March 2017

Optimal Bidding Strategy for a Strategic Power Producer Using Mixed Integer Programming

Sayed Abdullah Sadat
University of South Florida, s.abdullahs1@live.com

Optimal Bidding Strategy for a Strategic Power Producer Using Mixed Integer Programming

by

Sayed Abdullah Sadat

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering
College of Engineering
University of South Florida

Major Professor: Lingling Fan, Ph.D.
Selcuk Kose, Ph.D.
Zhixin Miao, Ph.D.

Date of Approval:
March 10, 2017

Keywords: Chance Constraints, MILP, Pool Strategy, Electricity Market, Nonlinear Optimization, MPEC, Almost Robust Optimization

DEDICATION

To my parents

Sayed Abdul Rahman and Suraya Sediqi

ACKNOWLEDGMENTS

Firstly, I would like to appreciate my advisor Dr. Lingling Fan who has immensely supported me in my research for the past two semesters. I would especially appreciate an excellent guidance that she has provided to me, which has not only been useful in doing research on popular topics and publish a high quality paper, but also to develop skills that would further enhance my research capacities. She has a wide range of research interests and sharp insights. She was always available to help and has been consistently motivating me to do better, which truly helped me.

Secondly, I would like extend my profound gratitude to Dr. Selcuk Kose who has not just accepted to serve in my thesis committee, but has also extended all the material support and facility for my work and research during my thesis. I found him to be a great mentor in developing my research skills.

I appreciate Dr. Zhixin Miao for his support as a power and energy track advisor, his advice, and helpful comments as the committee member for my thesis.

I would like to extend my profound appreciation to Dr. Hadi Charkhgard for his support on optimization techniques, and providing recommendations and insight on the use of different solvers and modeling techniques.

Finally, I would like to thank the U.S. State Department, particularly the Fulbright Student Program for their generous and valuable financial support for my Master of Science in Electrical Engineering degree at the University of South Florida.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The thesis focuses on a mixed integer linear programming (MILP) formulation for a bi-level mathematical program with equilibrium constraints (MPEC) considering chance constraints. The particular MPEC problem relates to a power producer's bidding strategy: maximize its total benefit through determining bidding price and bidding power output while considering an electricity pool's operation and guessing the rival producer's bidding price. The entire decision-making process can be described by a bi-level optimization problem. The contribution of our thesis is the MILP formulation of this problem considering the use of chance constrained mathematical program for handling the uncertainties.

First, the lower-level poor operation problem is replaced by Karush-Kuhn-Tucker (KKT) optimality condition, which is further converted to an MILP formulation except a bilinear item in the objective function. Secondly, duality theory is implemented to replace the bilinear item by linear items. Finally, two types of chance constraints are examined and modeled in MILP formulation. With the MILP formulation, the entire MPEC problem considering randomness in price guessing can be solved using off-shelf MIP solvers, e.g., Gurobi. A few examples and a case study is given to illustrate the formulation and show the case study results.

CHAPTER 1

INTRODUCTION

The electrical energy supply industry has gone through major restructuring process in many countries. It is becoming deregulated and competition based industry. The details of restructuring process and the framework for regulation can vary from country to country or from region to region. However, the overall organization in majority of the cases are based on the same principle, which is the generation part of the power system becomes separated and deregulated [1].

In this type of electricity market, the producers could either submit profit-maximizing bids to a pool or optimality self-schedule its generation in response to prices [2]. The basic feature of the deregulated market is the formation of wholesale energy market (WEM). All transactions related to electric power purchase takes place in WEM [1].

Many problems arising from engineering and economics are mathematical problems with equilibrium constraints (MPEC) [3]. In this thesis, a particular MPEC problem relates to a strategic power producer trying to maximize its total benefit through determining bidding price and bidding power output while considering an electricity pool's operation and guessing the rival producer's bidding price. The entire decision-making process can be described by a bi-level optimization problem. The upper-level tries to maximize the power producer's profit and minimize its cost while the lower-level emulates the decision making of a pool: minimize the total operation cost considering the bidding prices from the power producers while guaranteeing generation and load balance as well as enforcing the network limits. Optimal bidding strategy problem formulations have been seen in the literature, e.g., [4, 5, 6, 7, 8].

There are fundamentally two approaches for solving an MPEC problem [3]: (i) nonlinear optimization program solving techniques and (ii) MILP solving techniques.

The optimal bidding problem described by a bi-level program can be converted to a nonlinear programming problem after the lower-level optimization problem is replaced by the KKT optimality condition. Naturally, nonlinear program solving techniques, e.g., interior point method, can be applied to solve the problem. Such approach can be found in the literature [9, 10].

On the other hand, in the second approach, the nonlinear programming problem should be converted to an MILP first. MILP formulation has been adopted in [8] to express the complementary slackness condition in the KKT optimality by introducing binary variables and adopting the Big-M technique. Further, duality theory is used to linearize the objective function with a bilinear term by an MILP formulation.

Uncertainty is also considered for the optimal bidding problem in [8] where a stochastic programming problem is formulated and solved. [6] considers a similar problem with wind generation included for both day-ahead and real-time market. [6] also incorporates risk management and the uncertainty of other strategic power producers through stochastic programming approach.

The main idea is to provide a bidding strategy for the strategic power producer to be able to exercise market power in contrast to price taking behaviour. It could set its production quantity and/or prices at which it is willing to sell energy output in order to influence the market price [11]. By taking such actions, the firm risks selling less, but it raises the price it will get for all output that it does sell [11]. This bring into picture the idea of embedding a risk factor into the handling uncertainties approach such that the firm should be able to decide how far it is ready to go with risks involved in exercising the market power.

The other scenario where the integration of a risk factor into uncertainty model can be handy is when there is uncertainty in the total capacity of generation. This can be due to the combinational

use of renewable energy sources and conventional sources or some issues technical issue that would bring uncertainty to the total generation capacity of the bidding power producer.

Most recently, chance constrained mathematical program (CCMP) has found applications in power market to capture the randomness. For example, [12] investigated chance constrained unit commitment problems.

In this thesis, chance constrained optimal bidding strategy will be examined with chance constraints being modeled by MILP formulation using the method in [13]. The contribution of my thesis is the MILP formulation of a chance constrained optimal bidding strategy problem. First, the lower-level pool operation problem is replaced by Karush-Kuhn-Tucker (KKT) optimality condition, which is further converted to an MILP formulation except a bilinear term in the objective function. Secondly, duality theory is implemented to replace the bilinear term by its equivalent linear terms. Finally, two cases of chance constraints are examined and modeled in MILP formulation. The first case is for the bidding power producer that wants to exercise market power given the risks, and the second case is for the second scenario where the total generation capacity is affected by an uncertainty.

In the MILP formulation proposed, the entire MPEC problem considering randomness in price guessing can be solved using off-shelf MIP solvers, e.g., Gurobi. A few examples are given to illustrate the formulation and show the case study results.

CHAPTER 2

THE OPTIMAL BIDDING STRATEGY PROBLEM

In this chapter, a simple two-generator system is used to explain the optimal bidding strategy problem. The system is shown in Fig. 2.1. In this system, λ_s and λ_R are marginal costs of the two generators and are given. P_S, P_R, λ_1 and λ_2 are to be determined by the proposed strategy.

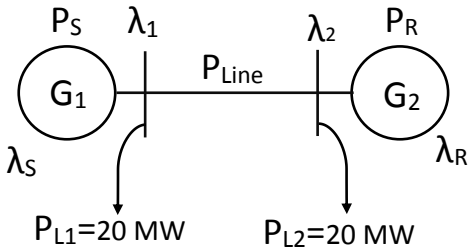

Figure 2.1. A simple power system with two producers.

Gen S will carry out the optimal bidding strategy decision making process to determine its bidding price α_s and bidding power P_S. Gen S will treat the marginal cost of the rival generator Gen R (λ_R) as known.

2.1 Upper Level Problem

The upper level problem is to determine the bidding price α_S and the power P_S. The objective is maximizing the total profit of the strategic producer, with the bidding price α_S not appearing. α_S will appear in the objective function of the lower-level problem. α_S will affect the bidding power P_S. $\min_{\alpha_S, P_S} \quad \lambda_S P_S - \lambda_1 P_S$ (2.1)

where λ_S is the marginal cost of the strategic power producer, and P_S is power produced by the strategic power producer. The first term sums the total cost. In the second term λ_1 is the locational marginal price at bus 1. The second term is the total revenue generated. λ_1 will be determined by the pool or the lower-level problem. Therefore, the objective function maximizes the total profit by minimizing the cost and maximizing the revenue. The generation P_S belongs to the feasible region defined by the lower level problem.

2.2 Lower Level Problem

The lower level problem is formulated to represent the market clearing process, and is shown as follows along with their dual variables:

$$\min_{P_S, P_R} \quad \alpha_S P_S + \lambda_R P_R \tag{2.2}$$

$$\text{subject to} \quad P_S - P_{L1} = P_{Line} : \lambda_1 \tag{2.3}$$

$$P_R - P_{L2} = -P_{Line} : \lambda_2 \tag{2.4}$$

$$P_{Line} \leq P_{Line}^{max} : \mu_{Line} \tag{2.5}$$

$$0 \leq P_S \leq P_S^{max} : \mu_S^{min}, \mu_S^{max} \tag{2.6}$$

$$0 \leq P_R \leq P_R^{max} : \mu_R^{min}, \mu_R^{max} \tag{2.7}$$

where λ_R is the marginal cost of the rival power producer, P_R is the power produced by the rival power producer, P_{Line} is the power flow in the Transmission line from bus 1 to bus 2, and P_{L1} and P_{L2} are constant loads at buses 1 and 2, respectively. P_{Line}^{max} is the transmission capacity of line 1-2, P_S^{max} is the upper limit of the strategic power producer and P_R^{max} is the upper limit of the rival power producer. Constraints (2.3) and (2.4) enforce power balance in the network, and constraints (2.5) - (2.7) are power bounds for their respective variables.

2.3 Model Conversion to MPEC

We first replace the lower-level problem by the KKT conditions and convert the bi-level problem to an MPEC problem.

The Lagrangian function of the lower-level problem is formulated as shown below:

$$\mathcal{L}\ (P_S, P_R, \lambda_1, \lambda_2, \mu_S^{min}, \mu_S^{max}, \mu_R^{min}, \mu_R^{max}, \mu_{Line}) = (\quad \alpha_S P_S + \lambda_R P_R + \lambda_1(P_{Line} + P_{L1} - P_S)$$

$$+\lambda_2(P_{L2} - P_{Line} - P_R) + \mu_S^{max}(P_S - P_S^{max}) - \mu_S^{min} P_S + \mu_{Line}(P_{Line} - P_{Line}^{max}) + \mu_R^{max}(P_R - P_R^{max}) -$$

$$\mu_R^{min} P_R)$$

The KKT conditions are as following:

$$\alpha_S - \lambda_1 - \mu_S^{min} + \mu_S^{max} = 0 \tag{2.8}$$

$$\lambda_R - \lambda_2 - \mu_R^{min} + \mu_R^{max} = 0 \tag{2.9}$$

$$P_S - P_{L1} = P_{Line} \tag{2.10}$$

$$P_R - P_{L2} = -P_{Line} \tag{2.11}$$

$$0 \leq P_{Line}^{max} - P_{Line} \perp \mu_{Line} \geq 0 \tag{2.12}$$

$$0 \leq P_S \perp \mu_S^{min} \geq 0 \tag{2.13}$$

$$0 \leq P_R \perp \mu_R^{min} \geq 0 \tag{2.14}$$

$$0 \leq P_S^{max} - P_S \perp \mu_S^{max} \geq 0 \tag{2.15}$$

$$0 \leq P_R^{max} - P_R \perp \mu_R^{max} \geq 0 \tag{2.16}$$

Hence, the MPEC model for the optimal bidding strategy of the first generator in the 2-bus problem becomes the constraints (2.8) to (2.16), and (2.1) as the objective function.

CHAPTER 3

NONLINEAR SOLUTION TECHNIQUES FOR MPECS

Mathematical programs with equilibrium constraints (MPECs) form a relatively new and interesting subclass of nonlinear programming problems [14]. Among the many techniques available for the numerical solutions of Mathematical Programs with Equilibrium Constraints, the widely used approaches are nonlinear programming approach and Mixed Integer Linear programming (MILP) [3]. If a formulated within the accepted range of accuracy could be achieved, MILP formulation remains the most preferred approach as it can be easily solved using the commercial sovers available in the market. The MILP approach is discussed in detail in the following chapter. In nonlinear programming approach, the equilibrium constraints are either converted to a smooth equation or augmented to the objective via a suitable error bound [3].

There are a number of nonlinear programming approaches including the following three approaches, which are mostly adopted in the literature.

1. SQP Methods

2. Artificial Intelligence Approach

3. Interior Point Methods

In the following sections we breifly introduce the aforementioned methods for solving nonlinear MPEC problems. In the available commercial ad non-commercial solvers, it is seen that the interior point method is preferred approach for the nonlinear optimization problems. Therefore, We apply

the the interior point method to our example problem and its MPEC formulation. We later examine the result.

