Decomposition Techniques for Large-Scale Energy Optimization Problems

Konstantinos Tavlaridis-Gyparakis

A thesis presented for the degree of Doctor of Philosophy

Supervised by: Antonio Frangioni
(University of Pisa)

Department Of Computer Science
University of Pisa
Italy
November 15, 2017
3.2.2 A DP-Procedure for the Non-linear 1UC with Ramp-Constraints . . . . . 33
3.2.3 Solving the Economic Dispatch Problem (EDP) .................................. 35
3.2.4 Improving the DP-Procedure for the Non-linear 1UC with Ramp-Constraints 39

3.3 New MILP Formulations for 1UC ................................................................. 42
3.3.1 The $p_{h,k}$ MILP Formulation ............................................................... 43
3.3.2 The $p_t$ MILP Formulation .................................................................. 46
3.3.3 Computational Results ........................................................................... 46

4 The EDF Unit Commitment Problem ......................................................... 49
4.1 Nomenclature ............................................................................................ 51
4.1.1 Sets ........................................................................................................ 51
4.1.2 Constants ............................................................................................... 52
4.1.3 Continuous variables ............................................................................. 54
4.1.4 Discrete and binary variables ............................................................... 55
4.2 Objective Function .................................................................................... 56
4.3 Global Constraints ................................................................................... 57
4.4 Thermal Constraints ................................................................................ 58
4.4.1 Production States .................................................................................. 58
4.4.2 Interval of primary power values .......................................................... 59
4.4.3 Power changes of primary power values ............................................. 61
4.4.4 Reserved power production .................................................................. 67
4.4.5 Start Up Costs ....................................................................................... 70
4.4.6 Minimum on/off duration ..................................................................... 71
4.4.7 Maximum number of starts .................................................................. 71
4.4.8 Deep decrease ....................................................................................... 71
4.4.9 Initial Conditions .................................................................................. 72
4.5 Hydro Constraints .................................................................................. 75
4.5.1 Power production .................................................................................. 75
4.5.2 Water volume conservation ................................................................. 76
4.5.3 Volume .................................................................................................... 77
4.5.4 Spillage .................................................................................................. 77
4.5.5 Flow ....................................................................................................... 78
CONTENTS

4.5.6 Pumping .................................................. 78
4.5.7 Monotonicity ........................................... 78
4.5.8 Volume dependent water values ....................... 79
4.6 Comparing the EDF and Academic UCs .................. 79

5 Lagrangian Approaches for UC ............................. 81
3.1 Integer Linear Programming ................................ 81
3.2 Lagrangian Relaxation .................................... 83
3.3 Cutting Plane methods, the Dantzig-Wolfe decomposition approach, and Column Generation .................. 83
3.4 Better versions of the Cutting Plane algorithm: Bundle methods .............................................. 86
3.5 Lagrangian Heuristics for UC .............................. 87
3.5.1 Existing Lagrangian Heuristics ......................... 87
3.5.2 Expanding the Lagrangian Heuristics .................. 89

6 A Structured Modeling System .............................. 93
6.1 Motivations and Design ................................... 93
6.2 The Core Elements of SMS++ ............................. 95
6.2.1 Block .................................................. 96
6.2.2 Variable .............................................. 98
6.2.3 Constraint ............................................ 99
6.2.4 Objective Function .................................. 101
6.2.5 Solver ............................................... 102
6.2.6 Modification ......................................... 105
6.3 An SMS++ Application for the UC ....................... 105

7 Conclusions .................................................. 113
Chapter 1

Introduction

The current Ph.D. Thesis has as its main focus and research area the field of Combinatorial Optimization. The aim is to further develop and improve some of the techniques and methods that are used to solve hard Combinatorial Optimization problems, among which are algorithmic Convex Analysis, Mixed Integer Linear Programming and Decomposition Approaches. We are interested in both exploring (and further enriching) the theoretical understanding of these fields, and also in creating efficient algorithmic tools in order to apply these methods in the most effective way. Hence, in the thesis we will try to properly balance the theoretical investigation with actual implementation of the proposed approaches and their testing on real-world optimization problems, in particular related to electrical energy production and distribution.

This consistent implementative aspect of the thesis makes it also well-suited for contributing to the development of existing optimization codes, and initiate the development entirely new ones. One particular aspect of this development, sitting squarely in the intersection between optimization and computer science, is the development of a new modeling system for the description and solution of structured optimization problems, with an initial focus on those that can be easily described under the form of Mixed-Integer Linear Problems (MILP), but with an eye to making the extension to other classes easy.

More specifically, our work is mainly directed to the development of decomposition methods for a real-life, large-scale, challenging industrial optimization problem related to optimal Electric Energy Production and Distribution (EEPĐ): the Unit Commitment (UC) problem. In the last few decades, the combination of huge development in computer hardware and in general-purpose solvers for large class of difficult optimization problems, mainly that of MILP, has made it possible to routinely solve instances of sized that could have previously been unimaginable. This has, rightly, made general-purpose solver a popular choice among the available solution alternatives for tackling large scale optimization problems. Still, in our opinion decomposition methods remain a compelling choice for tackling several families of large scale problems, justifying our interest in the theoretical study and the further development of techniques and efficient decomposition tools.

The main advantage that decomposition methods offer is the fact that they are better poised to exploit certain forms of underlying structure of the examined problem, possibly leasing significant reductions of the computational effort during the solution process. The existing literature describes a large number of specific structures and model classes for which the available decom-
position techniques prove to be more effective than general-purpose ones. Another significant advantage that decomposition methods offer is the potential to exploit the separable structure of problems in order to exploit parallel computing techniques. This may result in very substantial efficiency improvements, especially in this day and age where the increment of computational power of computing system is for the most part due to the increase in number of available cores, possibly specialized for specific tasks (GPU). Most of the available general-purpose solvers lack this ability, or at least have a very different profile w.r.t. the granularity at which the parallel techniques can be used. Because of all the above mentioned reasons we believe that our research focus is driven to the proper direction and we are confident that the results of our work will be a further validation of our belief. The rest of the Thesis is structured as follows. In Chapter 2, we start our study with a overview of the main problem under examination that is the Unit Commitment Problem, taking a look at both its main ingredients and the main solution techniques that have been proposed in the literature to tackle it. Thereafter in Chapter 3, we proceed with a detailed discussion of the academic Thermal Unit Commitment Problem, and providing an overview to already existing efficient formulations (developing also a new one in 3.3.2) and DP-approaches, and then we proceed with implementing and testing them with our developed modeling system SMS++. Moreover in Chapter 4, we proceed with a full detailed description of the real-life large scale Unit Commitment of EDF, proposing three different MILP formulations. Then we go on with Chapter 5, where we discuss some of the basic theoretical background of the methods used in order to tackle the UC and the chapter finishes with presenting a Lagrangian Heuristic for tackling the EDF-UC, for which no actual implementation for the Edf UC has developed yet. After this we proceed with Chapter 6 where we discuss our developed modeling system SMS++, which was used to implement all the formulations and DP-algorithms of ?? and finally in Chapter 7, we proceed with summarizing the main conclusions that derive from our work.
Chapter 2

The Unit Commitment Problem

One very famous and difficult optimization problem that has been studied extensively over the past decades is a problem of the electrical industry that focuses in the energy production and distribution systems. The very basic goal of this industrial energy problem is to schedule properly a set of different available production units meeting their physical and technical restrictions, while satisfying a series of linking constraints. The complexity of the constraints that express the physical and technical restrictions may be really high, leading to highly complexed and difficult to solve problems. The above described problem is referred in the industry as the Unit Commitment Problem and it can be considered to be of a high practical importance, as it has been of both the academic and industrial research interest for decades, leading to a series of developments and improvements in the efficiency of the several proposed solution methods. One of the main reasons that the Unit Commitment Problem is a highly interesting problem from a research perspective is the fact that it offers a large amount of variations of the original problem depending on the different electrical systems and production markets that can be found around the world, which in most cases are really difficult to be solved. In addition to the above, the continuous development of optimization methods has given the chance to the researchers to tackle problems that are even more realistic and as a result more complex than ever before.

Still even after all these intensive research efforts, the Unit Commitment cannot be considered a well solved problem yet. One of the main factors for this is the fact that the problem demands are changing throughout time, as new technological advantages are altering and making more complicated the technical constraints or even completely new categories of production units may be introduced (such as renewable energy units), or the change from a monopolistic to a free market can arise, etc. Another reason for this is the fact that the energy production companies are expecting to have good feasible production schedules for their units within an unreasonable small computational time, making the problem even more challenging and difficult. Moreover the methodological and technical developments that have been made and used in order to tackle successfully unit commitment problems in the past were based in models with simplifications of the original physical and technical particularities, and as a result all the successfully solved models are continuously being altered adapting more sophisticated and realistic models that withdraw the simplifications that have been made so far in several parts of the model.

As a result of all the above mentioned factors, it can be easily understood that there is still a lot of space for further research in the solution process of the Unit Commitment Problem, something that makes this problem a very good candidate for the methods examined and proposed
in this thesis. Before we proceed with the discussion of the ingredients and the methodologies concerning the problem, it is important that we indicate some basic simplifications and assumptions under which we consider the Unit Commitment Problem. These can be summarized in the following two elements:

1. the general market and distribution setting,
2. the level of the uncertainty.

