greece. (Proceedings Version of J2)

C4. Ngan, C.K., Brodsky, A., & Lin, J. (2011). *Multi-Event Decision Making over Multivariate Time Series*. Abstract Proceedings of the EWG-DSS London-2011

Workshop on Decision Systems. London, United Kingdom. (Proceedings Version of J3)

C5. Ngan, C.K., Brodsky, A., & Lin, J. (2011). *A Service Framework for Querying, Monitoring, and Learning Multivariate Time Series*. The 13th International Conference on Enterprise Information Systems. Beijing, China. (Proceedings Version of J4)

C6. Ngan, C.K., Brodsky, A., & Lin, J. (2010). *Decisions on Multivariate Time Series: Combining Domain Knowledge with Utility Maximization*. The 15th IFIP WG8.3 International Conference on Decision Support Systems. Lisbon, Portugal.

BIBLIOGRAPHY

[1]. Chatfield, C. (2001). *Time-Series Forecasting*. Chapman & Hall/CRC.

[2]. Bisgaard, S. & Kulahci, M. (2011). *Time Series Analysis and Forecasting by Example*. John Wiley & Sons, Inc.

[3]. Stack, J.B. (2009). *Technical and Monetary Investment Analysis*. https://www.investech.com/index.php.

[4]. Wikimedia Foundation. (2013). *Web Service*. Wikipedia, the Free Encyclopedia. http://en.wikipedia.org/wiki/Web_service.

[5]. Chamberlin, D. and Boyce, R. (1974). *SEQUEL: A Structured English Query Language*. Proceedings of the 1974 ACM SIGMOD Workshop on Data Description, Access, and Control.

[6]. Teorey, T., et al. (2008). *Database Design: Know It All*. Morgan Kaufmann Publishers.

[7]. Harrington, J. (2009). *Relational Database Design and Implementation (Third Edition)*. Morgan Kaufmann Publishers.

[8]. Cochrane, R., Pirahesh, H., and Mattos, N. (1996). *Integrating Triggers and Declarative Constraints in SQL Database Systems*. Proceedings of the 22th International Conference on Very Large Data Bases.

[9]. Snodgrass, R.T. (1995). *The TSQL2 Temporal Query Language*. The Springer International Series in Engineering and Computer Science, Vol. 330.

[10]. Combi, C., Montanari, A., and Pozzi, G. (2007). *The T4SQL Temporal Query Language*. Proceedings of the 16th ACM Conference on Information and Knowledge Management.

[11]. Carney, D., and others. (2002). *Monitoring Streams – A New Class of Data Management Applications*. Proceedings of the 28th International Conference on Very Large Data Bases.

[12]. Abadi, D., and others. (2003). *Aurora: A New Model and Architecture for Data Stream Management*. The International Journal on Very Large Data Bases, Vol. 12, Issue 2, Pages 120-139.

[13]. Lerner, A. & Shasha, D. (2003). *AQuery: Query Language for Ordered Data, Optimization Techniques, and Experiments*. Proceedings of the 29th International Conference on Very Large Data Bases.

[14]. Cook, D. & others. (2000). *Binary Response and Logistic Regression Analysis*. NSF/ILI Project: Beyond Traditional Statistical Methods. Iowa State University

[15]. Heij, D., De Boer, P., Franses, P.H., Kloek, T., and Van Dijk, H.K. (2004). *Econometric Methods with Applications in Business and Economics*. Oxford University Press.

[16]. Dougherty, C. (2011). *Introduction to Econometrics (Fourth Edition)*. Oxford University Press.

[17]. Bierens, J. (2008). *The Logit Model: Estimation, Testing, and Interpretation*. Department of Economics, Pennsylvania State University.

[18]. Hansen, B.E. (2013). *Econometrics*. University of Wisconsin. http://www.ssc.wisc.edu/~bhansen/econometrics/Econometrics.pdf.

[19]. Myung, J. (2003). *Tutorial on Maximum Likelihood Estimation*. Journal of Mathematical Psychology, Vol. 47, Issue 1, Pages 90–100.

[20]. Maheu, J. and McCurdy, T. (2000). *Identifying Bull and Bear Markets in Stock Returns*. Journal of Business & Economic Statistics, Vol. 18 Issue 1, Pages 100-112.

[21]. Bickel, P., Ritov, Y., and Ryden, T. (1998). *Asymptotic Normality of the Maximum-Likelihood Estimator for General Hidden Markov Models*. The Annuals of Statistics, Vol. 26, Issue 4, Pages 1614 – 1635.

[22]. Evans, J.R. (2012). *Statistics, Data Analysis, and Decision Modeling (Fifth Edition)*. Pearson Education, Inc.

[23]. Bellman, R. (2010). *Dynamic Programming*. Princeton University Press.