3.1 SQP Methods

Recently in some cases nonlinear programming solvers have been used to solve some MPEC problems. In [15], the author claims that the sequential quadratic programming (SQP) methods have been successful in solving nonlinear MPEC problems. [15] examines the local convergence properties of SP methods it has applied to an MPEC problem. It shows that the SQP converges super-linearly under reasonable assumptions near a strongly stationary points [15].

Suppose for a given nonlinear complementarity problem (NCP) shown below,

$$\min_{x \leq 0, y, s} \quad f(x, y) \tag{3.1}$$

$$\text{subject to} \quad g(x, y) \geq 0 \tag{3.2}$$

$$h(x, y) - s = 0 \tag{3.3}$$

$$0 \leq y \perp s \geq 0 \tag{3.4}$$

The complementary constraint (3.4) can be replaced by a nonlinear inequality, the optimization problem then becomes [3]:

$$\min_{x \leq 0, y, s} \quad f(x, y) \tag{3.5}$$

$$\text{subject to} \quad g(x, y) \geq 0 \tag{3.6}$$

$$h(x, y) - s = 0 \tag{3.7}$$

$$Ys \leq, \quad y \geq 0, \quad s \geq 0 \tag{3.8}$$

where Y=diag(y_i). The nonlinear equivalent of the MPEC problem can then be solved by applying standard NLP solvers.

3.2 Artificial Intelligence Approach

A number of algorithms have been developed that are also called as artificial intelligence approach. In this approach, an evolutionary algorithm is a subset of evolutionary computation, a generic population-based meta-heuristic optimization algorithm. The well-known example fo this approach is genetic algorithms (GAs).

GAs are a special sort of optimization algorithm as stated by [16]. It states that "all optimization algorithms can be thought of as ways of exploring the space of possible solutions to a problem, and selecting one (or several) possible solutions as being optimal. The GA uses a close analogy with Darwinian evolutionary search to select possible solutions: a number of solutions are evaluated for fitness, and the fitter solutions reproduce, recombine, and possibly mutate. The average fitness of solutions tends to increase, and the algorithm stops searching either after a specified number of generations, or once some other externally defined criterion is satisfied. Thus, GAs are a method of optimization which use an evolutionary process to generate increasingly good solutions to the problem posed. The driving idea behind their use is that natural evolution has solved some extraordinarily complex design optimization problems; simulating this process may allow us also to solve complex optimizations".

Application of genetic algorithms in bilevel programming solution has been presented in [17, 18]. In [17], the upper level of the bilevel programming problem consists of the objective of the leader, and the lower level problem consist of the follower. The KKT conditions are applied ot the lower level program and thereby converting a bilevel programming problem was transformed into a one level problem with complementary constraints. This is followed by the use of the genetic algorithm to solve the single level problem. The work has presented the Numerical results of the proposed GA against hybrid Tabu-ascent algorithm and showed its efficiency in terms of computation while maintaining the quality of solutions same for the both compared methods [3].

A good example of the application of co-evolutionary programming method can be found in [19]. This work has analyzed the supply function equilibrium (SFE) models of an oligopolistic electricity market where the affine function model and the piecewise affine function model were considered. The rapid convergence characteristics of the SFE in the affine function model is shown through the simulation results, and hence the method method has the great potential to be used to solve the real equilibrium problems of electricity markets [3].

3.3 Interior Point Methods

Interior point methods for nonlinear programs (NLP) are adapted for solution of mathematical programs with complementarity constraints (MPCCs). The constraints of the MPCC are suitably relaxed so as to guarantee a strictly feasible interior for the inequality constraints [20].

Interior-point methods solve this type of problems or their KKT conditions by applying Newtons method to a sequence of equality constrained problems, or to a sequence of modified versions of the KKT conditions [21].

As appeared in [20], let's consider the formulation as shown below:

$$\min_{x \geq 0, w, y} \quad f(x, w, y) \tag{3.9}$$

$$\text{subject to} \quad h(x, w, y) = 0 \tag{3.10}$$

$$0 \leq w \perp y \geq 0 \tag{3.11}$$

Equation (3.11) can also be written as $wy = 0$, which means either of the variables w or y must be zero.

Let's apply the interior point method to the example shown in 2.1. We use the MPEC model represented by the single level objective function (2.1) and constraints as (2.8) to (2.16) that can

also be written in the form as shown in (3.12) - (3.13).

$$\min_{\alpha_S, P_S} \quad \lambda_S P_S - \lambda_1 P_S \tag{3.12}$$

subject to

$$\alpha_S - \lambda_1 - \mu_S^{min} + \mu_S^{max} = 0 \tag{3.13}$$

$$\lambda_R - \lambda_2 - \mu_R^{min} + \mu_R^{max} = 0 \tag{3.14}$$

$$P_S - P_{L1} = P_{Line} \tag{3.15}$$

$$P_R - P_{L2} = -P_{Line} \tag{3.16}$$

$$\mu_{Line}(P_{Line}^{max} - P_{Line}) = 0 \tag{3.17}$$

$$P_S \ \mu_S^{min} = 0 \tag{3.18}$$

$$P_R \ \mu_R^{min} = 0 \tag{3.19}$$

$$\mu_S^{max}(P_S^{max} - P_S) = 0 \tag{3.20}$$

$$\mu_R^{max}(P_R^{max} - P_R) = 0 \tag{3.21}$$

CHAPTER 4

MILP FORMULATION

The MPEC problem will be formulated into an MILP problem. We carry out linearization in the objective function as well as in the constraints. The MPEC model obtained after applying the KKT conditions, includes the following nonlinearities:

1. The term $\lambda_1 P_S$ in the objective function.

2. The complementarity conditions in the constraints.

4.1 Bilinear Term in the Objective Function

To find a linear expression for $\lambda_1 P_S$, we can use the strong duality condition and some of the KKT equalities. The strong duality theorem says that if a problem is convex, the objective functions of the primal and dual problems have the same value at the optimum. The complementarity conditions in the constraints can be dealt with using the slack variables and the "big-M" method. Thus, an MILP formulation is achieved [8].

According to the strong duality theorem, at optimal point Z_{primal} is equal to Z_{dual}. Hence,

$$Z_{primal} = \alpha_S P_S + \lambda_R P_R \tag{4.1}$$

$$Z_{dual} = -\mu_{Line} P_{Line}^{max} - \mu_S^{max} P_S^{max} - \mu_R^{max} P_R^{max} \tag{4.2}$$

$$Z_{primal} = Z_{dual} \tag{4.3}$$

Let's consider equation (2.8) and multiply it with P_S

$$\alpha_S P_S - \lambda_1 P_S - \mu_S^{min} P_S + \mu_S^{max} P_S = 0 \tag{4.4}$$

From equation (2.12), we have

$$\mu_S^{min} P_S = 0 \tag{4.5}$$

From equation (2.14), we have

$$\mu_S^{max} P_S = \mu_S^{max} P_S^{max} \tag{4.6}$$

Substituting equations (4.4) - (4.6) in (4.3),

$$\lambda_1 P_S = -\lambda_R P_R - \mu_{Line} P_{Line}^{max} - \mu_R^{max} P_R^{max} \tag{4.7}$$

The objective function becomes,

$$\min \ \lambda_S P_S + \lambda_R P_R + \mu_{Line} P_{Line}^{max} + \mu_R^{max} P_R^{max} \tag{4.8}$$

and the linear constraints are (2.8)- (2.11).

4.2 Complementary Slackness

The complementary slackness conditions (12)-(16) can be converted to MILP formulations by introducing a binary variable for each complementary slackness condition.

$$0 \leq P_{Line}^{max} - P_{Line} \leq \omega_{max}^{Line} M \tag{4.9}$$

$$0 \leq \mu_{Line} \leq (1 - \omega_{max}^{Line}) M \tag{4.10}$$

$$0 \leq P_S \leq \omega_{min}^{S} M \tag{4.11}$$

$$0 \leq \mu_S^{min} \leq (1 - \omega_{min}^{S}) M \tag{4.12}$$

$$0 \leq P_R \leq \omega_{min}^{R} M \tag{4.13}$$

$$0 \leq \mu_R^{min} \leq (1 - \omega_{min}^{R}) M \tag{4.14}$$

$$\omega_{max}^{Line}, \omega_{min}^{S}, \omega_{min}^{R} \in \{0, 1\} \tag{4.15}$$

$$0 \leq P_S^{max} - P_S \leq \omega_{max}^{S} M \tag{4.16}$$

$$0 \leq \mu_S^{max} \leq (1 - \omega_{max}^S)M \tag{4.17}$$

$$0 \leq P_R^{max} - P_R \leq \omega_{max}^R M \tag{4.18}$$

$$0 \leq \mu_R^{max} \leq (1 - \omega_{max}^R)M \tag{4.19}$$

$$\omega_{max}^S, \omega_{max}^R \in \{0, 1\} \tag{4.20}$$

For example, ω_{max}^S is introduced to indicate if the generator's upper bound is hit ($\omega_{max}^S = 0$) or not ($\omega_{max}^S = 1$). M is a big number. When the limit is hit, $0 \leq P_S^{max} - P_S \leq \omega_{max}^S M$ is equivalent to $P_S = P_S^{max}$. When the limit is not hit, $0 \leq \mu_S^{max} \leq (1 - \omega_{max}^S)M$ is equivalent to $\mu_S^{max} = 0$.

The model is now a mixed integer linear program (MILP), with (2.1) as objective function, and (2.8)-(2.16), (4.9)-(4.20) as constraints.

4.3 Five Bus Example

Let's consider the 5 bus system shown below in Fig. 4.1. The formulation that can identify the best offering strategy for the strategic power producer can be stated using the following bilevel model. For the simplicity we assume that the system is uncontested, no startup or shutdown costs, and no ramp-up or ramp-down limitations.

$$\min_{\Gamma_{tib}^S, P_{tib}^S, \forall t, \forall i, \forall b} \sum_{tib} \lambda_{tib}^S P_{tib}^S - \sum_{t(i \in \Psi_n)b} \alpha_{tn} P_{tib}^S = f(\Gamma_{tib}^S, P_{tib}^S) \tag{4.21}$$

subject to:

$$\alpha_{tn} = \lambda_{tn}, \quad \forall t, \forall n \tag{4.22}$$

$$P_{tib}^S \in \arg \min_{P_{tib}^S, P_{tjb}^R, P_{tdk}^D} \sum_{tib} \Gamma_{tib}^S P_{tib}^S + \lambda_{tjb}^R P_{tjb}^R - \sum_{t(i \in \Psi_n)b} \lambda_{tdk}^D P_{tdk}^D \tag{4.23}$$

subject to:

$$\lambda_{tn} : \sum_{(i \in \Psi_n)b} P_{tib}^S + \sum_{(j \in \Psi_n)b} P_{tjb}^R = \sum_{(d \in \Psi_n)k} P_{tdk}^D \qquad \forall t, \forall n \tag{4.24}$$

$$\mu_{tib}^{S^{min}}, \mu_{tib}^{S^{max}} : 0 \leq P_{tib}^{S} \leq P_{tib}^{S^{max}} \qquad \forall t, \forall i, \forall b \qquad (4.25)$$

$$\mu_{tjb}^{O^{min}}, \mu_{tjb}^{O^{max}} : 0 \leq P_{tjb}^{R} \leq P_{tjb}^{O^{max}} \qquad \forall t, \forall j, \forall b \qquad (4.26)$$

$$\mu_{tdk}^{D^{min}}, \mu_{tdk}^{D^{max}} : 0 \leq P_{tdk}^{D} \leq P_{tdk}^{D^{max}} \qquad \forall t, \forall d, \forall k \qquad (4.27)$$

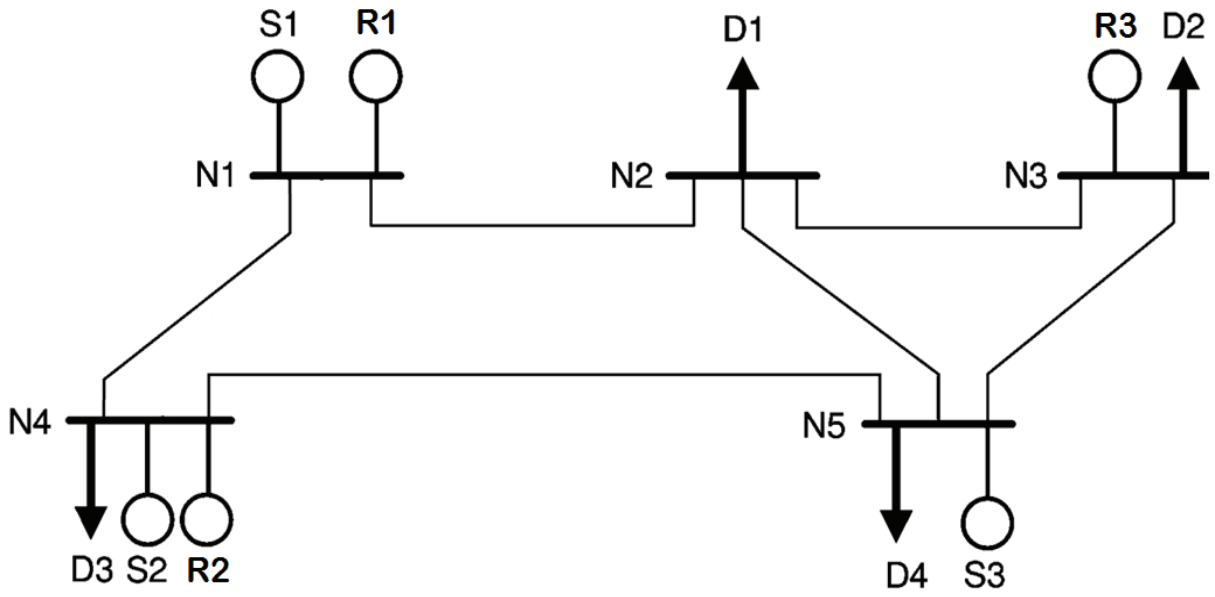

Figure 4.1. A five-bus test system, where S generators are the strategic power producers, R are the rival power producers and D are the demands in every bus.