As far as the first element is concerned historically speaking the first UC where being carried out in an era where one had to consider a Monopolistic Producer (MP), that was responsible for all the different kind of tasks combined with the energy management of the area (production, transmission, distribution etc). Nowadays that the energy market has been liberalized the above mention decision chain has transformed to a rather decentralized and complex setting. The setting for our discussion refers to the case, where the industries owning generation assets (GENCOs) have to proceed with bidding their energy production over a Power Exchange markets (PE). After this point companies that own transmission infrastructures are assigned the task of the energy transmission as Transmission System Operators (TSO), having to comply with a series of safety and other of nature regulations complied by the Power Exchange Manager (PEM). Yet again this system setting that we consider for this chapter as the basic framework within the different UCs are found is just a simplified version of a rather more complicated reality. The interested reader is directed to [164, 81, 109, 69], where several other elements and aspects that are found in several different real-life system settings are taken into account and discussed thoroughly.

As far as the second element is concerned, we will focus in the case of the UC where the stochastic elements of the problem are either being excluded or through different ways being transformed to have a deterministic nature. As a result of this we can consider that our UC problems refer to the case of the Deterministic Unit Commitment. For the case of the Stochastic UCs that has received scientific interest and examination the last few decades. We refer the interested reader to [166, 43, 145] for a first look in the some of the main elements that introduce stochasticity to the model, and to [22, 21, 162] for some of the main solution methodologies (such as Stochastic Optimisation, Robust Optimisation, Chance Constraints) that exist in order to tackle them.

Moreover, since -as it can be clear from the above- in our discussion we will focus on only some of the several different models and solution methodologies of the UC, namely in the ones connected one way or another with our research focus. And this is only because a through review and analysis in the history and the different models and solution methodologies of the Unit Commitment would require a thesis on itself. We refer the interested reader to [54, 136, 158] for well detailed surveys on the problem in order to take a full understanding of the history, the complexity, and the different forms and different solution methods of the UC problem.

2.1 General Overview of the UC Ingredients

In a general context the (deterministic) Unit Commitment Problem can be seen as a problem, where one is required to minimize the cost (or maximize the profit) of the schedule of the set of generating units over a given time horizon. The main parts that compose the optimization problem (i.e. its fundamental ingredients) are the objective function, its constraints (that are
2.1. GENERAL OVERVIEW OF THE UC INGREDIENTS

divided in several subsets), and the time horizon. In the following we will denote it by $T$, which is usually divided in a set of discrete time steps $t \in T$, that in most of the cases refers to hours or half-hours. The overall optimization time horizon $T$ may vary from 24 to a few hundred hours based on the way the production schedules are being fixed in each different company, leading to the short, middle and long term versions of the unit commitment problem.

In mathematical terms, UC has the general structure

$$
\min \{ f(x) : x \in X_1 \cap X_2, \}
$$

where $x \in \mathbb{R}^n$ is the decision vector.

In most of the cases $x$ is indexed over all generating units $k = 1, \ldots, m$ and the time steps $t \in T$. The main decisions that are of concern in a Unit Commitment problem can be divided in the following categories:

1. commitment decisions, which refer to the state (if the unit is off or on) of a generating unit in any given time step and take binary values.

2. production decisions, which denote the produced amount of energy of a generating unit in any given time step and take continuous values.

3. network decisions, such as the ones representing phase angle or voltage magnitudes, that describe the state of the transmission or distribution network.

The above mentioned decisions are subjected to a set of constraints that are mainly divided in two categories. $X_1$ refers to all the technical and physical constraints of the individual generating units and $X_2$ refers to the global constraints.

The set of $X_1$ can be a non-convex complex set, that is composed of the cartesian product of smaller sets i.e., $X_1 = \prod_{k=1}^m X_k^1$, with $X_k^1 \subseteq \mathbb{R}^n$ and $\sum_{k=1}^m n_k = n$. Also the objective function $f$ in most cases can be decomposed by the sets $X_k^1$, i.e., $f(x) = \sum_{k=1}^m f_k(x_k)$ and $x_k \in X_k^1$. Every subset $X_k^1$ roughly holds the feasible production schedules for a unit, and even though units can be very different between each other based on their individual operational and physical characteristics, usually there exist some specific type of units that share the same technical and physical characteristics and as a result large part of their constraints. The two most well studied and known types of generating units are the thermal and the hydro units. In addition to them there are also the renewable generating units, which won’t be presented here as they are not part of the unit commitment models that will be examined in this thesis. We refer the interested reader to [29, 122].

Finally the set $X_2$, refers to the linking constraints of the Unit Commitment Problem, which are concerned in most with the cases the satisfaction of demand and in most of the cases can be described through a convex or even polyhedral set. Some other types of global constraints may refer to some maximum level of allowed emission levels or some network transmission switching constraints. For a detailed demonstration of formulations of the overall UC we will continue the discussion throughout chapter 4 where the case of EDF Unit Commitment is fully discussed and formulated.
2.1.1 Objective Function

The objective function of the UC can vary depending on many reasons, with one of the most important ones to be the types of decision makers that define each individual problem. In principle, if we are in the case of a MP or of a GENCO that has already some fixed bids, a level of production needs to be specified, and thus the goal is the minimization of the production costs of the corresponding generating units. Yet again in the free-market regime the objective function can be altered to maximizing the profit of selling the produced power of the generating units. However in both cases the production costs have to be calculated and taken into account. Yet again in the latter case one has to estimate and calculate the production profits since, it is the difference of the two that gives the total revenue and needs to be maximized.

In a general context we could say that in the case of profit maximization we can see two main different cases for the problem. In the first one the GENCO is a big enough market-player that it can be considered to be a price-maker, and in this case can also include a bidding strategy for the prices of power production (see for example [47]). Yet again one has to take into consideration the fact that the individual decision of a price-maker are also affected by the strategies of the other players in the energy-market, which add an higher level of uncertainty. Plus it should be also taken into account the fact that there are some restrictive regulations been set by PEM in order to avoid behaviors that exclude smaller-players from the market and impose a cartel or a monopoly. Finally the second case concerns the case of the price-taker where the GENCO has an overall small portion of the production and as a result is unable to influence the market-prices. In this case a level of uncertainty is included by the fact that one has to predict as accurately as possible the selling price of energy for the different units for all the optimization horizon.

Moving on, as far as the production costs are concerned, which as mentioned above are present in all the different cases of the UC, these vary depending on the different types and characteristics of the generating units. The costs of fuel burning units are usually modeled as piece-wise linear convex, quadratic convex, or even non convex functions that are separable for each time step. Even though in most of the cases the potential non-linearities/non-convexities of fuel-consumption-power curve can be overcome with piecewise linear approximation or other similar approaches, there are other characteristics that can cause non-convexity in the objective function. The simplest case for such a situation can arise when there is a fixed cost that must be paid each time the unit is producing energy regardless of its generated amount.

In addition to the production costs, there are also the start-up costs that occur every time a unit switches from offline to online. The most simplified approach considers start-up costs to be fixed, but in most cases they are time-dependent from the period the unit has previously been offline, which can be something pretty complex, due to the fact that one has to consider the choice if the unit will be completely depower (usually referred as cooling), or being stable in an appropriate temperature, having to pay some fuel-cost, but ending up with a decreased start-up cost (usually referred as banking). Fortunately the start-up cost can be added to the problem without a significantly big increase of computational complexity via the appropriate MILP formulations such as [129, 33], or even with no computational additional complexity in the case of DP formulations.

A special category of the fuel burning units are nuclear units, which in general are considered to be thermal units with some particular specific technical and physical characteristics. In most cases the objective function is formulated as thermal units with some additional modulation
2.1. GENERAL OVERVIEW OF THE UC INGREDIENTS

costs for whenever a variation in the amount of generating power is required.

The hydro units in most of the cases are considered to not have energy production cost. In cases where one has to maximise profits, the units would attempt to use all of the available water, since in the short-term model like the one of the UC the units don’t take into account the mid-term horizon planning. In order to overcome this situation in many models a water value coefficient is placed in the objective function to denote the expected water volume expected to be left in the reservoirs after the end of the optimization horizon. Usually these values and the expected reservoir volumes are calculated in the mid-term optimization planning, we refer the interested reader to \[161, 36\] for further details on these issues.

Finally let us note the fact that the modeling of the objective function based on free-market regimes is not presented in this section, due to the fact that our examined UC models do not examine such cases. We refer the interested reader to \[44, 27, 12, 111\], which give detailed models and analysis of such problems. Also for a highly complex version of UC objective function the formulation presented in chapter 4.2 where the EDF UC objective functions are formulated, provides a good example. Before moving forward to the next ingredient let us proceed with a mathematical formulation that describes the most basic version of an objective function that can be found in the academic hydro-thermal versions of the UC. Following the above notation we further introduce the set of thermal generating units \(i \in I\) and hydro units \(h \in H\) that should produce energy over a set of discretized time steps \(t \in T\). The decision variables that are expected to exist in such a version of the problem can be summarized as follows:

- \(u_{i,t} \in \{0, 1\}\): commitment state of thermal unit \(i \in I\).
- \(p_{i,t} \in \mathbb{R}_{+}\): production level of thermal unit \(i \in I\).
- \(q_{j,t} \in \mathbb{R}_{+}\): water discharge for hydro unit \(j \in H(h)\) for valley \(h \in H\)

This results in an objective function that has the following form:

\[
f(p, u) = \sum_{i \in I} c_i(p_i, u_i) = \sum_{i \in I} \left( s_i(u_i) + \sum_{t \in T} c_{i,t}(p_{i,t}) \right) \tag{2.1.1.1}
\]

- production costs \((c_{i,t}(p_{i,t}))\), which as discussed above usually are nonlinear convex.
- time-dependent start-up costs \((s_i(u_i))\).
- Note that \(q_{j,t}\) of the hydro units -as discussed above- don’t impact the objective function.

