

ABSTRACT

DENAXAS, EVANGELOS. Decomposition of the Stochastic Market Clearing Problem with Lagrangian Relaxation. (Under the direction of Dr. Kay and Dr. Medhin).

The modern way of living has a subtle requirement for electricity. The uninterrupted flow of energy is an important quality metric for modern cities as almost everything depends on this. A closer look at the electricity network that ensures the production and delivery of electricity is given in this thesis. The field of energy economics provides a rich collection of large-scale models that help operators of the network manage the available resources with efficiency and safety. The problem of stochastic market clearing is studied as one of the models that provide a complete view of the energy market as a system. The intricacy of the problem results in extreme computational complexity. This thesis studies practical decomposition methods that curve the exponential complexity of the problem and provide near optimal solutions to large-scale models. Finally, this work aspires to serve as a practical guide to the application of decomposition and hence weight is also given in presenting the field with details in hope that future researchers may find practical answers to their problems.

Decomposition of the Stochastic Market Clearing Problem using Lagrangian Relaxation

by
Evangelos Denaxas

A thesis submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Master of Science

Operations Research

Raleigh, North Carolina

2015

APPROVED BY:

Dr. Michael Kay
Co-Chair of Advisory
Committee

Dr. Negash Medhin
Co-Chair of Advisory
Committee

Dr. Patiño-Echeverri

Dedication

I would like to dedicate this work to my family.

To my mother who constantly supports me and gives me positive energy to continue my journey a few thousand miles away from Greece.

To my father who has been my main source of motivation and courage all of these years. Even though he left this world early, he will always be with me.

Biography

I am an optimization engineer with a background in combinatorial optimization and large-scale applications of global optimization. I received my first Master's Degree in July 2013 in Electrical and Computer Engineering at Aristotle University of Thessaloniki, Greece where I successfully defended my thesis titled, "Solution of the multi-trip Vehicle Routing Problem with Time Window restriction using parallel Simulated Annealing method" under the supervision of Professor Nikos Pitsianis. In August 2013, I started my second Master's in Operations Research at North Carolina State University with a focus in non-linear and global optimization and theory of algorithms. Currently, I live in Framingham, Massachusetts where I work as a Software Engineer on Applied Optimization for the company MathWorks.

Acknowledgements

I would like to wholeheartedly thank Ali Daraeepour and Dr. Dalia Patino-Echeverri for all of the patience and support they have shown me. Ali's patience has been critical for the completion of this work as well as the strong, loving support of Dalia.

I would also like to thank Terri Thomas for editing this thesis and correcting all of my Greek errors.

Lastly, I would like to thank Dr. Kay and Dr. Medhin for their support throughout my studies at NCSU.

Table of Contents

List of Tables	vii
List of Figures	viii
1 Introduction	1
2 Decomposition techniques.....	4
2.1 Introduction	4
2.2 Decomposition in linear programming	4
3 Stochastic Optimization.....	9
4 Two stage stochastic programming	14
4.1 Introduction	14
4.2 Classification of problems.....	16
4.3 Decomposing M/M	18
4.4 Sub-gradient optimization	21
4.5 Phases of decomposition	23
4.5.1 Initial guess of multipliers	25
4.5.2 Solving the sub-problems	25
4.5.3 Gathering of distributed solutions.....	26
4.5.4 Stopping criteria.....	26
4.5.5 Updating the sub-gradient.....	27
4.5.6 Updating the step length	28
4.5.7 Updating the multipliers	30
5 Power systems engineering	34
5.1 Introduction to electricity market.....	34
5.2 Entities of the market	34
5.2.1 Consumers.....	35
5.2.2 Producers.....	36
5.2.3 Managing authorities	40
5.3 Operational overview	41
5.4 Stochastic market clearing under wind uncertainty	44
5.4.1 Model elements.....	45
5.4.2 Objective function.....	54

5.4.3	Constraints	55
6	Decomposition of the Stochastic Market Clearing problem	58
6.1	Formulating the Lagrangian	58
6.2	Formulating the decomposed sub-problems	62
6.3	Calculating sub-gradient directions.....	64
6.4	Strengthening the sub-problems.....	64
7	Results	66
7.1	Problem data.....	66
7.2	Experiment setting.....	69
7.2.1	Simple step method.....	69
7.2.2	Polyak step method.....	70
7.2.3	Weighted sub-gradient method.....	72
7.2.4	Overall comparison.....	74
7.2.5	Cutting plane method.....	74
7.3	Practical considerations.....	75
8	Conclusions and future work.....	77
9	Bibliography	78
10	Appendix.....	82
10.1	Main control flow.....	82
10.2	First stage sub-problem	90
10.3	Second stage sub-problem.....	95

List of Tables

<i>Table 5-1</i> Indices used in SMC model	45
<i>Table 5-2</i> Cost coefficients of the SMC problem	46
<i>Table 5-3</i> Constraint coefficients of SMC.....	47
<i>Table 5-4</i> First stage decision variables for the SMC	48
<i>Table 5-5</i> Second stage decision variables for the SMC	48
<i>Table 5-6</i> Lagrangian multipliers for the SMC	49

List of Figures

<i>Figure 2-1 The block angular structure</i>	5
<i>Figure 2-2 The block diagonal structure</i>	7
<i>Figure 4-1 Sub-gradients of a non-smooth function</i>	22
<i>Figure 4-2 Sub-differential function for $\varphi(\lambda)= \lambda$</i>	23
<i>Figure 4-3 Oscillation of sub-gradient methods</i>	31
<i>Figure 5-1 Entities of the energy market</i>	34
<i>Figure 5-2 Model days of load for summer and winter from the NE-ISO</i>	36
<i>Figure 5-3 Breaddown of power sources per country</i>	38
<i>Figure 5-4 Map of ISOs in the US</i>	41
<i>Figure 5-5 Timeline of the energy market</i>	42
<i>Figure 7-1 Average wind production</i>	67
<i>Figure 7-2 Maximum capacity of slow units</i>	67
<i>Figure 7-3 Maximum capacity of fast units</i>	68
<i>Figure 7-4 Total demand in a day</i>	68
<i>Figure 7-5 Simple step length method</i>	70
<i>Figure 7-6 Polyak step length for $\lambda=0.1, 1$</i>	71
<i>Figure 7-7 Polyak step length for $\lambda=10$</i>	72
<i>Figure 7-8 Weighted sub-gradient method</i>	73
<i>Figure 7-9 Comparison of sub-gradient methods</i>	74
<i>Figure 7-10 Comparison of production and demand after Phase 2</i>	76

1 Introduction

The field of Operations Research is tightly coupled with the study of optimization problems. Drawn from various industries, optimization problems are modeled carefully to represent real life processes; hence, the reduction of cost of the model can lead to cost reduction or robust hedging against risk. The general form of an optimization model is shown in the following form:

$$\begin{aligned} & \min_x f(x) \\ & \text{such that: } g(x) \leq 0 \\ & h(x) = 0 \end{aligned}$$

In this general sense, the cost of the process of interest under the form of a mathematical formula $f(x)$ that acts upon a domain of decision variables x is modeled. Constraints on the physical limits of the decision variables are modeled with the set of inequality functions $g(x)$ while equality constraints are modeled with $h(x)$.

The plurality of different combinations of these elements as well as the mathematical nature of the described functions spawns interesting branches on Operations Research with great complexity and importance. In the academic setting, in order to reduce the complexity of the analysis of the solving techniques for these problems, it is customary for certain assumptions to take place. It is often the case where optimization problems are studied under assumptions of convexity of the objective constraint functions, or assumptions on the smoothness of the problem space considering the decision variables as part of sets of numbers that provide the engineer with tangible tools for understanding and tackling the problem.

In a variety of applications such approach is sensible in the sense that, approximating the real problem with a small degree of error provides a practical path of dealing with the extreme computational burden that usually arises in Optimization Theory provided that the engineer has a systematic way of bounding the error under practical and application specific thresholds. Nevertheless, as real life processes evolve, effectively modeling systems under helpful assumptions becomes increasingly challenging and as the systems assume great size and

complexity it becomes difficult to formulate a successful approximation that its optimization will be able to provide a realistic impact.

In the current study, the application is drawn from the field of Electricity Economics and in particular the Stochastic Market Clearing problem. Following the model by [1] the problem can be mapped to a large mixed integer programming problem, a formulation that appears to be flexible and practical under the assumption of deterministic sources of data. It becomes a cumbersome computational task when introducing stochasticity in parts of the model that naturally involve probabilistic elements like forecasts on load demand or wind power production.

The nature of the problem brings forth practical considerations that make it hard to use any of the tools of approximation. The focus is directed towards addressing the difficulties that may be encountered in a practical manner and different ways to tackle the computational load that is required to efficiently provide a solution to the modeled function.

The following chapters are structured in a way that provides a guide to different solution techniques that are usually employed in practical settings. The application of these methods is also presented on a particular problem while exposing the different techniques and heuristics that can provide a sensible manner to dealing with the complexity.

In Chapter 2, an overview of the field of decomposition in linear programming is given. The lessons learned from the linear case in many cases can be helpful when dealing with integer or mixed integer problems. This is particularly true for the method of Lagrangian decomposition, which is the topic of this thesis.

In Chapter 3, a brief overview of the field of Stochastic Optimization is given where important notions on stochasticity and the value of a stochastic solution versus the deterministic are introduced.

In Chapter 4, a more specific view onto the particular field of interest is provided: the two stage stochastic programs and decomposition techniques relevant to the problem. The field of non-differential optimization is presented with the main focus on sub-gradient optimization. The methods presented in this chapter are the ones that have been used in the application.

Chapter 5 serves as introduction to the electricity market. Important terms are introduced and helpful insights on the problems that can be found in the field are given. Additionally, the problem under consideration is formally introduced. All the important elements of the model are described and a clear connection with real life application is attempted.

In Chapter 6, detailed information of the application of decomposition of the Stochastic Market Clearing problem is provided.

Chapter 7 provides a detailed overview of the results from applying different decomposition techniques on the problem and comments on the applicability and effectiveness of different methods.

Lastly, Chapter 8 provides thoughts on future work and improvements on the field.

2 Decomposition techniques

2.1 Introduction

The emphasis of this thesis will be on decomposition of a large-scale stochastic, two stage mixed integer problem with a specific application on the Stochastic Market Clearing problem.

The application of decomposition for this problem is not straightforward. Decomposition of this problem has not been studied in the literature. Additionally, decomposition literature involving a mix of integer and continuous variables in both stages of two stage stochastic programs is limited. The lack of sufficient support from the literature becomes an opportunity to provide a comprehensive exposition on the field.

An additional issue to the future researcher is the lack of extensive survey reports on best practices and applications of two stage stochastic programming. Applications and methodologies do exist in the literature for different kinds of formulations but only a few try to present the state of the art in a centralized manner [2] including the case of having mixed integers in both stages. As an effect, it is not intuitive to understand the inner connection of the algorithms that are used in the field to use this knowledge for future extensions.

In this chapter the main ideas on decomposition and seminal work in the field for the continuous case will be introduced. Building on this knowledge, in later chapter extensions of these methods for the case of two stage stochastic programming will be presented and justification for the choice of approaching the problem with Lagrangian relaxation will be explained.

2.2 Decomposition in linear programming

In earlier years of industrial engineering, modeling real life processes as linear problems quickly led engineers to deal with thousands or millions of variables. Following the most advanced techniques at the time led to the realization that dealing with large problems as a whole is not practical and in many cases impossible [3]. Hence, a large portion of the literature has since been devoted to investigating ways to tackle the rising complexity of the problems and provide solutions in a reasonable timeframe. With increasing power of modern computers and advances in algorithms, currently linear problems with hundreds of thousands or millions

of variables can be solved easily within a matter of minutes. Nevertheless, the lessons learned from this period become valuable since generalizations of the methods that will be presented can be applied directly in problems where linearity is not given.

The fundamental ideas that allow decomposition of a problem are the idea of column generation linked with the idea that problems oftentimes have exploitable structure.

Focusing on the constraints of a problem, equations are formulated in the following form:

$$Ax = b$$

In practical applications as the problem grows and variables x span across millions of elements the constraint matrix A becomes sparse. This is due to the natural control flow of the processes that the matrix is naturally modeling and its inner connection. With careful rearrangement of columns and rows the matrix can be reformulated to expose a certain structure that is easily exploitable by decomposition methods.

Every linear program can thus be reformulated to assume block angular structure as shown in the figure below.

$$A = \begin{bmatrix} B_1 & B_2 & \dots & B_p \\ C_1 & 0 & 0 & 0 \\ 0 & C_2 & 0 & 0 \\ 0 & 0 & \dots & 0 \\ 0 & 0 & 0 & C_p \end{bmatrix}$$

Figure 2-1 The block angular structure

If p equals to 1, any linear program in this general structure can be represented. In large applications, the constraints can be separated into sets. In the figure above, sets B refer to constraints that act upon all the variables x , while the sets C act upon only parts of the variables. This becomes obvious when working with modeling of large real life processes where the structure depicted in Figure 2-1 becomes an expression of the connection of the modeled subsystems.

In the seminal paper in the area by [4], the algorithm exploits the block angular structure of large systems to provide parallel execution and to avoid the increase of computational complexity of the full model.

From [5], we find the foundation of the method in the following theorem.

1. Assume $X \in R^n$ is a compact convex set, then $S(X)$ can represent the set of all extreme points of X .
2. If $Conv(S(X))$ is the convex hull of $S(X)$, then $C(E(X)) = X$

This theorem gives the ability to decompose a full problem that has been rearranged to have block angular structure according to the realization that any element in the variable vector x may be substituted by convex combination of the following form:

$$x^k = \sum_i \lambda_i x_i, x_i: C_j x = b_j, x \geq 0$$

The full problem is separated into a master problem that maintains the state of the variables across the whole model and a set of sub-problems that maintain information about the sub-matrices. In an iterative fashion the algorithm is guaranteed to converge to the optimum solution in a finite amount of steps.

The use of the convex combination of the variables leads to the formulation of the master problem and p sub-problems. The form of the master problem is shown below.

$$\begin{aligned} & \min c^T \sum_i \lambda_i x_i \\ & st: \sum_i B x_i \lambda_i = b \\ & \sum_i \lambda_i = 1, \lambda_i \geq 0 \end{aligned}$$

Note the implication of the notation used to describe the sub-matrices of the angular form in the formulation of the master problem. Matrix B accommodates the complicating constraints

while matrices C are implicitly present in the definition of the convex combination of the variables x_i .

Furthermore, the set of sub-problems can be formulated as follows.

$$\begin{aligned}
 SP^n: \min & (c_n - \pi_n B_n)x \\
 \text{st: } & C_n x = b_n \\
 & x \geq 0, n = 1, \dots, p
 \end{aligned}$$

The benefit of the method is that at each iteration, smaller problems are dealt with and hence the computational complexity is linked with the full matrix A .

As the process evolves new columns are added to the master problem as they become available from the sub-problems hence the categorization of this method to the column generation algorithms. In contrast with column generation, the next method adds constraints to the master problem across iterations. Bender's decomposition, or the L-shaped method should be viewed as the dual counterpart of the Dantzig-Wolfe decomposition where instead of columns, rows are added to the main problem.

If in addition to complicating constraints there are complicating variables in the original problem, as the problem grows in size the full matrix A may be rearranged in order to assume a block diagonal form shown in the following figure.

$$A = \begin{bmatrix} B_1 & B_2 & \dots & B_p & D_1 \\ C_1 & 0 & \dots & 0 & D_2 \\ 0 & C_2 & 0 & 0 & \dots \\ 0 & 0 & \dots & 0 & \dots \\ 0 & 0 & 0 & C_p & D_m \end{bmatrix}$$

Figure 2-2 The block diagonal structure

The connection between the two methods becomes apparent if considering the Dantzig-Wolfe master problem to contain the first row of the complicating constraints and the sub-problems referring to the sub-matrices of the complicating variables. Then in order to solve the sub-problems, optimize the dual counter parts that will effectively show block angular structure

rather than block diagonal structure. This method essentially describes the steps taken by Bender's decomposition.

Lastly, the final main path of decomposition formulates the Lagrangian of the original problem where the complicating constraints have been relaxed. In the case where the feasible region does not include any discontinuity points, the Karush-Kuhn-Tucker conditions can be used to solve the problem directly. In presence of discontinuities, iterative descent methods may be applied to reach optimality. The importance of the Lagrangian decomposition is that the method does not require any assumptions on convexity or linearity for the original problem. The price is paid with slow convergence rates to reach the optimal solution.

It is important to note the deep connection between these three main solution approaches to decomposition. In the continuous case all methods converge to the same solutions and observing the different formulations can make interesting conclusions. Convexity, guaranteed by linear programs allows the exposition in the case of Lagrangian relaxation of the dual formulation when applied to simple linear programs [6]. Also, it is instructive to see that Bender's decomposition, even though oftentimes presented in literature separate from other methods, is in fact the application of Dantzig-Wolfe decomposition to the dual formulation of the original problem [5].

The realization that the methods are similar becomes important when dealing with integer programs and the literature presents different variations of these original decomposition techniques. It should be expected to see the same computational problems and convergence issues found in methods that stem from Bender's decomposition as well as Dantzig-Wolfe decomposition.

Furthermore, in integer and mixed integer cases as discussed in later chapters, following the Lagrangian relaxation path is oftentimes criticized due to a slow convergence rate. Nevertheless the counterpart algorithms, based on the other two techniques suffer from similar convergence issues precisely because of strong connections of the methods.

3 Stochastic Optimization

In most practical applications, modeling real life processes involves modeling uncertainty. In reality, the majority of the data that engineers have to use in order to map the problem onto a solution space is probabilistic in nature or are results of stochastic processes. This element becomes obvious when dealing with dynamic systems like the electricity or stock market or with time evolving systems where the decision maker has limited information about the outcomes of their choices. Hence, it becomes of great importance to be able to navigate in the space of uncertainty in a practical way and be able to quantify or model the uncertain sources in effective models that will be able to be incorporated into the general optimization model.

In many cases it is possible for the decision maker to analyze the probabilistic nature of the uncertain sources that participate in the model, then the derived distribution function can be easily incorporated in the optimization procedure. One of the alternatives in dealing with these problems is working with expected values of the data and hence the minimization of the objective function leads to minimizing with respect to the mean.

In some practical applications, optimization is focused on the worst case scenario whereby the designer chooses to work with the extreme values of uncertain data, making sure that the feasibility of the worst case scenario qualifies application specific requirements [1].

Nevertheless, the mean value approach or the worst case scenario approach in many applications fail to deliver an optimal decision based on the fact that there is the danger of over or under estimation of the elements which can produce a sub optimal solution to the problem that in many cases may be too conservative.

Additionally to the reasons mentioned above, working directly with analytical probability functions may not be feasible in many cases, especially when the stochastic processes involved are of complex nature, time varying, or chaotic.

It is important to notice that in many practical settings of importance, the stochastic processes may be semi-Markovian continuous processes that approximate real life variance of signals. In that case, incorporating a full analytical description may prove to be cumbersome in analysis and computational complexity [7].

In the case where it is impossible or of little practical value to work directly with analytical formulas the notion of scenarios in the optimization procedure can be incorporated. The term scenario is defined as a full set of data corresponding to a particular realization of the stochastic process under consideration that is tied to a probability of realization. For example, in the financial engineering setting when studying the maximum return on the stock market, a scenario could be a full set of a forecasted trend of a stock for a particular time period tied to a specific probability of realization.

In the electricity market, whenever wind production penetration is under consideration, it is customary to generate scenarios for “model days” of the year based on past historical data along with the statistical probability of realization. Assuming some degree of knowledge of the past it is possible to generate models that imitate the statistical characteristics of wind in order to generate realizations of the underlying stochastic process, which in this case is the wind.

Working with scenarios oftentimes requires a reformulation of the original deterministic model in order to incorporate the values of the realizations of the stochastic variable as well as the probabilities of the realization.

In most cases, the objective function includes a weighing term over the probability space and the constraints are reformulated into two sets of variables; the “wait and see” variable and the “here and now” variable. The “here and now” variable includes all the variables of the model that remain unaffected by the stochasticity of the model. These may be cost variables of variables that refer to decentralized parts of the model that do not contain any probabilistic element.

The “wait and see” variable are the variables tied to the behavior of each scenario. In literature it is customary to encounter the terms scenario independent and scenario dependent to describe the two sets. An example of a general formulation can be found in the formula below.

$$\min_x c^T x_i + \sum_{s \in S} p_s (d_s^T x_d)$$

such that: $Ax = b$

$Wx_d = T$

Where S describes the set of all scenarios, p_s describes the realization probability for every scenario and the subscripts on the decision variables are used to designate scenario dependent variables and scenario independent variables. In this case we can imagine vector d_s^T describing costs related to each scenario realization.

Working with scenarios in stochastic optimization gives the decision maker a practical tool to incorporate tangible data sources into the model and deal with uncertainty in ways that in many practical applications is not possible. There is practical benefit of working with mathematical models that approximate weather patterns and incorporating the resulting data into the optimization rather than having to work with real data.

In many cases getting ahold of real data or real measurements becomes an issue of copyrights or proprietary information and engineers can be banned from being able to draw useful conclusions. Even in cases where actual data sets are easily accessible, in many cases the engineer needs to go through a cumbersome sanitation process of the data and deal with issues of data corruption that can either introduce sources of error in the statistical nature of the data set or even deem the source impractical to work altogether.

As practical as it may be, the technique of working with scenarios proves to be expensive in practice. The use of scenarios is equivalent practice to a discretization of the probability space for the sake of practicality in order to make feasible the analysis of a system. This procedure introduces approximation errors that will impact the quality and the effectiveness of the results. It is useful in that sense to include a wide range of scenarios to capture all of the stochastic characteristics, a notion very similar with increasing the sample space on a statistical experiment.

Assuming the decision maker is able to provide a large amount of sample scenarios, this will eventually prove to give little practical benefit as the complexity of the model becomes

exponentially difficult to solve with every addition of a scenario. Especially in the cases of dealing with problem formulations that require integer, binary, or mixed integer solution spaces, the problems belong to the NP-Hard family and are notoriously difficult to provide exact solutions and the difficulty explodes exponentially to the size of the problem.

Nevertheless, as discussed in further detail in Chapter 6, the architecture of a problem formulated with the use of scenarios has a very helpful structure that is easily exploitable.

Given the increased computational load of stochastic optimization programs, it is oftentimes helpful to quantify the merit of the stochastic solution versus the deterministic equivalent model that incorporates mean values of the stochastic elements and provides a straightforward approach to solve. The following definitions are widely accepted metrics to quantify the value of the stochastic approach in the literature:

- Expected Value Solution (EVS)
- Expected Value of Perfect Information (EVPI)
- Value of Stochastic Solution (VSS)

The metric of the Expected Value Solution (EVS) is the cost of the deterministic equivalent problem where it has been assumed that all stochastic variables will assume their mean values. This substitution in the optimization problem provides a straightforward simplification.

In order to calculate the metric of Expected Value of Perfect Information (EVPI), it needs to be assumed that the decision maker has perfect foresight with regards to uncertainty and is able to make optimal decisions at every step of the problem.

The difference between the optimal objective and the EVS is defined as the metric EVPI and essentially can be described as the negative impact of stochasticity on the decision making process. Note that in reality it is not assumed that there is perfect foresight but this calculation gives a quantitative view of the impact that the stochastic elements of the model can have in the end result.

The most revealing metric, the Value of Stochastic Solution (VSS) is calculated by taking the difference between the optimal cost of the stochastic problem where every scenario is

considered with the designated probabilities and the EVS. This metric reveals the merit of following a more computationally evolved method to provide a solution to the problem versus simply incorporating the expected values in the model. In many cases, the EVS provides sub-optimal approximations that prove to be overly conservative, depending on the specific probabilistic characteristics of the model.

Having the above terms defined it is advisable before adopting any of the solution techniques presented in the rest of this thesis to analyze the values of these metrics in order to gain clarity on the level that stochasticity may impact the final solution.

In many applications the decision maker may find the benefits obtained by the full stochastic model that takes every scenario under consideration does not yield great benefit in the optimal cost. In that case it will be justified to follow solution techniques that focus on the average values of the stochastic variables.

The interested reader can find a detailed example on the calculation of these metrics and an illustration of the impact of the sources on Stochastic Programming [7].

4 Two stage stochastic programming

4.1 Introduction

In this chapter the focus will be on presenting the family of two stage stochastic programming problems that is of interest for this application. In many practical settings decision-making is set in a continuous base where a series of decisions must be made sequentially with every step being connected to the next. Drawing from the field of Logistics Engineering, the problem of multi-period production scheduling would be an example where decisions form pipelined stages and the decision maker has to decide the optimal solution for a pre-specified time period [8].

These problems in general are classified as multi-stage optimization problems and when stochasticity is allowed to play a role in modeling, the methods belong to the family of multi-stage stochastic programming problems. The subset of two stage stochastic problems has a limitation on the number of stages, or periods, and it is required to be just two. Many practical applications may be mapped to these problems and the reason for this is intuitive.

In the setting being explored it is assumed that the decision maker has to make a decision. Nevertheless, it is impossible to achieve optimality with this decision because he or she is behind a veil of ignorance with regard to the effect of their decisions due to uncertainty on some part of the problem. Next, it is assumed that the uncertainty reveals itself the actual values of the stochastic elements become identifiable. After this revelation the decision maker needs to make corrective actions upon the decision that has already been taken.

In a sequential timeline, the decision maker goes in the following direction:

1. A decision needs to be taken at time t_0 assuming no information is available regarding the stochastic elements.
2. At time $t_1 > t_0$ the uncertainty is lifted and the decision maker has clarity on the actual values of the problem.
3. At time $t_2 > t_1$ the decision maker needs to adjust the decision he or she made at t_0 .

Under this general framework, there are a variety of cases that can fall within. In the financial engineering field, brokers need to make a decision about the movement of stocks before they become aware of the movement of stock. In the electricity market the generating units are committed a day in advance in the market without being aware of the level that the demand will reach or the level of load that will be able to be absorbed by renewable resources like wind or solar, since both these elements bare stochastic characteristics.

In two stage stochastic problems the definition of the terms “here and now” and “wait and see” variables becomes more intuitive as the decision maker at t_0 , is required to make a decision “here and now” while keeping in mind, it is only at t_2 that corrective actions represented by the “wait and see” variables may take place.

A more formal and widely used terminology for these sets of variables that will be adopted is the first stage decisions and the second stage decisions. In the first stage there is ignorance about the stochasticity of the model, therefore any scenario dependent values cannot be included. The second stage values are tied to the realization of each scenario and represent both the realization of the stochasticity as well as the corrective actions needing to take place.

As discussed earlier, stochastic problems pose great computational burden in practical applications. In reality the decision maker has to face millions of variables and hundreds of scenarios that effectively cause the complexity of the model to explode in an exponential degree. A large portion of the literature and research on the topic of stochastic problems is devoted to methods that try to deal with complexity since in many cases the solution of these problems, given a large enough data set is not computationally tractable.

This aspect can really affect the practicality of the solution methods especially in dynamic environments where the decision maker needs to provide an insight in a matter of hours. In this sense, having an alternative of substituting the stochastic variables of the problem by averaging over the elements makes sense, given that as shown earlier, the data provides a justified argument for this choice. Nevertheless, in many cases that is not true and other options need to be explored that can help deal with the complexity of these models.

4.2 Classification of problems

In an effort to shed light onto the field of two stage stochastic programming problems, in this section a formal classification of all the problems as well as the most practical application for solution via decomposition will be presented. This choice assumes that stochastic problems of this nature that are found in practical applications, always come with a high degree of complexity and a large set of scenarios that make the straightforward solution difficult when dealing with the full problem.

In the seminal paper by [9], there is a helpful classification notation that tries to segregate the family of two stage stochastic programming problems based on the nature of the variables in both stages. In a manner similar to the classification of queuing systems a three-letter classification system can be used where the first letter denotes the nature of the variables in the first stage, the second letter denotes the nature of the variables in the second stage, and the third letter denotes the nature of the stochasticity.

In details [9] use the following letters for classification:

- B to denote purely binary variables
- C to denote continuous variables
- M to denote that the variables are a mix of binary and continuous

The letters above may be used in the first two slots of the three-letter system fully capturing the nature of the variables involved in a particular method. Note that the case of pure integer variables is omitted since every integer program can be transformed to a 0/1 binary problem without loss of generality.

For the characterization of the stochasticity the authors use the following letters:

- D to denote using discrete distributions
- C to denote continuous stochastic variables

In most practical applications the case of dealing with continuous stochastic elements is rare. Usually due to the increased complexity of dealing with analytical formulas, as discussed

earlier, most applications deal with discretized stochastic domain and apply the notion of scenarios. Therefore, in the following presentation the last letter assuming the focus is on the discrete case will be omitted.

The pure continuous case, C/C following the notation, is the simplest family of problems and can be directly solved by following Bender's decomposition algorithm. The scenarios force the constraint matrix to assume a block diagonal structure that is exploitable by the method. Bender's decomposition is based on the strong duality that is linked with purely continuous variables. In cases where either the master problem or the sub-problems include integer, binary, or a mix of integer and binary variables the strong duality theorem is not applicable. Only in special cases it can be proved that problems of this kind may in fact behave within the prescription of the strong duality theorem [10] but this is not usually the case, especially in large models.

In lack of support from duality, [11], suggested a modification of the method in order to be applicable in a broader collection of problems, where convexity is not assumed. Most of the methods that deal with different combinations of variables in first and second stage have as initial point the Generalized Bender's decomposition.