4.3.1 MPEC Model

Converting the bilevel formulated model to mathematical program with equilibrium constraints (MPEC) using Lagrangian duality and KKT conditions (i.e. A similar procedure as defined in [8] is followed for a 5 bus system), we get:

$$\min_{\Gamma_{tib}^{S}, P_{tib}^{S}, \forall t, \forall i, \forall b} \sum_{tib} \lambda_{tib}^{S} P_{tib}^{S} - \sum_{t(i \in \Psi_n)b} \lambda_{tn} P_{tib}^{S} \qquad (4.28)$$

subject to

$$\Gamma_{tib}^{S} - \lambda_{tn} - \mu_{tib}^{S^{min}} + \mu_{tib}^{S^{max}} = 0 \qquad \forall t, \forall i \in \Psi_n, \forall b$$

$$\lambda^R_{tjb} - \lambda_{tn} - \mu^{R^{min}}_{tjb} + \mu^{O^{max}}_{tjb} = 0 \qquad \forall t, \forall j \in \Psi_n, \forall b \qquad (4.29)$$

$$-\lambda^D_{tdk} + \lambda_{tn} - \mu^{D^{min}}_{tdk} + \mu^{D^{max}}_{tdk} = 0 \qquad \forall t, \forall d \in \Psi_n, \forall k \qquad (4.30)$$

$$\sum_{(i\in\Psi_n)b} P^S_{t(i\in\Psi_n)b} + \sum_{(j\in\Psi_n)b} P^O_{t(j\in\Psi_n)b} = \sum_{(d\in\Psi_n)k} P^D_{t(d\in\Psi_n)k} \qquad \forall t, \forall n \qquad (4.31)$$

$$0 \le P^S_{tib} \perp \mu^{S^{min}}_{tib} \ge 0, \qquad \forall t, \forall i, \forall b \qquad (4.32)$$

$$0 \le P^R_{tjb} \perp \mu^{O^{min}}_{tjb} \ge 0, \qquad \forall t, \forall j, \forall b \qquad (4.33)$$

$$0 \le P^D_{tdk} \perp \mu^{D^{min}}_{tdk} \ge 0, \qquad \forall t, \forall d, \forall k \qquad (4.34)$$

$$0 \le P^{S^{max}}_{tib} - P^S_{tib} \perp \mu^{S^{max}}_{tib} \ge 0, \qquad \forall t, \forall i, \forall b \qquad (4.35)$$

$$0 \le P^{O^{max}}_{tjb} - P^R_{tjb} \perp \mu^{O^{max}}_{tjb} \ge 0, \qquad \forall t, \forall j, \forall b \qquad (4.36)$$

$$0 \le P^{D^{max}}_{tdk} - P^D_{tdk} \perp \mu^{D^{max}}_{tdk} \ge 0, \qquad \forall t, \forall d, \forall k \qquad (4.37)$$

4.3.2 Equivalent Linear Formulation

The MPEC model (4.29) to (4.37) includes the following nonlinearities:

1. The term $\lambda_1 P_S$ in the objective function.

2. The complementarity conditions in the constraints.

To find a linear expression for $\lambda_{tn} P^S_{tib}$, we use the strong duality condition and some of the KKT equalities. The strong duality theorem says that if a problem is convex, the objective functions of the primal and dual problems have the same value at the optimum [8]. The linearized form of the model is as below:

$$\min \sum_{tib} \lambda^S_{tib} P^S_{tib} + \sum_{tjb} \lambda^O_{tjb} P^R_{tjb} - \sum_{tdk} \lambda^D_{tdk} P^D_{tdk} + \sum_{tjb} P^{R^{max}}_{tjb} \mu^{R^{max}}_{tjb} + \sum_{tdk} P^{D^{max}}_{tdk} \mu^{D^{max}}_{tdk} \qquad (4.38)$$

subject to

$$\Gamma^S_{tib} - \lambda_{tn} - \mu^{S^{min}}_{tib} + \mu^{S^{max}}_{tib} = 0 \qquad \forall t, \forall i \in \Psi_n, \forall b \qquad (4.39)$$

$$\lambda^{O}_{tjb} - \lambda_{tn} - \mu^{O^{min}}_{tjb} + \mu^{O^{max}}_{tjb} = 0 \qquad \forall t, \forall j \in \Psi_n, \forall b \qquad (4.40)$$

$$-\lambda^{D}_{tdk} + \lambda_{tn} - \mu^{D^{min}}_{tdk} + \mu^{D^{max}}_{tdk} = 0 \qquad \forall t, \forall d \in \Psi_n, \forall k \qquad (4.41)$$

$$\sum_{(i \in \Psi_n)b} P^{S}_{t(i \in \Psi_n)b} + \sum_{(j \in \Psi_n)b} P^{O}_{t(j \in \Psi_n)b} = \sum_{(d \in \Psi_n)k} P^{D}_{t(d \in \Psi_n)k} \qquad \forall t, \forall n \qquad (4.42)$$

$$P^{S}_{tib} \geq 0 \qquad \forall t, \forall i, \forall b \qquad (4.43)$$

$$\mu^{S^{min}}_{tib} \geq 0 \qquad \forall t, \forall i, \forall b \qquad (4.44)$$

$$P^{S}_{tib} \leq (1 - \omega^{S^{min}}_{tib}) M^{P}, \qquad \forall t, \forall i, \forall b \qquad (4.45)$$

$$\mu^{S^{min}}_{tib} \leq \omega^{S^{min}}_{tib} M^{\mu P}, \qquad \forall t, \forall i, \forall b \qquad (4.46)$$

$$P^{R}_{tjb} \geq 0 \qquad \forall t, \forall j, \forall b \qquad (4.47)$$

$$\mu^{O^{min}}_{tjb} \geq 0 \qquad \forall t, \forall j, \forall b \qquad (4.48)$$

$$P^{R}_{tjb} \leq (1 - \omega^{O^{min}}_{tjb}) M^{P}, \qquad \forall t, \forall j, \forall b \qquad (4.49)$$

$$\mu^{O^{min}}_{tjb} \leq \omega^{O^{min}}_{tjb} M^{\mu P}, \qquad \forall t, \forall j, \forall b \qquad (4.50)$$

$$P^{D}_{tdk} \geq 0 \qquad \forall t, \forall d, \forall k \qquad (4.51)$$

$$\mu^{D^{min}}_{tdk} \geq 0 \qquad \forall t, \forall d, \forall k \qquad (4.52)$$

$$P^{D}_{tdk} \leq (1 - \omega^{D^{min}}_{tdk}) M^{P}, \qquad \forall t, \forall d, \forall k \qquad (4.53)$$

$$\mu^{D^{min}}_{tdk} \leq \omega^{D^{min}}_{tdk} M^{\mu P}, \qquad \forall t, \forall d, \forall k \qquad (4.54)$$

$$\omega^{S^{min}}_{tib}, \omega^{O^{min}}_{tjb}, \omega^{D^{min}}_{tdk} \in \{0, 1\} \qquad (4.55)$$

$$P^{S^{max}}_{tib} - P^{S}_{tib} \geq 0 \qquad \forall t, \forall i, \forall b \qquad (4.56)$$

$$\mu^{S^{max}}_{tib} \geq 0 \qquad \forall t, \forall i, \forall b \qquad (4.57)$$

$$P^{S^{max}}_{tib} - P^{S}_{tib} \leq (1 - \omega^{S^{max}}_{tib}) M^{P} \qquad \forall t, \forall i, \forall b \qquad (4.58)$$

$$\mu^{S^{max}}_{tib} \leq \omega^{S^{max}}_{tib} M^{\mu P} \qquad \forall t, \forall i, \forall b \qquad (4.59)$$

$$P_{tjb}^{O^{max}} - P_{tjb}^{R} \geq 0 \qquad \forall t, \forall j, \forall b \qquad (4.60)$$

$$\mu_{tjb}^{O^{max}} \geq 0 \qquad \forall t, \forall j, \forall b \qquad (4.61)$$

$$P_{tjb}^{O^{max}} - P_{tjb}^{R} \leq (1 - \omega_{tjb}^{O^{max}}) M^{P} \qquad \forall t, \forall j, \forall b \qquad (4.62)$$

$$\mu_{tjb}^{O^{max}} \leq \omega_{tjb}^{O^{max}} M^{\mu P} \qquad \forall t, \forall j, \forall b \qquad (4.63)$$

$$P_{tdk}^{D^{max}} - P_{tdk}^{D} \geq 0 \qquad \forall t, \forall d, \forall k \qquad (4.64)$$

$$\mu_{tdk}^{D^{max}} \geq 0 \qquad \forall t, \forall d, \forall k \qquad (4.65)$$

$$P_{tdk}^{D^{max}} - P_{tdk}^{D} \leq (1 - \omega_{tdk}^{D^{max}}) M^{P} \qquad \forall t, \forall d, \forall k \qquad (4.66)$$

$$\mu_{tdk}^{D^{max}} \leq \omega_{tdk}^{D^{max}} M^{\mu P} \qquad \forall t, \forall d, \forall k \qquad (4.67)$$

$$\omega_{tib}^{S^{max}}, \omega_{tjb}^{O^{max}}, \omega_{tdk}^{D^{max}} \in \{0, 1\} \qquad (4.68)$$

CHAPTER 5

UNCERTAINTIES IN THE POWER MARKET MODELS

5.1 Stochastic Programming

So far we have seen the conversion of a bilevel problem to an MPEC and then to a preferred format, which is mixed integer linear programming (MILP). For any particular price path, this solution is an upper bound of achievable profits given perfect information. Under particular market structures it may be possible to come close to achieving this upper bound in day ahead markets using MILPs on very accurate forecasts. However, in general, the real time price for the spot rate of electricity exhibits significant volatility and is therefore difficult, if not impossible, to predict accurately.

This motivates the use of probabilistic models for the derivation of arbitrage value, in particular the use of dynamic programming and stochastic programming. The MILP approach can also be used in a stochastic setting by setting the prices equal to the expected values of the price realizations over time. This approach to the storage problem with uncertainty can be used to quickly find a lower bound on profits, but otherwise it is unsophisticated and performs poorly relative to the alternative techniques [22].

Let's consider the example shown in Fig. 2.1. We incorporate the probabilities for the uncertain parameters and by using the stochastic programming, the formulation becomes:

$$\min \sum_{v} \pi_v(\lambda_S P_{Sv} + \lambda_R P_{Rv} - \lambda_{L1v} P_{L1v} - \lambda_{L2v} P_{L2v} + \mu_{Linev} P_{Linev}^{max} + \mu_{Rv}^{max} P_{Rv}^{max} + \mu_{L1v}^{max} P_{L1v}^{max} + \mu_{L2v}^{max} P_{L2v}^{max}) \tag{5.1}$$

The index υ is the value of the parameter in the scenario υ. The constraints are (2.8) - (2.16) and (4.9) - (4.20) with the uncertain parameters and corresponding decision variable having an additional index of υ.

Since in the stochastic programming we deal with different scenarios, additional constraints are needed to ensure that the resulting offer curves are increasing in price [8]. The following equations that higher productions correspond to higher prices while maintaining the linearity fo the model [8].

$$(1 - x_{\upsilon\upsilon'})M^x \geq \lambda_{1\upsilon} - \lambda_{1\upsilon'} \leq x_{\upsilon\upsilon'}M^x \tag{5.2}$$

$$(1 - y_{\upsilon\upsilon'})M^y \geq P_{S\upsilon} - P_{S\upsilon'} \leq y_{\upsilon\upsilon'}M^y \tag{5.3}$$

$$x_{\upsilon\upsilon'} + y_{\upsilon\upsilon'} = 2z_{\upsilon\upsilon'} \tag{5.4}$$

$$\omega^S_{max}, \omega^R_{max}, \omega^{L1}_{max}, \omega^{L1}_{max}, x_{\upsilon\upsilon'}, y_{\upsilon\upsilon'}, y_{\upsilon\upsilon'} \in \{0, 1\} \tag{5.5}$$

5.2 Almost Robust Optimization

In this section we briefly discuss about the almost robust optimization model, which is one of the effective approaches to address the uncertainty in data for discrete optimization models. It is a trade-off between the objective function value with robustness, to find optimal solutions that are almost robust (feasible under most realizations) [23]. The proposed model is attractive due to its simple structure, its ability to model dependence among uncertain parameters, and its ability to incorporate the decision makers attitude towards risk by controlling the degree of conservatism of the optimal solution [23]. Specifically, the Robustness Index that enables the decision maker to adjust the almost robust optimization to better suit his risk preference [23].