2.1.2 Thermal Units

A thermal unit refers to a power plant where the energy is mainly produced by steam. The technical and physical constraints are mainly divided in static (the ones that must hold in every single time step) and dynamic (the ones that link different time steps) constraints. In chapter 3.1 we describe a series of different formulations for the Thermal Units for the case of the simplified
academic UC and moreover in 4.4 we discuss the constraints concerning the Thermal Units of the much more complicated EDF Unit Commitment Problem.

The main static constraints are:

1. Production Capacity Constraints: the production capacity of the units, where there exist cases where the output has a discretized form with some specified allowed power intervals.

2. Spinning Reserve: that refers to the additional producing capacity which can be achieved by increasing the power output of the online unit.

3. Crew constraints: the maximum number of power generators allowed to be open at each time step.

The main different types of dynamic constraints are:

1. Min Up/Down Time: the minimum allowed up and down time of a generating unit.

2. Ramp Up/Down Rate: a constant that bounds the increases and decreases of power production of the units.

3. Min Stable Duration: the amount of time during which a unit cannot alter its production level.

4. Max Starts: the maximum number of allowed starts of the units for a given time interval.

5. Start/Stop Curves: the following of specific production curves when the units are switched on or off.

6. Modulations/Stability: the control of the number of changes in the production level during the time horizon through the existence of modulation and stable states.

In this subsection as well let us proceed with formulating some of the most common constraints that are expected to be found in the academic versions of the UC. Note that all the constraints below are valid for $i \in I, t \in T$:

- Maximum and minimum power output:

  \[ p_{i,t}^u u_{i,t} \leq p_{i,t} \leq \bar{p}_{i,t} u_{i,t} \]  \hspace{2cm} (2.1.2.1)

  Where $p_{i,t}$ and $\bar{p}_{i,t}$ are the constants defining the minimum and maximum output respectively and variable $u_{i,t}$ ensures the unit is not allowed to produced any amount of energy while being shut down (i.e. $u_{i,t} = 0$).

- Ramp-up/Down constraints:

  \[ p_{i,t} - p_{i,t-1} \leq u_{i,t-1} \Delta^+_{i,t} + p_{i,t-1} \]  \hspace{2cm} (2.1.2.2)

  \[ p_{i,t-1} - p_{i,t} \leq u_{i,t} \Delta^-_{i,t} + p_{i,t} \]  \hspace{2cm} (2.1.2.3)
Where $\Delta^+_{i,t}$ and $\Delta^-_{i,t}$ are the constants defining the ramp-up and ramp-down threshold respectively and $p^0_i$ and $p^0_i$ are constants referring respectively to start-up and shut-down limits of unit $i$. Moreover the way the rhs is formulated in (2.1.2.2) it ensures that in case the unit was switched on in the time step $t$ it will be able to arrive from zero to the minimum power output disregarding the ramp-up threshold. In a similar fashion $u_{i,t}$ in (2.1.2.3) is ensuring that the unit in case it switches off in time step $t$ will be able to draw its power output to zero disregarding the ramp-down threshold.

- Min up/down-time constraints:

$$u_{i,t} - u_{i,t-1} \leq u_{i,r} \quad r = \{t + 1, \cdots , t + \tau^+_{i}\}$$  (2.1.2.4)

$$u_{i,t-1} - u_{i,t} \leq 1 - u_{i,r} \quad r = \{t + 1, \cdots , t + \tau^-_{i}\}$$  (2.1.2.5)

Where $\tau^+_{i}$ and $\tau^-_{i}$ are the constants defining the min-up and min-down time for each thermal unit $i$ respectively respectively. The lhs of (2.1.2.4) is ensuring that in case the unit was switched on in time step $t$, it will be forced to remain online for a time-period equal to $\tau^+_{i}$, since the variable of rhs will be forced to be equal to 1 for all future time-steps $r$. And finally the lhs of (2.1.2.5) is ensuring that in case the unit was switched off in the time step $t$, it will be forced to remain offline for a time-period equal to $\tau^-_{i}$. Since in this case the lhs has a value equal to 1 and thus the variable of the rhs is obliged to keep its value equal to 0 for all future time-steps $r$ in order not to violate the constrain.

- Start up Costs:

$$c^0_{i}(u_i) = \sum_{t=1}^{T} c^0_{i,t}$$  (2.1.2.6)

$$c^0_{i,t} \geq \sigma_{i,\tau}(u_{i,t} - \sum_{j=1}^{\tau} u_{i,t-j}) \quad \tau \in \{\tau^-_{i}, \cdots , \tau^+_{i}\}$$  (2.1.2.7)

$$c^0_{i,t} \geq 0$$  (2.1.2.8)

Where $\sigma_{i,\tau}$ refers to the function that is used in order to calculate the corresponding start up costs. This function can be complicated and difficult to construct on its own relying, on several factors. Their calculation depends on the number of time steps the unit has been previously remained offline. We refer the interested reader to ([153][170][156]), where some of the most well known approaches to efficiently define it have been made. Moreover $\tau_i$ denotes a time constant that refers to the amount of time after which the cost of starting an offline unit is constant in time. The above constraints represent one of the main elements that adds complexity in the thermal part of most versions of UC.

### 2.1.3 Hydro Units

Hydro units are usually contained in hydro valleys, which are composed of several reservoirs, turbines and pumps that are connected to each other. These connections are expressed by several different constraints, that demonstrate the way the different parts of the hydro valleys...
influence each other. More specifically there exist the so called run-of-river (ROR) units, that consist of turbines that use water released from uphill reservoirs in order to produce energy. In a similar fashion the pumps follow the exact opposite process of the ROR units. Let us note that in 4.5 are discussed the constraints concerning the Hydro Units of the much more complicated EDF Unit Commitment.

Moving on and trying to take a look to the different characteristics of the hydro units we proceed with the same categorization of the constraints’ nature as we did with the thermal units. The most typical static constraints one is expected to find for hydro units can be summarized as:

1. Reservoir Level: Each reservoir is obliged at least for some physical capacity reasons to remain within certain boundaries. On top of the physical restrictions though, in a lot of cases these bounds also reflect the long-term decisions of the water management, which need to be also taken into account in the modeling of the problem. This is because keeping a certain amount of water in the reservoirs is one of the few ways to manage to store high-levels of energy. Since one of the issues with electrical energy is that it cannot be physically stored and kept for long-term and thus the storing of certain water level in reservoirs can be considered a way to overcome this. In

2. Generation Bounds: They usually refer to some lower and upper bounds on the water flowing in the pumps and to control of the number of changes in the production level during the time horizon through the existence of modulation and stable states.

As for the dynamic ones:

1. Flow Equations: These type of constraints refer to some equation that must hold for the water flow based on the water inflows, the turbined downhill water and the pumped water as well.

2. Flow delay: In these case there can arise constraints that refer to water delays from the uphill reservoir to the downhill.

3. Smooth Turbinining: There exist constraints that take care in some kind of smooth turbinining. In the sense that for a given time horizon the turbining must follow some specific line such as not being able to increase and in the exact next time step decrease, or the other way around.

4. Turbinining/Pumping Incompatibility: In some cases turbines can be reversible and as a result there is some incompatibility in the switching between pumping and turbinining water. On top of this one has to consider the fact that this switches have a certain time delay (of some time steps).

5. Forbidden Zones: There exist some highly complicated hydro units, where effects like mechanical vibrations and cavitation make the use of certain intervals of turbined water undesirable, as it would result in low efficiency and/or high output variation. As a result constraints are imposed to avoid the use of them in these specific zones.

Before finishing this subsection let us demonstrate some of the most common constraints that are found in the academic versions of the UC. Once again the constraint below are valid for $j \in H(h)$ and $t \in T$:
2.1. GENERAL OVERVIEW OF THE UC INGREDIENTS

- Maximum discharge:

\[ 0 \leq q_{j,t} \leq \overline{q}_j, \quad (2.1.3.1) \]

where \( \overline{q}_j \) is the constant that denotes the maximum discharge output for unit \( j \) at every time-step.

- Maximum and minimum reservoir volume:

\[ v_j \leq v_{j,t} \leq \overline{v}_j, \quad (2.1.3.2) \]

where \( v_j \) and \( \overline{v}_j \) are the constants that define the minimum and maximum accepted volumes of reservoir \( j \) at every time-step.

- Water conservation:

\[ v_{j,t} - v_{j,t-1} = \bar{w}_{j,t} - w_{j,t} - q_{j,t} + \sum_{k \in S(j)} (q_{k,t-t_{kj}} + w_{k,t-t_{kj}}), \quad (2.1.3.3) \]

where \( \bar{w}_{j,t} \) is the constant for the water inflows, \( w_{j,t} \) denotes the constant for water spillage, and \( t_{kj} \) is the time delay of water flowing from unit \( k \) to unit \( j \). The above water conservation equation is used in order to make sure that the reservoir’s volume in all the consecutive time steps is consistent with the inflows and outflows. That is achieved by having in the left side the difference of water volume between two consecutive periods which then is set to be equal to the sum of all external inflows and received water from upstream reservoirs (with respect to time delays) minus the spillage and the discharged water from the reservoir to the ones below.

2.1.4 Renewable Units

The renewable units mostly refers to solar generation, wind farms, geothermal production and some stand alone ROR hydro units. As far as the UC is concerned the main characteristics of concern of these type of units is their difficulty to be controlled. The produced energy, that in some cases can even be negative, (for instance there are situations in wind-farms, where in case that there is no actual wind, energy needs to be consumed in order to keep the blades of the wind turbines spinning within the security levels), mostly relies on external factors outside the decision makers’ choices. On top of that it is very difficult to get accurate production forecast can be very difficult to be accurate even when the time horizons are very short-term. On top of the above the correlation between the production of the renewable units and the energy load can be even negative. This is because there are cases where the renewable units are operating are under regulations that don’t allow them to be modulated by simply disconnecting them from the grid. Thereby the situation can arise when one is paid to consume the produced renewable energy regardless the real demand.