Decomposition of the class B/* can be approached by a method that is an extension of the Generalized Bender's and was suggested by [9]. In this approach, the rows added to the problem iteratively as feasibility or optimality cuts are able to account for a binary first stage.

The class M/I is studied in the work by [12], extending the work on B/* to include continuous variables along with integers in the first stage. Nevertheless, the approach is inapplicable for the current thesis since it only supports pure integer second stage.

The work by [13] is a loose extension of Bender's decomposition and makes use of facetial search for the mixed integer variables. Theory of disjunctive programming provides a solid framework to tackle problems of the form C/M and B/M. For a detailed review of decomposition methods based on disjunctive programming the interested reader is referred to [14] as well as [7].

An interesting method that tries to provide hierarchical convexifications of the dual space is provided by [15]. The method can successfully approach solution of problems of the class B/M nevertheless, applications on large scale systems using this method are not frequent. As noted by [16], the method requires complex approximations of the problem space that are not practical for real life applications.

Finally, a practical method that uses approximations of the stages to simplify the optimization steps is provided by [17]. The method provides an approximation scheme that makes use of linear programming relaxations to solve stochastic integer problems in a practical manner. The method is tested in large scale applications.

4.3 Decomposing M/M

Problems with mixed integer variables in both stages pose the greatest challenge in terms of decomposition. The lack of strong duality theorem makes approaches based on Dantzig-Wolfe and Bender's decomposition inapplicable. Additionally, extensions of the methods that are able to tackle the decomposition as in [9] lose ground as soon a mixture of binary and continuous variables in the first stage are considered. In a survey by [12] the algorithms are presented with the option of including mixed variables in both stages but no practical application is presented in order to assess the computational feasibility of the methods that are based in disjunctive programming.

Furthermore, the method's results do not prove to be beneficial in large-scale programs due to the computational bottleneck connected to the intermediate stages of the algorithm. Extensive work presented by [13], [14] makes the case of practical use of disjunctions in order to deal with mixed integers.

However, the practicality of the method is questionable since the notions used are far away from straightforward and pose a great computational burden as problems assume great sizes. Additionally, since mixed integer programs are tied with an optimality gap due to the lack of duality support, convergence is not guaranteed to provide the optimal solution.

With the methods that descend from Dantzig-Wolfe practically being inapplicable in the field of mixed integer stochastic two stage programming, the path left is Lagrangian decomposition. As seen in the survey by [15], applications in literature have considered the use of Lagrangian

as the only alternative. That is due to the flexibility of the method to accommodate a variety of problems since the applicability is not restricted by assumptions on convexity or linearity.

A great portion of applications in the literature is devoted to the use and study of Lagrangian decomposition. Nevertheless, work on two stage stochastic programming problems has been scarce [15]. It is with great hope that this study will be able to provide a practical guide with extending the method of Lagrangian decomposition to the M/M case.

Considering two stage stochastic programs that incorporate scenarios to represent uncertainty, the general formulation can be captured below.

$$\begin{aligned} \min c^T x + \sum_{s \in S} p_s (d^T y) \\ \text{st: } Ax \leq b \\ \\ Wy \leq T \\ \\ x \in X, y \in Y \end{aligned}$$

In this form, the problem variables are separated between the scenario dependent variables x and the scenario independent variables y . The set S describes the collection of all scenarios. Sets X, Y describe a junction between continuous and binary, in the general case integer numbers.

In [6] the authors present Lagrangian decomposition as a relaxation of the complicating constraints, and as an extension the presented formulation emphasizes in the scenario decomposition of a model. In that sense, following the work by [16], expand the definition of the scenario independent variables to assume an outcome for every scenario $s \in S$ as follows.

$$x^i = \sum_{s \in S} p_s x_s^i$$

Where the superscript i , selects the i^{th} element of the vector x . Notice that the scenario independent variable x^i , under this formulation assumes a scenario dependent counterpart for all scenarios. It is important to mention that with this substitution the problem is now decomposable by scenario. Nonetheless, the duplicated scenario independent variables are

incoherent as they are free to assume any value across scenarios, effectively solving as many independent problems as the number of scenarios.

In order to gain sanity in the first stage and complete the formulation, the non-anticipativity constraints need to be introduced in the model. Non-anticipativity constraints are a set of constraints linking all the duplicated scenario independent variables to assume the same value across scenarios as follows.

$$x_s^i == x^i$$

Notice the non-anticipative is not constraint to assume only the following form. Different suggestions exist in literature that may prove beneficial depending on the problem. For simplicity the above form will be adopted.

The newly formulated decomposable problem assumes the following form.

$$\min \sum_{s \in S} p_s (c^T x_s + d^T y)$$

$$st: \begin{bmatrix} \hat{A} \\ W \end{bmatrix} \begin{bmatrix} x_s \\ y \end{bmatrix} \leq \begin{bmatrix} \hat{b} \\ T \end{bmatrix}$$

$$x_s^i == x^i, \forall i$$

$$x_s, x \in X, y \in Y$$

Where \hat{A}, \hat{b} are the initial values of the matrix A and vector b , duplicated across scenarios. It is important to note that this form of the problem so far is completely equivalent to the original form. The only difference is scenario independence of the first stage variables is only expressed via the non-anticipativity constraint.

The addition of the non-anticipativity constraint provides a practical way to decompose the full problem in the cost of the expansion of the dual space proportionally to the number of scenarios.

In order to decompose the problem into separable modules that can be solved in parallel the non-anticipativity constraint is relaxed and placed in the objective function under the penalty of the respective Lagrangian multipliers as follows:

$$\min \sum_{s \in S} p_s(c^T x_s + d^T y) + \sum_{s \in S, i=1 \dots n} \mu_s(x_s^i - x^i)$$

$$st: \begin{bmatrix} \hat{A} \\ W \end{bmatrix} \begin{bmatrix} x_s \\ y \end{bmatrix} \leq \begin{bmatrix} \hat{b} \\ T \end{bmatrix}$$

$$x_s, x \in X, y \in Y$$

At the optimal point, the Lagrangian penalty will vanish and the solution to the decomposed problem will effectively follow the original, should the variables belong to the continuous realm.

Considering the dual problem, the formulation leads to the maximization over the values of Lagrangian multipliers [6].

$$\varphi(\lambda) = \min_x \sum_{s \in S} p_s(c^T x_s + d^T y) + \sum_{s \in S, i=1 \dots n} \mu_s(x_s^i - x^i)$$

$$x_s, x \in X, y \in Y$$

$$Dual\ problem: \max_{\mu} \varphi(\mu)$$

An important realization is that at this stage if the values in both stages were continuous, steepest descent directions can be calculated by differentiating the Lagrangian and methods like Newton's descent algorithm may be used directly to reach optimality. In the case of M/M, focusing the Lagrangian is non-differentiable due to integrality constraints posed in both sets X, Y . In this case using direct optimization methods is an option but due to the size that the problem assumes in large-scale applications, such approach would be difficult to reach the optimal solution.

4.4 Sub-gradient optimization

In order to solve the dual problem, methods for non-differentiable optimization need to be employed. In [17], the supporting theory of sub-gradient optimization is introduced for the case of non-differentiable dual functions.

The theory provides methods that can minimize the dual problem by use of sub-differentials that eventually transform the algorithm into a variation of the steepest descent method. The

sense of direction in this context is given by the calculation of the sub-gradient and the step size is theoretically calculated but in reality in many cases is problem specific.

Assuming the dual problem $\varphi(\lambda)$ acts on a domain $D \in R^n$ and is convex, then a function sg , may be called a sub-gradient of φ at a point $\hat{\lambda} \in D$ if:

$$\varphi(\lambda) \geq \varphi(\hat{\lambda}) + sg(\lambda - \hat{\lambda}), \forall \lambda \in D$$

In the case where the domain is concave the function is defined to be the super-gradient of φ .

Borrowing from [6], the following figure demonstrates the essence of the sub-gradients.

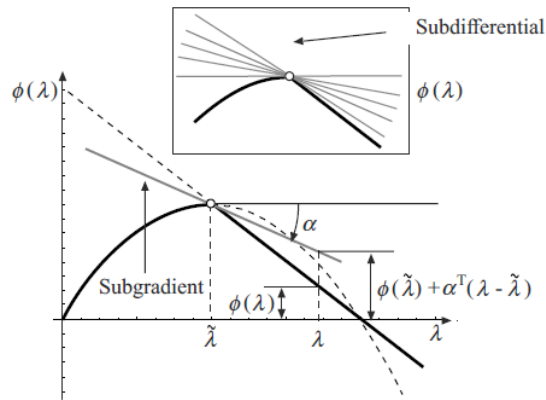


Figure 4-1 Sub-gradients of a non-smooth function

At any continuous point of the curve the definition of sub-gradient will follow the gradient definition. At discontinuity points there exist multiple sub-gradients. The collection of all possible sub-gradients at a point λ , is defined to be the sub-differential of the function at that point and is denoted by $\partial\varphi(\lambda)$. For example, as an example of sub-differential functions, the author in [17] uses the case where $\varphi(\lambda) = \|\lambda\|$ as shown below [17].

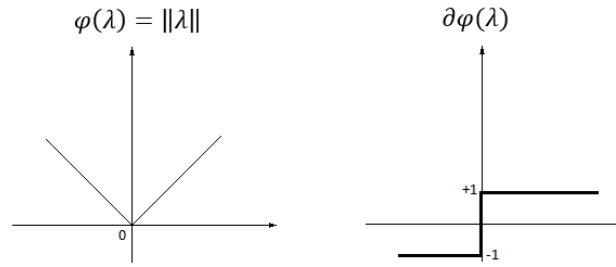


Figure 4-2 Sub-differential function for $\varphi(\lambda)=|\lambda|$

Observing the form of the dual problem, given some initial values for the multipliers the problem is solvable in the domain of the decision variables and also decomposable across scenarios. Once the solution for each scenario and the non-anticipativity part of the objective function has been found the solutions are aggregated and the values of the multipliers are updated. This process is repeated for multiple steps until certain metrics of convergence are met.

4.5 Phases of decomposition

Due to the lack of strong duality theorem Lagrangian decomposition of the M/M problems cannot prove that convergence will meet the optimal solution. The extensive work on decomposition techniques [6], lends the notation of phases that Lagrangian decomposition algorithms need to go through in cases of decomposing problems with integrality constraints in any of the two stages.

In the suggested notation, Phase 1 describes the process of the methods until sub-gradient optimization meets the pre-specified stopping criteria. Notice that the solution at this point need not be optimal in the sense that the full problem would require.

Taking into consideration the non-anticipativity constraints this notion transforms into the scenario independent variables that are duplicated across scenarios need not have equal values. A solution that is not consistent in this sense is not useful for the overall optimization problem and for that reason Phase 2 is employed. In Phase 2 the goal is to gain sanity in the first stage while maintaining close proximity to the optimal point reached in Phase 1.

Every application making use of Lagrangian decomposition in two stage stochastic problems is required to have both Phase 1 and Phase 2 since the final solution need to bare no artifacts of the decomposition procedure.

The steps for solving the decomposed Lagrangian problem can be summarized in the following steps.

Phase 1:

1. Assume initial values for the Lagrangian multipliers $\mu_s^0, s \in S$
2. Solve the scenario dependent sub-problems and the problem related to non-anticipativity. This step can be done in parallel; there is no communication between the sub-problems.
3. Gather the optimal solution values for each sub-problem and all the values of the optimal variables.
4. Check if the stopping criteria are met. If yes, stop go to step 1 of Phase 2.
5. Update the sub-gradient vector.
6. Update the step length.
7. Update the multipliers.
8. Go to step 2.

Phase 2

1. Incrementally update the sub-problems in order to converge to a single vector for the first stage scenario independent variables.
2. If stopping criteria are met, the process terminates. If not, go to step 1.

In the following section, a detailed exposition on the details of every step of the method will be given. Focus will be given on applicability and practicality rather than presentation of the theoretical proofs that support the method. Emphasis is aslo given on specific problems that may arise when applying some of the methods.

4.5.1 Initial guess of multipliers

In the first step of Phase 1 an initial guess on the multipliers is required. Based on the nature of the relaxed constraints this step may be an important one that defines the rate of convergence to the optimal solution.

In practice with trial and error a proper set of multipliers may be found. In the specific form where the only relaxed constraints are the non-anticipativity constraints, since the Lagrangian multipliers refer to relaxation of equality constraints, there is no bound that is required for the multipliers. That is not the case when the relaxed constraints are inequality constraints, in which case the multipliers need to be non-negative [17].

Once a set of multipliers is provided, by initial guess or by the updating procedure, the multipliers are then propagated into the sub-problems and are treated as constant parameters.

4.5.2 Solving the sub-problems

In step 2 of the method the sub-problems may be solved in parallel since all the variables are decomposable by scenario. The communication of information in this method is effectively realized by the values of the multipliers. In some application a normalization of the technology parameters related to the problem may be helpful to avoid dominance of variables with wide range.

In the case of M/M, all the sub-problems are mixed integer programs and the computational complexity remains in the class of NP-Hard. Nevertheless, the size of the problems is definitely smaller, hence the exponential explosion of complexity is avoided.

In [16] the integrality constraint on the binary variables of the sub-problems has been relaxed without important impact on the optimality or convergence of the solution. On that note it is interesting to point out that, similar relaxation techniques on integrality may give an opportunity for the addition of a second level of decomposition.

Depending on the size of the sub-problems at this point, relaxation of some of the integrality constraints may allow the use of simpler decomposition methods like Bender's or Generalized Bender's in presence of residual integer variables [18].

4.5.3 Gathering of distributed solutions

In the third step of the method, the solutions across sub-problems are aggregated into the main process of the algorithm. This step does not entail any interesting methodologies but remains important since it constitutes the algorithmic bottleneck of the method. Step 2 can be parallelized since the structure of the problem is embarrassingly parallel [19]. Step 3 on the other hand cannot be parallelized and it is important to notice and expect a high need for read access memory at this point of the program.

For problems of great size, step 3 may not be feasible to be completed on a single machine due to memory limitations and a distributed strategy of decentralized aggregation may need to be employed. In large-scale problems the physical computational needs of the algorithms used can limit the applicability of certain approaches and this becomes particularly evident in methods that require a high degree of memory storage as the cutting plane method that will be investigated.

4.5.4 Stopping criteria

The stopping criteria that can be used at step 4 of the method can assume many forms and are problem specific in a lot of areas. In the following some alternatives found in literature will be presented.

- Criteria based on bounds

If the optimum value of the Lagrangian dual is known, the difference of the current value of the Lagrangian from the optimum can be monitored. Setting a stopping threshold may employ a stopping criterion.

$$\text{if } \|L^* - L^k\| < e, \text{STOP}, e \sim 1\%$$

In practical applications the value of the optimum point is unknown a priori. It can be substituted by values of tight bounds on the objective. Pre-solving the linear relaxation of the full problem and storing the optimal value may acquire such bounds. In [17], the author suggests the use of an inflation parameter $0 < a < 2$ that can help provide successful upper bounds on the relaxed objective.

- Criteria based on norms

In a manner similar to the continuous case of steepest descent algorithms, the norm of the gradient is an indication of a local optimum. Under conditions of convexity this can be an assertion of global optimality. As an extension of this notion the norm of the sub-gradient can provide information about the state of convergence of the method. It is important to mention that since sub-gradient optimization is widely known to behave in an oscillating fashion special care is warranted when using this type of stopping criterion.

- Criteria based on number of iterations

Due to the oscillating nature of the algorithm, any methods that try to track a consistent minimization of a metric is sensitive to errors. In many practical applications it may be useful to set a maximum number of iterations that the algorithm should run to achieve a confidence level in the quality of the solution in Phase 1. In [16], the authors have pre-specified a maximum number of 100 iterations, after which Phase 1 is terminated and the result is given to Phase 2. Even though simple and practical, this method does not give any assurance on the state of the quality of the solution so a set of trial and error iterations need to run in order to discover the right amount of steps for each application.

4.5.5 Updating the sub-gradient

At step 5, the direction of descent is updated by collecting the optimal value of the decision variables from the sub-problems. In the formulation where the non-anticipativity constraints have been relaxed, it can be shown that the difference between the optimal values at step k between the scenario dependent variables and scenario independent variables is a valid sub-gradient for the problem [20], [6].

$$sg^k = x_s^k - \widehat{x}_k$$

Where the vector \widehat{x}_k is the values of the scenario independent variables at the current iteration duplicated to assume conformable size to the number of scenarios.

In step 3 of the method, this step does not contain any significant algorithmic methodology rather than posing a choking point of the algorithm since the collection of the values across scenarios requires significant run time memory when the application is large.

4.5.6 Updating the step length

At step 6, the length of the subsequent move in the dual space is calculated. Computational experience in non-linear optimization algorithms makes evident that the step length is an extremely sensitive element in the process.

In many of the approaches that will be presented, it is important to note that heuristic choices on parameters may be used. While in theory many of the approaches come with strong theoretical support that prove convergence in a finite amount of steps, in reality it is easy to deviate from the theoretical assumptions on optimality and the danger of over or under shooting is possible.

In [17], the author provides the theoretical support on convergence regarding step sizes. Considering at step k , the multipliers μ^k are not optimal. The value of step length that can guarantee that in step k the following inequality will hold $\|\mu^{k+1} - \mu^*\| < \|\mu^k - \mu^*\|$ should satisfy the following condition:

$$0 < \gamma^k < \frac{2(L(\mu^*) - L(\mu^k))}{\|sg^k\|^2}$$

Where $\|sg^k\|^2$ is the squared norm of the sub-gradient vector at step k . Theoretical support is easily lost when the optimal value of the problem, $L(\mu^*)$, is not known at step k . A close monitoring of the state of the step length can help understand the corrective actions that may need to be in place to fine tune the step length calculation.

A more general set of assumptions on the step lengths can also prove to lead to optimality. In [17] it is demonstrated that as long as one of the following conditions hold for the series of step sizes, the sub-gradient method will converge to the optimum.

- $\lim_{k \rightarrow \infty} \left\{ \sum_{i=0}^k (\gamma^i)^2 \right\} < \infty, \lim_{k \rightarrow \infty} \left\{ \sum_{i=0}^k \gamma^k \right\} = \infty$

- $\lim_{k \rightarrow \infty} \{\gamma^k\} = 0, \lim_{k \rightarrow \infty} \{\sum_{i=0}^k \gamma^i\} = \infty$

All step size rules that are applied in the field need to guarantee one of the conditions above in order to have an assurance that the vector of multipliers comes closer to the optimal at each step.

The following describes different approaches that focus on practicality of the calculation of the step length.

- Simple step length

The simplest form of the step length calculation comes from [6] as a parametric updating of the value with respect to the current step.

$$\gamma^k = \frac{1}{\alpha + \beta k}, \alpha, \beta \in R^+$$

This formulation allows for the successive lowering of the step length value while maintaining that the infinite sum of the values grows to infinity. For practical considerations, the parameters α , β , are problem specific and once more may be discovered heuristically. The drawback of this method is that it does not take under consideration the current state of the process and it is very easy to generate too long or too short step lengths. This phenomenon impacts the already difficult and oscillating nature of the process. In favor of this approach is the simplicity and practicality since it does not pose any great computational burden on the rest of the algorithm.

On a similar vein, in [21] the following step length updating methods are suggested.

$$\gamma^k = \frac{a}{\sqrt{k}}, a \in R^+$$

- Modified step length

Working directly with the convergence bound from [17], the realization that knowledge of the optimum value of the Lagrangian is unattainable, [22] suggests a practical manipulation of the past values of the Lagrangian as alternative. A weighing parameter α is introduced to update dynamically the upper bound in the calculation of the step length.

$$\gamma^k = \frac{\hat{L} - L(\mu^k)}{\|sg^k\|^2}$$

$$\hat{L} = a_r L^0 + (1 - \alpha)L^b$$

$$a_r = \begin{cases} \varepsilon_0, & \text{if } r \geq r_2 \\ e^{-0.69(r/r_1)^{3.2}} & \end{cases}$$

Where parameters ε_0, r_1, r_2 are determined by problem specific properties. This method alleviates the need for having prior knowledge of the optimum value of the Lagrangian and provides a dynamically updated step length that adapts to the oscillations of the process.

- Polyak step length

Working directly with the convergence bound, the Polyak step length is parameterizing the multiplication factor of 2 [23]. In this process the proper value of the parameter is left to the specific application.

$$\gamma^k = \frac{\lambda(L(\mu^*) - L(\mu^k))}{\|sg^k\|^2}, \lambda \in R^+$$

4.5.7 Updating the multipliers

The heart of the method lies in the updating mechanism of the Lagrangian multipliers. For this step one can follow an approach that resembles the steepest descent method or use more advanced methodologies that promise more stable behavior. In the following section the most notable approaches will be presented.

- Simple update

In the simple approach the new multipliers are calculated incorporating the information of the new sub-gradient direction and the step length.

$$\mu_{k+1} = \mu_k + \gamma_k s g_k$$

Following the Polyak step length, the updating formula for the multipliers would form as follows:

$$\mu_{k+1} = \mu_k + \frac{\lambda(L(\mu^*) - L(\mu^k))}{\|s g^k\|^2} s g_k, \lambda \in R^+$$

After updating, the new multipliers are used to calculate the new optimal point for all the sub-problems.

The most helpful advantages of this method are the ease of use and practicality since no significant computational effort is required for the update. On the other side of the spectrum, the method is known to suffer from oscillation causing the convergence of the method to perform in a zig-zag manner as seen in the figure below [22].

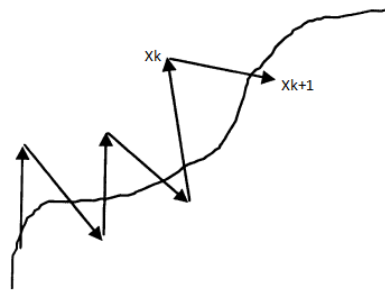


Figure 4-3 Oscillation of sub-gradient methods

One proposed heuristic approach to combat the oscillation in the method is to use weighted directions to calculate the update multipliers, thus including previous sub-gradient values in the process. A weight factor α can be introduced to accommodate the consolidation of directions as follows [24]:

$$\mu_{k+1} = \mu_k + \left\{ \frac{\lambda(L(\mu^*) - L(\mu^k))}{\|sg^k\|^2} \right\} \{a * sg_k + (1 - a) * sg_{k-1}\}, \lambda \in R^+, a \in [0,1]$$

- Cutting plane method

At every iteration the sub-gradients provide a polyhedral approximation of the dual problem [17]. In that sense working with the approximation, the cutting plane method at each iteration uses the information from all the past steps to provide a new and improved upper value for the dual problem.

In this method, information about the step length is not needed as the new multipliers for step k+1 are provided by the solution of the following linear problem [6] [17].

$$\begin{aligned} & \min w \\ & \text{st: } w \leq L(x^1) + sg_1(\mu^{k+1} - \mu^1) \\ & \quad w \leq L(x^2) + sg_2(\mu^{k+1} - \mu^2) \\ & \quad \dots \\ & \quad w \leq L(x^k) + sg_k(\mu^{k+1} - \mu^k) \\ & \quad \underline{\mu^{k+1}} \leq \mu^{k+1} \leq \overline{\mu^{k+1}} \end{aligned}$$

The method utilizes all previous knowledge of direction as it generates the approximation of the dual problem. The lower and upper bounds of the multipliers are dictated by the nature of the constraints. In the case of the non-anticipativity, the equation constraints allow the multipliers to be free.

Applications that have used the cutting plane method in different problems to solve the dual problem report faster convergence with diminished effects of oscillation [25]. The defect of the method is the need for storing all previous information about the sub-gradients and the multipliers. As the number of iterations increase, the linear problem required to provide the new multipliers becomes increasingly large and difficult to

tackle. Also memory management becomes a limiting factor for practical large-scale applications as the information is usually kept in memory, limiting the available resources for the rest of the method.

In their work, [25], suggest a modification of the method where only a subset of the previous step is used to calculate the new multipliers. As a preprocessing step, the suggested approach validates the importance of every residual past hyperplane and removes the ones that do not prove to provide useful bounds on the form of the approximated dual problem.

- Bundle method

On a similar tangent with the cutting plane method, the bundle method also provides a polyhedral approximation of the dual problem by the solution of a quadratic problem. The method includes a stabilizing factor in order to diminish the oscillations in a larger degree.

The method maintains the drawbacks of the cutting plane method in terms of applicability, due to the need of maintaining past information in memory. Additionally, the method's benefits are drawn at large by the quadratic factor of stabilization that acts as the center of gravity for the multipliers [17].

5 Power systems engineering

5.1 Introduction to electricity market

In the following chapters, the application field of power engineering will be presented. The field of stochastic forecasting and scheduling has been a key point of research for years in the field of power system engineering and electricity economics. The electricity market is a highly competitive and dynamic field of operations where decision makers are faced with extremely large optimization and decision problems in order to operate the network in an optimal manner. Enhancing the computational efficiency via decomposition has always been of the main points of research and in fact a great deal of new methods have stemmed from the field.

5.2 Entities of the market

In order to study the computational models that describe the problem, it is instructive to present the architecture of the electricity market in the United States and specifically the details of the market in California, where the datasets used in this study have been taken from.

The electricity market comprises of three main entities. In a high level, the market involves the producers, the consumers and the managing or regulating authorities. The inclusion of a managing authority enhances the classic producer/consumer paradigm with increased safety to avoid the boundary conditions of the system (starvation of customers/saturation caused by overproduction), and also to increase the efficiency of the process.

In non-technical aspect, the managing authority enforces regulations and promotes transparency of the operations to increase the social benefit. In the figure below a high level depiction of the system is shown.



Figure 5-1 Entities of the energy market

5.2.1 Consumers

On the consumer side, residential and industrial consumers are considered. Residential demand includes power delivered to houses and housing complexes and is usually consolidated through private trading agencies. Industrial consumers are manufacturing plants that rely on the electricity grid for their needs.

Both entities, residential and industrial, build a characteristic pattern on consumption rate. Even though the final load curve is stochastic, it is valid to assume that loads follow specific trends during the course of a day. This consistency on load demand is important in the electricity markets because successful forecasting of future demand leads to more efficient scheduling and minimal waste in the network.

Industrial consumers form what is known to be the base load as the need for industrial plants can be scheduled based on the operational planning of the enterprise where load is not anticipated to fluctuate rapidly.

Additionally, industrial consumers due to the extensive need for power consumption, form the majority of the total demand. Finally, due to the nature of the machinery used in industrial processes, industrial consumers need to undergo a process called cosine correction in order to reduce the amount of reactive power circulated in the network.

In certain markets industrial customers are bound to specific cosine thresholds in order to connect to the transmission grid. Residential consumers form the peak of the demand curve, which adds extra load on the base. Residential demand is often characterized by lower volume of required power output but with higher degree of fluctuation. Seasonal characteristics and whether phenomena can play a significant role in forming the demand curve in real time.

Based on historical records on load characteristics, it is feasible to forecast specific load zones that will be required. Beyond this safety threshold a great degree of stochasticity takes place in determining a close approximation to the actual future demand. In the following figures, the historical data from the ISO in New England show the characteristic curves based on patterns focused on the differences between a characteristic summer day and a characteristic winter day [26].

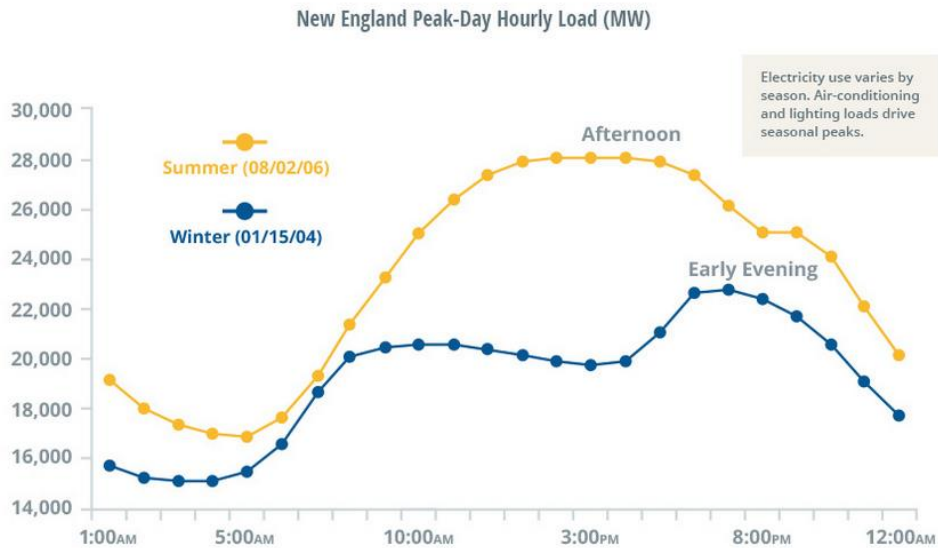


Figure 5-2 Model days of load for summer and winter from the NE-ISO

As shown by the figure, due to the extended use of air-conditioning during the summer, from 10AM until early evening, the summer day shows a distinct difference on the trend with comparison to the winter day.

5.2.2 Producers

On the production side, private and state agencies are tasked with power generation and distribution in the transmission network. The architecture of the power production line is fairly intricate as it is a heavily regulated area. Many agents both private and public help shape the final profile.

Power generation is the efficient transformation of various types of energy into electric energy. In most cases, thermal energy is transformed into electric energy with the use of turbines that drive electromagnetic generators. Wind power generation transforms the kinetic energy of the wind into electricity. In a similar manner it is possible to harvest the kinetic energy of tidal sea waves and transform the change of momentum into electric energy [27]. The photovoltaic effect makes possible the transformation of the energy of the sun into a steady flow of electricity.