5.3 Chance Constraints

In this section, we consider the uncertainty in λ_R, the guessed marginal cost of a rivalry generator. Two types of chance constraints will be examined. The first chance constraint is as follows. Given the randomness of the marginal cost of Gen R, we would like to obtain the bidding price and the bidding power that can represent more than $1-\epsilon$ chance. This condition can be represented by chance constraints. The representation is not straightforward, though. The second chance constraint is more straightforward. Given the randomness of the marginal cost of Gen R, we would like to obtain the bidding power that will be less than a certain value, say 22 MW with a chance greater than or equal to $1-\epsilon$.

Both types of chance constraints can be modeled in MILP formulation by enumerating scenarios and introducing a binary variable φ_υ [13] . $\varphi_\upsilon = 0$ means that Scenario υ is included in the representing scenarios. $\varphi_\upsilon = 1$ means that this scenario is not a representative scenario that can make the chance constraint be satisfied.

5.3.1 Case I

In Case 1, we use the chance constraint to represent the probability of the solution of the bidding strategy is more than $1-\epsilon$. Since this probability is difficult to be written in a mathematical format, we proceed to explain the MILP formulation.

We define auxiliary variables $P^C_{S\upsilon}$ for $P_{S\upsilon}$, $P^C_{R\upsilon}$ for $P_{R\upsilon}$, $\mu^C_{Line,\upsilon}$ for $\mu_{Line,\upsilon}$ and $\mu^{maxC}_{R\upsilon}$ for $\mu^{max}_{R\upsilon}$ in chance constrained mathematical program (CCMP). The objective function (21) is now replaced by the following objective function.

$$\min \quad \sum_{\upsilon} \pi_\upsilon (\lambda_S P^C_{S\upsilon} + \lambda_R P^C_{R\upsilon} + \mu^C_{Line\upsilon} P^{max}_{Line\upsilon} + \mu^{maxC}_{R\upsilon} P^{max}_{R\upsilon}) \tag{5.6}$$

where π_υ is the probability of scenario υ is happening.

Note that this objective function already considers enumeration of all scenarios due to the uncertainty of λ_R. The uncertainty of λ_R is represented by many scenarios, each with a probability.

The auxiliary variables will be 0 if this particular scenario is not a representing scenario. The auxiliary variables will be the same as the variables if the particular scenario is a representing scenario. This condition can then be expressed by MILP formulations with a binary variable φ_υ introduced.

By using a binary controller variable and "Big-M" coefficient, the chance constrained problem can be formulated with a set of linear constraints as shown in the following where M is a sufficiently large number.

$$\sum_{\upsilon \in \Upsilon} \pi_\upsilon \varphi_\upsilon \leq \epsilon \tag{5.7}$$

$$P^C_{S\upsilon} \geq P_{S\upsilon} - \varphi_\upsilon M \tag{5.8}$$

$$P^C_{R\upsilon} \geq P_{R\upsilon} - \varphi_\upsilon M \tag{5.9}$$

$$\mu^C_{Line\upsilon} \geq \mu_{Line\upsilon} - \varphi_\upsilon M \tag{5.10}$$

$$\mu^{maxC}_{R\upsilon} \geq \mu^{max}_{R\upsilon} - \varphi_\upsilon M \tag{5.11}$$

$$P^C_{S\upsilon}, P^C_{R\upsilon}, \mu^C_{Line\upsilon}, \mu^{maxC}_{R\upsilon} \geq 0 \tag{5.12}$$

$$\varphi_\upsilon \in \{0, 1\} \tag{5.13}$$

(5.7) represents the total probability of the non-representing scenarios (when $\varphi_\upsilon = 1$) should be less than ϵ. (5.8) -(5.11) represent the relationships between the auxiliary variables and the variables. For example, when $\varphi_\upsilon = 1$, or Scenario υ is not a representing one, then $P^C_{S\upsilon}$ should be greater than a large negative number. Since $P_{S\upsilon} \geq 0$, this constraint (5.8) is basically a relaxed one without imposing any constraint. Due to the minimization problem's objective function which includes a term $\lambda_S P^C_{S\upsilon}$, $P^C_{S\upsilon}$ will be 0.

5.3.2 Case II

For the second type of chance constraint, it is easy to write a mathematical expression as:

$$\text{Probability}\left\{ \; P_S(\upsilon) \leq P_S^0 \; \right\} \geq 1 - \epsilon$$

where P_S^0 is a given value.

This constraint can be expressed in MILP formulation by enumeration of scenarios and introducing the binary variable φ_υ.

$$\sum_{\upsilon \in \Upsilon} \pi_\upsilon \varphi_\upsilon \leq \epsilon \tag{5.14}$$

$$P_{S\upsilon} + \varphi_\upsilon M \leq P_S^0 \tag{5.15}$$

$$\varphi_\upsilon \in \{0, 1\} \tag{5.16}$$

For this type of chance constraint, there is no need to introduce auxiliary variables.

Hence, our model for the mixed integer linear programming model with chance constrained mathematical program has its objective function (34) and constraints (8)-(11),(22)-(33) for every scenario υ, and the additional constraints (35)-(41) or (42-44) related to chance constraints.

CHAPTER 6

ILLUSTRATIVE EXAMPLES

The Nonlinear IPOPT problems are solved using SCIP version 3.2.1 [24] and MILP problems are solved using GUROBI 6.5 [25], interfaced with Python 2.7 on an Intel(R) Core(TM) i7-6700 with processors at 3.4 GHz and 16 GB RAM.

6.1 Two-Generator System: Nonlinear Interior Point Method

The parameters given in table 6.1 are used to examine the equations derived in section 3.3. The results are tabulated in table 6.2. For the python code, please refer to the Appendix A.1.

Table 6.1. Parameters for two bus network shown in Fig. 2.1 for ipopt method

λ_S ($/MWh$)	P_S^{max} (MW)	λ_R ($/MWh$)	P_R^{max} (MW)	P_{L1} (MW)	P_{L1} (MW)	P_{Line}^{max} (MW)
12	25	13.25	25	18	30	10

Table 6.2. The solution for the nonlinear formulation using ipopt method

λ_1 ($/MWh$)	P_S (MW)	P_R (MWh)	P_{Line} (MW)
13	25	23	7

6.2 Two-Generator System: Case I

Using the parameters given in Table 6.3, we will solve our model example described in Fig. 2.1, first with CCMP and then without CCMP. The probabilities and the corresponding values for different scenarios are generated using the command uniform in python. Fig. 6.1 presents the seven scenarios with different probabilities. The model is solved for a time horizon of 24 hours,

considering different risk tolerance levels, and the results are tabulated in Table 6.4 and Table 6.5. For python code of the problem, please refer to the Appendix B.1.

Table 6.3. Parameters for two bus network shown in Fig. 2.1

λ_s ($/MWh$)	P_S^{max} (MW)	P_R^{max} (MW)
12	22.8	22.8

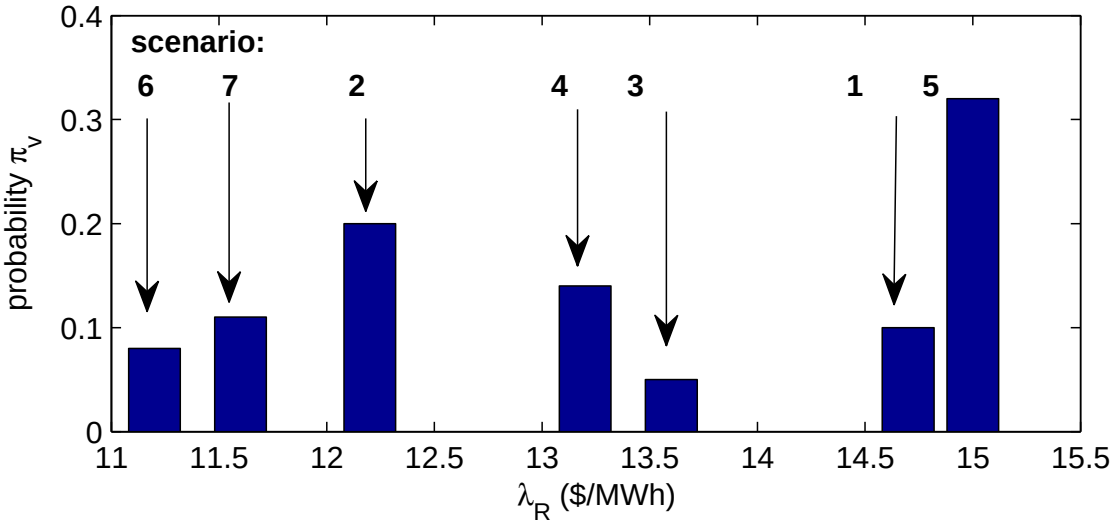

Figure 6.1. Generated λ_R ($/MWh) values for different scenarios.

Table 6.4. Result table for the solution of illustrative example case I

ϵ(%)	0%	8%	20%	30%	50%
profit ($)	7094	7318	7676	7997	8675

Table 6.4 shows that if $\epsilon = 0$, that is, all seven scenarios are included as the representative scenarios, the total profit is $7094. With ϵ increasing, that is, the chance of $1 - \epsilon$ decreasing, the worst scenarios will be excluded, which leads to the increase of the profit.

The scenario selection details are presented in Table 6.5. For example, where $\epsilon = 0$, all scenarios are selected. This problem is same as a stochastic programming problem. When $\epsilon = 0.08$, scenario 6 with $\lambda_R = 11.2$ ($/MWh) is not selected. This exclusion leads to an increase of the rival's marginal cost on average. Therefore, the strategic power producer's benefit will be improved.

Table 6.5. Scenario selection for the 2-bus system with CCMP case 1, 1.0 stands for not selected

ϵ	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	υ_7
0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.05	0.0	0.0	**1.0**	0.0	0.0	0.0	0.0
0.08	0.0	0.0	0.0	0.0	0.0	**1.0**	0.0
0.20	0.0	**1.0**	0.0	0.0	0.0	0.0	0.0
0.30	**1.0**	**1.0**	0.0	0.0	0.0	0.0	0.0
0.50	**1.0**	0.0	0.0	0.0	**1.0**	**1.0**	0.0

6.3 Two-Generator System: Case II

Using the parameters given in Table 6.6, we will solve our model example described in Fig. 2.1. The probabilities and the corresponding values for different scenarios are used from Fig. 6.1. The model is solved for a time horizon of 24 hours, considering different risk tolerance levels, and the results are tabulated in Table 6.7. Increasing ϵ means relaxing the constraint of $P_S \leq P_S^0$. Therefore, the profit increases with ϵ increasing. For python code of the problem, please refer to the Appendix B.2.

Table 6.6. Parameters for two bus network shown in Fig. 2.1

λ_s ($/MWh$)	P_S^{max} (MW)	P_R^{max} (MW)	P_0^S (MW)
12	40	40	30

Table 6.7. Result table for the solution of illustrative example case II

ϵ(%)	0%	8%	20%	30%	50%
profit ($)	7348	7571	7929	8277	8950

Table 6.8 presents the computed P_{Sv} values and Table 6.9 presents the selection of the representative scenarios. Table 6.9 can be compared with Table 6.8. When $P_{Sv} > 30$, this scenario will not be selected.

Table 6.8. P_{Sv} in every scenario for case II

ϵ	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	υ_7
0.00	30.0	30.0	30.0	30.0	30.0	25.0	25.0
0.08	30.0	30.0	30.0	30.0	30.0	25.0	25.0
0.20	**35.0**	30.0	30.0	30.0	30.0	30.0	30.0
0.30	**35.0**	**35.0**	30.0	30.0	30.0	30.0	30.0
0.50	**35.0**	**35.0**	**35.0**	**35.0**	30.0	30.0	30.0

Table 6.9. Scenario selection for the 2-bus system with CCMP case 2, 1.0 stands for not selected

ϵ	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	υ_7
0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.08	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.20	**1.0**	0.0	0.0	0.0	0.0	0.0	0.0
0.30	**1.0**	**1.0**	0.0	0.0	0.0	0.0	0.0
0.50	**1.0**	**1.0**	**1.0**	**1.0**	0.0	0.0	0.0

6.4 5-bus System 24-hour Example

This section presents the implementation of the model to network based on the fig. 4.1 and the section 4.3. The parameters are based on the values given in [8]. For python code of the problem and the output of it please refer to the Appendix B.3.