All the above makes it clear that the main problem that arise with the insertion of the renewable generation units is the increase of the uncertainty of the amount energy demand that needs to be satisfied. This leads to the situation that the already existing production units are forced to operate to a much greater level as back-up production units, in order to take care of the possible fluctuations that arise from the renewable units. Because of this the hydro and thermal
units are obliged to have an increased production flexibility, in order to be able to guarantee
the load’s satisfaction at short notice. This however usually results in larger production costs.
More information on the handling of the renewable units within the UC framework is given in
[29], [51], [118].

2.1.5 Global Constraints

Through the UC literature the most well-known and frequently observed global constraint that
links the different units together is the Global Demand Constraint. In the different versions of
the UC the demand satisfaction, even in cases that can be considered that it is not actually
present in the model such as the case of self-scheduling models is handled in different ways,
creates one way or an other create a link that is coupling all the different units together. More
specifically in the very basic case of MP things are pretty straightforward and all the generating
units need to satisfy the global demand summing up their overall independently produced power.
In the case of the liberalized market regimes things can be expected to be a bit different, since in
the self-scheduling cases of the UC units are reacting independently to price signals, but yet the
demand satisfaction can still link the different units together. More specifically after the phase
of the biddings and price settings in the energy market, GENCos can proceed with swapping the
demand between different units, in order to manage to handle better the production in respect
with the operating constraints of the different generating units [144]. In addition to this one can
go through an adjustment market, where energy can be bought/sold, that basically has the exact
same results [137]. As one can see in both case we return back to a global impact of the demand
constraints. In addition to the above cases, in the case of the UC-based bidding systems one
has to take into account and model the impact on the energy prices of the overall generation
production of a GENCo, which leads again to the situation where the demand constraint has
a global scope. Also let us note that in [13] are discuss the constraints concerning the Thermal
Units of the much more complicated EDF Unit Commitment.

Moving on another important element that needs to be taken into account, when discussing the
global demand constraints, is the transmission network. In general the transmission network can
be considered to be modeled in three ways, based on the level of accuracy in the approximation
of the overall characteristics. These can be considered to be:

1. The single bus model: This is the most simplified version, here all the different particular-
   ities and characteristics of the network are disregarded. This leads to a linear constraint,
   where one has to satisfy for each time instance the given demand, by setting its amount
equal to the total sum of the production of all the different operating units.

2. The DC model: In the case where the network structure is modelled, but still by proceeding
   with a simplified version of the Kirchoff laws. In this case the constraints might be more
   complex than the above case but remain linear [101], [89]. An interesting approach is
   followed in [9] where the concept of umbrella constraints is introduced, defining a subset
   of DC constraints that are active, thereby leading to a significant decrease of the number
   of constraints.

3. The AC model: In this case the modeling approach takes full consideration of the Kirchoff
   laws, leading to nonlinear and nonconvex constraints. This makes as one can imagine
   the corresponding problem more difficult to solve [125], [120], [152]. Recent research efforts
have resulted in the transformation of the AC Constraints to a quadratic form yielding a convex relaxations, that allows the implementation SemiDefinite Programming (SDP) approaches \cite{119}.

We now present the most basic and well-known global constraint that is found in the Academic UC versions, which is the first case of the single bus model. We assume the total energy demand $d_t$ is known for all time steps of the optimization horizon $t \in T$. On top of that we will also formulate the well known global constraint on Spinning Reserve:

\begin{align*}
\sum_{i \in P} p_{i,t} + \sum_{h \in H} \sum_{j \in H(h)} a_j q_{j,t} &= d_t \quad (2.1.5.1) \\
\sum_{i \in P} u_{i,t} r_{i,t} + \sum_{h \in H} \sum_{j \in H(h)} r_{j,t} &\geq R_t \quad (2.1.5.2)
\end{align*}

where in \[(2.1.5.1)\] $a_j$ is the constant power-to-discharged water and the left hand side of the equation denotes the sums of the power production of all the different units that needs to be equal to the forecasted energy demand for all the time steps. In \[(2.1.5.2)\] we have the so called spinning reserve $r_{i,t}$ and $r_{j,t}$, that are equal to the difference between the maximum output of all running units and the current output, which should in all time steps be greater or equal than a given threshold $R_t$.

2.1.6 Optimal Transmission Switching

In most of the literature of the UC the transmission network is disregarded from a modeling point of view, being simply considered just as the energy flow between the generating units and the demand. As a result of this in the most traditional UCs we don’t encounter any kind of "network variables". Yet again this very simplistic approach is rather unrealistic and misleading, since the transmission network in general is for sure not a trivial element in the system. As a result of this exclusive network variables are required in order for the network to be modeled accurately. A recent extension to the introduction of network variables has been the case where the system behavior is optimized by the dynamically change of the topology of the network. As a result of this a new class of problems have been introduced, that are called Optimal Transmission Switching (OTS) or System Topology Optimization (STO). In these cases every network line is linked with a binary decision variable for all the different time-steps of the optimization horizon that correspond to the switching on of each one of them. This extends the UC to the case of UCOTS \cite{58, 82, 130, 83, 133, 175} and makes the problem more complicated and difficult to tackle.

2.2 General Overview on Solution Methods

The following subsection gives a short introduction to the main solution methodologies that have been followed over the years to tackle the deterministic version of the UC. Before we start discussing the different types of solution methodologies the following clarification is needed. UC
CHAPTER 2. THE UNIT COMMITMENT PROBLEM

itself as already shown in the previous subsection is not really a specific problem but rather a
broad set of problems that share some common features. As a result of this, for each of the
different solution methodologies discussed below, we will use the defined in \[157\] basic version
of the UC (bUC). This can be considered to have the following characteristics:

1. offer-demand equilibrium,
2. minimum up/down times,
3. spinning reserve,
4. generation capacities.

Moving on as far as the different methodology classes are concerned in this subsection is decided
to proceed with the presentation of these categories that one way or another are connected with
our work and the solution approaches that were followed. And that is:

1. Dynamic Programming,
2. MIP approaches,
3. Decomposition approaches,
4. (Meta-)Heuristic approaches.

2.2.1 Dynamic Programming

Dynamic Programming \[15, 20\] is one of the oldest approaches for solving the UC. Even though
it is mostly used in order to solve sub-problems of the original UC that can arise after some
decomposition approach, (for which we will discuss in more details in chapter ?), one can find
in the literature also attempts to solve the overall UC via a DP solution scheme. The literature
contains several different approaches aimed at overcoming the curse of dimensionality of DPs,
for example a combination of DP with Priority Listing \[155, 86\], Variable Window Truncated
DP \[135\], Sequential Combination \[139\], Truncated Combination \[138\], or even some heuristics
such as \[135\]. Another approach can be considered to be the iteratively application of DP
(multi-pass DP \[173, 52\]), where for each iteration the previous solution is used in order to
provide a guidance for the discretization of state and time space.

In general the DP approaches might not have proven themselves as the best candidate for
solving the total UC, but yet again it can be considered to be an efficient method when dealing
with the subproblems of the UC that arise after the initial problem’s decomposition. In the
most basic decomposition approach one proceeds by relaxing the linking constraints and as
a result being left with independent subproblems that represent the single units, also known
as 1UC. In the most simple form of the 1UC where the start-up costs of the units are not
time-dependent, the solution can be obtain in a linear time over size of \(T\). In the case that the
costs are time-dependent (of the time steps the unit has been previously remained off) then the
this cost becomes quadratic \[176\]. In the case where the ramp-constraints are also included,
things become more complicated, due to the fact that the discretization of the power output
variables (which is crucial in order to keep the computational costs in the above mentioned
2.2. GENERAL OVERVIEW ON SOLUTION METHODS

levels) becomes a more difficult task. That is because in general in order to tackle the above
mentioned cases of UC via the use of DP, we have to set the optimal production decisions $p_{i,t}$
independently for each time instant, which requires the commitment decision $u_{i,t}$ to be fixed,
which is true if ramp rate constraints are not present. In [?] an efficient DP approach is
considered where the ramp constraints and the time-dependent start up costs are taken into
account and the power production costs are assumed to be piecewise linear. The approach is
further extended in [62] in order to take into account general convex function, which results in
computational costs being a cubic function of $T$.

Finally it should be mentioned that DP approaches have been implemented in [154] in order to
tackle the hydro valley subproblems, following a 3-stage procedure. There an expert system is
used in order to choose desirable solutions, then a DP approach is used for each unit and finally
a network optimization step takes care of the links between the reservoirs.

2.2.2 Mixed Integer Programming

In this section we will examine the Mixed Integer Programming approaches for tackling the UC.
In our case we focus in the MILP formulations (Mixed Integer Linear Programming) and the
MIQP ones, where for MIQPs we only focus to cases where all the constraint of the problem have
a linear form and only in the objective function exist some quadratic terms. That is because
the UCs we have studied and examined have the above mentioned nature. Let us mention
beforehand that the mathematical framework of MILP is further discussed in chapter 5.1 of the
thesis.

Attempts to tackle the UC via MIP formulations exist from the earliest years that optimization
techniques were introduced in order to tackle UC problems. These early approaches were
focusing in exhaustive enumeration of the solution space in order to pick the solution with the
lowest cost. We refer the interested reader to [94, 80] where such methods were used. Yet again,
as one can imagine, such approaches are rather difficult to apply to large scale problems and
as a result is seldom used nowadays. The only few exceptions are cases of hybrid approach like
decomposition methods under specific circumstances, like the case of [67] where enumeration is
applied to some subproblems in a decomposed hydro valley.