[24]. Bellman, R. (1966). *Adaptive Control Processes: a Guided Tour*. Princeton University Press.

[25]. Brodsky, A., Henshaw, S.M., and Whittle, J. (2008). *CARD: A Decision-Guidance Framework and Application for Recommending Composite Alternatives*. Proceedings of the 2008 ACM conference on Recommender Systems.

[26]. Brodsky, A. and Wang. X.S. (2008). *Decision-Guidance Management Systems (DGMS): Seamless Integration of Data Acquisition, Learning, Prediction, and Optimization*. Proceedings of the 41st Hawaii International Conference on System Sciences.

[27]. Brodsky, A., Bhot, M.M., Chandrashekar, M., Egge, N.E., and Wang, X.S. (2009). *A Decisions Query Language (DQL): High-Level Abstraction for Mathematical Programming over Databases*. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data.

[28]. Alrazgan, A., Ajay Nagarajan, A., Brodsky, A., and Egge, N. (2011). *Learning Occupancy Prediction Models with Decision-Guidance Query Language*. Proceedings of the 44th Hawaii International Conference on System Sciences.

[29]. Brodsky, A., Egge, N., and Wang, X. (2011). *Reusing Relational Queries for Intuitive Decision Optimization*. Proceedings of the 44th Hawaii International Conference on System Sciences.

[30]. Brodsky, A., Egge, N., and Wang, X. (2012). *Supporting Agile Organizations with a Decision Guidance Query Language*. Journal of Management Information Systems, Vol. 28, Issue 4, Pages 39 – 68.

[31]. Egge, N., Brodsky, A., and Griva, I. (2012). *Distributed Manufacturing Networks: Optimization via Preprocessing in Decision Guidance Query Language*. International Journal of Decision Support System Technology, Vol. 4, Issue 3, Pages 25 – 42.

[32]. Shao, G., Kibira, D., Brodsky, A., and Egge, N. (2011). *Decision Support for Sustainable Manufacturing Using Decision Guidance Query Language*. International Journal of Sustainable Engineering, Vol. 4, Issue 3, Pages 251 – 265.

[33]. Winston, W. (2003). *Operations Research: Applications and Algorithms (Fourth Edition)*. Cengage Learning, Inc.

[34]. Conejo, A.J., Castillo, E., Minguez, R., and Garcia-Bertrand, R. (2006) *Decomposition Techniques in Mathematical Programming*. Springer-Verlag Berin Heidelberg.

[35]. Sarker, R.A. and Newton, C.S. (2008). *Optimization Model: A Practical Approach*. CRC Press, Taylor & Francis Group.

[36]. ILOG S.A. and ILOG, Inc. (2007). *ILOG CPLEX 11.0 User's Manual*. http://www-eio.upc.es/lceio/manuals/cplex-11/html/.

[37]. Ngan, C.K. & Brodsky, A. (2013). *Optimal Event Monitoring through Internet Mash-up of Multivariate Time Series*. Special Issue on Collaborative Networking Environments and the Internet Technology. International Journal of Decision Support Systems Technology.

[38]. Wikimedia Foundation. (2013). *Web 2.0*. Wikipedia, the Free Encyclopedia. http://en.wikipedia.org/wiki/Web_2.0.

[39]. Ngan, C.K., Brodsky, A., and Lin, J. (2011). *A Service Framework for Learning, Querying and Monitoring Multivariate Time Series*. Proceedings of the 13th International Conference on Enterprise Information Systems.

[40]. Ngan, C.K., Brodsky, A., & Lin, J. (2012). *An Event-Based Service Framework for Querying, Monitoring, and Learning Multivariate Time Series*. Lecture Notes in Business Information Processing, Enterprise Information Systems, Vol. 102, Part 3, Pages 208-223.

[41]. Ngan, C.K., Brodsky, A., and Lin, J. (2010). *Decisions on Multivariate Time Series: Combining Domain Knowledge with Utility Maximization*. Proceedings of the 15th IFIP WG8.3 International Conference on Decision Support Systems.

[42]. Ngan, C.K., Brodsky, A., & Lin, J. (2011). *Multi-Event Decision Making over Multivariate Time Series*. Abstract Proceedings of the EWG-DSS London-2011 Workshop on Decision Systems.

[43]. Ngan, C.K., Brodsky, A., & Lin, J. (2013). *Multi-Event Decision Making over Multivariate Time Series*. Special Issue on Collaborative Decision Support Systems. International Journal of Information and Decision Sciences.

[44]. Ngan, C.K., Brodsky, A., & Lin, J. (2012). *R-Checkpoint Algorithm for Multi-Event Decision Making over Multivariate Time Series*. Fusing Decision Support Systems into the Fabric of the Context.