6.5 Six-bus System 24-hour Example

This section presents the implementation of the model to network based on the description given in [8]. The model including the CCMP type 1 chance constraints is applied to the network, considering the extensions such as, dc network model, unit ramp up and ramp down limits, different consumer bids in different hours of the day and susceptance of the transmission lines. We consider seven scenarios with different probabilities. The scenarios differ on rival producers offer λ_{Rv}, and on consumer bids λ_{Liv} for all i related to load buses. We can generate a set of scenarios by multiplying the above two terms given in [2] by the entries of vector [1.15, 1.1, 1.092, 1, 1.05, 0.89, 0.69]. The corresponding probabilities are assumed as [0.1, 0.2, 0.3, 0.1, 0.1, 0.1, 0.1]. Table 6.10 shows how

different risk factors ϵ affects the selection of the scenarios during the solution of the optimization CCMP model for the six-bus system.

Table 6.10. Scenario selection for the six-bus system, 1.0 stands for not selected

ϵ	υ_1	υ_2	υ_3	υ_4	υ_5	υ_6	υ_7
0.00	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.10	0.0	0.0	0.0	**1.0**	0.0	0.0	0.0
0.50	0.0	**1.0**	0.0	**1.0**	0.0	**1.0**	0.0

The sum of the profits for different hours corresponding to different scenarios is plotted in Fig. 6.2. It can be seen that some scenarios have been relaxed depending on the risk factor ϵ, and thereby increasing the corresponding profit to that scenario.

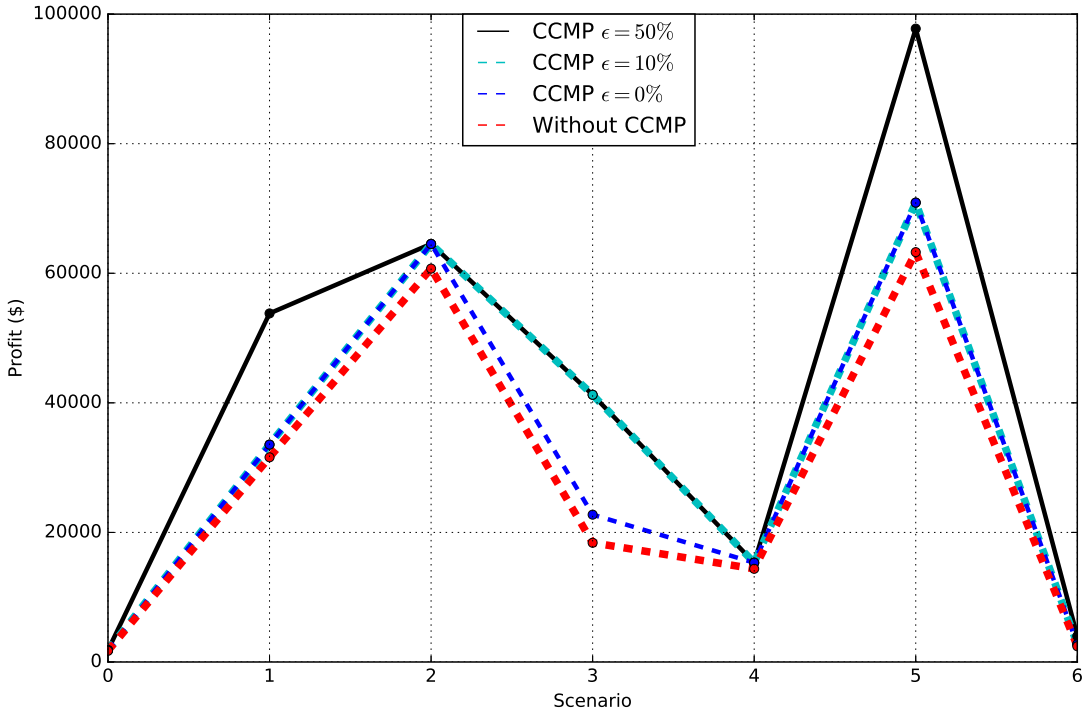

Figure 6.2. A comparison of profit contribution versus scenario for different risk tolerance levels

The sum of the profits for different hours and sum of the total generation with risk factor and without risk factor corresponding to percentage of risk factors are plotted in Fig. 6.3. This figure

is used to provide a visualized comparison of how increasing the risk factor can increase the market power, and hence more profit corresponding to less generation or same generation.

Figure 6.3. A plot of profit and generation vs different risk tolerance levels

CHAPTER 7

CONCLUSION

The thesis presents a mixed integer linear programming (MILP) formulation for a bi-level mathematical program with equilibrium constraints (MPEC) considering chance constraints.

The particular MPEC problem relates to a power producer's bidding strategy: maximize its total benefit through determining bidding price and bidding power output while considering an electricity pool's operation and guessing the rival producer's bidding price.

The entire decision-making process can be described by a bi-level optimization problem. The contribution of my thesis is the MILP with chance constraints formulation of this problem. First, the lower-level poor operation problem is replaced by Karush-Kuhn-Tucker (KKT) optimality condition, which is further converted to an MILP formulation except a bilinear item in the objective function. Secondly, duality theory is implemented to replace the bilinear item by linear items. Finally, two types of chance constraints are examined and modeled in MILP formulation. With the MILP formulation, the entire MPEC problem considering randomness in price guessing can be solved using off-shelf MIP solvers, e.g., Gurobi. An example is given to illustrate the formulation and show the case study results.

For the future, the implementation of almost robust optimization technique on MPEC problems can be further studied, which may handle the uncertainties more efficiently if a proper penalty value could be calculated.

REFERENCES

[1] M. Fampa, L. A. Barroso, D. Candal, and L. Simonetti, "Bilevel optimization applied to strategic pricing in competitive electricity markets," *Computational Optimization and Applications*, vol. 39, no. 2, pp. 121–142, 2008. [Online]. Available: http://dx.doi.org/10.1007/s10589-007-9066-4

[2] R. Rajaraman, C. Associates, F. Alvarado, V. President, and L. R. C. Associates, "Optimal bidding strategy in electricity markets under uncertain energy and reserve prices optimal bidding strategy in electricity markets under uncertain energy and reserve prices," 2003.

[3] X.-P. Zhang, *Restructured electric power systems: analysis of electricity markets with equilibrium models*. John Wiley & Sons, 2010, vol. 71.

[4] B. F. Hobbs, C. B. Metzler, and J.-S. Pang, "Strategic gaming analysis for electric power systems: An mpec approach," *IEEE transactions on power systems*, vol. 15, no. 2, pp. 638–645, 2000.

[5] L. A. Barroso, R. D. Carneiro, S. Granville, M. V. Pereira, and M. Fampa, "Nash equilibrium in strategic bidding: a binary expansion approach," *IEEE TRANSACTIONS ON POWER SYSTEMS PWRS*, vol. 21, no. 2, p. 629, 2006.

[6] T. Dai and W. Qiao, "Optimal bidding strategy of a strategic wind power producer in the short-term market," *IEEE Transactions on Sustainable Energy*, vol. 6, no. 3, pp. 707–719, 2015.

[7] S. J. Kazempour and H. Zareipour, "Equilibria in an oligopolistic market with wind power production," *IEEE Transactions on Power Systems*, vol. 29, no. 2, pp. 686–697, 2014.

[8] C. Ruiz and A. J. Conejo, "Pool strategy of a producer with endogenous formation of locational marginal prices," *IEEE Transactions on Power Systems*, vol. 24, no. 4, pp. 1855–1866, 2009.

[9] W. Xian, L. Yuzeng, and Z. Shaohua, "Oligopolistic equilibrium analysis for electricity markets: a nonlinear complementarity approach," *IEEE Transactions on Power Systems*, vol. 19, no. 3, pp. 1348–1355, 2004.

[10] S. Petoussis, X. Zhang, and K. Godfrey, "Electricity market equilibrium analysis based on nonlinear interior point algorithm with complementarity constraints," *IET Generation, Transmission & Distribution*, vol. 1, no. 4, pp. 603–612, 2007.

[11] S. Borenstein, "Understanding competitive pricing and market power in wholesale electricity markets," *The Electricity Journal*, vol. 13, no. 6, pp. 49–57, 2000. [Online]. Available: http://EconPapers.repec.org/RePEc:eee:jelect:v:13:y:2000:i:6:p:49-57

[12] Q. Wang, Y. Guan, and J. Wang, "A chance-constrained two-stage stochastic program for unit commitment with uncertain wind power output," *IEEE Transactions on Power Systems*, vol. 27, no. 1, pp. 206–215, 2012.

[13] B. Zeng, Y. An, and L. Kuznia, "Chance constrained mixed integer program: Bilinear and linear formulations, and benders decomposition," *arXiv preprint arXiv:1403.7875*, 2014.

[14] A. U. Raghunathan and L. T. Biegler, "Mathematical programs with equilibrium constraints (mpecs) in process engineering," *Computers and Chemical Engineering*, vol. 27, no. 10, pp. 1381 – 1392, 2003. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0098135403000929

[15] R. Fletcher, S. Leyffer, D. Ralph, and S. Scholtes, "Local convergence of sqp methods for mathematical programs with equilibrium constraints," *SIAM Journal on Optimization*, vol. 17, no. 1, pp. 259–286, 2006. [Online]. Available: http://dx.doi.org/10.1137/S1052623402407382

[16] T. C. Price, "Using co-evolutionary programming to simulate strategic behaviour in markets," *Journal of Evolutionary Economics*, vol. 7, no. 3, pp. 219–254, 1997. [Online]. Available: http://dx.doi.org/10.1007/s001910050042

[17] S. Hejazi, A. Memariani, G. Jahanshahloo, and M. Sepehri, "Linear bilevel programming solution by genetic algorithm," *Computers and Operations Research*, vol. 29, no. 13, pp. 1913 – 1925, 2002. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0305054801000661

[18] H. I. Calvete, C. Gal, and P. M. Mateo, "A new approach for solving linear bilevel problems using genetic algorithms," *European Journal of Operational Research*, vol. 188, no. 1, pp. 14 – 28, 2008. [Online]. Available: //www.sciencedirect.com/science/article/pii/S0377221707003773

[19] H. Chen, K. P. Wong, C. Y. Chung, and D. H. M. Nguyen, "A coevolutionary approach to analyzing supply function equilibrium model," *IEEE Transactions on Power Systems*, vol. 21, no. 3, pp. 1019–1028, Aug 2006.

[20] A. U. Raghunathan and L. T. Biegler, "An interior point method for mathematical programs with complementarity constraints (mpccs)," *SIAM Journal on Optimization*, vol. 15, no. 3, pp. 720–750, 2005. [Online]. Available: http://dx.doi.org/10.1137/S1052623403429081

[21] S. Boyd and L. Vandenberghe, *Convex Optimization*. New York, NY, USA: Cambridge University Press, 2004.

[22] P. Mokrian and M. Stephen, "M.: A stochastic programming framework for the valuation of electricity storage," in *In: 26th USAEE/IAEE North American Conference. International Association for Energy Economics, Ann Arbor, MI (2006.*

[23] O. Baron, O. Berman, and M. M. Fazel-zarandi, "Almost robust optimization with binary variables."

[24] T. Achterberg, "Scip: solving constraint integer programs," *Mathematical Programming Computation*, vol. 1, no. 1, pp. 1–41, 2009.

[25] G. Optimization, "Gurobi optimizer reference manual," 2014.