The development of efficient MIP solvers brought the MIP formulations for the first time among
the most preferable choices of tackling UCs. Some early MILP applications can be found in
[70, 124, 40], moreover some Hydro-Thermal UC MIP formulations can be found in [50, 150]
where several different hydro constraints are introduced in the formulation. Some interesting
combinations of decompositions approaches with MIP formulations can be found in [99, 140],
where in the first case the Lagrangian Relaxation is combined with a Branch and Bound (BnB)
procedure in order to obtain valid bounds to improve the branching procedure. In the latter
case a heuristic procedure is used in order to obtain a good quality integer feasible solution for
the warm-start of the BnB.

Yet again despite its existence in above mentioned literature, the MIP approaches face some
serious drawbacks and restrictions, causing their efficiency to drop significantly when the size
of the problem or the level of realistic accuracy of the modeling is increased. In order for this
issues to be overcome MILP approaches are combined with other techniques such as DP [24],
Logic Programming [87] and Quadratic Programming (QP) [149]. In [149] more specifically a
hydro-thermal UC with various constraints is tackled with the adoption of some decomposition
method and is reported to have a significant smaller computational size compared with other methods used to tackle the problem. In [64] it is demonstrated that with piecewise linearly approximated objective function the resulting MILPs are even more efficient than the direct use of a MIQP model. In [66] a combination of LR with MILP is proposed, where problems with 200 thermal units and 200 hydro units are solved within a few minutes.

Finally it should be mentioned that production curves of hydro-units have in many cases a non-convex and non-linear [34] (due to head-to-generated-power function), that makes the problem rather difficult to solve. Approaches to overcome this issue are the use of specified convex formulations [174] for particularly special cases, or the approximation of the original formulation [35]. However despite these efforts to improve the solution scheme, this element makes the UC significantly more difficult and can be consider one of the main challenges of the deterministic case of the problem.

However, despite the above mentioned difficulties, the fact that nowadays the MIP solvers have significantly increased their efficiency and started to include selected nonlinear features have resulted in them being one of the most popular choice of the available solution methodologies: they seem to be able to solve an increasingly number of different MILP formulations of UCs within reasonably time frameworks and with acceptable accuracy level [33]. This situation resulted in a focus on the development of more tight UC MILP formulations that provide a more realistic and detailed description of the operational and physical characteristics of the problems increasing its size and difficulty. [85, 112, 116].

As far as the tighter MIP formulations are concerned, the main focus has been put in the development of better representation of several part of the UC. Such as in [132] where a more efficient representation of the polyhedra of min up/down constraints and ramp up/down constraint is developed. Moreover in [65, 90] a better piecewise linear reformulation of the quadratic power cost function is introduced. In both occasions the efficiency of the MIP formulation has been increased.

Whereas the creation of more accurate and realistic models is concerned, the focus is put in doing away with certain previously existing simplifications of the model, that would potentially lead to production of solutions that are closer to the reality and thus can be more accurate. For the hydro units this can consider technical constraints such as in [37] or of the above mentioned water-to-produced-energy functions [141, 25]. Whereas for the thermal units this can refer to cases where is included the proper evaluation of the start-up and shut-down power trajectory[6] or the more detailed distinction between produced energy and the power trajectory of the units [121],

Finally let us note that in the above mentioned references the objective function is assigned with quadratic production costs, which means that we deal with a problem with linear constraints but quadratic objective function. For a different approach we refer the interested reader to [115] where the direct use of MIP formulation is combined with perspective cut approximation [65] and a special perturbation of the costs function that partially break down the symmetries. This combined with a heuristic guidance approach for the status variable leads to better results than the direct implementation of quadratic cost function.
2.2. GENERAL OVERVIEW ON SOLUTION METHODS

2.2.3 Lagrangian Decomposition

The UC exhibits a structure that has the potential to be exploited algorithmically in several different ways. One of the most well-known and widely used is the Lagrangian Relaxation (LR), which can be used in order to relax the global constraints. In our discussion with the term global constraints we refer to the demand and reserve power constraints that have been presented above. Note that we will further discuss the method in chapter 5.2 of the thesis. This is done by moving them to the objective function using lagrangian multipliers. As a result the relaxed problem consists of a series of 1UCs (as discussed above). The solution of all the independent 1UCs together with the Lagrangian multipliers provide a lower bound for the original problem. Moreover the arising maximization dual function is concave and thus one has to solve a convex optimization problem. In general we can distinguish two fundamental features in the solution scheme of LR approaches:

1. The way the Lagrangian Dual (LD) is tackled

2. The primal feasibility recovery. Since the original problem is expected to be non-convex the obtained relaxed solution may not even be feasible and as a result further techniques need to be developed and applied in order to recover the feasibility.

As far as the first point is concerned, the existing well studied field of non-differential optimization (NDO), offers several tractable solution choices. The most standard and the first ones to be used were the sub-gradient [142, 46] and the cutting plane [93] methods. For some early examples of their implementations we refer the interested reader in [59, 123, 19, 2]. Yet again since both methods have proven to be rather unstable and to lead to slow convergence, a series of different stabilization approaches have been introduced. In general the most successful stabilization approach is the use of Bundle Methods [103, 168]. There are several different Bundle approaches in the Literature [106, 105, 7] and one more recent and particularly interesting and useful approach is the use of inexact solutions of the LR [98, 48], since it is expected to be able to provide primary feasible solution even for the cases where the subproblems are not solved to optimality, something that is the case for many real life applications where the units have highly complicated nature and the optimisation periods are short. The history of Bundle use to tackle UC can be followed in [107, 114, 56, 26, 27].

As far as the second point is concerned a very important property of the LD is the fact that while they might be unable to guarantee primal feasible solution they can solve "convexified versions" of the problem [103, 61]. This means that they can produce solutions to the original problem that are feasible for all the constraints except the integrality condition. From that point onwards these convexified solutions, in which the integer variables end up with continuous values, can be manipulated by means of heuristic approaches in order to arrive at primal feasible solutions. Some heuristic approaches based on this are given in [13, 64, 147].

The literature for LR approaches to tackle the UC besides the older cases such as [59, 123, 117] also contains large-scale versions of UC that have been tackled via LR. In [117] we have one of the first attempts to obtain a primal feasible solution via means of LR rather than just using the LR for obtaining a Lower Bounds for the BnB. In [150] we observe a 3-stage approach to tackle another -for that time- large-scale UC (with 100 thermal and 6 hydros). Where in the first step a LR takes place, followed by the solution of the thermal-subproblems by means of
DP and of the hydro-subproblems by penalty multipliers method. Finally a specially tailored Newton's method is implemented. In \cite{150, 160} we can find a "unit-decommitment" approach, where originally all units are set to be committed for the entire optimization horizon. Then a decomposition procedure is followed, where units are decommitted one at a time using the LR results. An interesting comparison between a Primal MILP approach and an LR approach is carried in \cite{176}. For the LR approach a bundle method is firstly used in order to solve the LD and then two Lagrangian heuristics are implemented in an effort to recover primal feasibility. The first of the two is focused on searching the time steps with the highest demands violations and then carrying out a strategy proposed in \cite{176} for changing the commitment variables. The second one is fixing the values the commitment variables by using the Lagrangian multipliers that are near to integrality. Let us note that the LR-based method is proved to handle better the larger and more complex instances.

On this point it should be mentioned that LR is used also in relaxing constraints other than the global demand ones. The existing literature provides us with several cases where different types of thermal \cite{4, 172} or hydro constraints \cite{127, 57, 159} are relaxed. This yields to simpler subproblems to be solved, which is something vital in the cases where the units (notably the hydro ones) are highly complex units. More specifically, in \cite{128} a highly complex hydro-system that tries to approximate closely the operational and physical realities is considered. Forbidden zones and water head effect are introduced into the formulation. In order to be able to solve the problem, the demand constraints together with the reservoir balance equations are dualised, a sub-gradient method is used to solve the LD, and then a heuristic that fixes the hydro discrete variables is used to obtain primal feasible solutions.

2.2.4 Augmented Lagrangian Relaxation

As already discussed above, the LR has the drawback that doesn’t guarantee to find a primal feasible solution. A potential work around for this issue is the adoption of the Augmented Lagrangian Relaxation, where besides the linear penalisation via the Lagrange multipliers a quadratic penalisation is added. This approach can be expected to improve the chance of finding a feasible solution, yet again since as already mentioned the examined UC problems in most of the cases exhibit a non-convex nature the obtained solutions tend to be local optima. On top of this issue needs to be added the fact that this method makes it not possible to separate the original problem to independent subproblems. In order to achieve separability several cases such as \cite{11, 12} use the auxiliary problem principle. Another work around as shown in \cite{16} is the use of an alternative decomposition scheme that is based in block coordinate descent. In addition to the above, AL approaches have been used to tackle other ingredients of the model such as enviromental constraints \cite{167} and network transmission constraints \cite{16}.