Traditionally, the power production field consists of a mix of generating units, making use of different production technologies and natural resources. Generating units may be separated in two major categories based on the nature of the resources used. Non-renewable fossil resources like coal, oil and natural gas have been the traditional way of power production.

With the advent of technology of renewable energy generation, energy from the sun, wind, sea motion and geo-thermal energy penetrate the energy market in an increasingly higher degree every year. The increased risk related to the operation of nuclear resources, drives nuclear power generation to play a lesser part in the global map of power production.

Depending on the source used for power production, generating units exhibit specific characteristics and limitations. The most important facts about generators deal with the maximum (peak) power output, the minimum power output, the flexibility of the unit also known as ramp capability, start-up and shut-down times and operating costs. With increased environmental awareness, characteristics on environmental pollution also become increasingly relevant, as power generation is a major contributor in a country's carbon footprint.

In the recent years there has been motivation to penalize the excessive emission of greenhouse gases and as an effect of this movement, many countries have active carbon taxes, to set an incentive on greener solutions. Under this spectrum, specifications on pollution characteristics of generating units become extremely important in planning and scheduling. In that sense, the weight falls on the fossil fueled generators since renewable resources do not introduce pollutants into the atmosphere.

In the following graph the breakdown of power generation sources across the globe is shown [28].

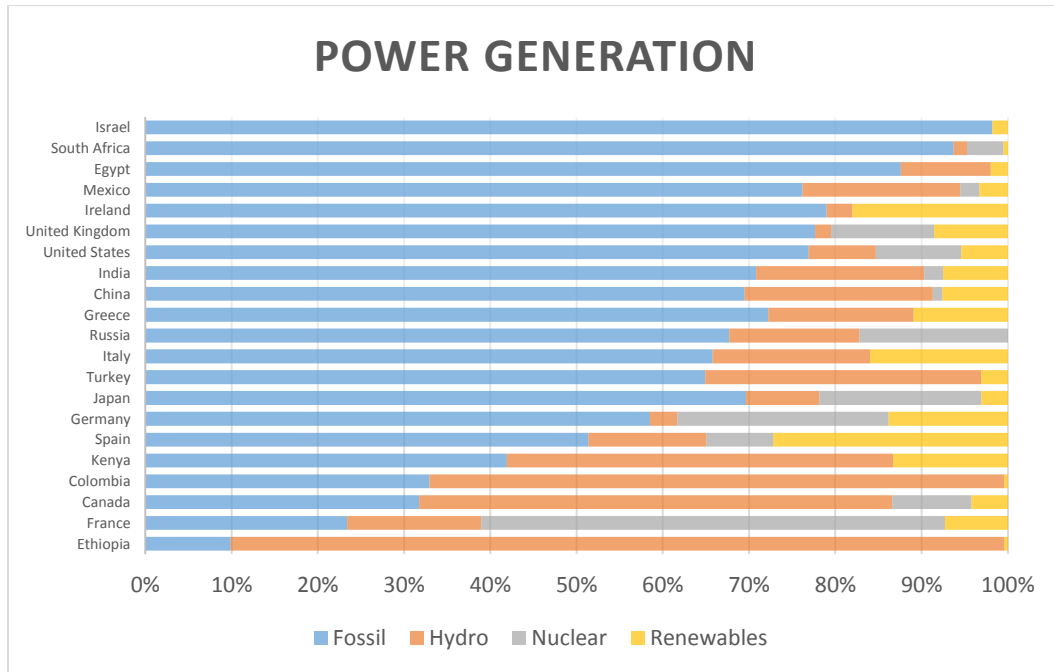


Figure 5-3 Breaddown of power sources per country

Countries like Canada and Ethiopia have managed to capture the majority of their power needs through hydroelectric generation. In other countries like the United States the graph shows a mix of resources with fossil fuel taking the majority share. Almost every country has a degree of penetration for modern renewable resources like wind and solar.

In the following section some of the major categories of power production units are presented.

- Coal and Oil power plants

Traditional fossil sources of energy production, coal and oil are used to transform chemical energy into kinetic energy via the generation of water steam that drives power generation turbines. These kinds of generators have been the traditional source of power. Usually making up for the largest portion of the produced electricity, fossil based generation units are usually bound by low flexibility on ramping and slow start-up periods. The operating cost of the units is relatively the cheapest in comparison with other non-renewable methods. The environmental impact is high with a high output of toxic emissions into the atmosphere. Traditionally, these generators are used to face the daily base load since ramping capability is low. In recent years energy producers are

making a shift away from fossil fueled generators due to the high environmental impact, low generation efficiency in the coal and oil cycle, and the advance in efficiency of cleaner sources of energy.

- Hydroelectric power plants

Hydroelectric power plants are transforming kinetic energy of streams of water into electric energy by driving water turbines. Water dams are specifically constructed to regulate the movement of the water according to seasonal patterns of rainfall and energy requirements. This source of energy is classified as renewable since no resources other than natural water movement are required for generation of electricity. Hydroelectric generators can participate in the network as base load facing units given a steady flow of energy on regular periods or as high rate reserve resources in cases where the power plant has been designed with water reserves [29]. The use of this method is restricted by the natural resources available. Usually countries or areas with large riverbeds and high rainfalls can integrate hydroelectric generation in a large degree in the market.

- Natural gas power plants

Natural gas generators harvest the chemical energy generated by internal combustion of the gas and drive power generation turbines. These units are characterized with fast ramp capabilities and low start-up periods. Traditionally, natural gas plants have been used for emergency energy requirements and real time network adjustments. In recent years many networks use natural gas units as the main source of generation as natural gas becomes more available. For example in 2004, the participation of natural gas plants in the total generation in New England had been 15% only to rise to 44% in 2014 [26]. These units also operate with high efficiency and lower emission rates compared to other fossil fueled sources. Nevertheless, the production price with natural gas is relatively more expensive than with coal or crude oil production.

- Wind power plants

Wind generators transform the kinetic energy of the wind to electric. Along with solar units, wind generation units are the cleanest forms of energy production since they impose no environmental impact. Additionally, operating costs are practically negligible compared with other non-renewable sources of energy, making wind power production a cost effective alternative to energy production. The drawbacks in wind production come from the intermittent behavior, since the accuracy in weather patterns and wind forecasts is low. Furthermore, the power output of wind generators is low. Recent advances have led to increased production efficiency of wind turbines and slowly this form of power production is gaining a presence in the electricity market. Across the United States, Europe and other countries large-scale wind farms increase the overall power output, thus introducing a clean and inexpensive form of electricity. Of course this comes at the cost of introducing a great level of stochasticity in the network since the variability of wind can impact the scheduling of units.

5.2.3 Managing authorities

Managing authorities are tasked with the operation of the electricity network. The entire process of the system is under the authority of these independent agencies that act on a regional level, usually overlooking statewide and in some case multi-state operations.

The Independent System Operators (ISO's), facilitate the efficient operation of the energy market with processes involving forecasting demand and managing production and demand while maintaining high levels of transparency that guarantees social accountability. In the United States most ISO's are under the direct authority of the Federal Energy Regulatory Commission (FERC), while some remain without this clearance.

The current map of the ISO's in the United States is shown in the figure below. In many cases, smaller scale public authorities that do not participate in the network co-exist in many states, maintaining the remainder of the traditional decentralized form of the network.

There Are Nine ISOs and RTOs in North America

ISO New England covers the six states of Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont.

- California ISO
- Alberta Electric System Operator
- Electricity Reliability Council of Texas
- Southwest Power Pool
- Midcontinent ISO
- Ontario Independent Electricity System Operator
- PJM Interconnection
- New York ISO
- ISO New England

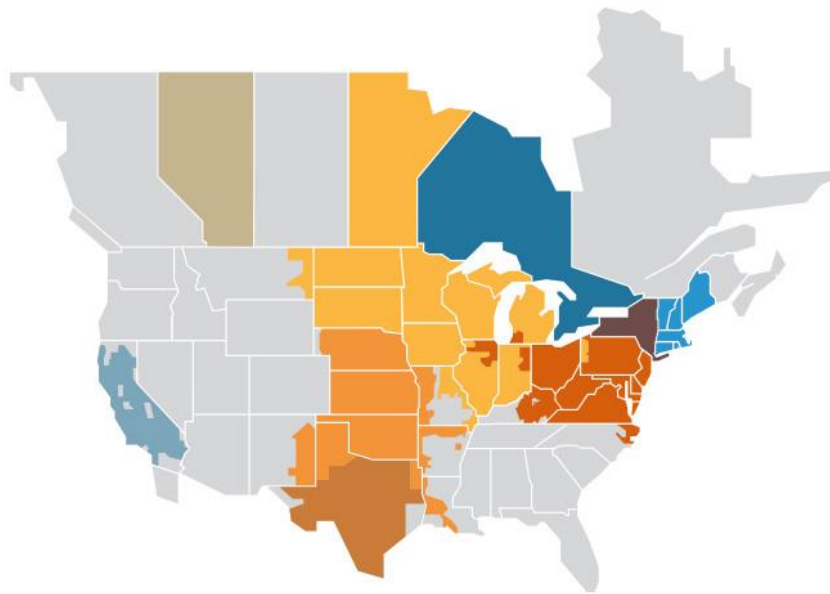


Figure 5-4 Map of ISOs in the US

The scheduling and operation of the generating units in order to meet daily demands is the main task of interest for this thesis. This is a complex process that involves large-scale models that are used to determine scheduling itineraries, market prices and safety levels in the network.

5.3 Operational overview

The operational timeline of the electricity market is complex. Energy as a commodity poses specific restrictions because at any given time the net amount of production must exactly meet demand. Other commodities give producers the ability to stock extra production but this is not the case with energy production. In the case of overproduction, the excess amount of energy is wasted. In order to plan efficiently for each day, managing authorities need to forecast demand curves and facilitate the communication between production and consumption.

The timeline of the operation of the market can be simplified into two stages. In the day-ahead market (DAM), decisions on scheduling of power production units are taking place based on forecasted demand curves. The market is cleared and this planning phase ensures that all demand needs to be met. Every day certain safety margins around the forecasted curves are set

and producers provide reserve services in the network in order to account for the readability of the system to face unexpected peaks. The DAM in most ISO's is cleared around noon the day before the transactions take place.

In the second stage, the ISO manage the real time realization of demand and need to adjust the specified schedule that was set in the DAM.

The time line is depicted as described by the California ISO in the figure below.

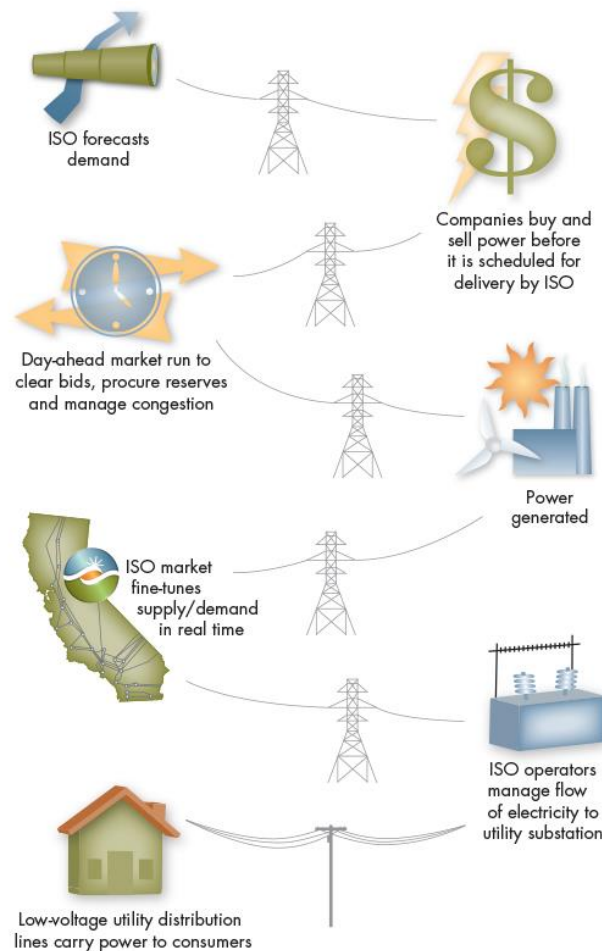


Figure 5-5 Timeline of the energy market

The operational safety margins are also scheduled and planned by the ISO's. Safety thresholds are set in order to avoid brown and black outages by ensuring that the operating capacity of the scheduled generators can withstand sudden peaks in demand or unscheduled unit failures

that can take scheduled generators offline at random times. The scheduling of spinning and non-spinning reserves by the ISO's does this.

Spinning reserves make sure that the power output of the scheduled units maintain some safe capacity threshold so units that are already supplying power can accommodate extra load without reaching the maximum output. In the California ISO, the spinning reserves are defined as the ability of online generators to ramp up the power output within 10 minutes from the dispatch instruction.

On the other hand, non-spinning reserves set an offline capacity threshold for generators. This effectively sets generators in a "ready" state, as they should be able to face demand within a time threshold, going from offline to online.

At this point it is useful to make the connection between the energy sources presented above under the spectrum of their response time. Traditionally slower generators, like coal power plants, are difficult to be used to provide time critical ramping of their power output. In [29], the Greek energy market is presented in which, gas turbines are used primarily to provide safety in the network by covering unexpected load fluctuations in short periods of time. In the problem that is relevant to this thesis the same separation of units exists making a clear distinction between slow generating units and fast generating units that have more flexibility and participate more actively in the reserves market.

The reserves side of the market plays an important role in the quality of service and the general social impact of the operation of the network. A robust, constant flow of electricity makes sure that the network is stable and that customers will not have to face outages. Nevertheless, depending on the utility of a customer it may prove impractical to meet demand for certain customers so load shedding is also part of the planning in the real time market. This is a planned action and not a result of the network being saturated.

The uncertainty in the demand side makes the scheduling of the network very demanding and dynamic. Other sources of uncertainty can refer to physical failures of the system due to generators faults or weather phenomena that can impact the transmission network. At some extent this random source of uncertainty needs to be included in the planning phase from the

ISO's. In this study the impact of equipment failures or fluctuation in the demand side will not be considered.

The inclusion of renewable sources of energy in the electricity network inadvertently adds an extra source of uncertainty in the planning effort. Accurately predicting the power output of wind or solar farms is a tedious task that can only be captured to a certain extent. These sources of energy have an intermittent nature since they are strongly correlated with weather patterns. As renewables take larger portions of the market with the increase on incentives for cleaner energy, the importance of robust stochastic models that can guide decision makers to clear the markets becomes ever so demanding.

In an effort to increase the ability of stochastic models to create robust schedules, there has been an extensive effort in the literature to generate wind and weather predictive models that can be used to simulate the performance of wind and solar farms. Historical data is used to gain insight on the statistical characteristics of wind patterns.

One of the main approaches in the field is the representation of the stochasticity of the wind as a Markovian phenomenon. In this sense, historical data is segmented based on periods of the day and it is possible to reproduce statistically accurate "model days" that can be used to enhance the robustness of stochastic models [30].

5.4 Stochastic market clearing under wind uncertainty

Following the formulation of the stochastic market clearing problem by [31], the mathematical program of the full problem will be presented in this section. For a clear definition of the problem the reader will be referred to [1] where the penetration of wind farms in the energy market is introduced. Nevertheless, [31] provides enhancements on the original problem with provisions on non-spinning reserves that make the approach more realistic and useful to the decision maker.

In the objective function of the problem, the net benefit of the operation of the network is calculated. The operating costs of the generators and the costs for maintaining reserves constitute the total cost, which is subtracted from the total utility of serving the daily load. The objective tries to maximize the total benefit earned in a day, or to minimize the negative benefit.

As described above, the problem involves the clearing of two markets. The Day Ahead Market (DAM) and the Real Time or Balancing Market (BM). The DAM is scenario independent as the decisions must be made without prior knowledge on the realization of the stochastic variable, in this problem, the wind production. The BM is shaped to encapsulate the effect of the stochasticity and the BM variables are scenario dependent.

5.4.1 Model elements

The following notation is used in the model to represent decision variables and constants that are considered input to the problem.

Table 5-1 Indices used in SMC model

<i>Symbol</i>	<i>Description</i>	<i>Domain</i>
h	Index referring to the slow generating units, $h \in [1, N_H]$, where N_H is the total number of slow units present in the network.	\mathbb{Z}
i	Index referring to the fast generating units, $i \in [1, N_I]$, where N_I is the total number of fast units present in the network.	\mathbb{Z}
j	Index referring to the loads present in the consumption side of the network, $j \in N_J$, where N_J is the total number of loads.	\mathbb{Z}
k	Index referring to the wind farms present in the network, $k \in [0, N_K]$, where N_K is the total number of wind farms.	\mathbb{Z}
n, m	Indices referring to the transmission buses of the network.	\mathbb{Z}
s	Index referring to the scenarios of the model, $s \in [1, S]$, where S is the total number of scenarios available.	\mathbb{Z}
t	Index refers to the time increments of the system, $t = [1, 24]$, representing a single day in planning.	\mathbb{Z}

In order to improve readability, the notation taken from [31] has been slightly modified according to the following rules:

- Decision variables are noted with boldface letters, i.e. \mathbf{Su}_{ht} , while constants and input data are noted with regular letters, i.e. Su_{ht} .

- Decision variables that belong in the DAM, the first stage of the model, begin with capital letter, i.e. Su_{ht} , while variables that refer to the second stage begin with lower letter, i.e. su_{ht} .
- The indices, presented in Table 5-1, can be used to distinguish between similarly named variables and constants in order to reduce the number of distinct symbols required to describe similar notions, i.e. Su_{ht} refers to the start-up variables of the slow units in the first stage, while Su_{it} refers to the start-up variables of the fast units in the first stage.

With the notation assumptions above, the families of symbols will be presented below.

Table 5-2 Cost coefficients of the SMC problem

<i>Symbol</i>	<i>Description</i>	<i>Domain</i>
Su_{xt}	Start-up cost coefficient for generating units at time t . Note that $x \in \{h, i\}$, referring to the slow and the fast units.	\mathbb{R}
Sd_{xt}	Shut-down cost coefficient for generating units at time t , $x \in \{h, i\}$.	\mathbb{R}
PC_{xt}	Production cost coefficient for generating units at time t , $x \in \{h, i\}$.	\mathbb{R}
Ru_{xt}	Ramp-up cost coefficient at time t , $x \in \{h, i, j\}$.	\mathbb{R}
Rd_{xt}	Ramp-down cost coefficient at time t , $x \in \{h, i, j\}$.	\mathbb{R}
U_{jt}	Utility of loads at time t .	\mathbb{R}
mc_{xts}	Marginal cost at time t , for scenario s , $x \in \{h, i, j\}$.	\mathbb{R}
amc_{jts}	Adjusted marginal cost of load j at time t , for scenario s .	\mathbb{R}
$VOLL_j$	Value of lost load, for load j .	\mathbb{R}
ξ_k	Wind spillage cost, for wind farm k .	\mathbb{R}

The following table defines the notation used for the coefficients of the constraints used in the model.

Table 5-3 Constraint coefficients of SMC

Symbol	Description	Domain
$\underline{P}_x, \overline{P}_x$	Minimum and maximum production bounds for generation units, $x \in \{h, i\}$.	\mathbb{R}
\overline{P}_{jt}	Maximum energy required by load j , at time t .	\mathbb{R}
\overline{P}_{kt}	Maximum energy output of wind farm k , at time t .	\mathbb{R}
\overline{Ru}_{xt}	Maximum upward spinning reserve at time t , $x \in \{h, i, j\}$.	\mathbb{R}
\overline{Rd}_{xt}	Maximum downward spinning reserve at time t , $x \in \{h, i, j\}$.	\mathbb{R}
\overline{Rn}_{it}	Maximum non-spinning reserve for fast unit i , at time t .	\mathbb{R}
$B(n, m)$	Transmission network susceptance.	\mathbb{R}
\overline{T}	Transmission bus, maximum capacity.	\mathbb{R}
$\overline{\rho}_x$	Maximum ramp up bound for generating units, $x \in \{h, i\}$.	\mathbb{R}
$\underline{\rho}_x$	Maximum ramp down bound for generating units, $x \in \{h, i\}$.	\mathbb{R}
μ_{x0}^U	Initial minimum up-time for generating units, $x \in \{h, i\}$.	\mathbb{Z}
μ_{x0}^D	Initial minimum down-time for generating units, $x \in \{h, i\}$.	\mathbb{Z}
μ_x^U	Middle periods, minimum up-time for generating units, $x \in \{h, i\}$.	\mathbb{Z}
μ_x^D	Middle periods, minimum down-time for generating units, $x \in \{h, i\}$.	\mathbb{Z}
wp_{kts}	Wind production of wind farm k at time t , for scenario s .	\mathbb{R}

The following tables present the notation used for the decision variables of the model. Table 5-4 presents the first stage decision variables while Table 5-5 presents the second stage decision variables.

Table 5-4 First stage decision variables for the SMC

Symbol	Description	Domain
Su_{xt}	Start-up variable for generating units at time t , $x \in \{h, i\}$.	$\{0,1\}$
Sd_{xt}	Shut-down variable for generating units at time t , $x \in \{h, i\}$.	$\{0,1\}$
Ct_{xt}	Commitment variable for generating units at time t , $x \in \{h, i\}$.	$\{0,1\}$
P_{xt}	Power output of generating units at time t , $x \in \{h, i\}$.	\mathbb{R}
Ru_{xt}	Upward spinning reserve for generating units at time t , $x \in \{h, i\}$.	\mathbb{R}
Rd_{xt}	Downward spinning reserve for generating units at time t , $x \in \{h, i\}$.	\mathbb{R}
Rn_{it}	Non-spinning reserve for fast generating unit i at time t .	\mathbb{R}
P_{jt}	Energy scheduled for load j at time t .	\mathbb{R}
Ru_{jt}	Upward reserve scheduled for load j at time t .	\mathbb{R}
Rd_{jt}	Downward scheduled for load j at time t .	\mathbb{R}
Δ_{xt}	Voltage angle of transmission buses, $x \in [n, m]$	\mathbb{R}

Table 5-5 Second stage decision variables for the SMC

Symbol	Description	Domain
su_{it}	Start-up variable for fast generating unit j , at time t and scenario s .	$\{0,1\}$
sd_{it}	Shut-down variable for fast generating unit j , at time t and scenario s .	$\{0,1\}$
ct_{it}	Commitment variable for fast unit j , at time t and scenario s .	$\{0,1\}$
p_{xts}	Power output of generating units at time t and scenario s , $x \in \{h, i\}$.	\mathbb{R}
ru_{xts}	Upward spinning reserve for generating units at time t and scenario s , $x \in \{h, i\}$.	\mathbb{R}
rd_{xts}	Downward spinning reserve for generating units at time t and scenario s , $x \in \{h, i\}$.	\mathbb{R}

Table 5-6 Continued

rn_{xts}	Non-spinning reserve for generating units at time t and scenario s , $x \in \{h, i\}$.	\mathbb{R}
p_{jts}	Energy scheduled for load j at time t and scenario s .	\mathbb{R}
ru_{jts}	Upward reserve scheduled for load j at time t and scenario s .	\mathbb{R}
rd_{jts}	Downward scheduled for load j at time t and scenario s .	\mathbb{R}
u_{jts}	Load shedding on load j at time t and scenario s .	\mathbb{R}
ws_{kts}	Wind spillage for wind farm k at time t and scenario s .	\mathbb{R}
δ_{xts}	Voltage angle at time t and scenario s , $x \in [n, m]$	\mathbb{R}

After relaxation the following Lagrangian multiplier terms are added in the problem.

Table 5-7 Lagrangian multipliers for the SMC

Symbol	Description	Domain
u_{xts}^{Su}	Lagrangian multiplier for the start-up variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^{Sd}	Lagrangian multiplier for the shut-down variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^{Ct}	Lagrangian multiplier for the commitment variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^{Ru}	Lagrangian multiplier for the upward spinning reserve variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^{Rd}	Lagrangian multiplier for the downward spinning reserve variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^{Rn}	Lagrangian multiplier for the non-spinning reserve variables, $x \in \{h, i\}$.	\mathbb{R}
u_{xts}^A	Lagrangian multiplier for the voltage angle variables, $y \in \{n, m\}$.	\mathbb{R}

Following the above notation, the Stochastic Market Clearing problem can then be formulated as follows.

$$SMC: \min c^T \mathbf{x} + \sum_{s \in S} p_s (d^T \mathbf{y} + w^T \mathbf{x}) =$$

$$\sum_{t=1}^{24} \sum_{h=1}^{N_H} \{ SuC_{ht} * \mathbf{Su}_{ht} + PC_{ht} * \mathbf{P}_{ht} + SdC_{ht} * \mathbf{Sd}_{ht} + RuC_{ht} * \mathbf{Ru}_{ht} + RdC_{ht} * \mathbf{Rd}_{ht} \} + \quad [a]$$

$$\sum_{t=1}^{24} \sum_{i=1}^{N_I} \{ SuC_{it} * \mathbf{Su}_{it} + PC_{it} * \mathbf{P}_{it} + SdC_{it} * \mathbf{Sd}_{it} + RuC_{it} * \mathbf{Ru}_{it} + RnC_{it} * \mathbf{Rn}_{it} + RdC_{it} * \mathbf{Rd}_{it} \} + \quad [b]$$

$$\sum_{t=1}^{24} \sum_{j=1}^{N_J} \{ RuC_{jt} * \mathbf{Ru}_{jt} + RdC_{jt} * \mathbf{Rd}_{jt} - U_{jt} * \mathbf{P}_{jt} \} + \quad [c]$$

$$\sum_{s=1}^S p_s * \left\{ \sum_{t=1}^{24} \left[\sum_{h=1}^{N_H} (mc_{hts} * \mathbf{r}_{hts}) \right. \right. \quad [d]$$

$$+ \sum_{\substack{i=1 \\ N_j}}^{N_i} (Suc_{it} * (\mathbf{su}_{its} - \mathbf{Su}_{it}) + Sdc_{it} * (\mathbf{sd}_{its} - \mathbf{Sd}_{it}) + mc_{its} * \mathbf{r}_{it}) \quad [e]$$

$$+ \sum_{j=1}^{N_j} (mc_{jts} * \mathbf{r}_{jts} + amc_{jts} * (\mathbf{ru}_{jts} - \mathbf{rd}_{jts}) + VOLL_j * \mathbf{ll}_{jts}) \quad [f]$$

$$\left. + \sum_{k=1}^{N_K} \mathbf{ws}_{kts} * \xi_k \right\} \quad [g]$$

such that:

First stage constraints:

$$\underline{P}_h * \mathbf{Ct}_{ht} \leq \mathbf{P}_{ht} + \mathbf{Ru}_{hts} - \mathbf{Rd}_{hts} \leq \overline{P}_h * \mathbf{Ct}_{ht}, \forall h, t \quad [1]$$

$$\underline{P}_i * \mathbf{Ct}_{it} \leq \mathbf{P}_{it} + \mathbf{Ru}_{its} + \mathbf{Rn}_{its} - \mathbf{Rd}_{its} \leq \overline{P}_i * \mathbf{Ct}_{it}, \forall i, t \quad [2]$$

$$0 \leq \mathbf{P}_{jt} < \overline{P}_{jt}, \forall j, t \quad [3]$$

$$0 \leq \mathbf{P}_{kt} < \overline{P}_{kt}, \forall k, t \quad [4]$$

$$0 \leq \mathbf{Ru}_{ht} \leq \overline{Ru}_{ht}, \forall h, t \quad [5]$$

$$0 \leq \mathbf{Rd}_{ht} \leq \overline{Rd}_{ht}, \forall h, t \quad [6]$$

$$0 \leq \mathbf{Ru}_{it} \leq \overline{Ru}_{it}, \forall i, t \quad [7]$$

$$0 \leq \mathbf{Rd}_{it} \leq \overline{Rd}_{it}, \forall i, t \quad [8]$$

$$0 \leq \mathbf{Rn}_{it} \leq \overline{Rn}_{it}, \forall i, t \quad [9]$$

$$0 \leq \mathbf{Ru}_{jt} \leq \overline{Ru}_{jt}, \forall j, t \quad [10]$$

$$0 \leq \mathbf{Rd}_{jt} \leq \overline{Rd}_{jt}, \forall j, t \quad [11]$$

$$\sum_{h=1}^{N_H} \mathbf{P}_{ht} + \sum_{i=1}^{N_I} \mathbf{P}_{it} + \sum_{K=1}^{N_K} \mathbf{P}_{kt} - \sum_{j=1}^{N_J} \mathbf{P}_{jt} - \sum_{n,m \in \Psi} B(n, m) * (\Delta_{nt} - \Delta_{mt}) = 0, \forall t \quad [12]$$

$$-\overline{T} \leq B(n, m) * (\Delta_{nt} - \Delta_{mt}) \leq \overline{T}, \forall n, m \in \Psi, t \quad [13]$$

$$-\pi \leq \Delta_{nt} \leq \pi, \forall n \in \Psi, t \quad [14]$$

$$-\pi \leq \Delta_{mt} \leq \pi, \forall m \in \Psi, t \quad [15]$$

$$(\mathbf{Ct}_{h,t-1} - \mathbf{Ct}_{ht}) + \mathbf{Su}_{ht} - \mathbf{Sd}_{ht} = 0, \forall h, t \quad [16]$$

$$(\mathbf{Ct}_{i,t-1} - \mathbf{Ct}_{it}) + \mathbf{Su}_{it} - \mathbf{Sd}_{it} = 0, \forall i, t \quad [17]$$

$$\sum_{t=1}^{\mu_{h0}^U} (1 - \mathbf{Ct}_{ht}) = 0, \forall h \quad [18]$$

$$\sum_{t=1}^{\mu_{h0}^D} \mathbf{Ct}_{ht} = 0, \forall h \quad [19]$$

$$\sum_{q=t}^{t+\mu_h^U-1} \mathbf{Ct}_{hq} \geq \mu_h^U * \mathbf{Su}_{ht}, \forall t = (\mu_{h0}^U + 1), \dots, (\mu_h^U + 1), \forall h \quad [20]$$

$$\sum_{q=t}^{t+\mu_h^D-1} (1 - \mathbf{Ct}_{hq}) \geq \mu_h^D * \mathbf{Sd}_{ht}, \forall t = (\mu_{h0}^D + 1), \dots, (\mu_h^D + 1), \forall h \quad [21]$$

$$\sum_{q=t}^{24} (\mathbf{Ct}_{hq} - \mathbf{Su}_{ht}) \geq 0, \forall t = (25 - \mu_h^U), \dots, 24, \forall h \quad [22]$$

$$\sum_{q=t}^{24} (1 - \mathbf{Ct}_{hq} - \mathbf{Sd}_{ht}) \geq 0, \forall t = (26 - \mu_h^D), \dots, 24, \forall h \quad [23]$$