APPENDIX A: PYTHON CODES FOR NONLINEAR FORMULATION

A.1 Python Code for the 2-bus System Using Ipopt Method

A.1.1 Code

```
from pyscipopt import *
t= range(1,2)
b= range(1,3)
PLmax=10
P_D=[18, 30]
Lam_D=18
Lam_O=13.25
P_Smax=25
Lam_S=12
P_Omax=25
model = Model()
A_S,P_S,P_O,Ltn,mu_Smax,mu_Omax,PL,mu_L,om_L,om_Smax,om_Omax,om_Smin,mu_Smin,
    om_Omin,mu_Omin,om_Lmin,mu_Lmin = [{} for i in range(17)]
for T in t:
    A_S[T] = model.addVar('A_S',vtype='C')
    P_S[T] = model.addVar('P_S',vtype='C')
    mu_Smax[T] = model.addVar('mu_Smax',vtype='C')
    om_Smax[T] = model.addVar('om_Smax',vtype='B')
    P_O[T] = model.addVar('P_O',vtype='C')
    mu_Omax[T] = model.addVar('mu_Omax',vtype='C')
    om_Omax[T] = model.addVar('om_Omax',vtype='B')
    PL[T] = model.addVar('PL',vtype='C')
    mu_L[T] = model.addVar('mu_L',vtype='C')
    om_L[T] = model.addVar('om_L',vtype='B')
    mu_Lmin[T] = model.addVar('mu_Lmin',vtype='C')
    om_Lmin[T] = model.addVar('om_Lmin',vtype='B')
    mu_Omin[T] = model.addVar('mu_Omin',vtype='C')
    om_Omin[T] = model.addVar('om_Omin',vtype='B')
    mu_Smin[T] = model.addVar('mu_Smin',vtype='C')
    om_Smin[T] = model.addVar('om_Smin',vtype='B')
for T in t:
    for N in b:
        Ltn[T, N] = model.addVar('Ltn',vtype='C', lb=0)
for T in t:
    model.addCons(P_S[T]-P_D[0] == PL[T])
    model.addCons(P_O[T]-P_D[1] == -PL[T])
    model.addCons((PLmax-PL[T])<=om_L[T]*PLmax)
    model.addCons(mu_L[T] <=(1-om_L[T])*PLmax)
    model.addCons(PLmax-PL[T] >= 0)
    model.addCons(PL[T]+PLmax >= 0)
    model.addCons(A_S[T]-Ltn[T, 1]+mu_Smax[T] == 0)
    model.addCons((P_Smax-P_S[T]) >= 0)
    model.addCons(P_Smax-P_S[T] <= (1-om_Smax[T])*P_Smax)
    model.addCons(mu_Smax[T] <= om_Smax[T]*P_Smax)
    model.addCons(Lam_O-Ltn[T, 2]+mu_Omax[T] == 0)
```

```
45      model.addCons(P_Omax-P_O[T] <= (1-om_Omax[T])*P_Omax)
46      model.addCons(mu_Omax[T] <= om_Omax[T]*P_Omax)
47      model.addCons(P_Omax-P_O[T] >= 0)
48 obj = quicksum(Lam_S*P_S[T]-Ltn[T,1]*P_S[T] for T in t)
49 model.setObjective(obj, "MINIMIZE")
50 model.optimize()
```

A.1.2 Output

```
 1 presolving:
 2 (round 1, fast)       14 del vars, 12 del conss, 0 add conss, 8 chg bounds, 2
       chg sides, 2 chg coeffs, 0 upgd conss, 1 impls, 1 clqs
 3 presolving (2 rounds: 2 fast, 1 medium, 1 exhaustive):
 4  18 deleted vars, 14 deleted constraints, 0 added constraints, 8 tightened
       bounds, 0 added holes, 2 changed sides, 2 changed coefficients
 5  1 implications, 0 cliques
 6 presolved problem has 0 variables (0 bin, 0 int, 0 impl, 0 cont) and 0
       constraints
 7 transformed objective value is always integral (scale: 1)
 8 Presolving Time: 0.00
 9
10  time | node  | left  |LP iter|LP it/n| mem |mdpt |frac |vars |cons |cols |rows
       |cuts |confs|strbr|  dualbound   | primalbound  |  gap
11 t 0.0s|     1 |     0 |     0 |     - | 224k|   0 |   - |   0 |   0 |   0 |   0
       |   0 |   0 |   0 |      --      |-2.500000e+01 |     Inf
12   0.0s|     1 |     0 |     0 |     - | 223k|   0 |   - |   0 |   0 |   0 |   0
       |   0 |   0 |   0 |-2.500000e+01 |-2.500000e+01 |   0.00%
13
14 SCIP Status        : problem is solved [optimal solution found]
15 Solving Time (sec) : 0.00
16 Solving Nodes      : 1
17 Primal Bound       : -2.50000000000000e+01 (1 solutions)
18 Dual Bound         : -2.50000000000000e+01
19 Gap                : 0.00 %
```

APPENDIX B: PYTHON CODES FOR MILP

B.1 Python Code for the 2-bus System Using CCMP Case 1

B.1.1 Code

```
from gurobipy import *
import numpy as np
import math as ma
import matplotlib
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import random as ran
Lam_D=[[19.802116822458622, 19.944456674560165, 19.813855317207416,
    17.823159917070598, 19.268975905242996, 18.85125499680762,
    17.925725187657715],
        [19.54706548020172, 17.697666942797742, 17.559463082394245,
    18.423892092263987, 18.845296754034834, 18.621628541265412,
    18.743515087182733]]
Lam_O=[14.747034109908812, 12.235933384118333, 13.558811420161696,
    13.247795352289096, 14.998683190872775, 11.246644280930704,
    11.643163301063465]
t= range(1,25)
i=range(1,3)
pik=[0.1,0.2,0.05,0.14,0.32,0.08,0.11]
w=range(1,8)
P_mline=10
M=10
ep=0.5
P_Dmax=[22, 23]
P_Smax=22.8
Lam_S=12
P_Rmax=22.8
model = Model()
G_S,P_S,P_SC,P_R,P_RC,P_D,P_DC,Lam_tn,Lam_tnC,mu_Smin,mu_Smax,mu_Rmin,mu_Rmax,
    mu_RmaxC,mu_Dmin,mu_Dmax=[{} for x in range(16)]
mu_DmaxC,om_Rmin,om_Smax,om_Smin,om_Dmax,om_Dmin,om_Rmax,X_S,Y_S,Z_SS,Z_S,Pline,
    om_line,mu_line,mu_lineC=[{} for x in range(15)]
for W in w:
    for T in t:
        G_S[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="G_S_{}_{}".
    format(W, T))
        P_S[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_S_{}_{}".
    format(W, T))
        P_SC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_SC_{}_{}".
    format(W, T))
        mu_Smin[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Smin_
    {}_{}".format(W, T))
        mu_Smax[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Smax_
    {}_{}".format(W, T))
```

```
32          om_Smin[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Smin_{}_{}
    ".format(W, T))
33          om_Smax[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Smax_{}_{}
    ".format(W, T))
34      Z_S[W] = model.addVar(vtype=GRB.BINARY, name="Z_S_{}".format(W))
35  for W in w:
36      for W1 in w:
37          for T in t:
38              X_S[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="X_S_{}_{}
    _{}".format(W, W1, T))
39              Y_S[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="Y_S_{}_{}
    _{}".format(W, W1, T))
40              Z_SS[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="Z_S_{}_
    {}_{}".format(W, W1, T))
41  for W in w:
42      for T in t:
43          P_R[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_R_{}_{}".
    format(W, T))
44          P_RC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_RC_{}_{}".
    format(W, T))
45          mu_Rmin[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Rmin_
    {}_{}".format(W, T))
46          mu_Rmax[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Rmax_
    {}_{}".format(W, T))
47          mu_RmaxC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_RmaxC_{}_{}".format(W, T))
48          om_Rmin[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Rmin_{}_{}
    ".format(W, T))
49          om_Rmax[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Rmax_{}_{}
    ".format(W, T))
50          Pline[W, T] = model.addVar(vtype=GRB.CONTINUOUS, name="Pline_{}_{}".
    format(W, T))
51          om_line[W, T] = model.addVar(vtype=GRB.BINARY, name="om_line_{}_{}".
    format(W, T))
52          mu_line[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_line_
    {}_{}".format(W, T))
53          mu_lineC[W, T]= model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_lineC_
    {}_{}".format(W, T))
54  for W in w:
55      for T in t:
56          for D in i:
57              P_D[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_D_{}
    _{}_{}".format(W, T, D))
58              P_DC[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_DC_
    {}_{}_{}".format(W, T, D))
59              mu_Dmin[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_Dmin_{}_{}_{}".format(W, T, D))
60              mu_Dmax[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_Dmax_{}_{}_{}".format(W, T, D))
61              mu_DmaxC[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_DmaxC_{}_{}_{}".format(W, T, D))
62              om_Dmin[W, T, D] = model.addVar(vtype=GRB.BINARY, lb=0, name="
    om_Dmin_{}_{}_{}".format(W, T, D))
63              om_Dmax[W, T, D] = model.addVar(vtype=GRB.BINARY, lb=0, name="
    om_Dmax_{}_{}_{}".format(W, T, D))
64  for W in w:
65      for T in t:
66          for N in i:
```

```
67                Lam_tn[W, T, N] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      Lam_tn_{}_{}_{}".format(W, T, N))
68                Lam_tnC[W, T, N] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      Lam_tnC_{}_{}_{}".format(W, T, N))
69 model.update()
70 for W in w:
71      for T in t:
72          model.addConstr(P_S[W, T]-P_D[W, T, 1] == Pline[W, T])
73          model.addConstr(P_R[W, T]-P_D[W, T, 2] == -Pline[W, T])
74          model.addConstr(P_mline-Pline[W, T] <= om_line[W,T]*2*P_mline)
75          model.addConstr(mu_line[W,T] <= (1-om_line[W,T])*2*P_mline)
76 for W in w:
77      for T in t:
78          model.addConstr(G_S[W, T]-Lam_tn[W, T, 1]-mu_Smin[W, T]+mu_Smax[W, T] ==
       0)
79          model.addConstr(P_S[W, T] <= (1-om_Smin[W, T])*P_Smax)
80          model.addConstr(mu_Smin[W, T] <= om_Smin[W, T]*P_Smax)
81          model.addConstr(P_Smax-P_S[W, T] >= 0)
82          model.addConstr(P_Smax-P_S[W, T] <= (1-om_Smax[W, T])*P_Smax)
83          model.addConstr(mu_Smax[W, T] <= om_Smax[W, T]*P_Smax)
84 for W in w:
85      for T in t:
86          model.addConstr(Lam_R[W-1]-Lam_tn[W, T, 2]-mu_Rmin[W, T]+mu_Rmax[W, T]
     == 0)
87          model.addConstr(P_R[W, T] <= (1-om_Rmin[W, T])*P_Rmax)
88          model.addConstr(mu_Rmin[W, T] <= om_Rmin[W, T]*P_Rmax)
89          model.addConstr(P_Rmax-P_R[W, T] >= 0)
90          model.addConstr(P_Rmax-P_R[W, T] <= (1-om_Rmax[W, T])*P_Rmax)
91          model.addConstr(mu_Rmax[W, T] <= om_Rmax[W, T]*P_Rmax)
92 for W in w:
93      for T in t:
94          for D in i:
95              model.addConstr(Lam_tn[W, T, D]-Lam_D[D-1][W-1]-mu_Dmin[W, T, D]+
     mu_Dmax[W, T, D] == 0)
96              model.addConstr(P_D[W, T, D] <= (1-om_Dmin[W, T, D])*P_Dmax[D-1])
97              model.addConstr(mu_Dmin[W, T, D] <= om_Dmin[W, T, D]*P_Dmax[D-1])
98              model.addConstr(P_Dmax[D-1]-P_D[W, T, D] >= 0)
99              model.addConstr(P_Dmax[D-1]-P_D[W, T, D] <= (1-om_Dmax[W, T, D])*
     P_Dmax[D-1])
100             model.addConstr(mu_Dmax[W, T, D] <= om_Dmax[W, T, D]*P_Dmax[D-1])
101 for W in w:
102     for T in t:
103         model.addConstr(P_S[W, T] + P_R[W, T] - quicksum(P_D[W,T,D] for D in i)
     ==0)
104 for W in w:
105     for W1 in w:
106         if W1>W:
107             for T in t:
108                 model.addConstr(Lam_tn[W, T, 1]-Lam_tn[W1, T, 1]<=X_S[W, W1, T]*
     M)
109                 model.addConstr(Lam_tn[W, T, 1]-Lam_tn[W1, T, 1]>=(1-X_S[W, W1,
     T])*M)
110                 model.addConstr(P_S[W, T]-P_S[W1, T] <=Y_S[W, W1, T]*M)
111                 model.addConstr(P_S[W, T]-P_S[W1, T]>=(1-Y_S[W, W1, T])*M)
112                 model.addConstr(X_S[W, W1, T]+Y_S[W, W1, T] == 2*Z_SS[W, W1, T])
113 model.addConstr(quicksum(pik[W-1]*Z_S[W] for W in w) <= ep, "chance Constraint")
114 for W in w:
115     for T in t:
116         model.addConstr(Lam_tnC[W, T, 1]>=Lam_tn[W, T, 1]-Z_S[W]*M)
```

```
for W in w:
    for T in t:
        for I in i:
            model.addConstr(P_DC[W, T, I] <=P_D[W, T, I]-Z_S[W]*M)
            model.addConstr(mu_DmaxC[W, T, I] >=mu_Dmax[W, T, I]-Z_S[W]*M)
        model.addConstr(P_SC[W, T]>=P_S[W, T]-Z_S[W]*M)
        model.addConstr(P_RC[W, T]>=P_R[W, T]-Z_S[W]*M)
        model.addConstr(mu_RmaxC[W, T]>=mu_Rmax[W, T]-Z_S[W]*M)
        model.addConstr(mu_lineC[W, T]>=mu_line[W, T]-Z_S[W]*M)
obj = (quicksum(Lam_S*P_SC[W, T]*pik[W-1] for T in t for W in w)
       + quicksum(Lam_O[W-1]*P_RC[W, T]*pik[W-1] for T in t for W in w)
       - quicksum(Lam_D[D-1][W-1]*P_DC[W, T, D]*pik[W-1] for D in i for T in t
    for W in w)
       + quicksum(P_Rmax*mu_RmaxC[W, T]*pik[W-1] for T in t for W in w)
       + quicksum(P_Dmax[D-1]*mu_DmaxC[W, T, D]*pik[W-1] for D in i for T in t
    for W in w)
       + quicksum(pik[W-1]*mu_lineC[W, T]*P_mline for T in t for W in w))
model.setObjective(obj, GRB.MINIMIZE)
model.optimize()
```