2.2.5 Meta-Heuristics

Since in general solving UC is a particularly difficult and delicate task, where so far no previously mentioned method seems to be a clear and global winner, several different heuristic approaches have been proposed to tackle the UC. However none of these approaches are considered to be particularly successful. On top of that in most cases these approaches only include some of the very basic characteristics of the problem, omitting some hydro and thermal constraints.
As a result of this we will not give a thorough account of their history and of the several different approaches that exist. We refer the interested reader to [3, 54, 148]. In chapter 5.5 of the thesis we will return to the discussion of heuristic approaches, but focusing in their implementation in order to obtain primal feasible solution after the use of decomposition methods.
Chapter 3

The Thermal Unit Commitment Problem

In this section we will focus our discussion on the restricted version of the UC that takes into consideration only the thermal part of the problem and thus can be considered as the Thermal Unit Commitment Problem. Our motivation to do so derives from the fact in that the last decade there has been extensive work in both deriving tighter formulations and more efficient solution algorithms for the Thermal UC. This is of significant interest and importance for the overall UC Problem, since even if other types of units are assigned to the Problem, it is common for the original problem to be decomposed into several independent subproblems. Thus exploring and achieving tight formulations and efficient solution approaches to tackle parts of the UC can be a critical step in improving the solution scheme for the more generalized and global problem.

The chapter is organized as follows, in 3.1 we proceed with a short review of some of the main studies that have been done for exploring and formulating tight formulations on the Thermal UC based on polyhedral analysis of its constraints. In 3.2 we proceed with demonstrating and testing the efficiency of some DP algorithms that are constructed to tackle the thermal subproblems. Finally in 3.3 we demonstrate and test the efficiency of two different new MILP formulation for the thermal subproblems that were inspired by the DP-algorithms presented in 3.2. Let us note that all the DP algorithms discussed in this chapter and one of out of the two MILP-formulations (namely 3.3.1) have been developed outside the work of this thesis. Thus our work has been focused in applying and testing the proposed formulations and DP-algorithms via the use of SMS++ that will be further discussed in detail in 6.

3.1 Tighter Formulations

Looking in the existing literature for tight formulation of the Thermal UC we need to take a look at the polyhedral analysis that has been focused on the different sets of constraints that come with the thermal units. More specifically we proceed with a look at the existing studies and formulations that have been done for the following sets of constraints, whose basic formulation has already presented in the 2.
• Minimum Up/Down Time Constraints

• Ramp Up/Down Time Constraints

• Power Output Constraints

• Time Dependent Start-Up Costs Constraints

Before proceeding with the description of the existing work on the independent sets of constraints, let us introduce two extra sets of binary variables that will be used throughout the rest of the section. This is done because the 3-binary formulations are proved to be more compact and efficient than the 1-binary ones for the UC, in the sense that they provide tighter formulations. The extra two binary variables are:

• $v_{i,t}$, variable denoting the start up of thermal unit $i \in I$ at time step $t$.

• $w_{i,t}$, variable denoting the shut down of unit $i \in I$ in time step $t$.

Finally note that for all the different sets of constraints in this chapter unless specified differently in the specific constraints, it is assumed that they are valid for $i \in I, t \in T$ and thus not being displayed in every independent formulation.

### 3.1.1 Minimum Up/Down Time Constraints

In [102] is provided a polyhedral description of the minimum up/down time constraints. Their proposed formulation used the 1-binary set approach (i.e. only commitment variables) with an exponential number of constraints, that can be separated in polynomial time. In [143] an extended formulation of the minimum up/down time constraints is presented, that has a polynomial number of constraints and variables. Their approach uses the 3-binary variables and the following constraints are introduced, which are used in place of (2.1.2.4)-(2.1.2.5) and are also proven to be facets of the convex hull of the examined min up/down time constraints polytope:

$$\sum_{s \in T(t - \tau_i^u + 1, t)} v_{i,s} \leq u_{i,t} \quad t \in \{\tau_i^u + 1, T\} \quad (3.1.1.1)$$

$$\sum_{s \in T(t - \tau_i^d + 1, t)} w_{i,s} \leq 1 - u_{i,t} \quad t \in \{\tau_i^d + 1, T\} \quad (3.1.1.2)$$

$$u_{i,t} - u_{i,t-1} = v_{i,t} - w_{i,t} \quad t \in \{2, T\} \quad (3.1.1.3)$$

### 3.1.2 Ramp Up/Down Time Constraints

An other set of constraints where several research has been reported in the literature are the ramp constraints. Let us begin with [131] where it is shown that (2.1.2.2)-(2.1.2.3) can be strengthened by the following formulation:
3.1. Tighter Formulations

\begin{align*}
p_{i,t} - p_{i,t-1} & \leq \Delta^+_{i,t} u_{i,t-1} + p^0_{i,t} v_{i,t} \quad (3.1.2.1) \\
p_{i,t-1} - p_{i,t} & \leq \Delta^-_{i,t} u_{i,t} + p^0_{i,t} w_{i,t} \quad (3.1.2.2)
\end{align*}

In the same paper the above constraints are further strengthened with the adoption of the following sets of constraints:

\begin{align*}
p_{i,t} - p_{i,t-1} & \leq \Delta^+_{i,t} u_{i,t} - p^0_{i,t} w_{i,t} - (\Delta^+_{i,t} - p^0_{i,t} + p^0_{i,t}) w_{i,t+1} + (p^0_{i,t} - \Delta^+_{i,t}) v_{i,t} & t = \{1, T-1\} \quad (3.1.2.3) \\
p_{i,t-1} - p_{i,t} & \leq \Delta^-_{i,t} - u_{i,t} + p^0_{i,t} w_{i,t} - (\Delta^-_{i,t} - p^0_{i,t} + p^0_{i,t}) v_{i,t-1} - (\Delta^-_{i,t} + p^0_{i,t}) v_{i,t} & t = \{2, T\} \quad (3.1.2.4)
\end{align*}

We note first that the above inequalities are valid under the condition that $\Delta^+_{i,t} > p^0_{i,t} - p^0_{i,t}$. Now let us break down the constraints and explain their validity. To begin with it is true that in $(3.1.2.3)$ at most one of $w_{i,t}$, $w_{i,t-1}$, and $v_{i,t}$ can be equal to one. In case $w_{i,t} = 1$, then $u_{i,t} = p_{i,t} = 0$ and thus we have that $p_{i,t-1} \geq p^0_{i,t}$. If $w_{i,t-1} = 1$ then $u_{i,t} = 1$ and the constraint takes the form $p_{i,t} - p_{i,t-1} \leq \Delta^+_{i,t} - p^0_{i,t} - p^0_{i,t}$, which is valid due to the fact that then we have that $p_{i,t} \leq \Delta^+_{i,t}$ and $p_{i,t-1} \geq \Delta^+_{i,t}$. Finally, if $w_{i,t} = w_{i,t-1} = 1$ and $v_{i,t} = 0$, then constraint $(3.1.2.3)$ reduces to $p_{i,t} - p_{i,t-1} \leq \Delta^+_{i,t}$. In a similar fashion we can prove the validity of $(3.1.2.4)$.

An other interesting study has been done in [45]. In this where the ramp-up polytopes and ramp-down polytopes are defined and studied separately, and the following constraints are proposed and are shown to be valid for $t = \{1, T-1\}$:

\begin{align*}
p_{i,t+1} - p_{i,t} & \leq (p^0_{i,t} - p^0_{i,t} - \Delta^+_{i,t}) v_{i,t+1} + (p^0_{i,t} + \Delta^+_{i,t}) u_{i,t+1} - p^0_{i,t} u_{i,t} \quad (3.1.2.5) \\
p_{i,t} - p_{i,t+1} & \leq (p^0_{i,t} - p^0_{i,t} - \Delta^-_{i,t}) w_{i,t+1} + (p^0_{i,t} + \Delta^-_{i,t}) u_{i,t} - p^0_{i,t} u_{i,t+1} \quad (3.1.2.6)
\end{align*}

Is straightforward to observe that $(3.1.2.5)$ & $(3.1.2.6)$ are stronger than $(3.1.2.1)$ & $(3.1.2.2)$ respectively. That is because $(3.1.2.1)$ can be obtained from the sum of $(3.1.2.5)$ together with $v_{i,t+1} \geq u_{i,t+1} - u_{i,t}$ multiplied by $p^0_{i,t} + \Delta^+_{i,t}$. Similarly, $(3.1.2.2)$ can be obtained as the sum of $(3.1.2.6)$ alongside with $w_{i,t+1} \geq u_{i,t} - u_{i,t+1}$ multiplied by $p^0_{i,t} + \Delta^-_{i,t}$ in the same paper a generalization of the above two constraints is proposed, that derive families of inequalities of exponential size and in the paper provides exact polynomial separation algorithms in order to deal with them.

3.1.3 Power Output Constraints

Another type of constraint that has been studied and tighter formulations proposed is the upper bound constraints in the power output. More specifically in [131] the following formulation is proposed:

\begin{equation}
p_{i,t} \leq \bar{p}_{i,t} u_{i,t + K_i(t)} + \sum_{\tau = 1}^{K_i(t)} (p^0_{i,t} + (\tau - 1) \Delta^-_{i,t}) w_{i,t + \tau} - \sum_{\tau = 1}^{K_i(t)} \bar{p}_{i,t} v_{i,t + \tau} \quad , \quad (3.1.3.1)
\end{equation}
where $K_i(t) = \max\{k \in N : k \leq \tau_i^+ + p_{i,t}^0 + (k - 1)\Delta_{i,t} < p_{i,t} - k < T\}$. This constraint is making the maximum power output $p_{i,t}$ being a function of variables $v$ and $w$. More specifically $K_i(t)$ via constraining its upper bound to be equal to maximum up time ($k \leq \tau_i^+$) ensures that at most one of $w$ variables and one of $v$ variables within the time interval $[t + 1, \ldots, t + K_i(t)]$ can be equal to $1$. And if such case occurs, then also $u_{i,t+K_i(t)} = 1$. Finally the above constraint doesn’t substitute (2.1.2.1).