[45]. Ngan, C.K., Brodsky, A., & Lin, J. (2012). *R-Checkpoint Algorithm for Multi-Event Decision Making over Multivariate Time Series*. Proceedings of the 16th IFIP WG8.3 International Conference on Decision Support Systems.

[46]. Ypma, T. (1995). *Historical Development of the Newton-Raphson Method*. SIAM Review, Vol. 37, Issue 4, Pages 531-551.

[47]. Gradshteyn, I. S. and Ryzhik, I. M. (2000). *"Hessian Determinants." §14.314 in Tables of Integrals, Series, and Products (Sixth Edition)*. Academic Press.

[48]. Wolsey, L. and Nemhauser, G. (1999). *Integer and Combinatorial Optimization*. Wiley-Interscience, Inc.

[49]. Feuerstein, S. and Bill Pribyl, B. (2009). *Oracle PL/SQL Programming (Fifth Edition)*. O'Reilly Media.

[50]. Alonso, G., Casati, F., Kuno, H., & Machiraju, V. (2004). *Web Sevices: Concepts, Architectures and Applications*. Springer-Verlag Berlin Heidelberg.

[51]. Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, YH., Simmen, D., and A. Singh, A. (2007). *A Data Mashup Fabric for Intranet Applications*. Proceedings of the 33rd International Conference on Very Large Data Bases.

[52]. Hentenryck, P.V. (1999). *The OPL Optimization Programming Language*. The MIT Press.

[53]. The IBM Corporation. (2012). *Optimization Programming Language (OPL)*. http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp?topic=%2Filog.odms.ide.help%2FOPL_Studio%2Fmaps%2Fgroupings_Eclipse_and_Xplatform%2Fps_opl_Language_1.html.

[54]. Bentley, J.L. (1975). *Multidimensional Binary Search Trees Used for Associative Searching*. Communications of the ACM, Vol. 18, Issue 9, Pages 509-517.

[55]. Bentley, J.L. (1979). *Multidimensional Binary Search Trees in Database Applications*. The IEEE Transactions on Software Engineering, Vol. 5, Issue 4, Pages 333-340.

[56]. Samet, H. (2006). *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers.

[57]. Candelon, B., Piplack, J., & Straetmans, S. (2007). *On Measuring Synchronization of Bulls and Bears: the Case of East Asia*. Journal of Banking & Finance, Vol. 32, Issue 6, Pages 1022–1035.

[58]. Minka, T. (2007). *A Comparison of Numerical Optimizers for Logistic Regression*. http://research.microsoft.com/en-us/um/people/minka/papers/logreg/minka-logreg.pdf.

[59]. Ahsan, M.Q., Chowdhury, A.H., Ahmed, S.S., Bhuyan, I.H., Haque, M.A., & Rahman, H. (2012). *Technique to Develop Auto Load Shedding and Islanding Scheme to Prevent Power System Blackout*. The IEEE Transactions on Power Systems, Vol. 27, Issue 1, Pages 198 – 205.

[60]. Mahat, P., Chen, Z., & Bak-Jensen, B. (2010). *Underfrequency Load Shedding for an Islanded Distribution System with Distributed Generators*. The IEEE Transactions on Power Delivery, Vol. 25, Issue 2, Pages 911 – 918.

[61]. Ding, Z., Cartes, D.A., & Srivastava, S. (2006). *New Load Shedding Scheme for Islanded Power Systems*. Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering.

[62]. You, H., Vittal, V., & Yang, Z. (2003). *Self-Healing in Power Systems: An Approach Using Islanding and Rate of Frequency Decline-Based Load Shedding*. The IEEE Transactions on Power Systems, Vol. 18, Issue 1, Pages 174 – 181.

[63]. Delfino, B., Massucco, S., Morini, A., Scalera, P., & Silvestro, F. (2001). *Implementation and Comparison of Different under Frequency Load-shedding Schemes*. Proceedings of Power Engineering Society Summer Meeting, Vol. 1, Pages 307 – 312.

[64]. Ameren Illinois Power Company. (2010). *Real-time Pricing for Residential Customers*. http://www.ameren.com/sites/aiu/ElectricChoice/Pages/ResRealTimePricing.aspx.

[65]. Centolella, P. (2010). *The Integration of Price Responsive Demand into Regional Transmission Organization (RTO) Wholesale Power Markets and System Operations*. Energy, Vol. 35, Issue 4, Pages 1568 - 1574.

[66]. Mohsenian-Rad, A-H. & Leon-Garcia, A. (2010). *Optimal Residential Load Control with Price Prediction in Real-Time Electricity Pricing Environments*. The IEEE Transactions on Smart Grid, Vol. 1, Issue 2, Pages 120 - 133.