Second stage constraints:

$$\underline{P}_h * \mathbf{Ct}_{ht} \leq \mathbf{p}_{hts} \leq \overline{P}_h * \mathbf{Ct}_{ht}, \forall h, t, s \quad [24]$$

$$\underline{P}_i * \mathbf{Ct}_{it} \leq \mathbf{p}_{its} \leq \overline{P}_i * \mathbf{Ct}_{it}, \forall i, t, s \quad [25]$$

$$\mathbf{p}_{hts} = \mathbf{P}_{ht} + \mathbf{ru}_{hts} - \mathbf{rd}_{hts}, \forall h, t, s \quad [26]$$

$$\mathbf{p}_{its} = \mathbf{P}_{it} + \mathbf{ru}_{its} + \mathbf{rn}_{its} - \mathbf{rd}_{its}, \forall i, t, s \quad [27]$$

$$\mathbf{ru}_{hts} + \mathbf{rn}_{hts} - \mathbf{rd}_{hts} = \overline{P}_h - \mathbf{P}_{ht}, \forall h, t, s \quad [28]$$

$$\mathbf{ru}_{its} + \mathbf{rn}_{its} - \mathbf{rd}_{its} = \overline{P}_i - \mathbf{P}_{it}, \forall i, t, s \quad [29]$$

$$0 \leq \mathbf{ru}_{hts} \leq \mathbf{Ru}_{ht}, \forall h, t, s \quad [30]$$

$$0 \leq \mathbf{rd}_{hts} \leq \mathbf{Rd}_{ht}, \forall h, t, s \quad [31]$$

$$0 \leq \mathbf{ru}_{its} \leq \mathbf{Ru}_{it}, \forall i, t, s \quad [32]$$

$$0 \leq \mathbf{rd}_{its} \leq \mathbf{Rd}_{it}, \forall i, t, s \quad [33]$$

$$0 \leq \mathbf{rn}_{its} \leq \mathbf{Rn}_{it}, \forall i, t, s \quad [34]$$

$$\mathbf{p}_{jts} + \mathbf{ru}_{jts} - \mathbf{rd}_{jts} \leq \mathbf{P}_{jt}, \forall j, t, s \quad [35]$$

$$0 \leq \mathbf{ru}_{jts} \leq \mathbf{Ru}_{jt}, \forall j, t, s \quad [36]$$

$$0 \leq \mathbf{rd}_{jts} \leq \mathbf{Rd}_{jt}, \forall j, t, s \quad [37]$$

$$\mathbf{ll}_{jts} \leq \mathbf{p}_{jts}, \forall j, t, s \quad [38]$$

$$\mathbf{ws}_{kts} < \mathbf{wp}_{kts}, \forall j, t, s \quad [39]$$

$$\mathbf{p}_{hts} - \mathbf{p}_{h,t-1,s} \leq \overline{\rho}_h (\mathbf{Ct}_{h,t-1} + \mathbf{Su}_{ht}), \forall h, t, s \quad [40]$$

$$\mathbf{p}_{its} - \mathbf{p}_{i,t-1,s} \leq \overline{\rho}_i (\mathbf{Ct}_{i,t-1} + \mathbf{Su}_{it}), \forall i, t, s \quad [41]$$

$$\mathbf{p}_{h,t-1,s} - \mathbf{p}_{hts} \leq \underline{\rho}_h (\mathbf{Ct}_{ht} + \mathbf{Sd}_{ht}), \forall h, t, s \quad [42]$$

$$\mathbf{p}_{i,t-1,s} - \mathbf{p}_{its} \leq \underline{\rho}_i (\mathbf{Ct}_{it} + \mathbf{Sd}_{it}), \forall i, t, s \quad [43]$$

$$\sum_{h=1}^{N_H} \mathbf{p}_{hts} + \sum_{i=1}^{N_I} \mathbf{p}_{its} + \sum_{K=1}^{N_K} (\mathbf{p}_{kts} - \mathbf{w}_{s_{kts}}) - \sum_{j=1}^{N_J} (\mathbf{p}_{jts} - \mathbf{u}_{jts}) - \quad [44]$$

$$\sum_{n,m \in \Psi} B(n, m) * (\delta_{nts} - \delta_{mts}) = 0, \forall t, s$$

$$-\bar{T} \leq B(n, m) * (\delta_{nts} - \delta_{mts}) \leq \bar{T}, \forall n, m \in \Psi, t, s \quad [45]$$

$$-\pi \leq \delta_{nts} \leq \pi, \forall n \in \Psi, t, s \quad [46]$$

$$-\pi \leq \delta_{mts} \leq \pi, \forall m \in \Psi, t, s \quad [47]$$

$$(\mathbf{ct}_{i,t-1,s} - \mathbf{ct}_{its}) + \mathbf{su}_{its} - \mathbf{sd}_{its} = 0, \forall i, t, s \quad [48]$$

$$\mathbf{su}_{its} - \mathbf{Su}_{it} \geq 0, \forall i, t, s \quad [49]$$

$$\mathbf{sd}_{its} - \mathbf{Sd}_{it} \geq 0, \forall i, t, s \quad [50]$$

$$\sum_{t=1}^{\mu_{i0}^U} (1 - \mathbf{ct}_{its}) = 0, \forall i, s \quad [51]$$

$$\sum_{t=1}^{\mu_{i0}^D} \mathbf{ct}_{its} = 0, \forall i, s \quad [52]$$

$$\sum_{q=t}^{t+\mu_i^U-1} \mathbf{ct}_{iqs} \geq \mu_i^U * \mathbf{su}_{its}, \forall t = (\mu_{i0}^U + 1), \dots, (\mu_i^U + 1), \forall i, s \quad [53]$$

$$\sum_{q=t}^{t+\mu_i^D-1} (1 - \mathbf{ct}_{its}) \geq \mu_i^D * \mathbf{sd}_{its}, \forall t = (\mu_{i0}^D + 1), \dots, (\mu_i^D + 1), \forall i, s \quad [54]$$

$$\sum_{q=t}^{24} (\mathbf{ct}_{its} - \mathbf{su}_{its}) \geq 0, \forall t = (25 - \mu_i^U), \dots, 24, \forall i, s \quad [55]$$

$$\sum_{q=t}^{24} (1 - \mathbf{ct}_{its} - \mathbf{sd}_{its}) \geq 0, \forall t = (26 - \mu_i^D), \dots, 24, \forall i, s \quad [56]$$

5.4.2 Objective function

The objective function contains two main terms that refer to the first and second stage of the model. Note that the second stage term is averaged across the probability space and that no connections between scenarios exist.

The terms of the objective function [a-c] correspond to the first stage. Term [a] of the objective function calculates the total operating cost of the slow units in the first stage. The cost can be broken down to start-up, shut-down, and production costs for each generator. Additionally, the cost of the scheduled safety reserves is added in the cost. Similarly, term [b] calculates the operating costs of all fast units that participate in the network. The breakdown of the cost coefficients is similar to the slow units with the addition of scheduled non-spinning reserves for fast units.

Term [c] provides the net contribution of loads in the model. The total monetary benefit received by serving each load is reduced by the cost of scheduling safety reserves. The net sum is effectively a negative number, corresponding to the negative gain that the model tries to minimize.

Terms [d-g] correspond to the second stage balancing market expected cost. Term [d] sums the marginal cost referring to the reserves deployed by slow units in the balancing stage. The objective term [e] exposes the flexibility allowed by the model to the fast units by calculating the adjusted operating cost. It is important to notice that fast units are allowed to adjust the commitment schedule in the balancing stage of the clearing process. This is relevant to the fact that the balancing stage requires fast adjustments to the realization of demand and other stochastic elements thus, fast units are used for this purpose due to their fast response time.

Term [f] captures the adjustment effect of the balancing stage on the participating loads. The cost of the deployed reserves is accrued as cost. Additionally, in cases where the loads were not served due to network saturation or due to economic decision tied to low utility, a penalty is added to the objective function.

Term [g] penalizes the loss of opportunity to include the power production of wind farms in the planning. Wind spillage cost is added in the objective, giving incentive to the model to make use of all the available wind resources.

5.4.3 Constraints

- First stage constraints

Constraints [1-23] refer to the first stage decision variables. Terms [1,2] impose bounds on the production of electricity for slow and fast units in the DAM. Term [3] represents the a priori expectation of the demand. It is important to notice that the constraint is deterministic in this model as it considers the demand to be fixed. This assumption is equivalent to optimizing against the maximum forecasted demand for the following day. This approach simplifies the problem but potentially can introduce approximation errors.

Constraint [4] bounds the wind production decision variable to some maximum amount. In this case, the maximum installed capacity of the wind farms may be used. Nevertheless, the maximum capacity in many cases is not met due to weather conditions. A more reasonable approach is the use of mean value of the forecasted wind production scenarios, inflated by a safety margin as follows:

$$\overline{P_{kt}} = a * E[P_{kt}] \leq \text{Max}[P_k], a \geq 1$$

The constraint terms [5-11] are imposing restrictions on the reserves of the units as well as reserves required by the loads. The values of these bounds are a matter of policy and are left to the experience and guidelines of the ISO's to decide.

Constraint [12] serves the most important role in the first stage since it enforces the balancing of production and demand. Power output from all the generating units along with the wind power production must be equal to the demand at all times.

Transmission network capacity and voltage angle constraints are captured with constraints [13-15].

Constraints [16, 17] enforce the logical relationship between the binary variables of the first stage. At every time period the binary variables that refer to the start-up, the commitment and the shut-down of a unit must conform to these rules. Additionally to [16,17], constraints [18-23] are enforcing the required minimum and maximum up and down times for all units. The planning horizon of a day is separated in three sub-periods, namely the initial period, middle

periods and final periods. Within these sub-periods operating constraints related to every generating unit are modeled.

- Second stage constraints

The second stage constraints are terms [24-56]. In the balancing stage, the block diagonal form of the problem becomes evident with a subset of the constraints referring only to scenario dependent variables, while others complicate first and second stage variables. In this sense, the decision on commitment of slow units is honored while the scheduling of the actual power output per unit may be adjusted in real time. For the fast units, flexibility on the commitment is allowed additionally to the adjustment of power output.

An additional set of constraints that is found in the second stage alone are the ramping constraints that act upon the scheduled power output of the units, enforcing operational bounds on the scheduled plan.

Constraints [24-29] set bounds on the scheduled power output in the BM by honoring the DAM production schedule and commitment. The terms [30-34] are setting the upper bound of the second stage reserve variables to be equal to the scheduled reserves as decided in the first stage.

In a similar manner, constraints [35-37] are setting up upper and lower bounds on load consumption and reserves per load based on values of the corresponding variables of the first stage.

Constraint terms [40-43] effectively honor the ramping capabilities of the generators, by enforcing the maximum upward and downward steps units are allowed to take during the operation.

In a similar fashion to the first stage, constraint [44] is the BM clearing constraint, enforcing all production to meet demand at all times. In this constraint unserved load and wind spillage are also introduced in the balancing equation, following the realization of uncertainty.

Constraint terms [45-47] bound the transmission model to conform to the capacity and voltage angle requirements across the modeled network in the second stage.

Constraints [48-50] give the fast units the ability to adjust their schedule in the second stage. This enhancement in the assumption on fast units is important because it is the main reason why the two sets of generators are modeled separately. It is not only a quantitative difference that differentiates the two but also a qualitative.

Finally, constraints [51-56] enforce the commitment requirements on the different periods of the planning horizon on the second stage variables.

By observing the formulation of the model and the definition domain of the decision variables it is evident that the SMC problem should be classified as a two stage stochastic problem with mixed integer first stage and mixed integer second stage. As described in previous chapters, the nature and the complexity of the problem can be practically addressed via decomposing the full model into smaller parts. In order to provide a result for the decomposed problem, the solution procedure as well as the decomposition steps will be described in the following chapter.

6 Decomposition of the Stochastic Market Clearing problem

6.1 Formulating the Lagrangian

Due to the mixed integer nature of the problem and the intricate structure, the computational complexity of the SMC grows exponentially with the increase of the input model. Mixed integer problems belong in the NP-Hard family of problems so dealing with large-scale systems becomes a tedious task.

Following the discussion of Chapter 4.3, the decomposition of the SMC will follow the formulation of Lagrangian decomposition. Research suggests that this technique has never been used to solve this type of problem and additionally, little research is published on applications of Lagrangian decomposition on two stage stochastic with both stages involving both binary and continuous variables. This thesis aims to present the application of the technique in a field that has received relatively little exposure in practical applications.

The proposed technique closely follows the steps presented in the work by [16], where Lagrangian relaxation has been employed to solve in a distributed manner the stochastic unit commitment problem which is a simplified look on the planning of the energy market. In this problem only the scheduling of the commitment and power output of the generators are considered without a general overview of the market's ecosystem. The steps presented in this work are easily transferred in the domain of the SMC problem due to the flexibility of the Lagrangian method.

The first step on decomposition focuses on decoupling the second stage from the first. Given that the increase of the number of scenarios will effectively provide a better approximation of the stochastic variables involved, the second stage also has a decoupled nature on its own with the second stage variables being independent from each other. The coupling happens on the first stage that provides the basis of operations and planning of the DAM.

In order to decouple the second stage, the first stage variables that are involved in the second stage need to be duplicated across all scenarios. The duplicated variables are then substituting the scenario dependent initial variables and effectively decouple the second stage from the first. For example, the first stage commitment variable for fast units Ct_{it} is scenario dependent and need to be substituted by the duplicated variable Ct_{its} which represents the decoupled first

stage constraint. Effectively, constraint [24] that refers to this variable will get reformulated as follows:

$$\underline{P}_i * \mathbf{Ct}_{its} \leq \mathbf{p}_{its} \leq \overline{P}_i * \mathbf{Ct}_{its}, \forall i, t, s$$

In essence, if the same procedure is followed for all coupling first stage variables, the second stage problems become independent from each other and the problem will be decomposed by scenarios.

A closer look reveals that the duplication of the first stage variables is not adequate to guarantee that the decomposed problem is equivalent to the original since the duplicated first stage variables are allowed to assume any value irrespective of each other. This artifact is corrected with the introduction of linking constraints on the duplicated variables that enforce them to be equal to each other and more importantly, equal to the original scenario independent first stage variables. This set of constraints, namely the non-anticipativity constraints enforce sanity and cohesiveness in the first stage. Examining the same variable as above, the non-anticipativity constraint for the first stage commitment variable for fast units will be as follows:

$$\mathbf{Ct}_{its} - \mathbf{Ct}_{it} = 0, \forall i, t, s$$

The non-anticipativity constraints concentrate the coupling nature of the full problem into a manageable set of the newly introduced duplicated variables. It is important to notice that at this stage of decomposition, after the introduction of the duplicated variables and the non-anticipativity constraints, the new problem is completely equivalent to the original. Of course the new problem as of now does not offer any advantages since the number of variables have increased by the number of scenarios. The benefit comes with the relaxation of the linking constraints in the next step but it is a helpful sanity check at this point of the process to verify that the two problems perform with exactly the same manner giving the same results.

In the next step, the non-anticipativity constraints are relaxed and enter the objective function under penalty of the Lagrangian multipliers. It is important to notice that at this step all the relaxed constraints are equality constraint so the related multipliers are not bounded below by zero as in the case of inequality constraints. Additionally, the difference between the relaxed variables constitutes a valid sub-gradient for this problem [6].

The Lagrangian problem, after the relaxation of the non-anticipativity constraints can be formulated as follows for the SMC problem.

$$\begin{aligned}
\text{Lagrangian: } \min \mathbf{c}^T \mathbf{x} + \sum_{s \in S} p_s (d^T \mathbf{y} + w^T \mathbf{x}_s) + u^T (\mathbf{x}_s - \mathbf{x}) = \\
\sum_{t=1}^{24} \sum_{h=1}^{N_H} \{ Su_{C_{ht}} * \mathbf{S}u_{ht} + PC_{ht} * \mathbf{P}_{ht} + Sd_{C_{ht}} * \mathbf{S}d_{ht} + Ru_{C_{ht}} * \mathbf{R}u_{ht} + Rd_{C_{ht}} * \mathbf{R}d_{ht} \} + \\
\sum_{t=1}^{24} \sum_{i=1}^{N_I} \{ Su_{C_{it}} * \mathbf{S}u_{it} + PC_{it} * \mathbf{P}_{it} + Sd_{C_{it}} * \mathbf{S}d_{it} + Ru_{C_{it}} * \mathbf{R}u_{it} + Rn_{C_{it}} * \mathbf{R}n_{it} \\
+ Rd_{C_{it}} * \mathbf{R}d_{it} \} + \\
\sum_{t=1}^{24} \sum_{j=1}^{N_J} \{ Ru_{C_{jt}} * \mathbf{R}u_{jt} + Rd_{C_{jt}} * \mathbf{R}d_{jt} - U_{jt} * \mathbf{P}_{jt} \} + \\
\sum_{s=1}^S p_s * \left\{ \sum_{t=1}^{24} \left[\sum_{h=1}^{N_H} (mc_{hts} * \mathbf{r}_{hts}) \right. \right. \\
+ \sum_{i=1}^{N_I} (Su_{C_{it}} * (\mathbf{su}_{its} - \mathbf{S}u_{its}) + Sd_{C_{it}} * (\mathbf{sd}_{its} - \mathbf{S}d_{its}) + mc_{its} * \mathbf{r}_{its}) \\
+ \sum_{j=1}^{N_J} (mc_{jts} * \mathbf{r}_{jts} + amc_{jts} * (\mathbf{ru}_{jts} - \mathbf{rd}_{jts}) + VOLL_j * \mathbf{l}_{jts}) \\
\left. \left. + \sum_{k=1}^{N_K} w_{s_{kts}} * \xi_k \right] \right\} \\
+ \sum_{t=1}^{24} \sum_{s=1}^S \sum_{x \in X, \ell=1, \dots, N_\ell, y \in Y} \{ u_{xts}^{Su} * (\mathbf{S}u_{xts} - \mathbf{S}u_{xt}) + u_{xts}^{Sd} \\
* (\mathbf{S}d_{xts} - \mathbf{S}d_{xt}) + u_{xts}^{Ct} * (\mathbf{C}t_{xts} - \mathbf{C}t_{xt}) + u_{xts}^P * (\mathbf{P}_{xts} - \mathbf{P}_{xt}) + u_{xts}^{Ru} \\
* (\mathbf{R}u_{xts} - \mathbf{R}u_{xt}) + u_{xts}^{Rd} * (\mathbf{R}d_{xts} - \mathbf{R}d_{xt}) + u_{its}^{Rn} * (\mathbf{R}n_{its} - \mathbf{R}n_{it}) + u_{yts}^A \\
* (\mathbf{\Delta}_{yts} - \mathbf{\Delta}_{yt}) \}
\end{aligned}$$

such that:

First stage constraints: [1 – 23]

Second stage constraints: [38,39], [44 – 48], [51 – 56]

$$\underline{P}_h * \mathbf{Ct}_{hts} \leq \mathbf{p}_{hts} \leq \overline{P}_h * \mathbf{Ct}_{hts}, \forall h, t, s \quad [57]$$

$$\underline{P}_i * \mathbf{Ct}_{its} \leq \mathbf{p}_{its} \leq \overline{P}_i * \mathbf{Ct}_{its}, \forall i, t, s \quad [58]$$

$$\mathbf{p}_{hts} = \mathbf{P}_{hts} + \mathbf{ru}_{hts} - \mathbf{rd}_{hts}, \forall h, t, s \quad [59]$$

$$\mathbf{p}_{its} = \mathbf{P}_{its} + \mathbf{ru}_{its} + \mathbf{rn}_{its} - \mathbf{rd}_{its}, \forall i, t, s \quad [60]$$

$$\mathbf{ru}_{hts} + \mathbf{rn}_{hts} - \mathbf{rd}_{hts} = \overline{P}_h - \mathbf{P}_{hts}, \forall h, t, s \quad [61]$$

$$\mathbf{ru}_{its} + \mathbf{rn}_{its} - \mathbf{rd}_{its} = \overline{P}_i - \mathbf{P}_{its}, \forall i, t, s \quad [62]$$

$$0 \leq \mathbf{ru}_{hts} \leq \mathbf{Ru}_{hts}, \forall h, t, s \quad [63]$$

$$0 \leq \mathbf{rd}_{hts} \leq \mathbf{Rd}_{hts}, \forall h, t, s \quad [64]$$

$$0 \leq \mathbf{ru}_{its} \leq \mathbf{Ru}_{its}, \forall i, t, s \quad [65]$$

$$0 \leq \mathbf{rd}_{its} \leq \mathbf{Rd}_{its}, \forall i, t, s \quad [66]$$

$$0 \leq \mathbf{rn}_{its} \leq \mathbf{Rn}_{its}, \forall i, t, s \quad [67]$$

$$\mathbf{p}_{jts} + \mathbf{ru}_{jts} - \mathbf{rd}_{jts} \leq \mathbf{P}_{jts}, \forall j, t, s \quad [68]$$

$$0 \leq \mathbf{ru}_{jts} \leq \mathbf{Ru}_{jts}, \forall j, t, s \quad [69]$$

$$0 \leq \mathbf{rd}_{jts} \leq \mathbf{Rd}_{jts}, \forall j, t, s \quad [70]$$

$$\mathbf{p}_{hts} - \mathbf{p}_{h,t-1,s} \leq \overline{\rho}_h (\mathbf{Ct}_{h,t-1,s} + \mathbf{Su}_{ht,s}), \forall h, t, s \quad [71]$$

$$\mathbf{p}_{its} - \mathbf{p}_{i,t-1,s} \leq \overline{\rho}_i (\mathbf{Ct}_{i,t-1,s} + \mathbf{Su}_{its}), \forall i, t, s \quad [72]$$

$$\mathbf{p}_{h,t-1,s} - \mathbf{p}_{hts} \leq \underline{\rho}_h (\mathbf{Ct}_{hts} + \mathbf{Sd}_{hts}), \forall h, t, s \quad [73]$$

$$\mathbf{p}_{i,t-1,s} - \mathbf{p}_{its} \leq \underline{\rho}_i (\mathbf{Ct}_{its} + \mathbf{Sd}_{its}), \forall i, t, s \quad [74]$$

$$\mathbf{su}_{its} - \mathbf{Su}_{its} \geq 0, \forall i, t, s \quad [75]$$

$$\mathbf{sd}_{its} - \mathbf{Sd}_{its} \geq 0, \forall i, t, s \quad [76]$$

In this formulation, in order to improve readability, the sets of all slow units, $\{1, \dots, N_H\}$, all fast units, $\{1, \dots, N_I\}$ and all loads, $\{1, \dots, N_J\}$ have been compacted into the superset X . For transmission variables the set Y comprises of all the sets of buses in the system.

6.2 Formulating the decomposed sub-problems

Relaxing the non-anticipativity constraints allows the decomposition of the SMC problem into separate sub-problems. The solution procedures for the Lagrangian problem described in Chapter 4.4 are iterative in nature. For each iteration of the algorithms, the number of sub-problems will be equal to the number of scenarios plus one problem that will represent the first stage. Traditionally, the problem that refers to the first stage is considered to be the master problem, while the sub-problems that are scenario dependent are simply referred to as sub-problems, following the notation used in Bender's decomposition.

Expanding on the Lagrangian problem presented above, the decomposed first stage Lagrangian sub-problem is formulated as follows:

First stage Lagrangian sub – problem: $\min c^T \mathbf{x} - u^T \mathbf{x} =$

$$\sum_{t=1}^{24} \sum_{h=1}^{N_H} \{SuC_{ht} * \mathbf{S}u_{ht} + PC_{ht} * \mathbf{P}_{ht} + SdC_{ht} * \mathbf{S}d_{ht} + RuC_{ht} * \mathbf{R}u_{ht} + RdC_{ht} * \mathbf{R}d_{ht}\} +$$

$$\sum_{t=1}^{24} \sum_{i=1}^{N_I} \{SuC_{it} * \mathbf{S}u_{it} + PC_{it} * \mathbf{P}_{it} + SdC_{it} * \mathbf{S}d_{it} + RuC_{it} * \mathbf{R}u_{it} + RnC_{it} * \mathbf{R}n_{it} \\ + RdC_{it} * \mathbf{R}d_{it}\} +$$

$$\sum_{t=1}^{24} \sum_{j=1}^{N_J} \{RuC_{jt} * \mathbf{R}u_{jt} + RdC_{jt} * \mathbf{R}d_{jt} - U_{jt} * \mathbf{P}_{jt}\} -$$

$$\sum_{t=1}^{24} \sum_{s=1}^S \sum_{x \in X, t=1, \dots, N_t, y \in Y} \{u_{xts}^{Su} * \mathbf{S}u_{xt} + u_{xts}^{Sd} * \mathbf{S}d_{xt} + u_{xts}^{Ct} * \mathbf{C}t_{xt} + u_{xts}^P * \mathbf{P}_{xt} + u_{xts}^{Ru} * \mathbf{R}u_{xt} \\ + u_{xts}^{Rd} * \mathbf{R}d_{xt} + u_{its}^{Rn} * \mathbf{R}n_{it} + u_{yts}^A * \mathbf{A}_{yt}\}$$

such that:

First stage constraints: [1 – 23]

By analyzing the problem it is evident that there is no dependence on scenarios. The first stage sub-problem can be solved independently given an initial guess on the values of the Lagrangian multipliers.

The scenario dependent sub-problems after decomposition are presented below.

Scenario dependent Lagrangian sub – problems:

$$\begin{aligned}
\min \sum_{s \in S} p_s (d^T \mathbf{y} + w^T \mathbf{x}_s) + u^T \mathbf{x}_s = \\
\sum_{s=1}^S p_s * \left\{ \sum_{t=1}^{24} \left[\sum_{h=1}^{N_H} (mc_{hts} * \mathbf{r}_{hts}) \right. \right. \\
\left. \left. + \sum_{i=1}^{N_i} (Suc_{it} * (\mathbf{su}_{its} - \mathbf{Su}_{its}) + Sdc_{it} * (\mathbf{sd}_{its} - \mathbf{Sd}_{its}) + mc_{its} * \mathbf{r}_{it}) \right. \right. \\
\left. \left. + \sum_{j=1}^{N_j} (mc_{jts} * \mathbf{r}_{jts} + amc_{jts} * (\mathbf{ru}_{jts} - \mathbf{rd}_{jts}) + VOLL_j * \mathbf{ll}_{jts}) \right. \right. \\
\left. \left. + \sum_{k=1}^{N_K} \mathbf{ws}_{kts} * \xi_k \right\} \\
+ \sum_{t=1}^{24} \sum_{s=1}^S \sum_{x \in X, \ell=1, \dots, N_\ell, y \in Y} \left\{ u_{xts}^{Su} * \mathbf{Su}_{xts} + u_{xts}^{Sd} * \mathbf{Sd}_{xts} + u_{xts}^{Ct} * \mathbf{Ct}_{xts} + u_{xts}^P * \mathbf{P}_{xts} + u_{xts}^{Ru} \right. \\
\left. * \mathbf{Ru}_{xts} + u_{xts}^{Rd} * \mathbf{Rd}_{xts} + u_{its}^{Rn} * \mathbf{Rn}_{its} + u_{yts}^A * \mathbf{A}_{yts} \right\}
\end{aligned}$$

such that:

Second stage constraints: [38,39], [44 – 48], [51 – 56], [57 – 76]

This formulation presents the entire collection of scenario dependent sub-problems but in closer inspection there is no connection between the problems across scenarios. Every different scenario may be optimized independently as a separate mixed integer problem.

In summary, the SMC problem following the method of Lagrangian relaxation has been decomposed to independent mixed integer problems. Following iterative sub-gradient methods, the decomposed solution may be found.

6.3 Calculating sub-gradient directions

The sub-gradient direction for the SMC may be calculated from the slack on the non-anticipativity constraints [6]. At the optimal point, the sub-gradient vector should have norm close to zero, forcing all the duplicated scenario dependent variables to assume the same value, which in turn is equal to the first stage variables.