Moreover in [43] the following constraints for the upper bound values of power output are proposed:

\begin{align*}
    p_{i,t} &\leq p_{i,t}u_{i,t} - (p_{i,t} - p_{i,t}^0)v_{i,t} & (3.1.3.2) \\
    p_{i,t} &\leq p_{i,t}u_{i,t} - (p_{i,t} - p_{i,t}^0)w_{i,t+1} & (3.1.3.3)
\end{align*}

Finally an even stronger formulation than the above was presented in [72], where the convex hull of 1UC with only min-up/down time constraints was described. In this paper the constraints for the upper bound power output that dominates the constraints (3.1.3.2) - (3.1.3.3), which are split based on the value of $\tau_i^+$. More specifically in case $\tau_i^+ \geq 2$ the following constraint is introduced, which is valid for $t = \{2, \ldots, T - 1\}$:

\begin{align*}
    p_{i,t} &\leq p_{i,t}u_{i,t} - (p_{i,t} - p_{i,t}^0)v_{i,t} - (p_{i,t} - p_{i,t}^0)w_{i,t+1} & (3.1.3.4)
\end{align*}

and in the case $\tau_i^+ = 1$, we have:

\begin{align*}
    p_{i,t} &\leq p_{i,t}u_{i,t} - (p_{i,t} - p_{i,t}^0)v_{i,t} - \max(p_{i,t}^0 - p_{i,t}^0, 0)w_{i,t+1} & (3.1.3.5) \\
    p_{i,t} &\leq p_{i,t}u_{i,t} - (p_{i,t} - p_{i,t}^0)v_{i,t} - \max(p_{i,t}^0 - p_{i,t}^0, 0)w_{i,t+1} & (3.1.3.6)
\end{align*}

\section{3.1.4 Time Dependent Start-Up Costs Constraints}

Looking at existing work on improving the time dependent start up constraints one can start with [131] where start-up and shut-down variables are used in order to strengthen the start up constraint resulting in the following formulation:

\begin{align*}
    s_{i,t} \geq \sigma_{i,t}(v_{i,t} - \sum_{j=2}^{\tau} w_{i,t-j+1}) \quad \tau \in \{\tau_i^-, \tau_i^+\} & (3.1.4.1)
\end{align*}

Moving forward we proceed with [126], where a new formulation is introduced, which uses a new binary variable $\delta_{i,t,\tau}$, which denotes the paying of start-up cost $\sigma_{i,t}$ by the unit $i$ in the time instance $t$. Then the following constraints are introduced:

\begin{align*}
\sum_{\tau = \tau_i^-}^{\tau_i^+} \delta_{i,t,\tau} &= v_{i,t} \\
\delta_{i,t,\tau} &\leq w_{i,t-\tau} \quad \tau = \{\tau_i^-, \ldots, \tau_i^+ - 1\} \\
c_{i,t}^0 &\geq \sum_{\tau = \tau_i^-}^{\tau_i^+} \sigma_{i,t}\delta_{i,t,\tau}
\end{align*}

The first constraint ensures that only one specific start-up cost from the possible ones will be paid each time the unit starts, and the second one ensure that the min-down time will be respected.
3.2 DP Approaches for the 1UC

As already shown above the thermal version of the Unit Commitment Problem has received a lot of attention and research throughout the years. Apart from polyhedral analysis and formulations another important part of the research has been on solution approaches on the so called self-scheduling UC or the single unit Unit Commitment (1UC) problem. In the broader scheme of the overall UC these kind of problems arise after the original problem is decomposed leading to a series of independent subproblems (1UCs) that one needs to solve. In the rest of the section we will discuss for DP-approaches that tackle the following thermal-1UC, which as mentioned above have not been developed as part of our work and thus corresponding citations will be given:

\[
\min c(u) + \sum_{t \in T} f^t(p_t) \tag{3.2.0.1}
\]

\[
\text{s.t.}
\]

\[
p^*_t u_t \leq p_t \leq \bar{p}_t u_t \tag{3.2.0.2}
\]

\[
p_t - p_{t-1} \leq +u_{t-1}\Delta^+_t + p^0_t(1 - u_{t-1}) \tag{3.2.0.3}
\]

\[
p_{t-1} - p_t \leq u_t\Delta^-_t + p^0_t(1 - u_t) \tag{3.2.0.4}
\]

\[
u_t \leq 1 - u_{t-1} + u_r \quad r \in [t - \tau_+, t - 1] \tag{3.2.0.5}
\]

\[
u_t \geq 1 - u_{t-1} - u_r \quad r \in [t - \tau_-, t - 1] \tag{3.2.0.6}
\]

where the \(i\) index has been dropped since we are in the case of the 1UC and on top of this in the objective function \(c(u)\) refers to the cost of schedule that can include fixed generating cost and time-dependent or time-invariant start-up cost. Moreover \(f^t(p_t)\) is a closed convex power-generating cost function, where all its cost are connected with the power production (i.e. \(f^t(0) = 0\)). For our problems in instances solved in the current subsection we select a diagonal quadratic of the form \(f^t(p_t) = a \cdot p^2_t + b \cdot p_t\). On top of that power production bounds and ramp constraints are included via \((3.2.0.2) - (3.2.0.4)\). Finally \((3.2.0.5) - (3.2.0.6)\) refers to the min up/down constraints, which are also taken into account.

### 3.2.1 A DP-Procedure for simplified / UC without Ramp-Constraint

A well known efficient approach in the simplified case where no ramp constraints exist proposed in [63] is to proceed with the following two stage solution scheme. First compute for each time step the optimal generated unconstrained power:

\[
\hat{p}_t = \arg\min \{ f^t(p) : p \in \mathbb{R} \} \tag{3.2.1.1}
\]

Note that the above unconstrained function is not in all the cases leading to 0 because there are expected cases where costs are usually modified by Lagrangian multipliers and thus they end up with negative values or there can be incomes from sales. Thereafter proceed with using this result in order to compute the optimal power production level in case the unit is on, which is achieved by just projecting \(\hat{p}_t\) into the feasible set \([\underline{p}, \bar{p}]\):

\[
p^*_t = \min \{ \underline{p}, \max \{ \hat{p}_t, p^*_t \} \} = \arg\min \{ f^t(p) : p_t \leq p \leq \bar{p}_t \} \tag{3.2.1.2}
\]
Then we have that value $z^t = f^t(p_t^*\cdot)$ is the contribution of variable $p_t$ to the objective function value if the unit is committed, whereas in the case the unit has been off in $t$ the corresponding contribution is set to 0. As a result one has $z^t$ as the optimal cost of operating the unit at time instant $t$ if it is on and together with the fixed and start-up costs, this provide the overall costs of problem. This leads to the elimination of the $p_t$ variables from the problem. Thereafter with the use of a DP procedure using the first step’s results allows the computation of the optimal set of time periods where the unit has to remain on.

More specifically in the DP approach for the most simple case where start-up cost are not depending of the time-period the unit has been previously remained down, the state space of the dynamic programming is made up of $2\cdot T$ nodes, labeled $(t, ON)$ and $(t, OFF)$ for $t \in T$, representing respectively the unit’s commitment state at $t$, together with a source $s$ and a sink $d$ node. Moreover there are arcs between nodes $(t, ON)$ and $(t + 1, ON)$ for all $t < T$, denoting the decision of unit, that after passing the $\tau^+$ periods of mandatory commitment, to remain online at time instant $t$, which is also labeled with the sum of the corresponding $z^t$ and any other fixed costs. In the same fashion there are arcs between nodes $(t, OFF)$ and $(t + 1, OFF)$ for all $t < T$, denoting the decision of unit to remain offline for time step $t$ after having passed the $\tau^-$ time-steps of compulsory uncommitment, these arcs are labeled with zero cost. On top of that there exist also arcs for state switches, where more specifically we have to take into consideration all the arcs from $(t, ON)$ to $(t + \tau^-, OFF)$, with zero cost, which demonstrate that the unit has been shut down at time instant $t + 1$ and will be required to remain offline for the following $\tau^-$ periods. And also the arcs $(t, OFF)$ to $(t + \tau^+, ON)$, denoting the start-up of the unit at time instant $t + 1$ and its compulsory commitment for the following $\tau^+$ time steps, where these arcs are assigned with the corresponding start-up cost together with the generating and fixed costs for the overall considered time-period.

On top of the above one has to also take into consideration that for the nodes where $t + \tau^+ (\tau^-)$ is larger than $T$, the arcs are linked directly to the sink $d$, and the cost is properly modified. Then, there are arcs from the source $s$ to the nodes compatible with the initial state of the unit. That is, if the unit is initially uncommitted since $\tau^0$ time periods, there is an arc from $s$ to $(\max\{\tau^- - \tau^0, 1\}, 0)$; if $\tau^- > \tau^0$ this indicates that the unit has to remain uncommitted for the first $\tau^- - \tau^0$ time periods. The corresponding equivalent holds for the case that the unit has been initially committed since $\tau^0$ time periods, where we will have an arc from $s$ to $(\max\{\tau^+ - \tau^0, 1\}, 1)$; if $\tau^0 < \tau^+$ this indicates that the unit has to remain committed for the first $\tau^+ - \tau^0$ time periods, with appropriate cost. Finally, there are zero-cost arcs from $(T, ON)$ and $(T, OFF)$ to the sink $d$.