B.1.2 Output

```
Optimize a model with 7561 rows, 9583 columns and 18487 nonzeros
Coefficient statistics:
  Matrix range     [5e-02, 4e+01]
  Objective range  [5e-01, 1e+01]
  Bounds range     [1e+00, 1e+00]
  RHS range        [5e-01, 4e+01]
Found heuristic solution: objective -3637.6
Presolve removed 4608 rows and 8400 columns
Presolve time: 0.03s
Presolved: 2953 rows, 1183 columns, 6487 nonzeros
Variable types: 840 continuous, 343 integer (343 binary)

Root relaxation: objective -6.667364e+03, 552 iterations, 0.01 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    -6667.364361 -6667.3644  0.00%     -    0s

Explored 0 nodes (552 simplex iterations) in 0.05 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective -6.667364361278e+03, best bound -6.667364361278e+03, gap 0.0%
```

B.2 Python Code for the 2-bus System Using CCMP Case 2

B.2.1 Code

```
from gurobipy import *
Lam_R=[14.747034109908812, 12.235933384118333, 13.558811420161696,
    13.247795352289096, 14.998683190872775, 11.246644280930704,
    11.643163301063465]
Lam_D=[[19.802116822458622, 19.944456674560165, 19.813855317207416,
    17.823159917070598, 19.268975905242996, 18.85125499680762,
    17.925725187657715],
```

```
4           [19.54706548020172, 17.697666942797742, 17.559463082394245,
      18.423892092263987, 18.845296754034834, 18.621628541265412,
      18.743515087182733]]
5 t= range(1,25)
6 i=range(1,3)
7 pik=[0.1,0.2,0.05,0.14,0.32,0.08,0.11]
8 w=range(1,8)
9 P_mline=10
10 M=5
11 ep=0.50
12 P_Dmax=[22, 23]
13 P_Smax=40
14 Lam_S=12
15 P_Rmax=40
16 model = Model()
17 G_S,P_S,P_SC,P_R,P_RC,P_D,P_DC,Lam_tn,Lam_tnC,mu_Smin,mu_Smax,mu_Rmin,mu_Rmax,
      mu_RmaxC,mu_Dmin,mu_Dmax=[{} for x in range(16)]
18 mu_DmaxC,om_Rmin,om_Smax,om_Smin,om_Dmax,om_Dmin,om_Rmax,X_S,Y_S,Z_SS,Z_S,Pline,
      om_line,mu_line,mu_lineC=[{} for x in range(15)]
19 for W in w:
20      for T in t:
21          G_S[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="G_S_{}_{}".
      format(W, T))
22          P_S[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_S_{}_{}".
      format(W, T))
23          P_SC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_SC_{}_{}".
      format(W, T))
24          mu_Smin[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Smin_
      {}_{}".format(W, T))
25          mu_Smax[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Smax_
      {}_{}".format(W, T))
26          om_Smin[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Smin_{}_{}
      ".format(W, T))
27          om_Smax[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Smax_{}_{}
      ".format(W, T))
28      Z_S[W] = model.addVar(vtype=GRB.BINARY, name="Z_S_{}".format(W))
29 for W in w:
30      for W1 in w:
31          for T in t:
32              X_S[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="X_S_{}_{}
      _{}".format(W, W1, T))
33              Y_S[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="Y_S_{}_{}
      _{}".format(W, W1, T))
34              Z_SS[W, W1, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="Z_S_{}_
      {}_{}".format(W, W1, T))
35 for W in w:
36      for T in t:
37          P_R[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_R_{}_{}".
      format(W, T))
38          P_RC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_RC_{}_{}".
      format(W, T))
39          mu_Rmin[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Rmin_
      {}_{}".format(W, T))
40          mu_Rmax[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_Rmax_
      {}_{}".format(W, T))
41          mu_RmaxC[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_RmaxC_{}_{}".format(W, T))
42          om_Rmin[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Rmin_{}_{}
      ".format(W, T))
```

```
43          om_Rmax[W, T] = model.addVar(vtype=GRB.BINARY, lb=0, name="om_Rmax_{}_{}
    ".format(W, T))
44          Pline[W, T] = model.addVar(vtype=GRB.CONTINUOUS, name="Pline_{}_{}".
    format(W, T))
45          om_line[W, T] = model.addVar(vtype=GRB.BINARY, name="om_line_{}_{}".
    format(W, T))
46          mu_line[W, T] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_line_
    {}_{}".format(W, T))
47          mu_lineC[W, T]= model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="mu_lineC_
    {}_{}".format(W, T))
48 for W in w:
49      for T in t:
50          for D in i:
51              P_D[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_D_{}
    _{}_{}".format(W, T, D))
52              P_DC[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_DC_
    {}_{}_{}".format(W, T, D))
53              mu_Dmin[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_Dmin_{}_{}_{}".format(W, T, D))
54              mu_Dmax[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_Dmax_{}_{}_{}".format(W, T, D))
55              mu_DmaxC[W, T, D] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    mu_DmaxC_{}_{}_{}".format(W, T, D))
56              om_Dmin[W, T, D] = model.addVar(vtype=GRB.BINARY, lb=0, name="
    om_Dmin_{}_{}_{}".format(W, T, D))
57              om_Dmax[W, T, D] = model.addVar(vtype=GRB.BINARY, lb=0, name="
    om_Dmax_{}_{}_{}".format(W, T, D))
58 for W in w:
59      for T in t:
60          for N in i:
61              Lam_tn[W, T, N] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    Lam_tn_{}_{}_{}".format(W, T, N))
62              Lam_tnC[W, T, N] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
    Lam_tnC_{}_{}_{}".format(W, T, N))
63 model.update()
64 for W in w:
65      for T in t:
66          model.addConstr(P_S[W, T]-P_D[W, T, 1] == Pline[W, T])
67          model.addConstr(P_R[W, T]-P_D[W, T, 2] == -Pline[W, T])
68          model.addConstr(P_mline-Pline[W, T] <= om_line[W,T]*2*P_mline)
69          model.addConstr(mu_line[W,T] <= (1-om_line[W,T])*2*P_mline)
70 for W in w:
71      for T in t:
72          model.addConstr(G_S[W, T]-Lam_tn[W, T, 1]-mu_Smin[W, T]+mu_Smax[W, T] ==
     0)
73          model.addConstr(P_S[W, T] <= (1-om_Smin[W, T])*P_Smax)
74          model.addConstr(mu_Smin[W, T] <= om_Smin[W, T]*P_Smax)
75          model.addConstr(P_Smax-P_S[W, T] >= 0)
76          model.addConstr(P_Smax-P_S[W, T] <= (1-om_Smax[W, T])*P_Smax)
77          model.addConstr(mu_Smax[W, T] <= om_Smax[W, T]*P_Smax)
78 for W in w:
79      for T in t:
80          model.addConstr(Lam_R[W-1]-Lam_tn[W, T, 2]-mu_Rmin[W, T]+mu_Rmax[W, T]
    == 0)
81          model.addConstr(P_R[W, T] <= (1-om_Rmin[W, T])*P_Rmax)
82          model.addConstr(mu_Rmin[W, T] <= om_Rmin[W, T]*P_Rmax)
83          model.addConstr(P_Rmax-P_R[W, T] >= 0)
84          model.addConstr(P_Rmax-P_R[W, T] <= (1-om_Rmax[W, T])*P_Rmax)
85          model.addConstr(mu_Rmax[W, T] <= om_Rmax[W, T]*P_Rmax)
```

```
for W in w:
    for T in t:
        for D in i:
            model.addConstr(Lam_tn[W, T, D]-Lam_D[D-1][W-1]-mu_Dmin[W, T, D]+
    mu_Dmax[W, T, D] == 0)
            model.addConstr(P_D[W, T, D] <= (1-om_Dmin[W, T, D])*P_Dmax[D-1])
            model.addConstr(mu_Dmin[W, T, D] <= om_Dmin[W, T, D]*P_Dmax[D-1])
            model.addConstr(P_Dmax[D-1]-P_D[W, T, D] >= 0)
            model.addConstr(P_Dmax[D-1]-P_D[W, T, D] <= (1-om_Dmax[W, T, D])*
    P_Dmax[D-1])
            model.addConstr(mu_Dmax[W, T, D] <= om_Dmax[W, T, D]*P_Dmax[D-1])
for W in w:
    for T in t:
        model.addConstr(P_S[W, T] + P_R[W, T] - quicksum(P_D[W,T,D] for D in i)
    ==0)
for W in w:
    for W1 in w:
        if W1>W:
            for T in t:
                model.addConstr(Lam_tn[W, T, 1]-Lam_tn[W1, T, 1]<=X_S[W, W1, T]*
    M)
                model.addConstr(Lam_tn[W, T, 1]-Lam_tn[W1, T, 1]>=(1-X_S[W, W1,
    T])*M)
                model.addConstr(P_S[W, T]-P_S[W1, T] <=Y_S[W, W1, T]*M)
                model.addConstr(P_S[W, T]-P_S[W1, T]>=(1-Y_S[W, W1, T])*M)
                model.addConstr(X_S[W, W1, T]+Y_S[W, W1, T] == 2*Z_SS[W, W1, T])
model.addConstr(quicksum(pik[W-1]*Z_S[W] for W in w) <= ep, "chance Constraint")
for W in w:
    for T in t:
        model.addConstr(P_S[W, T]<=23.0+Z_S[W]*M)
obj = (quicksum(Lam_S*P_S[W, T]*pik[W-1] for T in t for W in w)
      + quicksum(Lam_R[W-1]*P_R[W, T]*pik[W-1] for T in t for W in w)
      - quicksum(Lam_D[D-1][W-1]*P_D[W, T, D]*pik[W-1] for D in i for T in t
    for W in w)
      + quicksum(P_Rmax*mu_Rmax[W, T]*pik[W-1] for T in t for W in w)
      + quicksum(pik[W-1]*mu_line[W, T]*P_mline for T in t for W in w))
model.setObjective(obj, GRB.MINIMIZE)
model.optimize()
```

B.2.2 Output

```
rectangle(1.873     |2.129  0.316  0.316)
rectangle(4.402     |2.129  0.316  0.316)
rectangle(3.23      |4.177  0.316  0.316)
rectangle(2.914     |6.524  0.316  0.316)
rectangle(4.521     |9.116  0.316  0.316)
rectangle(1.848     |4.202  0.316  0.316)
rectangle(4.903     |4.502  0.316  0.316)
rectangle(4.402     |2.125  0.316  0.316)
rectangle(4.402     |2.125  0.316  0.316)
rectangle(4.402     |2.125  0.316  0.316)
rectangle(4.402     |2.125  0.316  0.316)
rectangle(4.402     |2.125  0.316  0.316)
rectangle(4.558     |7.674  0.316  0.316)
rectangle(2.25      |7.595   0.316  0.316)
rectangle(1.552     |4.375  0.316  0.316)
rectangle(2.193     |9.063  0.316  0.316)
rectangle(1.552     |4.375  0.316  0.316)
rectangle(3.23      |4.177   0.316  0.316)
```

```
19 rectangle(4.629     |4.217  0.316  0.316)
20 rectangle(3.23      |4.177   0.316  0.316)
21 rectangle(4.738|4.177  0.316  0.316)
22 rectangle(|-5  6  3,5)
23 rectangle(|-5  6  3,5)
24 rectangle(|-5  6  3,5)
25 rectangle(|-5  6  3,5)
26 rectangle(|-5  6  3,5)
```