The sub-gradient vector has the following form.

$$sg^T = [sg_{x_{ts}}^{Su} \mid sg_{x_{ts}}^{Sd} \mid sg_{x_{ts}}^{Ct} \mid sg_{x_{ts}}^P \mid sg_{x_{ts}}^{Ru} \mid sg_{x_{ts}}^{Rd} \mid sg_{its}^{Rn} \mid sg_{y_{ts}}^A]$$

$$\forall x \in h, i, j, \forall y \in n, m$$

$$sg_{x_{ts}}^{Su} = \mathbf{S}u_{x_{ts}} - \mathbf{S}u_{x_t}, \forall x \in h, i$$

$$sg_{x_{ts}}^{Sd} = \mathbf{S}d_{x_{ts}} - \mathbf{S}d_{x_t}, \forall x \in h, i$$

$$sg_{x_{ts}}^{Ct} = \mathbf{C}t_{x_{ts}} - \mathbf{C}t_{x_t}, \forall x \in h, i$$

$$sg_{x_{ts}}^P = \mathbf{P}_{x_{ts}} - \mathbf{P}_{x_t}, \forall x \in h, i, j$$

$$sg_{x_{ts}}^{Ru} = \mathbf{R}u_{x_{ts}} - \mathbf{R}u_{x_t}, \forall x \in h, i, j$$

$$sg_{x_{ts}}^{Rd} = \mathbf{R}d_{x_{ts}} - \mathbf{R}d_{x_t}, \forall x \in h, i, j$$

$$sg_{its}^{Rn} = \mathbf{R}n_{its} - \mathbf{R}n_{it}$$

$$sg_{y_{ts}}^A = \mathbf{A}_{y_{ts}} - \mathbf{A}_{y_t}, \forall y \in n, m$$

The sub-gradient vector is calculated at the end of every iteration while collecting the optimal decision variables from the decomposed sub-problems.

6.4 Strengthening the sub-problems

At this stage of the decomposition, for some problems it is necessary to go through a phase of strengthening the sub-problems. The introduction of duplicated first stage scenario dependent

variables in the model leaves them without bounds. Hence, all the first stage constraints that impose physical bounds on the variables may be carried over to the Lagrangian sub-problems in order to avoid unboundedness.

On a similar vein, the sub-problems may be strengthened furthermore in order to maintain a higher level of consistency by imposing more first stage restrictions upon them. As discussed in Chapter 4.4 sub-gradient methods that deal with mixed integer problems are always bounded by the duality gap so the convergence of Phase 1 will not necessarily bring the solution to a point where all the scenario dependent sub-problems are feasible. That is to say that there may be inconsistencies across scenarios for the duplicated first stage constraints. In this sense Phase 2 is employed in order to restore sanity to the first stage and maintain close proximity to the optimal point of Phase 1.

Strengthening the model of the scenario dependent sub-problems may improve the resulting quality of the solution after Phase 1. By transferring more first stage constraints into the second stage the inconsistencies are avoided. Nevertheless, the cost incurs by much richer formulation for the decomposed sub-problems. Effectively a modeling trade off exists between the computational burden faced during iterations of Phase 1 and the inconsistencies that Phase 2 has to face.

In the present solution method, all the constraints that refer to physical bounds, i.e. maximum power output of the generating units, have been transferred over to the second stage since without this restriction the problems become unbounded at individual runs. Additionally, in order to strengthen the problems and structurally avoid inconsistencies, the constraints referring to minimum and maximum up and down times for the generators have also been transferred over to the second stage.

7 Results

7.1 Problem data

Input data for this work has been from [31], since this thesis is an extension of this work with addition of decomposition methods for the model. The model describes a scaled version of the Pennsylvania, New Jersey, and Maryland Interconnection regional transmission organization (PJM RTO).

The scaled network consists of 59 generators connected on two transmission buses. Out of the total, 36 generators are classified as slow, while 23 are classified as fast. The network includes a single windfarm that provides power from wind production in the area. The consumer side of the market is represented by 17 loads connected to the transmission network. The planning horizon of the model is 24 hours and for simplicity, an empty initial condition was assumed.

The original model in [31] uses the notion of a block to aggregate regional generators and loads in order to achieve a hierarchical segmentation of the full network. Nevertheless, the problem data only referred to a single block and thus the notion has been removed from the formulation and tests, without loss of generality, in order to simplify the calculations.

Furthermore, as discussed in Chapter 6, the day ahead consideration of wind production in the original model is set to the maximum capacity of the wind farm. In the present dataset, the installed wind capacity is set to 5GW. This assumption is unrealistic given that on an average day the maximum capacity is rarely reached. Instead of using the maximum capacity of the wind farm in this approach, the expected output has been used. This choice is a valid because it is based on the assumption that the decision maker has prior knowledge of the expected power output of the uncertain source and does not assume any explicit information about the exact realization of the variable.

To capture the stochasticity of the wind, 25 wind scenarios have been used in the model. In the original model, all scenarios are considered to assume equal probability and hence in this work similar approach has been followed.

The following figure depicts the average wind production of the 25 scenarios.

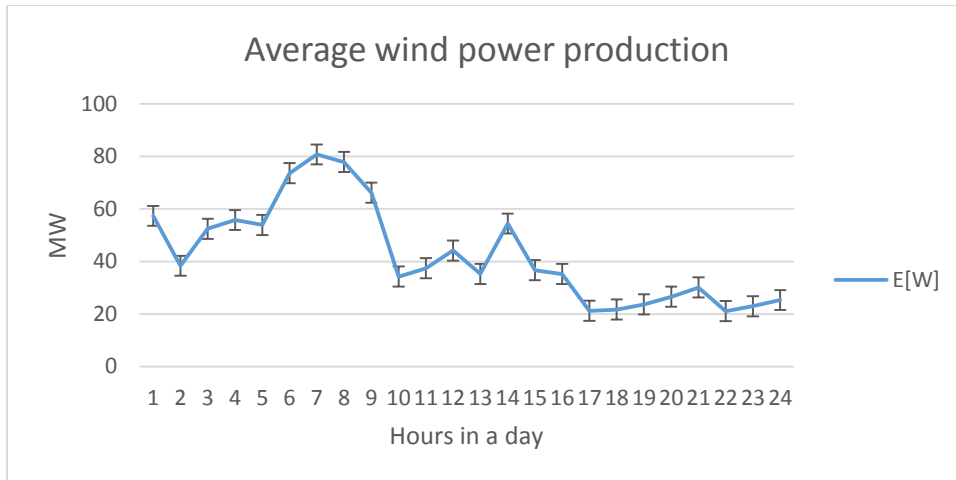


Figure 7-1 Average wind production

Observing the curve, it becomes evident that modeling against the installed wind capacity is unrealistic since expected production averages in levels far lower than the maximum.

The figures below depict the maximum capacity of the generating units that are present in the network.

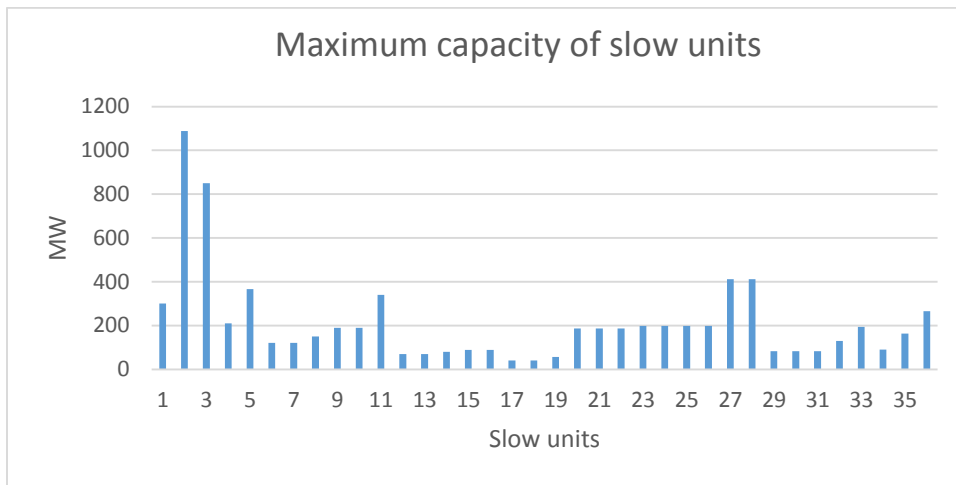


Figure 7-2 Maximum capacity of slow units

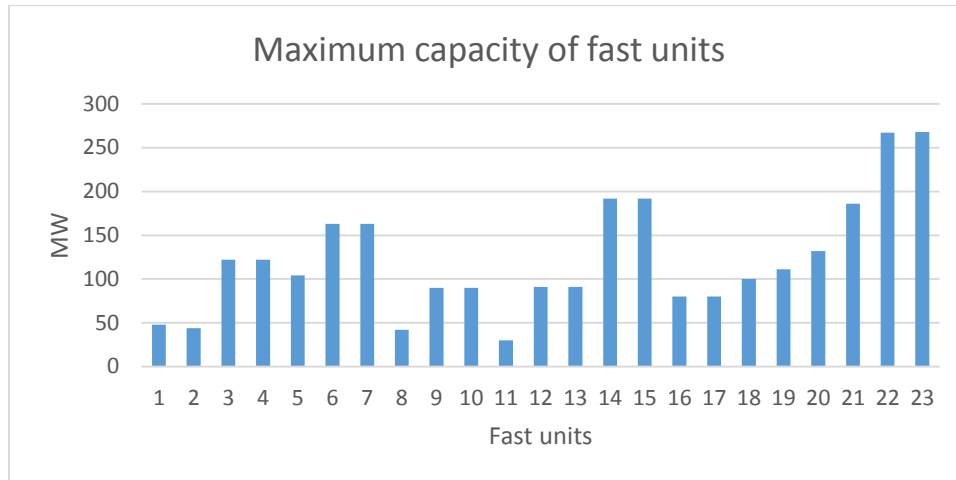


Figure 7-3 Maximum capacity of fast units

Summarizing the data, the total maximum power output that can come from slow units at any time is 7718MW, while for fast units is 2808MW. In total, the maximum power output of the generators is 10526MW.

The following figure depicts the sum of all 17 loads during the planning horizon.

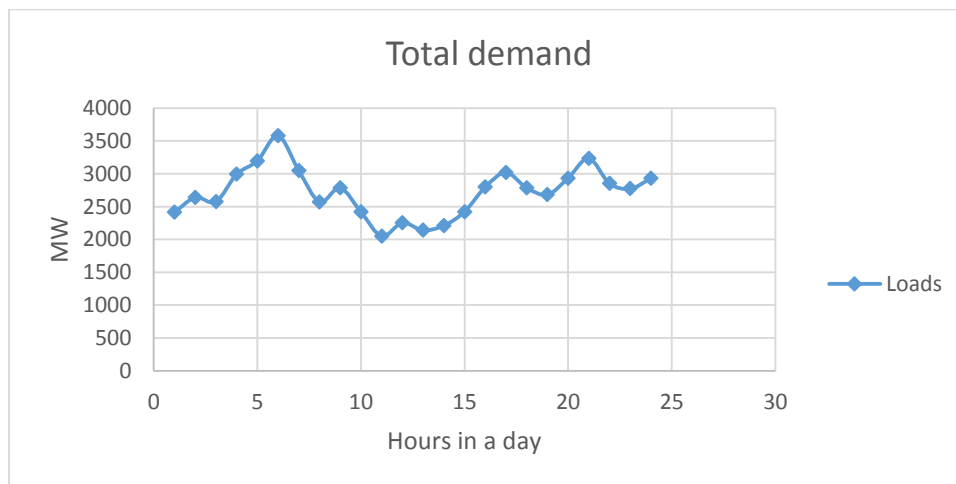


Figure 7-4 Total demand in a day

The maximum demand in the day is 3578.96MW. Comparing to the maximum power output of the generator network, it is evident that the network is not in a heavily congested state. From this realization it follows that the optimal solution should not shed any load.

This realization is helpful when analyzing the results since the model as presented in chapter 5 does not allow for infeasible solutions. The load shedding is modeled with a penalty so even at the extreme case where the entire load has been rejected the solution is feasible but extremely sub-optimal. In the current data set, the network can face demand with ease and border cases should not arise.

7.2 Experiment setting

The numerical experiments test different variations of the sub-gradient method in order to assess the differences of each approach. Additionally, different parameters were tested to demonstrate the impact on the convergence and the result. Finally the methods were tested using 25 scenarios in order to assess the impact of a more detailed discretization of the stochastic variable.

The stopping criterion of the method has been set as a maximum number of iterations given the non-monotonic behavior of the algorithm. Phase 1 and Phase 2 of the method were run for 120 iterations, with individual phases running for 60 each.

Two methods for updating the Lagrangian multipliers were tested. Methods include approaches that iteratively follow the steepest descent sub-gradient direction as well as the cutting plane method.

For methods requiring step length calculation, multiple approaches were examined. Constant step length with different combinations of parameters and the Polyak step length were tested in order to compare the impact of the step length choice in the convergence behavior of the algorithm.

Finally, the weighted sub-gradient method was also examined with parametric weight factors.

In the following section, results from different approaches will be presented. The optimal cost of each outcome will be compared against the optimal cost of the full problem in order to evaluate the method.

7.2.1 Simple step method

For the simple step method the following formula for the step length was tested.

$$\gamma_k = \frac{1}{\alpha + \beta k}, a, b \in R$$

The following parameters were tested for this approach.

$$\{a, b\} = \{[1, 0.1] \quad [5, 0.1] \quad [10, 0.1]\}$$

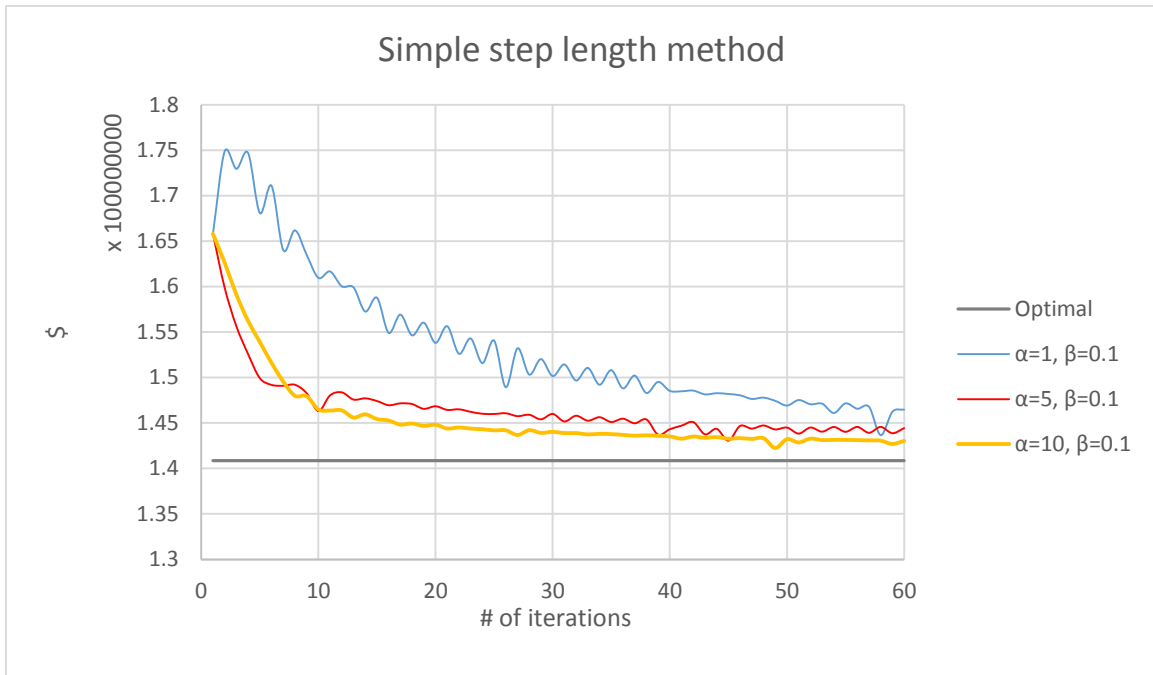


Figure 7-5 Simple step length method

The behavior of the method of $\alpha=1, \beta=0.1$ exposes the oscillating nature of the sub-gradient methods. By increasing the constant term of the denominator, it is possible to improve the convergence of the algorithm.

7.2.2 Polyak step method

For this approach, different values of the multiplication factor λ were tested. The following graphs shows the behavior of the Phase 1 method with Polyak step length for $\lambda = [0.1, 1, 10]$.

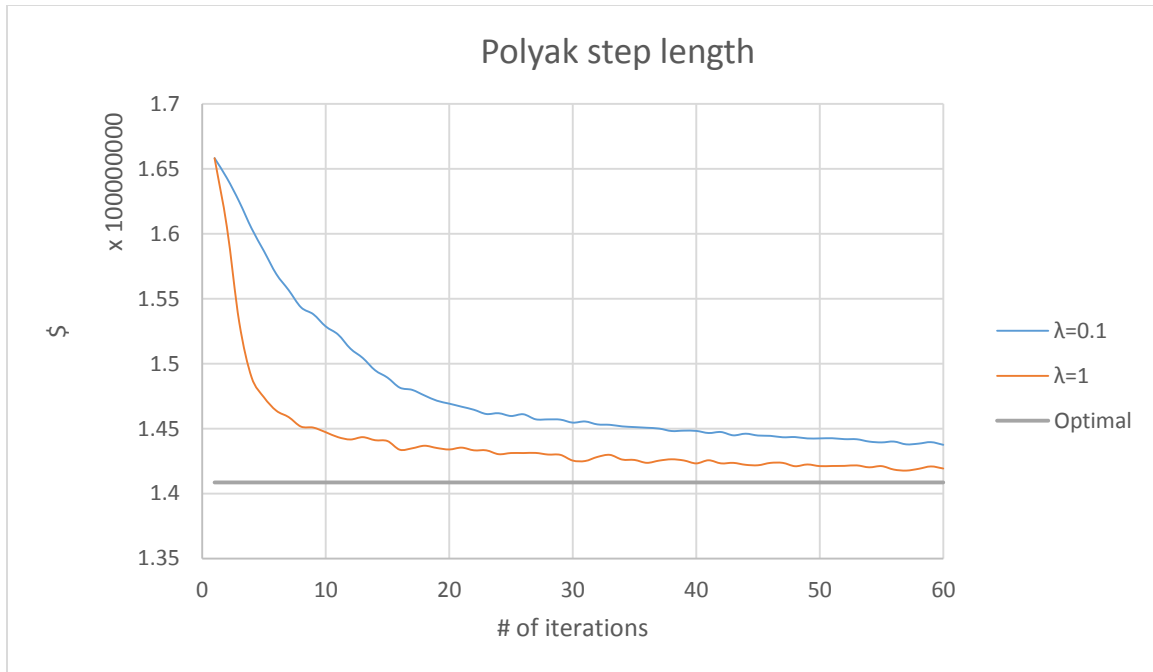


Figure 7-6 Polyak step length for $\lambda=0.1, 1$

In the figure above, the results from different multiplication factors are contrasted. It is evident that convergence is faster with use of the value $\lambda=1$ for the Polyak step method. Also after 60 iterations, it is important to notice that the method maintains a gap from the optimal value, which is also depicted.

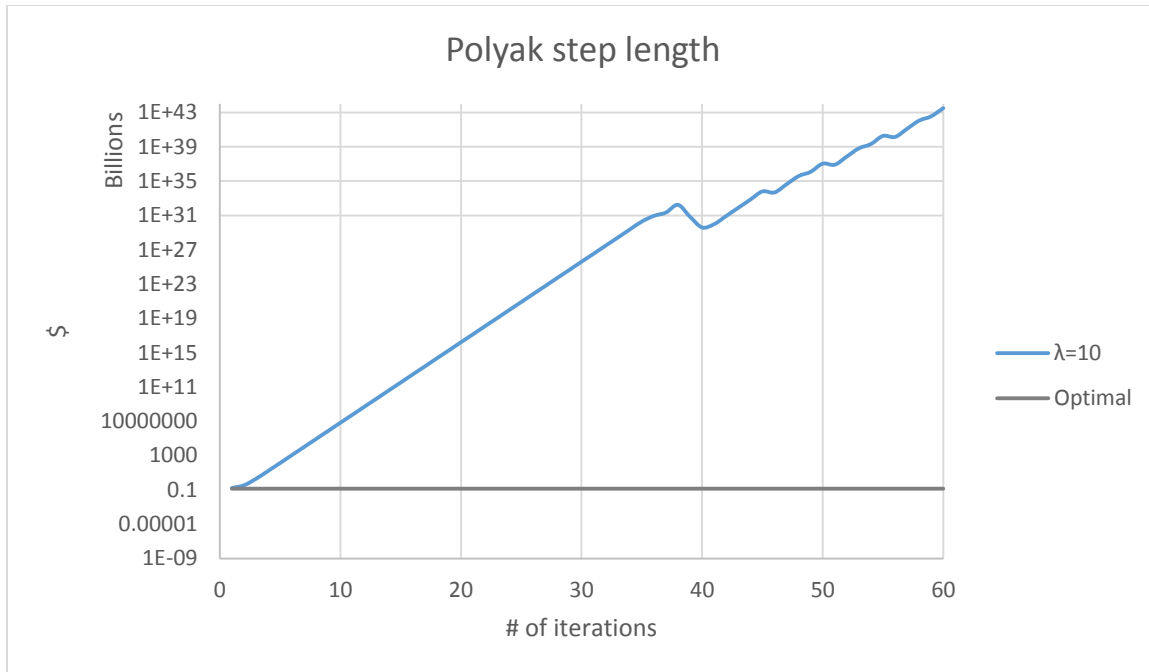


Figure 7-7 Polyak step length for $\lambda=10$

The above figure shows a divergent result when using $\lambda=10$ for the Polyak step length method. The graph appears to be in close proximity in the beginning of the plot but that is due to the almost exponential divergence of the experiment. As discussed above, the sub-gradient method appears to be sensitive to the choice of parameters and hence a trial and error procedure is important for the proper behavior of the algorithm.

7.2.3 Weighted sub-gradient method

In this approach, the weighted sum of the directions was limited to include only the most recent past direction. The new direction vector was calculated as follows:

$$sg_w^k = a * sg^k + b * sg^{k-1}, a + b = 1$$

The parameters a, b dictate the level that the previously calculated direction is allowed to affect the new direction. The following sets of parameters were tested:

$$\{a, b\} = \{0.9, 0.1\} \quad \{0.8, 0.2\} \quad \{0.7, 0.3\}$$

The following graphs show the behavior of the Phase 1 method for the set of parameters above.

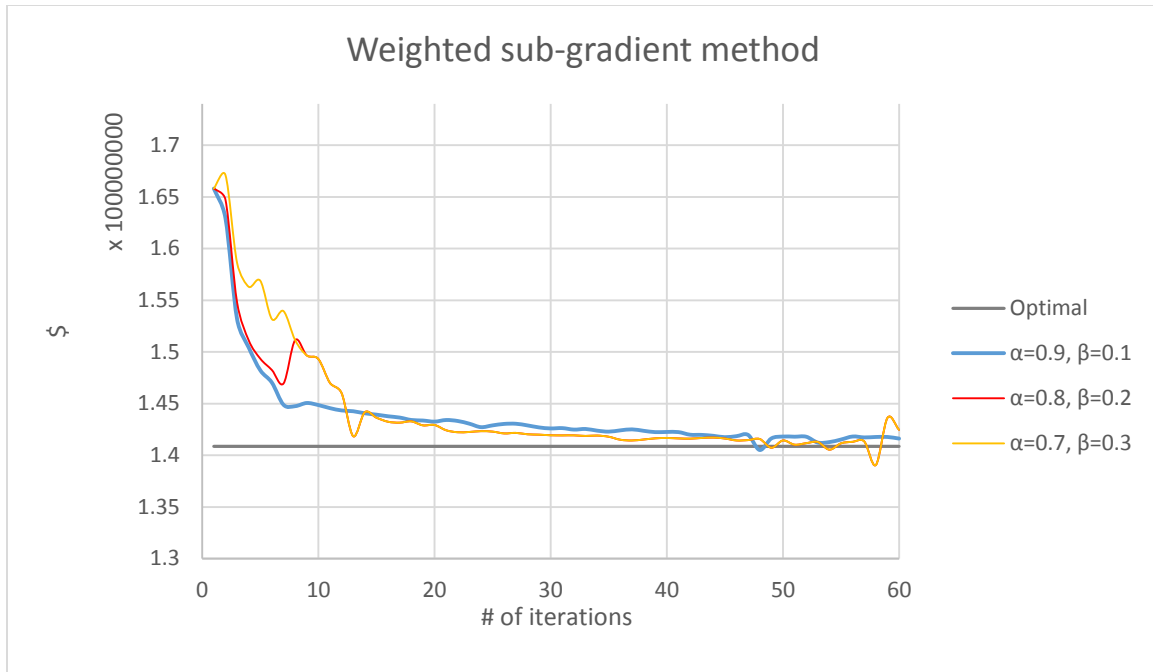


Figure 7-8 Weighted sub-gradient method

The weighted method follows a divergent path, as more weight is placed to the last known direction, as can be seen in the figure above. For the sets of parameters, $\alpha, \beta=(0.8, 0.2)$ and $\alpha, \beta=(0.7, 0.3)$, the method proves to have a slower convergence rate and more enhanced oscillating behavior.

7.2.4 Overall comparison

In the following, the best results from all of the methods are compiled in a single graph to provide insight for the different convergence rates of each method.

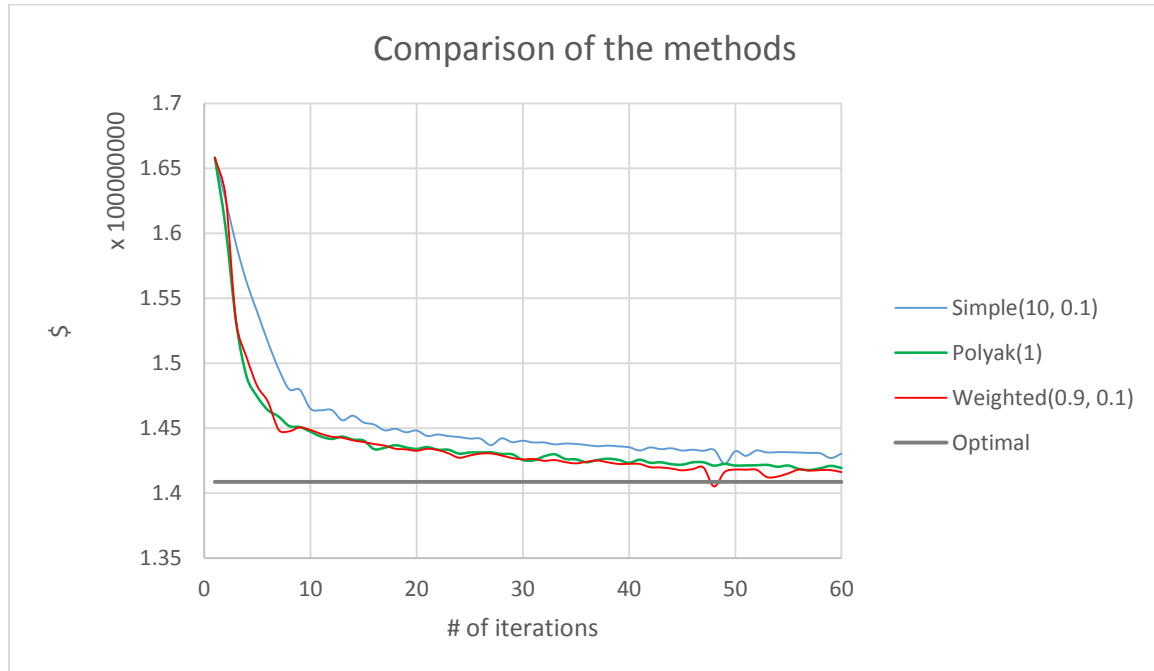


Figure 7-9 Comparison of sub-gradient methods

Observing the convergence curves of different approaches, it is evident that methods that follow the Polyak step updating procedure have a clear impact on convergence as well as the final optimality gap. Additionally, the weighted sub-gradient method proves to provide a small performance boost to the method, correcting the oscillation and following the optimal value closer than the simple method.

Finally, from this graph it is evident that the method follows a non-monotonic path to optimality. A similar approach to the one presented needs to take place in order to determine the appropriate number of iterations and parameters to be used.

7.2.5 Cutting plane method

We were not able to retrieve useful results for the cutting plane method. Almost all combinations of parameters and bounds for the multipliers lead to divergent solutions. Many

of the experiments resulted in a trivial behavior where the multipliers were assuming the upper or the lower bounds possible by the experiment without having any consideration for the state of the method at any time.

The method requires further investigation in order to identify the underlying reasons for this erratic behavior.

7.3 Practical considerations

Working with Lagrangian decomposition on two stage stochastic program brings forth some important practical considerations when analyzing the results for validity. The main reasons are the following:

- The method inherently involves inconsistencies. Phase 1 of the method does not guarantee to provide feasible solutions [6]. Infeasibility in the current context translates to differences in the duplicated first stage variables. Hence, validating the quality of the results as they come from Phase 1 become inherently difficult.
- The method has non-monotonic behavior. As discussed in Chapter 4, the solution of the Lagrangian dual with sub-gradient methods is following an oscillating path towards the optimal solution. This behavior increases the difficulty of assessing the quality of a solution at any given step.
- The method is sensitive to parameters and starting points. Baring close resemblance to the steepest descent algorithm, the parameters used in the calculation of step length and the initial values of Lagrangian multipliers affect the behavior of the method. Furthermore, it becomes difficult to distinguish erratic behaviors of the method that are caused by ill conditioned starting points rather than sub-optimal approaches.

In the problem of Stochastic Market Clearing, the most important aspect is the balance of the market. In every time-period in the planning horizon, production must be equal to demand, given that the network has enough capacity. In this scope, validation of the result should ensure that the loads that are present in the system are served at any point.