[67]. Burke, W. & Auslander, D. (2009). *Residential Electricity Auction with Uniform Pricing and Cost Constraints*. Proceedings of North American Power Symposium.

[68]. Ontario Newsroom for Residents. (2009). *Smarter Electricity Pricing Coming to Ontario: McGuinty Government Rolls Out Time-of-Use Rates*. http://news.ontario.ca/mei/en/2009/05/smarter-electricity-pricing-coming-to-ontario.html.

[69]. Holland, S., & Mansur, E. (2008). *Is Real-time Pricing Green? The Envirobmental Impacts of Electricity Demand Variance*. The Review of Economics and Statistics, Vol. 90, Issue 3, Pages 550 – 561.

[70]. Alexander, B. (2007). *Smart Meters, Real Time Pricing, and Demand Response Programs: Implications for Low Income Electric Customers*. Oak Ridge National Laboratory Technical Report.

[71]. Wolak, F. (2006). *Residential Customer Response to Real-time Pricing: The Anaheim Critical Peak Pricing Experiment*. Center for the Study of Energy Markets, University of California Energy Institute, UC Berkeley.

[72]. Borenstein, S. (2004). *The Long-run Effects of Real-time Electricity Pricing*. Center for the Study of Energy Markets, University of California Energy Institute, UC Berkeley.

[73]. Broccard, M., Girdinio, P., Moccia, P., Molfino, P., Nervi, M., and Pini Prato, A., (2010). *Quasi Static Optimized Management of a Multinode CHP Plant*. Journal of Energy Conversion and Management, Vol. 51, Issue 11, Pages 2367–2373.

[74]. M. Bojić, M. and Stojanović, B., (1998). *MILP Optimization of a CHP Energy System*. Energy Conversion and Management, Vol. 39, Issue 7, Pages 637–642.

[75]. SAS Institute, Inc., (2012). *SAS/OR(R) 9.22 User's Guide: Mathematical Programming*. http://support.sas.com/documentation/cdl/en/ormpug/63352/HTML/default/viewer.htm#ormpug_milpsolver_sect001.htm.

[76]. Savola, T. and Keppo, I. (1997). *Off-design Simulation and Mathematical Modeling of Small-scale CHP plants at part loads*. Journal of Applied Thermal Engineering, Vol. 25, Issues 8-9, Pages 1219–1232.

[77]. Tuula Savola, T. and Fogelholm, C. (2007). *MINLP Optimization Model for Increased Power Production in Small-scale CHP Plants*. Journal of Applied Thermal Engineering, Vol. 27, Issue 1, Pages 89–99.

[78]. Tuula Savola, T., Tveit, T., and Fogelholm, C. (2007). *A MINLP Model Including the Pressure Levels and Multiperiods for CHP Process Optimization*. Journal of Applied Thermal Engineering, Vol. 27, Issues 11–12, Pages 1857–1867.

[79]. Biezma, M. and San Cristobal, J. (2006). *Investment Criteria for the Selection of Cogeneration Plants – a State of the Art Review*. Journal of Applied Thermal Engineering, Vol. 26, Issues 5-6, Pages 583–588.

[80]. American Electric Power Inc., (2012). *CCS Front End Engineering & Design Report American Electric Power Mountaineer CCS II Project Phase 1*. http://cdn.globalccsinstitute.com/sites/default/files/publications/32481/ccs-feed-report-gccsi-final.pdf

BIOGRAPHY

Chun-Kit Ngan is a Ph.D. graduate in Information Technology in the Department of Computer Science at George Mason University (GMU). He received his MBA in Management Information Systems from California State University, Chico and his BEng in Electronic Engineering from the Hong Kong University of Science and Technology.

Currently, Dr. Ngan is an Adjunct Professor in the Department of Applied Information Technology at GMU, where he has taught a number of undergraduate courses, such as Object-Oriented Programming, Event-Driven Programming, and Program Design and Data Structures. His teaching interests include database management systems, software application programming, optimization models, data structures, and computational algorithms.

His research interests are Decision Guidance, Support and Optimization (DGSO) systems and the DGSO applications over multivariate time series. He has published a number of research papers in various conferences and journals. He received a Best Paper Award in 2013 and a Best Student Paper Award in 2011 at the 13th and 15th International Conference on Enterprise Information Systems respectively. He is also an active member of the EWG-DSS EURO Working Group, the IEEE Communications Society, and the IEEE Computer Society, where he was awarded the 2012 Upsilon Pi Epsilon/Computer Science Award for academic excellence.

Dr. Ngan is presently working as a Software Engineer for Syneren Technologies Corporation, which is located in Arlington, VA. In the spring 2014, he will join the Division of Engineering and Information Science in the School of Graduate Professional Studies at the Pennsylvania State University as an Assistant Professor of Information Science.