B.3 Python Code for the 5-bus System

B.3.1 Code

```
1 from gurobipy import *
2 t= range(1,25)
3 t1=range(1,24)
4 i= range(1,4)
5 si_S= [1, 4, 5]
6 si_R= [1, 3, 4]
7 si_D= [2, 3, 4, 5]
8 j= range(1,4)
9 d= range(1,5)
10 b= range(1,5)
11 k= list(range(1,6))
12 n= list(range(1,6))
13 Lam_D =[[[17.430, 17.250, 17.216, 16.886, 16.790], [17.430, 17.250, 17.216,
       16.886, 16.790], [17.430, 17.250, 17.216, 16.886, 16.790], [17.430, 17.250,
       17.216, 16.886, 16.790]],
14         [[17.250, 17.216, 16.886, 16.790, 16.380], [17.250, 17.216, 16.886,
       16.790, 16.380], [17.250, 17.216, 16.886, 16.790, 16.380], [17.250, 17.216,
       16.886, 16.790, 16.380]],
15         [[17.216, 16.886, 16.790, 16.380, 16.320], [17.216, 16.886, 16.790,
       16.380, 16.320], [17.216, 16.886, 16.790, 16.380, 16.320], [17.216, 16.886,
       16.790, 16.380, 16.320]],
16         [[17.216, 16.886, 16.790, 16.380, 16.320], [17.216, 16.886, 16.790,
       16.380, 16.320], [17.216, 16.886, 16.790, 16.380, 16.320], [17.216, 16.886,
       16.790, 16.380, 16.320]],
17         [[16.886, 16.790, 16.380, 16.320, 16.130], [16.886, 16.790, 16.380,
       16.320, 16.130], [16.886, 16.790, 16.380, 16.320, 16.130], [16.886, 16.790,
       16.380, 16.320, 16.130]],
18         [[16.886, 16.790, 16.380, 16.320, 16.130], [16.886, 16.790, 16.380,
       16.320, 16.130], [16.886, 16.790, 16.380, 16.320, 16.130], [16.886, 16.790,
       16.380, 16.320, 16.130]],
19         [[17.250, 17.216, 16.886, 16.790, 16.380], [17.250, 17.216, 16.886,
       16.790, 16.380], [17.250, 17.216, 16.886, 16.790, 16.380], [17.250, 17.216,
       16.886, 16.790, 16.380]],
20         [[17.940, 17.612, 17.430, 17.250, 17.216], [17.940, 17.612, 17.430,
       17.250, 17.216], [17.940, 17.612, 17.430, 17.250, 17.216], [17.940, 17.612,
       17.430, 17.250, 17.216]],
21         [[19.232, 18.932, 18.806, 19.344, 18.152], [19.232, 18.932, 18.806,
       19.344, 18.152], [19.232, 18.932, 18.806, 19.344, 18.152], [19.232, 18.932,
       18.806, 19.344, 18.152]],
22         [[20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922, 19.532,
       19.232, 18.932], [20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922,
       19.532, 19.232, 18.932]],
23         [[24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628, 20.876,
       20.606, 20.378], [24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628,
       20.876, 20.606, 20.378]],
```

```
24          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
25          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
26          [[24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628, 20.876,
        20.606, 20.378], [24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628,
        20.876, 20.606, 20.378]],
27          [[20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922, 19.532,
        19.232, 18.932], [20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922,
        19.532, 19.232, 18.932]],
28          [[20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922, 19.532,
        19.232, 18.932], [20.378, 19.922, 19.532, 19.232, 18.932], [20.378, 19.922,
        19.532, 19.232, 18.932]],
29          [[20.876, 20.606, 20.378, 19.922, 19.532], [20.876, 20.606, 20.378,
        19.922, 19.532], [20.876, 20.606, 20.378, 19.922, 19.532], [20.876, 20.606,
        20.378, 19.922, 19.532]],
30          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
31          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
32          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
33          [[25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968, 22.628,
        20.876, 20.606], [25.000, 24.968, 22.628, 20.876, 20.606], [25.000, 24.968,
        22.628, 20.876, 20.606]],
34          [[24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628, 20.876,
        20.606, 20.378], [24.968, 22.628, 20.876, 20.606, 20.378], [24.968, 22.628,
        20.876, 20.606, 20.378]],
35          [[19.532, 19.232, 18.932, 18.806, 18.344], [19.532, 19.232, 18.932,
        18.806, 18.344], [19.532, 19.232, 18.932, 18.806, 18.344], [19.532, 19.232,
        18.932, 18.806, 18.344]],
36          [[17.940, 17.612, 17.430, 17.250, 17.216], [17.940, 17.612, 17.430,
        17.250, 17.216], [17.940, 17.612, 17.430, 17.250, 17.216], [17.940, 17.612,
        17.430, 17.250, 17.216]]]
37 P_Dmax=[[[900, 25, 25, 25, 25]]*4]*24
38 P_Smax=[[[54.25, 38.75, 31.00, 31.00], [25.00, 25.00, 20.00, 20.00], [54.25,
        38.75, 31.00, 31.00], [68.95, 49.25, 39.40, 39.40]]]*24
39 Lam_S =[[[9.92, 10.25, 10.68, 11.26], [18.60, 20.03, 21.67, 22.72], [9.92,
        10.25, 10.68, 11.26], [10.08, 10.66, 11.09, 11.72]]]*24
40 P_Rmax=[[[140.00, 97.50, 52.50, 70.00], [68.95, 49.25, 39.40, 39.40], [68.95,
        49.25, 39.40, 39.40], [54.25, 38.75, 31.00, 31.00]]]*24
41 Lam_R = [[[19.20, 20.32, 21.22, 22.13], [10.08, 10.66, 11.09, 11.72], [10.08,
        10.66, 11.09, 11.72], [9.92, 10.25, 10.68, 11.26]]]*24
42 model = Model()
43 G_S,P_S,P_R,P_D,Lam_tn,mu_Smax,mu_Rmax,mu_Dmax,om_Smax,om_Rmax,om_Dmax,mu_Dmin,
        om_Dmin,mu_Smin,om_Smin,mu_Rmin,om_Rmin = [{} for x in range(17)]
44 for T in t:
45      for I in i:
46          for B in b:
47              G_S[T, I, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="G_S_{}
        _{}_{}".format(T, I, B))
48              P_S[T, I, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_S_{}
        _{}_{}".format(T, I, B))
```

```
49                mu_Smax[T, I, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_Smax_{}_{}_{}".format(T, I, B))
50                om_Smax[T, I, B] = model.addVar(vtype=GRB.BINARY, name="om_Smax_{}_
      {}_{}".format(T, I, B))
51                mu_Smin[T, I, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_Smin_{}_{}_{}".format(T, I, B))
52                om_Smin[T, I, B] = model.addVar(vtype=GRB.BINARY, name="om_Smin_{}_
      {}_{}".format(T, I, B))
53 for T in t:
54     for J in j:
55         for B in b:
56             P_R[T, J, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_R_{}
      _{}_{}".format(T, J, B))
57             mu_Rmax[T, J, B] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_Rmax_{}_{}_{}".format(T, J, B))
58             om_Rmax[T, J, B] = model.addVar(vtype=GRB.BINARY, name="om_Rmax_{}_
      {}_{}".format(T, J, B))
59             mu_Rmin[T, J, B] = model.addVar(vtype=GRB.CONTINUOUS, name="mu_Rmin_
      {}_{}_{}".format(T, J, B))
60             om_Rmin[T, J, B] = model.addVar(vtype=GRB.BINARY, name="om_Rmin_{}_
      {}_{}".format(T, J, B))
61 for T in t:
62     for D in d:
63         for K in k:
64             P_D[T, D, K] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="P_D_{}
      _{}_{}".format(T, D, K))
65             mu_Dmax[T, D, K] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_Dmax_{}_{}_{}".format(T, D, K))
66             om_Dmax[T, D, K] = model.addVar(vtype=GRB.BINARY, name="om_Dmax_{}_
      {}_{}".format(T, D, K))
67             om_Dmin[T, D, K] = model.addVar(vtype=GRB.BINARY, name="om_Dmin_{}_
      {}_{}".format(T, D, K))
68             mu_Dmin[T, D, K] = model.addVar(vtype=GRB.CONTINUOUS, lb=0, name="
      mu_Dmin_{}_{}_{}".format(T, D, K))
69 for T in t:
70     for N in n:
71         Lam_tn[T, N] = model.addVar(vtype=GRB.CONTINUOUS, name="Lam_tn_{}_{}".
      format(T, N))
72 model.update()
73 for T in t:
74     for I in i:
75         for B in b:
76             model.addConstr(G_S[T, I, B]-Lam_tn[T, si_S[I-1]]+mu_Smax[T, I, B]-
      mu_Smin[T, I, B] == 0)
77             model.addConstr(P_Smax[T-1][I-1][B-1]-P_S[T, I, B] >= 0)
78             model.addConstr(P_Smax[T-1][I-1][B-1]-P_S[T, I, B] <= (1-om_Smax[T,
      I, B])*P_Smax[T-1][I-1][B-1])
79             model.addConstr(mu_Smax[T, I, B] <= om_Smax[T, I, B]*P_Smax[T-1][I
      -1][B-1])
80             model.addConstr(P_S[T, I, B] <= (1-om_Smin[T, I, B])*P_Smax[T-1][I
      -1][B-1])
81             model.addConstr(mu_Smin[T, I, B] <= om_Smin[T, I, B]*P_Smax[T-1][I
      -1][B-1])
82 for T in t:
83     for J in j:
84         for B in b:
85             model.addConstr(Lam_R[T-1][J-1][B-1]-Lam_tn[T, si_R[J-1]]+mu_Rmax[T,
       J, B]-mu_Rmin[T, J, B] == 0)
86             model.addConstr(P_Rmax[T-1][J-1][B-1]-P_R[T, J, B] >= 0)
```

```
87                model.addConstr(P_Rmax[T-1][J-1][B-1]-P_R[T, J, B] <= (1-om_Rmax[T,
      J, B])*P_Rmax[T-1][J-1][B-1])
88                model.addConstr(mu_Rmax[T, J, B] <= om_Rmax[T, J, B]*P_Rmax[T-1][J
      -1][B-1])
89                model.addConstr(P_R[T, J, B] <= (1-om_Rmin[T, J, B])*P_Rmax[T-1][J
      -1][B-1])
90                model.addConstr(mu_Rmin[T, J, B] <= om_Rmin[T, J, B]*P_Rmax[T-1][J
      -1][B-1])
91 for T in t:
92      for D in d:
93          for K in k:
94                model.addConstr(Lam_tn[T, si_D[D-1]]-Lam_D[T-1][D-1][K-1]+mu_Dmax[T,
       D, K]-mu_Dmin[T,D,K] == 0)
95                model.addConstr(P_Dmax[T-1][D-1][K-1]-P_D[T, D, K] >= 0)
96                model.addConstr(P_Dmax[T-1][D-1][K-1]-P_D[T, D, K] <= (1-om_Dmax[T,
      D, K])*P_Dmax[T-1][D-1][K-1])
97                model.addConstr(mu_Dmax[T, D, K] <= om_Dmax[T, D, K]*P_Dmax[T-1][D
      -1][K-1])
98                model.addConstr(mu_Dmin[T,D,K] <= om_Dmin[T, D, K]*P_Dmax[T-1][D-1][
      K-1]*1000000)
99                model.addConstr(P_D[T, D, K] <= (1-om_Dmin[T, D, K])*P_Dmax[T-1][D
      -1][K-1])
100 for T in t:
101     model.addConstr(quicksum(P_D[T, 1, K] for K in k)==0.19*quicksum(P_D[T, D, K
      ] for K in k for D in d))
102     model.addConstr(quicksum(P_D[T, 2, K] for K in k)==0.27*quicksum(P_D[T, D, K
      ] for K in k for D in d))
103     model.addConstr(quicksum(P_D[T, 4, K] for K in k)==0.27*quicksum(P_D[T, D, K
      ] for K in k for D in d))
104 for T in t:
105     for N in n:
106         model.addConstr(quicksum(P_S[T, I, B] for B in b for I in i)+quicksum(
      P_R[T, J, B] for B in b for J in j)==quicksum(P_D[T, D, K] for K in k for D
      in d))
107 obj = (quicksum(Lam_S[T-1][I-1][B-1]*P_S[T, I, B] for B in b for I in i for T in
       t)
108        +quicksum(Lam_R[T-1][J-1][B-1]*P_R[T, J, B] for B in b for J in j for T
      in t)
109        -quicksum(Lam_D[T-1][D-1][K-1]*P_D[T, D, K] for K in k for D in d for T
      in t)
110        +quicksum(P_Rmax[T-1][J-1][B-1]*mu_Rmax[T, J, B] for B in b for J in j
      for T in t)
111        +quicksum(P_Dmax[T-1][D-1][K-1]*mu_Dmax[T, D, K] for K in k for D in d
      for T in t))
112 model.setObjective(obj, GRB.MINIMIZE)
113 model.optimize()
```

B.3.2 Output

```
1 Optimize a model with 6528 rows, 5688 columns and 19680 nonzeros
2 Coefficient statistics:
3   Matrix range     [2e-01, 9e+08]
4   Objective range  [1e+01, 9e+02]
5   Bounds range     [1e+00, 1e+00]
6   RHS range        [1e+01, 9e+02]
7 Warning: Model contains large matrix coefficients
8          Consider reformulating model or setting NumericFocus parameter
9          to avoid numerical issues.
10 Presolve removed 6517 rows and 5676 columns
```

```
Presolve time: 0.03s
Presolved: 11 rows, 12 columns, 28 nonzeros
Variable types: 8 continuous, 4 integer (4 binary)

Root relaxation: objective -9.257819e+04, 0 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

*    0     0               0    -92578.19000 -92578.190  0.00%     -    0s

Explored 0 nodes (0 simplex iterations) in 0.06 seconds
Thread count was 8 (of 8 available processors)

Optimal solution found (tolerance 1.00e-04)
Best objective -9.257819000000e+04, best bound -9.257819000000e+04, gap 0.0000%
```

ABOUT THE AUTHOR

Sayed Abdullah Sadat was born in 1989 in Kunduz, Afghanistan. He obtained his bachelor degree in Electrical and Electronics Engineering from Osmania University, India in 2013. He received his Master of Science in Electrical Engineering degree from the University of South Florida, USA in May 2017. His research interests include: electricity market, optimization, distributed control of microgrids, predictive control in active distribution systems, modeling of renewable energy sources, stability analysis of microgrids and real-time digital simulation.