The following graph shows the production as it is captured in the duplicated first stage variables against the load. The two graphs match exactly verifying that the optimal solution from Phase 2 actually provides the optimal solution to the problem.

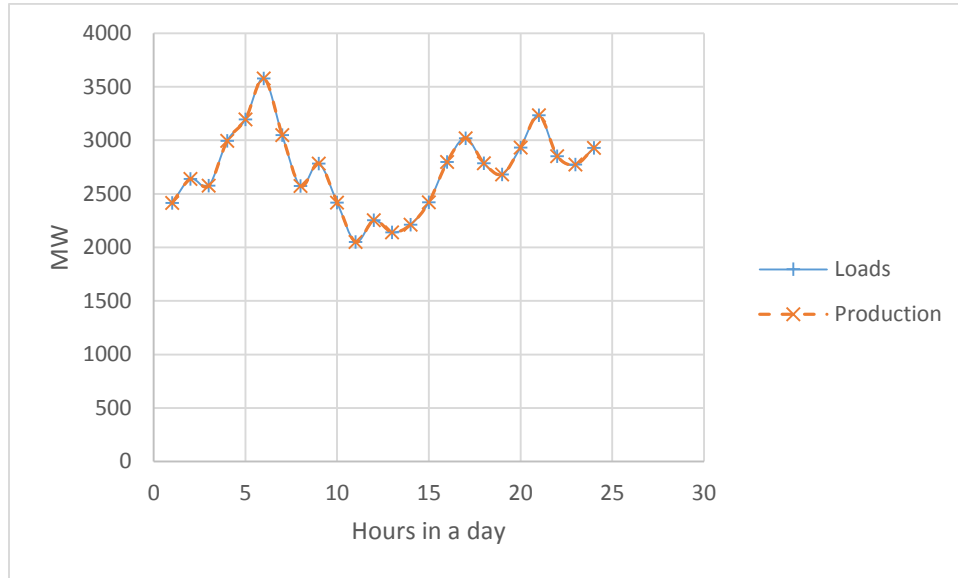


Figure 7-10 Comparison of production and demand after Phase 2

8 Conclusions and future work

In summary, the Stochastic Market Clearing problem has been decomposed with Lagrangian decomposition. Due to the discontinuities of the dual space, sub-gradient methods have been employed to solve the dual problem. Different approaches for the convergent phase of the method were tried and convergence to an optimal solution was observed.

The Lagrangian relaxation method can provide a practical approach in decomposing large mixed integer two stage stochastic problems. Nevertheless, the validity of the results is difficult to be verified as the method operates in an oscillating, non-monotonic fashion.

In order to achieve smooth operation of the method, trial and error procedures must take place in order to select the proper values of the parameters required by the methods. Every set of parameters is problem specific and special care is warranted in order to identify proper conditions due to the erratic behavior of the optimization method.

In future work, different multiplier updating methodologies will be examined. The cutting plane method proved to be inapplicable in this case with the method providing divergent results, contrary to the reported behavior in literature [6].

Furthermore, the available input data provided a representation of the network that was able to withstand the load in any case. In future work, the sensitivity of the result will be examined with different states of saturation of the network.

Lastly, systematic relaxation schemes may be able to provide a practical solution to the SMC in the same sense [17] is suggesting. The method balances between heuristic procedures and could potentially prove useful for the case of M/M problems.

9 Bibliography

- [1] A. Conejo, M. Carrion and J. Morales, Decision making under uncertainty in Electricity Markets, Springer, 2010.
- [2] A. Shabbir, "Two stage stochastic programming: A brief introduction," 2012.
- [3] L. Lasdon, Optimization Theory for Large Systems, New York: Mac Millan , 1970.
- [4] Dantzig and Wolfe, "The decomposition algorithm for linear programming," *Econometrica* 9, 1961.
- [5] D. Bertsimas and J. Tsitsiklis, Introduction to linear optimization, Nashua, NH: Athena Scientific, 1997.
- [6] A. Conejo, E. Castillo, R. Minguez and R. Garcia-Bertand, Decomposition Techniques in Mathematical Programming, Springer, 2006.
- [7] J. Birge and F. Louveaux, Introduction to Stochastic Programming, Springer, 2011.
- [8] S. Petkov and C. Maranas, "Multiperiod Planning and Scheduling of Multiproduct Batch Plants under Demand Uncertainty," *Industrial & Engineering Chemistry Research*, vol. 36, no. 11, pp. 4864-4881, 1997.
- [9] G. Laporte and F. Louveaux, "The integer L-shaped method for stochastic integer programs with complete recourse," *Operations Research Letters* 13, pp. 133-142, 1993.
- [10] M. Slater, "Lagrangian multipliers revisited," Yale University, New Haven, CT, 1980.
- [11] A. Geoffrion, "Generalized Bender's decomposition," *Journal of Optimization Theory and Applications*, pp. 237-260, 1972.
- [12] S. Sen and L. Ntaimo, "A comparative study of decomposition algorithms for stochastic combinatorial optimization," *Computational Optimization Applications*, pp. 299-319, 2008.

- [13] S. Sen and C. Smith, "Decomposition with Branch and Cut approaches for two stage stochastic mixed integer programming," 2002.
- [14] S. Sen and J. Hige, "The C3 theorem and a D2 algorithm for large scale stochastic mixed integer programming: set convexification," *Journal of Mathematical Programming*, 2005.
- [15] X. Zhu, "Discrete two stage stochastic mixed integer programs with applications to airline fleet assignment and workforce planning problems," University of Virginia, Virginia, 2006.
- [16] A. Papavasiliou, S. Oren and R. O'Neill, "Reserve Requirements for Wind Power Integration: A Scenario-Based Stochastic Programming Framework," *IEEE Transactions on Power Systems*, vol. 26, no. 4, pp. 2197-2206, 2011.
- [17] D. Bertsekas, *Nonlinear Programming*, Belmont, Massachusetts: Athena Scientific, 2008.
- [18] J. Naoum-Sawaya and S. Elhedhli, "A nested benders decomposition approach for telecommunication network planning," *Naval Research Logistics*, vol. 57, no. 6, pp. 519-539, 2010.
- [19] D. Bertsekas and J. Tsitsiklis, *Parallel and Distributed Computation Numerical Methods*, New York: Prentice-Hall, 1989.
- [20] Fisher, "An applications oriented guide to Lagrangian relaxation," *Interfaces*, vol. 15, no. 2, pp. 10-21, 1985.
- [21] J. Douchi, "Stanford E364b lecture notes," [Online]. Available: http://stanford.edu/class/ee364b/lectures/subgrad_method_slides.pdf. [Accessed 11 10 2015].
- [22] F. Fumero, "A modified subgradient algorithm for Lagrangean relaxation," *Computers & Operations Research*, vol. 28, pp. 33-52, 2001.

- [23] B. Polyak, "Minimization of unsmooth functionals," Moscow, 1968.
- [24] A. H. Kim S, "Convergence of a generalized subgradient method for non-differentiable convex optimization.," *Mathematical Programming*, pp. 75-80, 1991.
- [25] A. C. Jimknez Redondo, "Short-term hydro-thermal coordination by lagrangian relaxation: solution of the dual problem," *IEEE Transaction on Power Systems*, vol. 14, no. 1, pp. 89-95, 1999.
- [26] "New England ISO," [Online]. Available: <http://www.iso-ne.com/>.
- [27] "Renewable North West," [Online]. Available: <http://www.rnp.org/node/wave-tidal-energy-technology>.
- [28] "Federal Bureau of Investigation Fact Book," FBI, Washington DC, 2013.
- [29] D. Lampridis, P. Dokopoulos and G. Papagiannis, *Power System Engineering (Συστήματα Ηλεκτρικής Ενέργειας)*, Thessaloniki, Greece: Ziti Publications, 2006.
- [30] E. Denaxas, R. Bandyopadhyay, D. Patino-Echeverri and N. Pitsianis, "SynTiSe: A modified multi-regime MCMC approach for generation of wind power synthetic time series," in *IEEE Systems Conference*, Vancouver, Canada, 2014.
- [31] A. Daraeepour and D. P. Echeverri, "Stochastic market clearing in electricity markets with high penetration of wind energy: air emissions reductions and economic savings," in *20th Conference of the International Federation of Operational Research Societies*, Barcelona, Spain, 2014.
- [32] K. Frauendorfer, *Stochastic Two-Stage Programming*, Berlin: Springer-Verlag, 1992.
- [33] C. Caroe and J. Tind, "L-Shaped decomposition of two-stage stochastic programs with integer recourse," *Mathematical Programming* 83, pp. 451-464, 1998.

APPENDIX

10 Appendix

The code used in this thesis is provided here. The program has been developed in IBM CPLEX

12.5. The program is separated in three sections, the main control flow, the first stage sub-problem and the second stage sub-problems. Each part is shown below.

10.1 Main control flow

```
{string} Scenar = ...;
float PiScen[Scenar] = ...;
{string} Units_S = ...;
{string} Units_F = ...;
{string} Buses_N = ...;
{string} WindF = ...;
{string} Loads = ...;
int T = ...;
range TimePeriods = 1..T;
range TimePeriods0 = 0..T;
// container variables to hold information between iterations
int    DA_Commit_S [Scenar][TimePeriods0][Units_S];
int    DA_Startup_S [Scenar][TimePeriods0][Units_S];
int    DA_Shutdown_S [Scenar][TimePeriods0][Units_S];
float  DA_Gen_S    [Scenar][TimePeriods0][Units_S] ;
int    DA_Commit_F [Scenar][TimePeriods0][Units_F];
int    DA_Startup_F [Scenar][TimePeriods0][Units_F];
int    DA_Shutdown_F [Scenar][TimePeriods0][Units_F];
int    DA_IndNonSres_F [Scenar][TimePeriods0][Units_F];
float  DA_Gen_F    [Scenar][TimePeriods0][Units_F] ;
float  DA_SresUp_S [Scenar][TimePeriods][Units_S];
float  DA_SresDown_S [Scenar][TimePeriods][Units_S];
float  DA_SresUp_F [Scenar][TimePeriods][Units_F];
float  DA_SresDown_F [Scenar][TimePeriods][Units_F];
float  DA_NonSres_F [Scenar][TimePeriods][Units_F];
float  DA_WindP [Scenar][TimePeriods][WindF];
float  DA_Load [Scenar][TimePeriods][Loads];
float  DA_LoadRU [Scenar][TimePeriods][Loads];
float  DA_LoadRD [Scenar][TimePeriods][Loads];
float  DA_Delta [Scenar][TimePeriods][Buses_N];
int    DAL_Commit_S [TimePeriods0][Units_S];
int    DAL_Startup_S [TimePeriods0][Units_S];
int    DAL_Shutdown_S [TimePeriods0][Units_S];
float  DAL_Gen_S [TimePeriods0][Units_S] ;
int    DAL_Commit_F [TimePeriods0][Units_F];
int    DAL_Startup_F [TimePeriods0][Units_F];
int    DAL_Shutdown_F [TimePeriods0][Units_F];
int    DAL_IndNonSres_F [TimePeriods0][Units_F];
float  DAL_Gen_F [TimePeriods0][Units_F] ;
float  DAL_SresUp_S [TimePeriods][Units_S];
float  DAL_SresDown_S [TimePeriods][Units_S];
float  DAL_SresUp_F [TimePeriods][Units_F];
float  DAL_SresDown_F [TimePeriods][Units_F];
float  DAL_NonSres_F [TimePeriods][Units_F];
float  DAL_WindP [TimePeriods][WindF];
float  DAL_Load [TimePeriods][Loads];
float  DAL_LoadRU [TimePeriods][Loads];
float  DAL_LoadRD [TimePeriods][Loads];
float  DAL_Delta [TimePeriods][Buses_N];
// subgradient
float  sg_Commit_S [Scenar][TimePeriods0][Units_S];
float  sg_Startup_S [Scenar][TimePeriods0][Units_S];
float  sg_Shutdown_S [Scenar][TimePeriods0][Units_S];
float  sg_Gen_S [Scenar][TimePeriods0][Units_S] ;
float  sg_Commit_F [Scenar][TimePeriods0][Units_F];
```

```

float sg_Startup_F [Scenar][TimePeriods0][Units_F];
float sg_Shutdown_F [Scenar][TimePeriods0][Units_F];
float sg_IndNonSres_F [Scenar][TimePeriods0][Units_F];
float sg_Gen_F [Scenar][TimePeriods0][Units_F];
float sg_SresUp_S [Scenar][TimePeriods][Units_S];
float sg_SresDown_S [Scenar][TimePeriods][Units_S];
float sg_SresUp_F [Scenar][TimePeriods][Units_F];
float sg_SresDown_F [Scenar][TimePeriods][Units_F];
float sg_NonSres_F [Scenar][TimePeriods][Units_F];
float sg_WindP [Scenar][TimePeriods][WindF];
float sg_Load [Scenar][TimePeriods][Loads];
float sg_LoadRU [Scenar][TimePeriods][Loads];
float sg_LoadRD [Scenar][TimePeriods][Loads];
float sg_Delta [Scenar][TimePeriods][Buses_N];
// lagrangian multipliers
float m_Commit_S [Scenar][TimePeriods0][Units_S] = ...;
float m_Startup_S [Scenar][TimePeriods0][Units_S]= ...;
float m_Shutdown_S [Scenar][TimePeriods0][Units_S]= ...;
float m_Gen_S [Scenar][TimePeriods0][Units_S]= ...;
float m_Commit_F [Scenar][TimePeriods0][Units_F]= ...;
float m_Startup_F [Scenar][TimePeriods0][Units_F]= ...;
float m_Shutdown_F [Scenar][TimePeriods0][Units_F]= ...;
float m_IndNonSres_F [Scenar][TimePeriods0][Units_F]= ...;
float m_Gen_F [Scenar][TimePeriods0][Units_F] = ...;
float m_SresUp_S [Scenar][TimePeriods][Units_S]= ...;
float m_SresDown_S [Scenar][TimePeriods][Units_S]= ...;
float m_SresUp_F [Scenar][TimePeriods][Units_F]= ...;
float m_SresDown_F [Scenar][TimePeriods][Units_F]= ...;
float m_NonSres_F [Scenar][TimePeriods][Units_F]= ...;
float m_WindP [Scenar][TimePeriods][WindF]= ...;
float m_Load [Scenar][TimePeriods][Loads]= ...;
float m_LoadRU [Scenar][TimePeriods][Loads]= ...;
float m_LoadRD [Scenar][TimePeriods][Loads]= ...;
float m_Delta [Scenar][TimePeriods][Buses_N]= ...;
int NITER = 120;
range iter = 1..NITER;
float first_cost[iter];
float second_cost[Scenar];
float lamda_const = 0.1;
int modfile = 5; // if verbose > 0, print files on mod of this number
string lrf_source_file = "lr_first.mod";
string lrs_source_file = "lr_second.mod";
int numofScenarios = 25;
// modify these to run small/large
string lp_source_file = ...;
string lp_data_file = ...;
string lr_problem_data_file = ...;
int verbose = ...;
int m0 = ...;
int phaselMaxIterations =60;
execute{
thisOplModel.settings.mainEndEnabled = true;
for(var s in thisOplModel.Scenar){
thisOplModel.PiScen[s] = 1 / thisOplModel.numOfScenarios;
}
//-----
// initialize lagrangian multipliers -----
//-----
for(var s in thisOplModel.Scenar){
for(var t0 in thisOplModel.TimePeriods0){
for(var h in thisOplModel.Units_S){
thisOplModel.m_Commit_S[s][t0][h] = thisOplModel.m0;
thisOplModel.m_Startup_S[s][t0][h] = thisOplModel.m0;
thisOplModel.m_Shutdown_S[s][t0][h] = thisOplModel.m0;
thisOplModel.m_Gen_S[s][t0][h] = thisOplModel.m0;
}
for(var i in thisOplModel.Units_F){
thisOplModel.m_Commit_F[s][t0][i] = thisOplModel.m0;
thisOplModel.m_Startup_F[s][t0][i] = thisOplModel.m0;
}
}
}

```

```

thisOplModel.m_Shutdown_F[s][t0][i] = thisOplModel.m0;
thisOplModel.m_IndNonSres_F[s][t0][i] = thisOplModel.m0;
thisOplModel.m_Gen_F[s][t0][i] = thisOplModel.m0;
}
}
for(var t in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.m_SresUp_S[s][t][h] = thisOplModel.m0;
thisOplModel.m_SresDown_S[s][t][h] = thisOplModel.m0;
}
for(var i in thisOplModel.Units_F){
thisOplModel.m_SresUp_F[s][t][i] = thisOplModel.m0;
thisOplModel.m_SresDown_F[s][t][i] = thisOplModel.m0;
thisOplModel.m_NonSres_F[s][t][i] = thisOplModel.m0;
}
for(var k in thisOplModel.WindF){
thisOplModel.m_WindP[s][t][k] = thisOplModel.m0;
}
for(var j in thisOplModel.Loads){
thisOplModel.m_Load[s][t][j] = thisOplModel.m0;
thisOplModel.m_LoadRU[s][t][j] = thisOplModel.m0;
thisOplModel.m_LoadRD[s][t][j] = thisOplModel.m0;
}
for(var n in thisOplModel.Buses_N){
thisOplModel.m_Delta[s][t][n] = thisOplModel.m0;
}}
main{
function lrlp(){
// linear relaxation for the original problem definition
var lrlp_source = new IloOplModelSource(thisOplModel.lp_source_file);
var lrlp_solver = new IloCplex();
var lrlp_def = new IloOplModelDefinition(lrlp_source);
var lrlp_data = new IloOplDataSource(thisOplModel.lp_data_file);
var lrlp_model = new IloOplModel(lrlp_def, lrlp_solver);
var Lstar;
lrlp_model.addDataSource(lrlp_data);
lrlp_model.generate();
//lrlp_model.convertAllIntVars();
if(lrlp_solver.solve()){
Lstar = lrlp_solver.getObjValue();
// Print results
if(thisOplModel.verbose > 0){
// Gen S
var ofile = new IloOplOutputFile("./results/full_DA_Gen_S.csv");
for(var tt in thisOplModel.TimePeriods0){
for(var hh in thisOplModel.Units_S){
ofile.write(lrlp_model.DA_Gen_S[tt][hh] + ",");
}
ofile.writeln();
}
ofile.close();
// Load
ofile = new IloOplOutputFile("./results/full_DA_Load.csv");
for(var tt in thisOplModel.TimePeriods){
for(var jj in thisOplModel.Loads){
ofile.write(lrlp_model.DA_Load[tt][jj] + ",");
}
ofile.writeln();
}
ofile.close();
}
// Print results
lrlp_model.end();
lrlp_def.end();
lrlp_solver.end();
lrlp_source.end();
}
return Lstar;
}
}

```

```

var cofile = new IloOplOutputFile("./results/a_convergence.txt");
var ofile = new IloOplOutputFile();
// get the upper bound for the problem
var Lstar = lrlp() * 1.2;
writeln("0" + Lstar);
var dataSet = thisOplModel.lr_problem_data_file;
// first stage model definition
var lrf_source = new IloOplModelSource(thisOplModel.lrf_source_file);
var lrf_solver = new IloCplex();
var lrf_def = new IloOplModelDefinition(lrf_source);
var problem_data = new IloOplDataSource(dataSet);
var shared_data = thisOplModel.dataElements;
var lrf_model = new IloOplModel(lrf_def, lrf_solver);
lrf_model.addDataSource(shared_data);
lrf_model.addDataSource(problem_data);
lrf_model.generate();
problem_data.end();
// second stage model definition
var lrs_source = new IloOplModelSource(thisOplModel.lrs_source_file
var lrs_solver = new IloCplex();// declare a solver for the model
var lrs_def = new IloOplModelDefinition(lrs_source);
var scenarios_data = Array(thisOplModel.numOfScenarios);
for(var i = 1; i <= thisOplModel.numOfScenarios; i++){
scenarios_data[i] = "../windScenarios/wind_scenario_" + i + ".dat";
}
}
//-----
// [run K iterations for lagrangian relaxation, observe convergence] -----
//-----
for(var iter = 1; iter <= thisOplModel.NITER; iter++){
if(iter <= thisOplModel.phaselMaxIterations){
// update multipliers for first stage
var lrf_data = lrf_model.dataElements;
lrf_data.m_Commit_S = thisOplModel.m_Commit_S; // 1
lrf_data.m_Startup_S = thisOplModel.m_Startup_S; // 2
lrf_data.m_Shutdown_S = thisOplModel.m_Shutdown_S; // 3
lrf_data.m_Gen_S = thisOplModel.m_Gen_S; // 4
lrf_data.m_Commit_F = thisOplModel.m_Commit_F; // 5
lrf_data.m_Startup_F = thisOplModel.m_Startup_F; // 6
lrf_data.m_Shutdown_F = thisOplModel.m_Shutdown_F; // 7
lrf_data.m_IndNonSres_F = thisOplModel.m_IndNonSres_F; // 8
lrf_data.m_Gen_F = thisOplModel.m_Gen_F; // 9
lrf_data.m_SresUp_S = thisOplModel.m_SresUp_S; // 10
lrf_data.m_SresDown_S = thisOplModel.m_SresDown_S; // 11
lrf_data.m_SresUp_F = thisOplModel.m_SresUp_F; // 12
lrf_data.m_SresDown_F = thisOplModel.m_SresDown_F; // 13
lrf_data.m_NonSres_F = thisOplModel.m_NonSres_F; // 14
lrf_data.m_WindP = thisOplModel.m_WindP; // 15
lrf_data.m_Load = thisOplModel.m_Load; // 16
lrf_data.m_LoadRU = thisOplModel.m_LoadRU; // 17
lrf_data.m_LoadRD = thisOplModel.m_LoadRD; // 18
lrf_data.m_Delta = thisOplModel.m_Delta; // 19
// end first stage model
lrf_model.end();
// update the model definition
lrf_model = new IloOplModel(lrf_def, lrf_solver);
lrf_model.addDataSource(lrf_data);
// generate and solve updated first stage model
lrf_model.generate();
lrf_data.end();
if(lrf_solver.solve()){
thisOplModel.first_cost[iter] = lrf_solver.getObjValue();
// Print results
if(thisOplModel.verbose > 0 && (iter % thisOplModel.modfile == 0)){
ofile = new IloOplOutputFile("./results/f_" + iter + ".dat");
ofile.writeln(lrf_model.printSolution());
ofile.close();
}
// Print results
}else{

```



```

lrs_data.m_IndNonSres_F = thisOplModel.m_IndNonSres_F;
lrs_data.m_Gen_F = thisOplModel.m_Gen_F;
lrs_data.m_SresUp_S = thisOplModel.m_SresUp_S;
lrs_data.m_SresDown_S = thisOplModel.m_SresDown_S;
lrs_data.m_SresUp_F = thisOplModel.m_SresUp_F;
lrs_data.m_SresDown_F = thisOplModel.m_SresDown_F;
lrs_data.m_NonSres_F = thisOplModel.m_NonSres_F;
lrs_data.m_WindP = thisOplModel.m_WindP;
lrs_data.m_Load = thisOplModel.m_Load;
lrs_data.m_LoadRU = thisOplModel.m_LoadRU;
lrs_data.m_LoadRD = thisOplModel.m_LoadRD;
lrs_data.m_Delta = thisOplModel.m_Delta;
lrs_model.end()
lrs_model = new IloOplModel(lrs_def, lrs_solver)
lrs_model.addDataSource(lrs_data);
lrs_model.generate();
lrs_data.end();
// solve with updated multipliers
if(lrs_solver.solve()){
thisOplModel.second_cost[s] = lrs_solver.getObjValue();
if(thisOplModel.verbose > 0 && (iter % thisOplModel.modfile ==0)){
ofile = new IloOplOutputFile("./results/s_" + iter + "_" + (count-1) + ".dat");
ofile.writeln(lrs_model.printSolution());
ofile.close();
}
}else{
writeln("Error in 2nd stage");
}
// gather information from scenario s
for(var t0 in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.DA_Commit_S[s][t0][h]= lrs_model.DA_Commit_S[t0][h];
thisOplModel.DA_Startup_S[s][t0][h]= lrs_model.DA_Startup_S[t0][h];
thisOplModel.DA_Shutdown_S[s][t0][h]= lrs_model.DA_Shutdown_S[t0][h];
thisOplModel.DA_Gen_S[s][t0][h] = lrs_model.DA_Gen_S[t0][h];
}
for(var i in thisOplModel.Units_F){
thisOplModel.DA_Commit_F[s][t0][i] = lrs_model.DA_Commit_F[t0][i];
thisOplModel.DA_Startup_F[s][t0][i]= lrs_model.DA_Startup_F[t0][i];
thisOplModel.DA_Shutdown_F[s][t0][i] = lrs_model.DA_Shutdown_F[t0][i];
thisOplModel.DA_IndNonSres_F[s][t0][i] = lrs_model.DA_IndNonSres_F[t0][i];
thisOplModel.DA_Gen_F[s][t0][i] = lrs_model.DA_Gen_F[t0][i];
}
}
for(var t in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.DA_SresUp_S[s][t][h] = lrs_model.DA_SresUp_S[t][h]; // 10
thisOplModel.DA_SresDown_S[s][t][h]= lrs_model.DA_SresDown_S[t][h]; // 11
}
for(var i in thisOplModel.Units_F){
thisOplModel.DA_SresUp_F[s][t][i] = lrs_model.DA_SresUp_F[t][i]; // 12
thisOplModel.DA_SresDown_F[s][t][i]= lrs_model.DA_SresDown_F[t][i]; // 13
thisOplModel.DA_NonSres_F[s][t][i] = lrs_model.DA_NonSres_F[t][i]; // 14
}
for(var k in thisOplModel.WindF){
thisOplModel.DA_WindP[s][t][k] = lrs_model.DA_WindP[t][k]; // 15
}
for(var j in thisOplModel.Loads){
thisOplModel.DA_Load[s][t][j] = lrs_model.DA_Load[t][j];
thisOplModel.DA_LoadRU[s][t][j]= lrs_model.DA_LoadRU[t][j];
thisOplModel.DA_LoadRD[s][t][j]= lrs_model.DA_LoadRD[t][j];
}
for(var n in thisOplModel.Buses_N){
thisOplModel.DA_Delta[s][t][n] = lrs_model.DA_Delta[t][n]; // 19
}
}
}
lrs_model.end();
}

```

```

//-----
// [Update multiplier information using Subgradient methods]-----
//-----
// calculate subgradient
// calculate step length
// update lagrangian multipliers
var denominator = 0;
for(s in thisOplModel.Scenar){
for(var t0 in thisOplModel.TimePeriods0){
for(var h in thisOplModel.Units_S){
thisOplModel.sg_Commit_S[s][t0][h] = thisOplModel.DA_Commit_S[s][t0][h] -
thisOplModel.DAL_Commit_S[t0][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Commit_S[s][t0][h], 2); // 1

thisOplModel.sg_Startup_S[s][t0][h] = thisOplModel.DA_Startup_S[s][t0][h] -
thisOplModel.DAL_Startup_S[t0][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Startup_S[s][t0][h], 2); // 2

thisOplModel.sg_Shutdown_S[s][t0][h]= thisOplModel.DA_Shutdown_S[s][t0][h] -
thisOplModel.DAL_Shutdown_S[t0][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Shutdown_S[s][t0][h], 2);
// 3

thisOplModel.sg_Gen_S[s][t0][h] = thisOplModel.DA_Gen_S[s][t0][h] -
thisOplModel.DAL_Gen_S[t0][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Gen_S[s][t0][h], 2); // 4
}
for(var i in thisOplModel.Units_F){
thisOplModel.sg_Commit_F[s][t0][i] = thisOplModel.DA_Commit_F[s][t0][i] -
thisOplModel.DAL_Commit_F[t0][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Commit_F[s][t0][i], 2); // 5

thisOplModel.sg_Startup_F[s][t0][i] = thisOplModel.DA_Startup_F[s][t0][i] -
thisOplModel.DAL_Startup_F[t0][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Startup_F[s][t0][i], 2); // 6

thisOplModel.sg_Shutdown_F[s][t0][i] = thisOplModel.DA_Shutdown_F[s][t0][i] -
thisOplModel.DAL_Shutdown_F[t0][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Shutdown_F[s][t0][i], 2);
// 7

thisOplModel.sg_IndNonSres_F[s][t0][i] = thisOplModel.DA_IndNonSres_F[s][t0][i] -
thisOplModel.DAL_IndNonSres_F[t0][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_IndNonSres_F[s][t0][i], 2);
// 8

thisOplModel.sg_Gen_F[s][t0][i] = thisOplModel.DA_Gen_F[s][t0][i] -
thisOplModel.DAL_Gen_F[t0][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Gen_F[s][t0][i], 2); // 9
}
}
for(var t in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.sg_SresUp_S[s][t][h] = thisOplModel.DA_SresUp_S[s][t][h] -
thisOplModel.DAL_SresUp_S[t][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_SresUp_S[s][t][h], 2); // 10

thisOplModel.sg_SresDown_S[s][t][h] = thisOplModel.DA_SresDown_S[s][t][h] -
thisOplModel.DAL_SresDown_S[t][h];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_SresDown_S[s][t][h], 2); // 11
}
}
for(var i in thisOplModel.Units_F){

```

```

thisOplModel.sg_SresUp_F[s][t][i] = thisOplModel.DA_SresUp_F[s][t][i] -
thisOplModel.DAL_SresUp_F[t][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_SresUp_F[s][t][i], 2); //
12

thisOplModel.sg_SresDown_F[s][t][i] = thisOplModel.DA_SresDown_F[s][t][i] -
thisOplModel.DAL_SresDown_F[t][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_SresDown_F[s][t][i], 2); //
13

thisOplModel.sg_NonSres_F[s][t][i] = thisOplModel.DA_NonSres_F[s][t][i] -
thisOplModel.DAL_NonSres_F[t][i];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_NonSres_F[s][t][i], 2); //
14
}
for(var k in thisOplModel.WindF){
thisOplModel.sg_WindP[s][t][k] = thisOplModel.DA_WindP[s][t][k] -
thisOplModel.DAL_WindP[t][k];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_WindP[s][t][k], 2); // 15
}
for(var j in thisOplModel.Loads){
thisOplModel.sg_Load[s][t][j] = thisOplModel.DA_Load[s][t][j] - thisOplModel.DAL_Load[t][j];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Load[s][t][j], 2); // 16

thisOplModel.sg_LoadRU[s][t][j] = thisOplModel.DA_LoadRU[s][t][j] -
thisOplModel.DAL_LoadRU[t][j];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_LoadRU[s][t][j], 2); // 17

thisOplModel.sg_LoadRD[s][t][j] = thisOplModel.DA_LoadRD[s][t][j] -
thisOplModel.DAL_LoadRD[t][j];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_LoadRD[s][t][j], 2); // 18
}
for(var n in thisOplModel.Buses_N){
thisOplModel.sg_Delta[s][t][n] = thisOplModel.DA_Delta[s][t][n] -
thisOplModel.DAL_Delta[t][n];
denominator += thisOplModel.PiScen[s] * Math.pow(thisOplModel.sg_Delta[s][t][n], 2); // 19
}
}
}

if(iter == thisOplModel.phaselMaxIterations){
var mmm = 0;
for(s in thisOplModel.Scenar){
for(var t0 in thisOplModel.TimePeriods0){
for(var h in thisOplModel.Units_S){
thisOplModel.m_Commit_S[s][t0][h] = mmm;
thisOplModel.m_Startup_S[s][t0][h] = mmm;
thisOplModel.m_Shutdown_S[s][t0][h] = mmm;
thisOplModel.m_Gen_S[s][t0][h] = mmm;
}
for(var i in thisOplModel.Units_F){
thisOplModel.m_Commit_F[s][t0][i] = mmm;
thisOplModel.m_Startup_F[s][t0][i] = mmm;
thisOplModel.m_Shutdown_F[s][t0][i] = mmm;
thisOplModel.m_IndNonSres_F[s][t0][i] = mmm;
thisOplModel.m_Gen_F[s][t0][i] = mmm;
}
}
for(var t in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.m_SresUp_S[s][t][h] = mmm;
thisOplModel.m_SresDown_S[s][t][h] = mmm;
}
for(var i in thisOplModel.Units_F){
thisOplModel.m_SresUp_F[s][t][i] = mmm;
thisOplModel.m_SresDown_F[s][t][i] = mmm;
thisOplModel.m_NonSres_F[s][t][i] = mmm;
}
}
for(var k in thisOplModel.WindF){

```

```

thisOplModel.m_WindP[s][t][k]          = mmm;
}
for(var j in thisOplModel.Loads){
thisOplModel.m_Load[s][t][j] = mmm;
thisOplModel.m_LoadRU[s][t][j]      = mmm;
thisOplModel.m_LoadRD[s][t][j]      = mmm;
}
for(var n in thisOplModel.Buses_N){
thisOplModel.m_Delta[s][t][n]       = mmm;
}}
var lr_total_cost = thisOplModel.first_cost[iter];
for(var s in thisOplModel.Scenar){
lr_total_cost += thisOplModel.PiScen[s] * thisOplModel.second_cost[s]; }
var nominator = (lamdaSecond + thisOplModel.lamda_const) * Math.abs((Lstar - lr_total_cost));
var stepSize = nominator / denominator;
var gap = Math.abs((Lstar - lr_total_cost)/Lstar);
var sgnorm = Math.sqrt(denominator);
writeln(iter + " " + lr_total_cost + " " + gap + " " + sgnorm);
cofile.writeln(iter + " " + lr_total_cost + " " + gap + " " + sgnorm);
for(s in thisOplModel.Scenar){
for(var t0 in thisOplModel.TimePeriods0){
for(var h in thisOplModel.Units_S){
thisOplModel.m_Commit_S[s][t0][h] += stepSize * thisOplModel.sg_Commit_S[s][t0][h];
thisOplModel.m_Startup_S[s][t0][h] += stepSize * thisOplModel.sg_Startup_S[s][t0][h];
thisOplModel.m_Shutdown_S[s][t0][h] += stepSize * thisOplModel.sg_Shutdown_S[s][t0][h];
thisOplModel.m_Gen_S[s][t0][h] += stepSize * thisOplModel.sg_Gen_S[s][t0][h];
}
for(var i in thisOplModel.Units_F){
thisOplModel.m_Commit_F[s][t0][i] += stepSize * thisOplModel.sg_Commit_F[s][t0][i];
thisOplModel.m_Startup_F[s][t0][i] += stepSize * thisOplModel.sg_Startup_F[s][t0][i];
thisOplModel.m_Shutdown_F[s][t0][i] += stepSize * thisOplModel.sg_Shutdown_F[s][t0][i];
thisOplModel.m_IndNonSres_F[s][t0][i] += stepSize * thisOplModel.sg_IndNonSres_F[s][t0][i];
thisOplModel.m_Gen_F[s][t0][i] += stepSize * thisOplModel.sg_Gen_F[s][t0][i];
}}
}
for(var t in thisOplModel.TimePeriods){
for(var h in thisOplModel.Units_S){
thisOplModel.m_SresUp_S[s][t][h] += stepSize * thisOplModel.sg_SresUp_S[s][t][h];
thisOplModel.m_SresDown_S[s][t][h] += stepSize * thisOplModel.sg_SresDown_S[s][t][h];
}for(var i in thisOplModel.Units_F){
thisOplModel.m_SresUp_F[s][t][i] += stepSize * thisOplModel.sg_SresUp_F[s][t][i];
thisOplModel.m_SresDown_F[s][t][i] += stepSize * thisOplModel.sg_SresDown_F[s][t][i];
thisOplModel.m_NonSres_F[s][t][i] += stepSize * thisOplModel.sg_NonSres_F[s][t][i];
}
for(var k in thisOplModel.WindF){
thisOplModel.m_WindP[s][t][k] += stepSize * thisOplModel.sg_WindP[s][t][k];
}
for(var j in thisOplModel.Loads){
thisOplModel.m_Load[s][t][j] += stepSize * thisOplModel.sg_Load[s][t][j];
thisOplModel.m_LoadRU[s][t][j] += stepSize * thisOplModel.sg_LoadRU[s][t][j];
thisOplModel.m_LoadRD[s][t][j] += stepSize * thisOplModel.sg_LoadRD[s][t][j];
}
for(var n in thisOplModel.Buses_N){
thisOplModel.m_Delta[s][t][n] += stepSize * thisOplModel.sg_Delta[s][t][n];
}}}}
lrs_def.end();
lrs_solver.end();
lrs_source.end();
cofile.close();
}

```

10.2 First stage sub-problem

```

// external data required to have agnostic data initialization
string lp_source_file = ...;
string lp_data_file = ...;
string lr_problem_data_file = ...;
int verbose = ...;
int m0 = ...;

```

```

{string} Units_S = ...;
{string} Units_F = ...;
{string} Buses_N = ...;
{string} Buses_M = ...;
{string} WindF = ...;
{string} Loads = ...;
{string} Scenar = ...;
float PiScen [Scenar] = ...;

int MapT[Buses_N][Buses_M] = ...;
int MapGS[Buses_N][Units_S] = ...;
int MapGF[Buses_N][Units_F] = ...;
int MapL[Buses_N][Loads] = ...;
int MapW[Buses_N][WindF] = ...;
int T=...;
range TimePeriods = 1..T;
range TimePeriods0 = 0..T;

float CarbTx = ...;
float NOTx = ...; //NOx emission price
float SOTx = ...; //SO2 emission price

float HeatRate_S[Units_S] = ...; // Heat rate slow units//
float CarbEmissionR_S [Units_S] = ...; // CO2 Emission Rates from Slow Units//
float NOEmissionR_S [Units_S] = ...; //NOx emission rates for slow units
float SOEmissionR_S [Units_S] = ...; //SO2 emission rates for slow units
float FuelPrice_S[Units_S] = ...; // Fuel price for slow units ($/mmbtu)
float StartC_S [Units_S] = ...; // Fixed Start-up costs of slow units ($/start)//
float HeatRateSU_S [Units_S] = ...; //Startup heat input parameter as a fraction of capacity
(mmbtu/(MW*start)//
float SpinCap_S [Units_S] = ...; // Cap of up Spin Reserve offer of slow units//
float MaxCap_S [Units_S] = ...; // Max power output of slow unit//
float MinCap_S [Units_S] = ...; // Min power output of slow unit//
float RampRate_S [Units_S] = ...; // Pos ramp rate of slow units//
int MinUpTime_S [Units_S] = ...; // Min up time of slow units//
int MinDownTime_S [Units_S] = ...; // Min down time of slow units//
int TUI_S[Units_S] = ...; //Minimum of InitUpIntsReq and T (to prevent range errors)*/
int TDI_S[Units_S] = ...; //Minimum of InitDownIntsReq and T (to prevent range errors)*/

float HeatRate_F[Units_F] = ...; // Heat rate Fast Units//
float CarbEmissionR_F [Units_F] = ...; // CO2 Emission Rates from Fast Units//
float NOEmissionR_F [Units_F] = ...;
float SOEmissionR_F [Units_F] = ...;
float FuelPrice_F[Units_F] = ...; // Fuel price for fast units($/mmbtu)
float StartC_F [Units_F] = ...; // Start-up costs of fast units//
float HeatRateSU_F [Units_F] = ...; //Startup heat input parameter as a fraction of capacity
(mmbtu/(MW*start)//
float SpinCap_F [Units_F] = ...; // Cap of up Spin Reserve offer of fast units//
float MaxCap_F [Units_F] = ...; // Max power output of fast unit//
float MinCap_F [Units_F] = ...; // Min power output of fast unit//
float RampRate_F [Units_F] = ...; // Pos ramp rate of fast units//
int MinUpTime_F [Units_F] = ...; // Min up time of fast units//
int MinDownTime_F [Units_F] = ...; // Min down time of fast units//
int TUI_F[Units_F] = ...; //Minimum of InitUpIntsReq and T (to prevent range errors)*/
int TDI_F[Units_F] = ...; //Minimum of InitDownIntsReq and T (to prevent range errors)*/
float MaxLoad [TimePeriods][Loads] = ...; // Maximum demand of load j in period t//
float MarginalU[Loads] = ...; // marginal utility of loads//
float LoadRUC [Loads] = ...; // Spinning reserve up offer prices of loads//
float LoadRDC [Loads] = ...; // Spinning reserve down offer prices of loads//
float MaxLoadRU [TimePeriods][Loads] = ...; // Cap of up Spin Reserve offer of loads//
float MaxLoadRD [TimePeriods][Loads] = ...; // Cap of Down Spin Reserve offer of loads//
// Other input parameters
float InstWindCap [WindF] = ...; // Installed Wind Capacity in Wind farm k
float Pi = ...; // Pi constraint for voltage angles
float B[Buses_N][Buses_M]= ...; // Susceptance of line between n and m
float TransCap[Buses_N][Buses_M] = ...; // Transmission line capacity
// precalculated costs for units
float CostSlow [Units_S];
float CostFast [Units_F];

```

```

float CostStartUpSlow[Units_S];
float CostStartUpFast[Units_F];
float CostProductionSlow[Units_S];
float CostProductionFast[Units_F];
// lagrangian multipliers
float m_Commit_S[Scenar][TimePeriods0][Units_S] = ...;
float m_Startup_S[Scenar][TimePeriods0][Units_S] = ...;
float m_Shutdown_S[Scenar][TimePeriods0][Units_S] = ...;
float m_Gen_S[Scenar][TimePeriods0][Units_S] = ...;
float m_SresUp_S[Scenar][TimePeriods][Units_S] = ...;
float m_SresDown_S[Scenar][TimePeriods][Units_S] = ...;
float m_Commit_F[Scenar][TimePeriods0][Units_F] = ...;
float m_Startup_F[Scenar][TimePeriods0][Units_F] = ...;
float m_Shutdown_F[Scenar][TimePeriods0][Units_F] = ...;
float m_IndNonSres_F[Scenar][TimePeriods0][Units_F] = ...;
float m_Gen_F[Scenar][TimePeriods0][Units_F] = ...;
float m_SresUp_F[Scenar][TimePeriods][Units_F] = ...;
float m_SresDown_F[Scenar][TimePeriods][Units_F] = ...;
float m_NonSres_F[Scenar][TimePeriods][Units_F] = ...;
float m_WindP[Scenar][TimePeriods][WindF] = ...;
float m_Load[Scenar][TimePeriods][Loads] = ...;
float m_LoadRU[Scenar][TimePeriods][Loads] = ...;
float m_LoadRD[Scenar][TimePeriods][Loads] = ...;
float m_Delta[Scenar][TimePeriods][Buses_N] = ...;
range TimePeriodsA = 1..(T-MinUpTime_F["Unit 37"]+1); // Middle_NonSres_periods_F
range TimePeriodsB = (TDI_F["Unit 37"]+1)..(T-MinDownTime_F["Unit 37"]+1);
//Middle_Mindown_periods_F
range TimePeriodsC = (T-MinDownTime_F["Unit 37"]+2)..T; //Final_Mindown_periods_F
range TimePeriodsD = 1..TUI_S["Unit 1"];
range TimePeriodsJ = 1..TDI_S["Unit 1"];
range TimePeriodsE = (TUI_S["Unit 1"]+1)..(T-MinUpTime_S["Unit 1"]+1);
range TimePeriodsF = (T - MinUpTime_S["Unit 1"] + 1)..T;
range TimePeriodsK = (TDI_S["Unit 1"]+1)..(T-MinDownTime_S["Unit 1"]+1);
range TimePeriodsL = (T-MinDownTime_S["Unit 1"]+2)..T;
range TimePeriodsP = (T-MinUpTime_F["Unit 37"]+2)..T;
//*****PREPROCESSING*****/
execute CPLX_PARAM {
cplex.epgap = 0.1;
}
execute INITIALIZE {
for(var h in Units_S){
CostSlow[h]
(FuelPrice_S[h]+CarbEmissionR_S[h]*CarbTx+NOEmissionR_S[h]*NOTx+SOEmissionR_S[h]*SOTx);
}
for(var i in Units_F){
CostFast[i]
(FuelPrice_F[i]+CarbEmissionR_F[i]*CarbTx+NOEmissionR_F[i]*NOTx+SOEmissionR_F[i]*SOTx);
}
for(var h in Units_S){
CostStartupSlow[h] = (StartC_S[h] + (HeatRateSU_S[h] * MaxCap_S[h]) * CostSlow[h]);
CostProductionSlow[h] = (HeatRate_S[h] * CostSlow[h]);
}
for(var i in Units_F){
CostStartupFast[i] = (StartC_F[i] + (HeatRateSU_F[i] * MaxCap_F[i]) * CostFast[i]);
CostProductionFast[i] = (HeatRate_F[i] * CostFast[i]);
}
}

//*****Declaration of decision variables
//first-stage decision variables */
dvar boolean DAL_Commit_S [TimePeriods0][Units_S];
dvar boolean DAL_Startup_S [TimePeriods0][Units_S];
dvar boolean DAL_Shutdown_S [TimePeriods0][Units_S];
dvar boolean DAL_Commit_F [TimePeriods0][Units_F];
dvar boolean DAL_Startup_F [TimePeriods0][Units_F];
dvar boolean DAL_Shutdown_F [TimePeriods0][Units_F];
dvar boolean DAL_IndNonSres_F [TimePeriods0][Units_F];
dvar float+ DAL_Gen_S [TimePeriods0][Units_S] ;
dvar float+ DAL_SresUp_S [TimePeriods][Units_S];

```

```

dvar float+ DAL_SresDown_S [TimePeriods][Units_S];
dvar float+ DAL_Gen_F [TimePeriods0][Units_F];
dvar float+ DAL_SresUp_F [TimePeriods][Units_F];
dvar float+ DAL_SresDown_F [TimePeriods][Units_F];
dvar float+ DAL_NonSres_F [TimePeriods][Units_F];
dvar float+ DAL_WindP [TimePeriods][WindF];
dvar float+ DAL_Load [TimePeriods][Loads];
dvar float+ DAL_LoadRU [TimePeriods][Loads];
dvar float+ DAL_LoadRD [TimePeriods][Loads];
dvar float DAL_Delta [TimePeriods][Buses_N] in -Pi..Pi;
// addition of shut down cost
dexpr float ShutDownSlow = sum(t in TimePeriods, h in Units_S) CostStartUpSlow[h]
* DAL_Shutdown_S[t][h];
dexpr float ShutDownFast = sum(t in TimePeriods, i in Units_F)
CostStartUpFast[i] * DAL_Shutdown_F[t][i];
dexpr float StartUpSlow = sum(t in TimePeriods, h in Units_S) CostStartUpSlow[h]
* DAL_Startup_S[t][h];
dexpr float ProductionSlow = sum(t in TimePeriods, h in Units_S) (DAL_Gen_S[t][h] *
CostProductionSlow[h]);
dexpr float ReservesSlow = sum(t in TimePeriods, h in Units_S)
((DAL_SresUp_S[t][h] + DAL_SresDown_S[t][h]) * (CostProductionSlow[h]/5));
dexpr float StartUpFast = sum(t in TimePeriods, i in Units_F) CostStartUpFast[i]
* DAL_Startup_F[t][i];
dexpr float ProductionFast = sum(t in TimePeriods, i in Units_F) (DAL_Gen_F[t][i] *
CostProductionFast[i]);
dexpr float ReservesFast = sum(t in TimePeriods, i in Units_F) (DAL_SresUp_F[t][i]
+ DAL_SresDown_F[t][i] + DAL_NonSres_F[t][i]) * (CostProductionFast[i]/5);
dexpr float profit = sum(t in TimePeriods, j in Loads)
(DAL_Load[t][j] * MarginalU[j] - DAL_LoadRU[t][j] * LoadRUC[j] - DAL_LoadRD[t][j] *
LoadRDC[j]);

// lagrangian expressions
dexpr float expr_Commit_S = sum(s in Scenar, t in TimePeriods0, h in Units_S)
(PiScen[s] * m_Commit_S[s][t][h] * DAL_Commit_S[t][h]);
dexpr float expr_Startup_S = sum(s in Scenar, t in TimePeriods0, h in Units_S)
(PiScen[s] * m_Startup_S[s][t][h] * DAL_Startup_S[t][h]);
dexpr float expr_Shutdown_S = sum(s in Scenar, t in TimePeriods0, h in Units_S) (PiScen[s]
* m_Shutdown_S[s][t][h] * DAL_Shutdown_S[t][h]);
dexpr float expr_Gen_S = sum(s in Scenar, t in TimePeriods0, h in
Units_S) (PiScen[s] * m_Gen_S[s][t][h] * DAL_Gen_S[t][h]);
dexpr float expr_SresUp_S = sum(s in Scenar, t in TimePeriods, h in Units_S)
(PiScen[s] * m_SresUp_S[s][t][h] * DAL_SresUp_S[t][h]);
dexpr float expr_SresDown_S = sum(s in Scenar, t in TimePeriods, h in Units_S)
(PiScen[s] * m_SresDown_S[s][t][h] * DAL_SresDown_S[t][h]);
dexpr float expr_Commit_F = sum(s in Scenar, t in TimePeriods0, i in Units_F)
(PiScen[s] * m_Commit_F[s][t][i] * DAL_Commit_F[t][i]);
dexpr float expr_Startup_F = sum(s in Scenar, t in TimePeriods0, i in Units_F)
(PiScen[s] * m_Startup_F[s][t][i] * DAL_Startup_F[t][i]);
dexpr float expr_Shutdown_F = sum(s in Scenar, t in TimePeriods0, i in Units_F) (PiScen[s]
* m_Shutdown_F[s][t][i] * DAL_Shutdown_F[t][i]);
dexpr float expr_IndNonSres_F = sum(s in Scenar, t in TimePeriods0, i in Units_F)
(PiScen[s] * m_IndNonSres_F[s][t][i] * DAL_IndNonSres_F[t][i]);
dexpr float expr_Gen_F = sum(s in Scenar, t in TimePeriods0, i in
Units_F) (PiScen[s] * m_Gen_F[s][t][i] * DAL_Gen_F[t][i]);
dexpr float expr_SresUp_F = sum(s in Scenar, t in TimePeriods, i in Units_F)
(PiScen[s] * m_SresUp_F[s][t][i] * DAL_SresUp_F[t][i]);
dexpr float expr_SresDown_F = sum(s in Scenar, t in TimePeriods, i in Units_F)
(PiScen[s] * m_SresDown_F[s][t][i] * DAL_SresDown_F[t][i]);
dexpr float expr_NonSres_F = sum(s in Scenar, t in TimePeriods, i in Units_F)
(PiScen[s] * m_NonSres_F[s][t][i] * DAL_NonSres_F[t][i]);
dexpr float expr_WindP = sum(s in Scenar, t in TimePeriods, k in WindF)
(PiScen[s] * m_WindP[s][t][k] * DAL_WindP[t][k]);
dexpr float expr_Load = sum(s in Scenar, t in TimePeriods, j in Loads)
(PiScen[s] * m_Load[s][t][j] * DAL_Load[t][j]);
dexpr float expr_LoadRU = sum(s in Scenar, t in TimePeriods, j in Loads)
(PiScen[s] * m_LoadRU[s][t][j] * DAL_LoadRU[t][j]);
dexpr float expr_LoadRD = sum(s in Scenar, t in TimePeriods, j in Loads)
(PiScen[s] * m_LoadRD[s][t][j] * DAL_LoadRD[t][j]);

```

```

dexpr float expr_Delta = sum(s in Scenar, t in TimePeriods, n in
Buses_N) (PiScen[s] * m_Delta[s][t][n] * DAL_Delta[t][n]);dexpr float lagrangian =
(expr_Commit_S +
expr_Startup_S +
expr_Shutdown_S +
expr_Gen_S +
expr_SresUp_S +
expr_SresDown_S +
expr_Commit_F +
expr_Startup_F +
expr_Shutdown_F +
expr_IndNonSres_F +
expr_Gen_F +
expr_SresUp_F +
expr_SresDown_F +
expr_NonSres_F +
expr_WindP +
expr_Load +
expr_LoadRU +
expr_LoadRD +
expr_Delta);
minimize
StartUpSlow + ProductionSlow + ReservesSlow + StartUpFast + ProductionFast + ReservesFast -
lagrangian - profit+ ShutDownSlow + ShutDownFast;
subject to {
forall(h in Units_S){
DAL_Gen_S[0][h] == 0;
DAL_Commit_S[0][h] == 0;
DAL_Startup_S[0][h] == 0;
DAL_Shutdown_S[0][h] == 0;
}
forall(i in Units_F){
DAL_Gen_F [0][i] == 0;
DAL_Commit_F [0][i] == 0;
DAL_Commit_F [0][i]==0;
DAL_Startup_F[0][i]==0;
DAL_Shutdown_F[0][i]==0;
DAL_IndNonSres_F[0][i]==0;
}
// first stage
forall( t in TimePeriods, h in Units_S) // MaxProductionLimitFirst_S
DAL_Gen_S [t][h] + DAL_SresUp_S[t][h] <= MaxCap_S [h] * DAL_Commit_S [t][h];
forall( t in TimePeriods, h in Units_S) // MinProductionLimitFirst_S
DAL_Gen_S [t][h] - DAL_SresDown_S[t][h] >= MinCap_S [h] * DAL_Commit_S [t][h];
forall( t in TimePeriods, i in Units_F) // MaxProductionLimitFirst_F
DAL_Gen_F [t][i] + DAL_SresUp_F[t][i] <= MaxCap_F [i] * DAL_Commit_F [t][i];
forall( t in TimePeriods, i in Units_F) // MinProductionLimitFirst_F
DAL_Gen_F [t][i] - DAL_SresDown_F [t][i]>= MinCap_F [i] * DAL_Commit_F [t][i];
forall( t in TimePeriods, j in Loads) // MaxLoadsLimitFirst
DAL_Load [t][j] <= MaxLoad [t][j];
forall( t in TimePeriods, j in Loads) // LoadsSpinResUpLimit
DAL_LoadRU[t][j] <= MaxLoadRU[t][j];
forall( t in TimePeriods, j in Loads) // LoadsSpinResDownLimit
DAL_LoadRD[t][j] <= MaxLoadRD[t][j];
forall ( t in TimePeriods, k in WindF) // WindProductionBounds
DAL_WindP [t][k] <= InstWindCap [k];
forall( t in TimePeriods, h in Units_S)// SpinResUpLimit_S
DAL_SresUp_S[t][h] <= (SpinCap_S [h]*DAL_Commit_S [t][h]);
forall( t in TimePeriods, h in Units_S) // SpinResDownLimit_S
DAL_SresDown_S[t][h] <= (SpinCap_S [h]*DAL_Commit_S [t][h]);
forall( t in TimePeriods, i in Units_F) // SpinResUpLimit
DAL_SresUp_F[t][i] <= (SpinCap_F [i]*DAL_Commit_F[t][i]);
forall( t in TimePeriods, i in Units_F) // SpinResDownLimit_F
DAL_SresDown_F[t][i] <= (SpinCap_F [i]*DAL_Commit_F [t][i]);
forall( t in TimePeriods, i in Units_F) // NonSpinResLimitI
DALCommit_F[t][i]+DAL_IndNonSres_F[t][i]<=1;
forall( t in TimePeriods, i in Units_F) // NonSpinResLimitII
DAL_NonSres_F[t][i] <= SpinCap_F [i]*DAL_IndNonSres_F[t][i];
forall( t in TimePeriods, i in Units_F) //14// NonSpinResLimitIII[t][i]:

```

```

-DAL_NonSres_F[t][i] <= -(MinCap_F [i] * DAL_IndNonSres_F[t][i]);
forall( t in TimePeriods, n in Buses_N, m in Buses_M) // DA_TransmissionLineCapI
B[n][m]*(DAL_Delta[t][n]-DAL_Delta[t][m])<=TransCap[n][m];
forall( t in TimePeriods, n in Buses_N, m in Buses_M) // DA_TransmissionLineCapII
B[n][m]*(DAL_Delta[t][m]-DAL_Delta[t][n])<=TransCap[n][m];
// first stage equality constraints
forall( t in TimePeriods, h in Units_S) // DA_LogicalConstrSlowUnits
DAL_Commit_S[t-1][h] -DAL_Commit_S[t][h] + DAL_Startup_S[t][h] - DAL_Shutdown_S[t][h] == 0 ;
forall( t in TimePeriods, i in Units_F) // DA_LogicalConstrFastUnits
DAL_Commit_F[t-1][i] -DAL_Commit_F[t][i] + DAL_Startup_F[t][i] - DAL_Shutdown_F[t][i] == 0 ;
forall( t in TimePeriods, n in Buses_N) // DA_BusBalanceConstr
sum(h in Units_S) (DAL_Gen_S[t][h]*MapGS[n][h])
+sum(i in Units_F) (DAL_Gen_F[t][i]*MapGF[n][i])
+sum(k in WindF) (DAL_WindP[t][k]*MapW[n][k])
-sum(j in Loads) (DAL_Load[t][j]*MapL[n][j])
-sum(m in Buses_M) (B[n][m]*(DAL_Delta[t][n]-DAL_Delta[t][m])*MapT[n][m]) ==0 ;
forall( t in TimePeriods) // DAREfVoltageAngleConstr
DAL_Delta[t]["Bus 1"]==0;
forall(i in Units_F, t in TimePeriodsA) // Middle_NonSres_periods_F
sum (k in t..(t+MinUpTime_F[i]-1)) DAL_IndNonSres_F [k][i] >= MnUpTime_F[i]*(DAL_IndNonSres_F
[t][i]-DAL_IndNonSres_F[t-1][i]);
forall(i in Units_F, t in TimePeriodsP) // Final_NonSres_periods_F
sum (k in t..T) (DAL_IndNonSres_F[k][i]-DAL_IndNonSres_F [t][i] + DAL_IndNonSres_F [t-1][i] )
>= 0;
forall(i in Units_F, t in TimePeriodsB) // Middle_Mindown_periods_F
sum (k in t..(t+1)) (1 - DAL_Commit_F [k][i]-DAL_IndNonSres_F[k][i]) >=
MinDownTime_F[i]*DAL_Shutdown_F [t][i];
forall(i in Units_F, t in TimePeriodsC) // Final_Mindown_periods_F
sum (k in t..T) (1 - DAL_Commit_F [k][i] -DAL_IndNonSres_F[k][i]- DAL_Shutdown_F [t-1][i]) >=
0;
forall(h in Units_S) // InitialMinupSlow
sum (t in TimePeriodsD) (1 - DAL_Commit_S [t][h]) == 0;
forall(h in Units_S, t in TimePeriodsE) // MiddleMinupSlow
sum (k in t..(t+MinUpTime_S[h]-1)) (DAL_Commit_S [k][h]) >=
MinUpTime_S[h]*DAL_Startup_S[t][h];
forall(h in Units_S, t in TimePeriodsF) // FinalMinupSlow
sum (k in t..T) (DAL_Commit_S [k][h] - DAL_Startup_S [t][h]) >= 0;
forall(h in Units_S) // InitialMindownSlow
sum (t in TimePeriodsJ) (DAL_Commit_S[t][h]) == 0;
forall(h in Units_S, t in TimePeriodsL) // FinalMindownSlow
sum (k in t..T) (1 - DAL_Commit_S [k][h] - DAL_Shutdown_S [t][h]) >= 0;
forall(h in Units_S, t in TimePeriodsK) // MiddleMindownSlow
sum (k in t..(t+MinDownTime_S[h]-1)) (1 - DAL_Commit_S[k][h]) >=
MinDownTime_S[h]*DAL_Shutdown_S [t][h];
}

```

10.3 Second stage sub-problem

```

// external data required to have agnostic data initialization
string lp_source_file = ...;
string lp_data_file = ...;
string lr_problem_data_file = ...;
int verbose = ...;
int m0 = ...;
//~
{string} Scenar = ...;
float PiScen [Scenar] = ...;
string currentScenar = ...;
{string} Units S = ...;

```

```

{string} Units_F = ...;
{string} Buses_N = ...;
{string} Buses_M = ...;
{string} WindF = ...;
{string} Loads = ...;
int MapT[Buses_N][Buses_M] = ...;
int MapGS[Buses_N][Units_S] = ...;
int MapGF[Buses_N][Units_F] = ...;
int MapL[Buses_N][Loads] = ...;
int MapW[Buses_N][WindF] = ...;
int T=...;
range TimePeriods = 1..T;
range TimePeriods0 = 0..T;
//Ending status of previous period//
float CarbTx = ...;
float NOTx = ...; //NOx emission price
float SOTx = ...; //SO2 emission price
float HeatRate_S[Units_S] = ...; // Heat rate slow units//
float CarbEmissionR_S [Units_S] = ...; // CO2 Emission Rates from Slow Units//
float NOEmissionR_S [Units_S] = ...; //NOx emission rates for slow units
float SOEmissionR_S [Units_S] = ...; //SO2 emission rates for slow units
float FuelPrice_S[Units_S] = ...; // Fuel price for slow units ($/mmbtu)
float StartC_S [Units_S] = ...; // Fixed Start-up costs of slow units ($/start)//
float HeatRateSU_S [Units_S] = ...; //Startup heat input parameter as a fraction of
capacity (mmbtu/(MW*start))//
float SpinCap_S [Units_S] = ...; // Cap of up Spin Reserve offer of slow units//
float MaxCap_S [Units_S] = ...; // Max power output of slow unit//
float MinCap_S [Units_S] = ...; // Min power output of slow unit//
float RampRate_S [Units_S] = ...; // Pos ramp rate of slow units//
int MinUpTime_S [Units_S] = ...; // Min up time of slow units//
int MinDownTime_S [Units_S] = ...; // Min down time of slow units//
int TUI_S[Units_S] = ...; //Minimum of InitUpIntsReq and T (to prevent range errors)*//
int TDI_S[Units_S] = ...; //Minimum of InitDownIntsReq and T (to prevent range errors)*//
float HeatRate_F[Units_F] = ...; // Heat rate Fast Units//
float CarbEmissionR_F [Units_F] = ...; // CO2 Emission Rates from Fast Units//
float NOEmissionR_F [Units_F] = ...;
float SOEmissionR_F [Units_F] = ...;
float FuelPrice_F[Units_F] = ...; // Fuel price for fast units($/mmbtu)
float StartC_F [Units_F] = ...; // Start-up costs of fast units//
float HeatRateSU_F [Units_F] = ...; //Startup heat input parameter as a fraction of
capacity (mmbtu/(MW*start))//
float SpinCap_F [Units_F] = ...; // Cap of up Spin Reserve offer of fast units//
float MaxCap_F [Units_F] = ...; // Max power output of fast unit//
float MinCap_F [Units_F] = ...; // Min power output of fast unit//
float RampRate_F [Units_F] = ...; // Pos ramp rate of fast units//
int MinUpTime_F [Units_F] = ...; // Min up time of fast units//
int MinDownTime_F [Units_F] = ...; // Min down time of fast units//
int TUI_F[Units_F] = ...; //Minimum of InitUpIntsReq and T (to prevent range errors)*//
int TDI_F[Units_F] = ...; //Minimum of InitDownIntsReq and T (to prevent range errors)*//
float MaxLoad [TimePeriods][Loads] = ...; // Maximum demand of load j in period t//
float MarginalU[Loads] = ...; // marginal utility of loads//
float LoadRUC [Loads] = ...; // Spinning reserve up offer prices of loads//
float LoadRDC [Loads] = ...; // Spinning reserve down offer prices of loads//
float MaxLoadRU [TimePeriods][Loads] = ...; // Cap of up Spin Reserve offer of loads//
float MaxLoadRD [TimePeriods][Loads] = ...; // Cap of Down Spin Reserve offer of loads//
// Other input parameters
float VOLL = ...; //Value of lost load for Loads j
float WspillageC [WindF] = ...; //Cost of wind power spillage in Wind farm k [$/MWh].
float InstWindCap [WindF] = ...; // Installed Wind Capacity in Wind farm k
float Pi = ...; // Pi constraint for voltage angles
float B[Buses_N][Buses_M]= ...; // Susceptance of line between n and m
float TransCap[Buses_N][Buses_M] = ...; // Transmission line capacity
// precalculated costs for units
float CostSlow [Units_S];
float CostFast [Units_F];
float CostStartUpSlow[Units_S];
float CostStartUpFast[Units_F];
float CostProductionSlow[Units_S];
float CostProductionFast[Units_F];

```

```

//random variable
float WindP [TimePeriods][WindF] = ...; //Wind power generation of Wind farm k in period t
and in scenario W.
// lagrangian multipliers
float m_Commit_S[Scenar][TimePeriods0][Units_S] =...;
float m_Startup_S[Scenar][TimePeriods0][Units_S]=...;
float m_Shutdown_S[Scenar][TimePeriods0][Units_S]=...;
float m_Gen_S[Scenar][TimePeriods0][Units_S]=...;
float m_SresUp_S[Scenar][TimePeriods][Units_S]=...;
float m_SresDown_S[Scenar][TimePeriods][Units_S]=...;
float m_Commit_F[Scenar][TimePeriods0][Units_F]=...;
float m_Startup_F[Scenar][TimePeriods0][Units_F]=...;
float m_Shutdown_F[Scenar][TimePeriods0][Units_F]=...;
float m_IndNonSres_F[Scenar][TimePeriods0][Units_F]=...;
float m_Gen_F[Scenar][TimePeriods0][Units_F]=...;
float m_SresUp_F[Scenar][TimePeriods][Units_F]=...;
float m_SresDown_F[Scenar][TimePeriods][Units_F]=...;
float m_NonSres_F[Scenar][TimePeriods][Units_F]=...;
float m_WindP[Scenar][TimePeriods][WindF]=...;
float m_Load[Scenar][TimePeriods][Loads]=...;
float m_LoadRU[Scenar][TimePeriods][Loads]=...;
float m_LoadRD[Scenar][TimePeriods][Loads]=...;
float m_Delta[Scenar][TimePeriods][Buses_N]=...;

range TimePeriodsN = (TDI_F["Unit 37"]+1)..(T-MinDownTime_F["Unit 37"]+1);
range TimePeriodsG = 1..TUI_F["Unit 37"];
range TimePeriodsM = 1..TDI_F["Unit 37"];
range TimePeriodsH = (TUI_F["Unit 37"]+1)..(T-MinUpTime_F["Unit 37"]+1);
range TimePeriodsI = (T-MinUpTime_F["Unit 37"]+1)..T;
range TimePeriodsO = (T-MinDownTime_F["Unit 37"]+2)..T;
//*****PREPROCESSING*****/
execute CPLX_PARAM {
cplex.epgap = 0.1;
}
execute INITIALIZE {
for(var h in Units_S){
CostSlow[h] =
(FuelPrice_S[h]+CarbEmissionR_S[h]*CarbTx+NOEmissionR_S[h]*NOTx+SOEmissionR_S[h]*SOTx);
}
for(var i in Units_F){
CostFast[i] =
(FuelPrice_F[i]+CarbEmissionR_F[i]*CarbTx+NOEmissionR_F[i]*NOTx+SOEmissionR_F[i]*SOTx);
}
for(var h in Units_S){
CostStartupSlow[h] = (StartC_S[h] + (HeatRateSU_S[h] * MaxCap_S[h]) * CostSlow[h]);
CostProductionSlow[h] = (HeatRate_S[h]* CostSlow[h]);
}
for(var i in Units_F){
CostStartupFast[i] = (StartC_F[i] + (HeatRateSU_F[i] * MaxCap_F[i]) * CostFast[i]);
CostProductionFast[i] = (HeatRate_F[i]* CostFast[i]);
//first-stage decision variables */
dvar boolean DA_Commit_S [TimePeriods0][Units_S]; // first-stage commitment variable for
Slow units//
dvar boolean DA_Startup_S [TimePeriods0][Units_S]; // first-stage startup variable for
Slow units//
dvar boolean DA_Shutdown_S [TimePeriods0][Units_S]; // first-stage shutdown variable for
Slow units//
dvar boolean DA_Commit_F [TimePeriods0][Units_F]; // first-stage commitment variable for
fast units//
dvar boolean DA_Startup_F [TimePeriods0][Units_F]; // first-stage startup variable for
fast units//
dvar boolean DA_Shutdown_F [TimePeriods0][Units_F]; // first-stage shutdown variable for
fast units//
dvar boolean DA_IndNonSres_F [TimePeriods0][Units_F];
dvar float+ DA_Gen_S [TimePeriods0][Units_S] ; // Power output scheduled for slow unit h
in t period[MW]//
dvar float+ DA_SresUp_S [TimePeriods][Units_S]; //Up spinning reserve scheduled for slow
unit h in t//

```

```

dvar float+ DA_SresDown_S [TimePeriods][Units_S]; //Down spinning reserve scheduled for
slow unit h in t//
dvar float+ DA_Gen_F [TimePeriods0][Units_F]; // Power output scheduled for fast unit i
in t period[MW]//
dvar float+ DA_SresUp_F [TimePeriods][Units_F]; // Up spinning reserve scheduled for fast
unit i in t//
dvar float+ DA_SresDown_F [TimePeriods][Units_F]; // Down spinning reserve scheduled for
fast unit i in t//
dvar float+ DA_NonSres_F [TimePeriods][Units_F]; // Non-spinning reserve scheduled for
fast unit i in t//
dvar float+ DA_WindP [TimePeriods][WindF]; // Scheduled wind power for wind farm k in
period t [MW]//
dvar float+ DA_Load [TimePeriods][Loads]; // Scheduled power for load j in period t//
dvar float+ DA_LoadRU [TimePeriods][Loads]; // Up spinning reserve scheduled for load j in
t//
dvar float+ DA_LoadRD [TimePeriods][Loads]; // Down spinning reserve scheduled for load j
in t//
dvar float DA_Delta [TimePeriods][Buses_N] in -Pi..Pi; // Voltage angle at Bus n in
t//
//Second-stage decision variables//
dvar boolean BM_Commit_F [TimePeriods0][Units_F]; // second-stage commitment variable for
fast units//
dvar boolean BM_Startup_F [TimePeriods][Units_F]; // second-stage startup variable for
fast units//
dvar boolean BM_Shutdown_F [TimePeriods][Units_F]; // second-stage shutdown variable for
fast units//
dvar float+ BM_Gen_S [TimePeriods0][Units_S]; // Power output of slow unit h in period t
and scenario s [MW]//
dvar float+ BM_SresUp_S [TimePeriods][Units_S]; // Spinning reserve up deployed by slow
unit h in period t and scenario s [MW]//
dvar float+ BM_SresDown_S [TimePeriods][Units_S]; // Spinning reserve down deployed by
slow unit h in period t and scenario s [MW]//
dvar float+ BM_RGenB_S [TimePeriods][Units_S]; // Reserve deployed from the b-th block of
energy offered by slow unit h in period t and scenario s [MW]//

dvar float+ BM_Gen_F [TimePeriods0][Units_F]; // Power output of fast unit i in period t
and scenario s [MW]//
dvar float+ BM_SresUp_F [TimePeriods][Units_F]; // Spinning reserve up deployed by fast
unit i in period t and scenario s [MW]//
dvar float+ BM_SresDown_F [TimePeriods][Units_F]; // Spinning reserve down deployed by
fast unit i in period t and scenario s [MW]//
dvar float+ BM_NonSres_F [TimePeriods][Units_F]; // Non-Spinning reserve deployed by fast
unit i in period t and scenario s [MW]//
dvar float+ BM_RGenB_F [TimePeriods][Units_F]; // Reserve deployed from the b-th block of
energy offered by fast unit i in period t and scenario s [MW]//
dvar float+ BM_Load [TimePeriods][Loads]; // Power consumed by load j in period t and
scenario s [MW]//
dvar float+ BM_LoadRU [TimePeriods][Loads]; // Spinning reserve up deployed by load j in
period t and scenario s [MW]//
dvar float+ BM_LoadRD [TimePeriods][Loads]; // Spinning reserve down deployed by load j in
period t and scenario s [MW]//
dvar float+ BM_Lshed [TimePeriods][Loads]; // Load shedding imposed on consumer j in period
t and scenario s [MW]//
dvar float+ BM_WindSpilg [TimePeriods][WindF]; // Wind power generation spillage of wind
farm k in period t and scenario s [MW]//
dvar float BM_Delta [TimePeriods][Buses_N] in - Pi..Pi; // Voltage angle at Bus N in t
and scenario s//
dvar float BM_TF [TimePeriods][Buses_N][Buses_M]; // Transmission Flow in balancing market
in scenario s at time t from Bus n to Bus m [MW]//
dexpr float productionSlow = sum(t in TimePeriods, h in Units_S)
(CostProductionSlow[h] * BM_RGenB_S[t][h]);
dexpr float productionFast = sum(t in TimePeriods, i in Units_F)
(CostProductionFast[i] * BM_RGenB_F[t][i]);
dexpr float adjStartUpFast = sum(t in TimePeriods, i in Units_F) ((StartC_F[i] +
(HeatRateSU_F[i] * MaxCap_F[i]) * CostFast[i]) * (BM_Startup_F[t][i] -
DA_Startup_F[t][i]));
dexpr float marginalUtility = sum(t in TimePeriods, j in Loads) (MarginalU[j] *
(BM_LoadRU[t][j] - BM_LoadRD[t][j]));

```

```

dexpr float costLoadShed      =      sum(t in TimePeriods, j in Loads) (BM_Lshed[t][j] *
VOLL );
dexpr float costWindSpill    =      sum(t in TimePeriods, k in WindF) (BM_WindSpilg[t][k]
* WspillageC[k]);
// lagrangian expressions
dexpr float expr_Commit_S    = sum(t in TimePeriods0, h in Units_S)
(m_Commit_S[currentScenar][t][h] * DA_Commit_S[t][h]);
dexpr float expr_Startup_S   = sum(t in TimePeriods0, h in Units_S)
(m_Startup_S[currentScenar][t][h] * DA_Startup_S[t][h]);
dexpr float expr_Shutdown_S = sum(t in TimePeriods0, h in Units_S)
(m_Shutdown_S[currentScenar][t][h] * DA_Shutdown_S[t][h]);
dexpr float expr_Gen_S      = sum(t in TimePeriods0, h in Units_S)
(m_Gen_S[currentScenar][t][h] * DA_Gen_S[t][h]);
dexpr float expr_SresUp_S   = sum(t in TimePeriods, h in Units_S)
(m_SresUp_S[currentScenar][t][h] * DA_SresUp_S[t][h]);
dexpr float expr_SresDown_S = sum(t in TimePeriods, h in Units_S)
(m_SresDown_S[currentScenar][t][h] * DA_SresDown_S[t][h]);
dexpr float expr_Commit_F   = sum(t in TimePeriods0, i in Units_F)
(m_Commit_F[currentScenar][t][i] * DA_Commit_F[t][i]);
dexpr float expr_Startup_F  = sum(t in TimePeriods0, i in Units_F)
(m_Startup_F[currentScenar][t][i] * DA_Startup_F[t][i]);
dexpr float expr_Shutdown_F = sum(t in TimePeriods0, i in Units_F)
(m_Shutdown_F[currentScenar][t][i] * DA_Shutdown_F[t][i]);
dexpr float expr_IndNonSres_F = sum(t in TimePeriods0, i in Units_F)
(m_IndNonSres_F[currentScenar][t][i] * DA_IndNonSres_F[t][i]);
dexpr float expr_Gen_F     = sum(t in TimePeriods0, i in Units_F)
(m_Gen_F[currentScenar][t][i] * DA_Gen_F[t][i]);
dexpr float expr_SresUp_F   = sum(t in TimePeriods, i in Units_F)
(m_SresUp_F[currentScenar][t][i] * DA_SresUp_F[t][i]);
dexpr float expr_SresDown_F = sum(t in TimePeriods, i in Units_F)
(m_SresDown_F[currentScenar][t][i] * DA_SresDown_F[t][i]);
dexpr float expr_NonSres_F  = sum(t in TimePeriods, i in Units_F)
(m_NonSres_F[currentScenar][t][i] * DA_NonSres_F[t][i]);
dexpr float expr_WindP     = sum(t in TimePeriods, k in WindF)
(m_WindP[currentScenar][t][k] * DA_WindP[t][k]);
dexpr float expr_Load      = sum(t in TimePeriods, j in Loads)
(m_Load[currentScenar][t][j] * DA_Load[t][j]);
dexpr float expr_LoadRU    = sum(t in TimePeriods, j in Loads)
(m_LoadRU[currentScenar][t][j] * DA_LoadRU[t][j]);
dexpr float expr_LoadRD    = sum(t in TimePeriods, j in Loads)
(m_LoadRD[currentScenar][t][j] * DA_LoadRD[t][j]);
dexpr float expr_Delta     = sum(t in TimePeriods, n in Buses_N)
(m_Delta[currentScenar][t][n] * DA_Delta[t][n]);
dexpr float lagrangian = expr_Commit_S +
expr_Startup_S +
expr_Shutdown_S +
expr_Gen_S +
expr_SresUp_S +
expr_SresDown_S +
expr_Commit_F +
expr_Startup_F +
expr_Shutdown_F +
expr_IndNonSres_F +
expr_Gen_F +
expr_SresUp_F +
expr_SresDown_F +
expr_NonSres_F +
expr_WindP +
expr_Load +
expr_LoadRU +
expr_LoadRD +
expr_Delta;
minimize
adjStartUpFast + productionSlow + productionFast + marginalUtility + costLoadShed +
costWindSpill + lagrangian;
subject to {
// strengthening the first stage variables so they are not free
forall( t in TimePeriods, n in Buses_N) //19//      DA_BusBalanceConstr[t][n]:
sum(h in Units_S) (DA_Gen_S[t][h]*MapGS[n][h])

```

```

+sum(i in Units_F) (DA_Gen_F[t][i]*MapGF[n][i])
+sum(k in WindF) (DA_WindP[t][k]*MapW[n][k])
-sum(j in Loads) (DA_Load[t][j]*MapL[n][j])
-sum(m in Buses_M) (B[n][m]*(DA_Delta[t][n]-DA_Delta[t][m])*MapT[n][m]) ==0 ;
forall( t in TimePeriods, i in Units_F) //13//      NonSpinResLimitII[t][i]:
DA_NonSres_F[t][i] <= SpinCap_F [i]*DA_IndNonSres_F[t][i];
forall( t in TimePeriods, h in Units_S) // 1//      MaxProductionLimitFirst_S[t][h]:
DA_Gen_S [t][h] + DA_SresUp_S[t][h] <= MaxCap_S [h] * DA_Commit_S [t][h];
forall( t in TimePeriods, h in Units_S) // 2//      MinProductionLimitFirst_S[t][h]:
DA_Gen_S [t][h] - DA_SresDown_S[t][h] >= MinCap_S [h] * DA_Commit_S [t][h];
forall( t in TimePeriods, i in Units_F) // 4//
      MaxProductionLimitFirst_F[t][i]:
DA_Gen_F [t][i] + DA_SresUp_F[t][i] <= MaxCap_F [i] * DA_Commit_F [t][i];
forall( t in TimePeriods, i in Units_F) // 5 //      MinProductionLimitFirst_F[t][i]:
DA_Gen_F [t][i] - DA_SresDown_F [t][i]>= MinCap_F [i] * DA_Commit_F [t][i];
forall( t in TimePeriods, j in Loads) //6//      MaxLoadsLimitFirst[t][j]:
DA_Load [t][j] <= MaxLoad [t][j];
forall( t in TimePeriods, j in Loads) //15//      LoadsSpinResUpLimit[t][j]:
DA_LoadRU[t][j] <= MaxLoadRU[t][j];
forall( t in TimePeriods, j in Loads) //16//      LoadsSpinResDownLimit[t][j]:
DA_LoadRD[t][j] <= MaxLoadRD[t][j];
forall(t in TimePeriods, k in WindF)
DA_WindP[t][k] <= InstWindCap[k];
forall(h in Units_S){
DA_Gen_S[0][h] == 0;
BM_Gen_S[0][h] == 0;
DA_Commit_S[0][h] == 0;
DA_Startup_S[0][h] == 0;
DA_Shutdown_S[0][h] == 0;
}
forall(i in Units_F){
DA_Gen_F [0][i] == 0;
DA_Commit_F [0][i] == 0;
BM_Gen_F [0][i] == 0;
BM_Commit_F [0][i] == 0;
DA_Commit_F [0][i]==0;
DA_Startup_F[0][i]==0;
DA_Shutdown_F[0][i]==0;
DA_IndNonSres_F[0][i]==0;
}
// Second stage constraint
forall(t in TimePeriods, h in Units_S) //26//      MaxProductionLimSecnd_S[t][h]:
BM_Gen_S [t][h] <= MaxCap_S [h]*DA_Commit_S [t][h];
forall(t in TimePeriods, h in Units_S) //27//      MinProductionLimSecnd_S[t][h]:
-BM_Gen_S [t][h] <= -(MinCap_S [h]*DA_Commit_S[t][h]);
forall(t in TimePeriods, i in Units_F) //28//      MaxProductionLimSecnd_F[t][i]:
BM_Gen_F [t][i] <= MaxCap_F [i]*BM_Commit_F [t][i];
forall(t in TimePeriods, i in Units_F) //29//      MinProductionLimSecnd_F[t][i]:
-BM_Gen_F [t][i] <= -(MinCap_F [i]*BM_Commit_F [t][i]);
forall(t in TimePeriods, h in Units_S)//      RampupSlow[t][h]:
BM_Gen_S [t][h]-BM_Gen_S [t-1][h]<= RampRate_S [h]*DA_Commit_S[t-1][h] + RampRate_S
[h]*DA_Startup_S[t][h];
forall(t in TimePeriods, h in Units_S)//      RampdownSlow[t][h]:
BM_Gen_S [t-1][h]- BM_Gen_S [t][h]<= RampRate_S [h]*DA_Commit_S[t][h] + RampRate_S
[h]*DA_Shutdown_S[t][h];
forall(t in TimePeriods, i in Units_F)//      RampupFast[t][i]:
BM_Gen_F [t][i]- BM_Gen_F [t-1][i]<= RampRate_F [i]*BM_Commit_F[t-1][i] + RampRate_F
[i]*BM_Startup_F[t][i];
forall(t in TimePeriods, i in Units_F)//      RampdownFast[t][i]:
BM_Gen_F [t-1][i]- BM_Gen_F [t][i]<= RampRate_F [i]*BM_Commit_F[t][i] + RampRate_F
[i]*BM_Shutdown_F[t][i];
forall(t in TimePeriods, h in Units_S) //35 //      LeftCapLimits_S[t][h]:
BM_RGenB_S [t][h] <= MaxCap_S [h] - DA_Gen_S [t][h];
forall(t in TimePeriods, h in Units_S) //36 //      Blocklimits_S[t][h]:
-BM_RGenB_S [t][h] <= DA_Gen_S [t][h];
forall(t in TimePeriods, i in Units_F) //38//      LeftCapLimits_F[t][i]:
BM_RGenB_F[t][i] <= MaxCap_F [i]- DA_Gen_F [t][i];
forall(t in TimePeriods, i in Units_F) //39 //      Blocklimits_F[t][i]:
-BM_RGenB_F [t][i] <= DA_Gen_F [t][i];

```

```

forall(t in TimePeriods, h in Units_S) //41// SpinReserveUpDeploym_S[t][h]:
BM_SresUp_S [t][h] <= DA_SresUp_S[t][h];
forall(t in TimePeriods, h in Units_S) //42// SpinReserveDownDeploym_S[t][h]:
BM_SresDown_S [t][h] <= DA_SresDown_S[t][h];
forall(t in TimePeriods, n in Buses_N, m in Buses_M) //55//
BM_TransmissionLineCapI[t][n][m]:
BM_TF[t][n][m] <= TransCap[n][m];
forall(t in TimePeriods, n in Buses_N, m in Buses_M) //56//
BM_TransmissionLineCapII[t][n][m]:
-BM_TF[t][n][m] <= TransCap[n][m];
forall(t in TimePeriods, i in Units_F) //44// SpinReserveUpDeploym_F[t][i]:
BM_SresUp_F [t][i] <= DA_SresUp_F [t][i];
forall(t in TimePeriods, i in Units_F) //45// SpinReserveDownDeploym_F[t][i]:
BM_SresDown_F [t][i] <= DA_SresDown_F [t][i];
forall(t in TimePeriods, i in Units_F) //46// NonSpinReserveDeploym[t][i]:
BM_NonSres_F [t][i] <= DA_NonSres_F [t][i];
forall(t in TimePeriods, j in Loads) //47// LoadSheddingBound[t][j]:
BM_Lshed [t][j] <= BM_Load [t][j];
forall(t in TimePeriods, k in WindF) //48// WindSpillageBound[t][k]:
BM_WindSpilg [t][k] <= WindP[t][k];
forall(t in TimePeriods, j in Loads) //49// LoadsResUpConstr[t][j]:
BM_LoadRU[t][j]<=DA_LoadRU[t][j];
forall(t in TimePeriods, j in Loads) // 50// LoadsResDownConstr[t][j]:
BM_LoadRD[t][j]<=DA_LoadRD[t][j];
forall(t in TimePeriods, h in Units_S) //34// DecompA_S[t][h]:
BM_RGenB_S [t][h] == BM_SresUp_S [t][h] - BM_SresDown_S [t][h];

forall(t in TimePeriods, i in Units_F) //37// DecompA_F[t][i]:
BM_RGenB_F [t][i] == BM_SresUp_F [t][i]+ BM_NonSres_F [t][i] -BM_SresDown_F [t][i];
forall(t in TimePeriods, h in Units_S) //40// DecompositionGenPowrOutput_S[t][h]:
BM_Gen_S [t][h] == DA_Gen_S[t][h] + BM_SresUp_S [t][h] - BM_SresDown_S [t][h];
forall(t in TimePeriods, i in Units_F) //43// DecompositionGenPowrOutput_F[t][i]:
BM_Gen_F [t][i] == DA_Gen_F[t][i] + BM_SresUp_F [t][i]+ BM_NonSres_F [t][i] -BM_SresDown_F
[t][i];
forall(t in TimePeriods, j in Loads) //51// DecompLoadsConstr[t][j]:
BM_Load [t][j]== DA_Load [t][j] - BM_LoadRU[t][j] + BM_LoadRD[t][j] ;
forall(t in TimePeriods, i in Units_F)// BM_LogicalConstrFastUnits[t][i]:
BM_Commit_F[t-1][i] -BM_Commit_F[t][i] + BM_Startup_F[t][i] - BM_Shutdown_F[t][i] == 0 ;
//TransConstraints
forall(t in TimePeriods, n in Buses_N) //53// BM_BusBalanceConstr[t][n]:
sum(h in Units_S) (BM_Gen_S[t][h]*MapGS[n][h])
+sum(i in Units_F) (BM_Gen_F[t][i]*MapGF[n][i])
+sum(k in WindF) ((WindP[t][k]-BM_WindSpilg[t][k])*MapW[n][k])
-sum(j in Loads) ((BM_Load[t][j]-BM_Lshed[t][j])*MapL[n][j])
-sum(m in Buses_M) (B[n][m]*(BM_Delta[t][n]-BM_Delta[t][m])*MapT[n][m]) ==0 ;
forall(t in TimePeriods, n in Buses_N, m in Buses_M) //54//
BM_TransmissionFlow[t][n][m]:
BM_TF[t][n][m] == B[n][m]*(BM_Delta[t][n]-BM_Delta[t][m]);
// Min up-time constraints for fast units
forall(i in Units_F)// InitialMinupFast[i]:
sum (t in TimePeriodsG) (1 - BM_Commit_F [t][i]) == 0;
forall( i in Units_F, t in TimePeriodsH)// MiddleMinupFast[i][t]:
sum (k in t..(t+MinUpTime_F[i]-1)) BM_Commit_F [k][i] >= MinUpTime_F[i]* BM_Startup_F
[t][i];
forall(i in Units_F, t in TimePeriodsI)// FinalMinupFast[i][t]:
sum (k in t..T)(BM_Commit_F [k][i] - BM_Startup_F [t][i]) >= 0;
forall(i in Units_F)// InitialMindownFast[i]:
sum (t in TimePeriodsM) BM_Commit_F [t][i] == 0;
forall(i in Units_F, t in TimePeriodsN)// MiddleMindownFast[i][t]:
sum (k in t..(t+MinDownTime_F[i]-1)) (1-BM_Commit_F [k][i]) >= MinDownTime_F[i]*
BM_Shutdown_F[t][i];
forall(i in Units_F, t in TimePeriodsO )// FinalMindownFast[i][t]:
sum (k in t..T)(1 - BM_Commit_F [k][i] -BM_Shutdown_F [t][i]) >= 0;

```