It is straightforward to see that every $s - d$ path on this graph represents a feasible solution, and thus (1UC) is reduced to a shortest path problem on an acyclic graph, that can be solved in linear time on the number of arcs, i.e., in $O(T)$. The only modification to this that one has to consider is the case where time-dependent start-up costs are added to the problem. In this case we have a new graph denoted as $H$, that is an extention of the above described in order to include nodes $(t, -k)$, which are displaying that the unit has remained uncommitted for the last $k$ consecutive time-steps. Of course the corresponding arcs need to be included as well. The maximum value of $k$ in this graph refers to the amount of time steps after which no further increase of offline timesteps increases the start-up costs. This is usually referred as ‘cool off’, and in most of the cases is smaller than the overall time-horizon. Finally this modification results to a change of both the complexity and the graph’s size from $O(T)$ to $O(T^2)$ in the worst case.
The above described procedure clearly fails to compute feasible solutions that include ramp
constraints. By taking a look in the existing literature we find in [14] the ramp constraints
having been included to the DP process by the discretization of the power variables space but
this method cannot be considered to be efficient since the derived solutions are only approximate
ones and the computational size increases significantly. An efficient DP approach that takes
ramp-constraint into consideration has been proposed in [53]. The main idea in this approach is
the redefinition of the state space of the dynamic programming procedure so that computation of
the state costs reduces to a convex problem. The derived problem is the well-known economic
dispatch problem with ramping constraints (ED). The ED Problems are tackled with yet an
other DP procedure proposed by the authors of the paper, that solves them with a piecewise-
linear cost function. And so all in all a two nested DP procedure needs to be followed in order
to come-up with an overall efficient method for the problem. In this subsection we will focus
on an efficient DP procedure that was originally proposed in [63], which manages to solve the
1UC with ramp constraints with general convex cost functions. The proposed solution method
manages to solve the overall problem in $O(T^3)$, once all the costs of the nodes of the graph
has been first computed. In the following paragraphs we proceed with presenting the main
ingredients and the solution scheme of this method.

We start with the introduction of the following directed graph $G = (N, A)$. More specifically $N$
consists of all the nodes of the form $(h, k)$, where $h$ denoted the time period in which the unit
switched from offline to online state and $k$ denotes the last time period that it remained active
before being again offline in time period $k + 1$. Of course due to the min-up constraints the states
where $k < h + \tau_+ - 1$ are excluded since they are infeasible. As far as the arcs of the graph are
concerned, there are arcs to connect all nodes $(h, k)$ and nodes $(r, q)$, where $r > k + \tau_+ + 1$. In
this way it is assured that the Graph has only forward arcs satisfying the min-down constraints.

All the arcs are assigned with the corresponding time-dependent start-up costs for all the time-
steps between $k$ and $r$. Moreover there are arcs from the source $s$ that arrive to all different
nodes with respect to the initial state of the unit. That is in case the unit has been initially
uncommitted for a number of $\tau_0$ time periods then there is an arc for all nodes $(h, k)$ with
$h \geq \tau_- - \tau_0 + 1$. Also these arcs are assigned with the equivalent start-up costs. If in contrast the
unit has been initially committed for $\tau_0$ units time steps, then all the arcs $(1, k)$ such that $k \geq \tau_- - \tau_0$
are included. Moreover there is also an arc from each node to the sink $d$ that is assigned zero
costs. In order to include the fixed production costs we associate with each node $(h, k)$ the
corresponding fixed costs for the time period $h - k$, since the unit will remain committed to
producing power for all the time-steps between $h$ and $k$. Finally it is straightforward to see that
every $s - d$ path of the graph $G$ represents a feasible solution to the 1UC.

The next important thing to be considered is the optimal power production $p_t$ for each node
$(h, k)$ of each variable $p_t$. This is computed in polynomial time for each time interval $[h, k]$ with
the following economic dispatch with ramp constraints problem:
CHAPTER 3. THE THERMAL UNIT COMMITMENT PROBLEM

\[
\min \sum_{t=h}^{k} f^t(p_t) \tag{3.2.2.1}
\]

s.t.
\[
\begin{align*}
\underline{p}_t \leq p_t \leq \overline{p}_t & \quad h \leq t \leq k \tag{3.2.2.2} \\
\underline{p}_k & \leq \overline{p}_k \tag{3.2.2.3} \\
\underline{p}_t & \leq \overline{p}_t & \tag{3.2.2.4} \\
p_{t+1} - p_t & \leq \Delta^+_t & t = \{h, \cdots, k-1\} \tag{3.2.2.5} \\
p_t - p_{t+1} & \leq \Delta^-_t & t = \{h, \cdots, k-1\} \tag{3.2.2.6}
\end{align*}
\]

The problem (3.2.2.1) - (3.2.2.6), in which all the binary committed variables are fixed, will be denoted by \((ED_{hk})\) and as a result we remain with an optimization problem with linear constraints and convex objective function, whose optimal solution will be referred as \(z^*_{h,k}\). It can be solved in polynomial time for any arbitrary convex function \(f^t\). Moreover if the costs of the obtained \(z^*_{h,k}\) are added to the nodes, then the \(s-d\) path of the graph \(G\) represents the overall cost of the feasible solution. As a result the 1UC is being reduced to a shortest-path problem on an acyclic graph with \(O(T^2)\) nodes and \(O(T^4)\) arcs. So the overall problem can be solved in \(O(T^4)\), after all its corresponding data has been computed.

In order to further decrease the complexity of the solution process the authors of [63] take advantage of the following properties of the state space of graph \(G\). Starting point is the partitioning of the set of nodes \((h,k)\) into the following levels:

\[V_k = \{(h,k) : 1 \leq h \leq k\} \text{ for } k \geq 1 \text{ (}V_0\text{ contains starting node } s \text{ only)}\]

From the definition of \(G\) it follows that:

- all nodes in \(V_k\) have the same set of adjacent nodes
- the cost of the arc between \((h,k)\) and \((r,q)\) depends only on \(k\) and \(r\).

As a result of the above \(G\) can be visited in an ascending order of level \(k\), without having to explicitly explore the forward \(z^*\) of all but one node at each level \(k = 1, \cdots, n\). The proposed procedure can be described in the following way. For each level \(k\) there a list \(S_k \subset V_k\) exists that stores the visited nodes \((h,k) \in V_k\), with the length of the shortest path obtained so far being held in \(d_{h,k}\) (\(S_0\) contains the starting node \(s\) with zero cost). Then one has to repeat over \(k = 1, \cdots, n\) the following steps:

1. calculation of the \((ED_{hk})\) for all nodes in \(S_k\).
2. Find node \((h,k)\) in \(S_k\) with the smallest value of \(z^*_{h,k} + d_{h,k}\)
3. Visit all adjacent nodes \((r,q)\) of \((h,k)\) and compute \(d_{r,q}\), which is set equal to the sum of \(z^*_{h,k} + d_{h,k}\) with the cost of the arc that connects the examined node \((r,q)\) with \((h,k)\).
   - If node is visited for first time it is inserted to \(S_q\)
• Otherwise if the new value is smaller than the existing one, its label is upgraded.

The above order is a valid one, and the visit terminates, having obtained a shortest s âL S d path. In principle, all $O(T^2)$ nodes of the Graph $G$ are visited, and thus one has to compute all the $(ED_{hk})$ for all. Yet again for each $k$ one has to take into consideration only the node $(h, k)$ that is associated with the shortest path from $s$, which leads to the result that only $O((T-k)^2)$ forward arcs need to be considered. As a result the complexity of the visit drops to $O(T^3)$ together with the cost of solving the $O(T^2)$ $(ED_{hk})$.

### 3.2.3 Solving the Economic Dispatch Problem (EDP)

Since the EDP constitutes an important part of the overall solution scheme of tackling the 1UC via a DP approach, we now consider how it is constructed and solved. The proposed approach has originally been presented and proposed in [63]. The main idea behind the proposed algorithm for tackling sequences of $(ED_{hk})$ problems, lies in the exploiting of the solution of $(ED_{hk})$ in order to tackle the next one in sequence problem $(ED_{h(k+1)})$. This is achieved by the construction of the parametric problem $(ED_{hk}({\bar{p}}))$, that represents the restriction to $(ED_{hk})$ where the last variable $p_k$ is set to be fixed to the value $\bar{p}$ (i.e. imposing the extra constraint $p_k = \bar{p}$). In order for the overall procedure to be clearly understood, we need to follow and present here the properties of the optimal objective function value of $(ED_{hk}({\bar{p}}))$ as a function of the parameter $\bar{p}$. We start with a simplification of the examined function, which will be defined as follows:

$$z_{hk}(\bar{p}) = \begin{cases} \min \{ f^h(p_h) : \text{[3.2.2.2], [3.2.2.3], } p_h = \bar{p} \} & \text{if } h = k, \\ \min \{ \sum_{t=h}^k f^t(p_t) : \text{[3.2.2.2], [3.2.2.3], [3.2.2.5], [3.2.2.6], } p_k = \bar{p} \} & \text{otherwise.} \end{cases}$$

In the above formulation one can see that $\bar{p}$ is allowed to take any value inside $[p_t^l, p_t^u]$, including the values of fixing $p_k = \bar{p}$, that would result in infeasibility due to (3.2.2.4). The reason for this choice lies in the fact that $z_{hk}$ will be use for the computation of $z_{h(k+1)}$, and in this case (3.2.2.4) corresponds to variable $p_{k+1}$, and as a result is no longer binding for $p_k$.

Furthermore some part of the algorithm depends on some general properties of the function, which were proven to be valid in [63]. Since the described procedure has been implemented besides [63] also via SMS++ as part of our work in this thesis, we will proceed with presenting these properties and demonstrating their corresponding proof. The first property that one has to take into consideration it that the function $z_{hk}$ is convex. The convexity of the function derives from well-known general properties which won’t be further explained here, and we only point out the fact that $z_{hk}$ is the value function of the convex program $(ED_{hk}(\bar{p}_k))$ with respect to the right-hand side $\bar{p}_k$ of its constraint $p_k = \bar{p}_k$.

Moreover let us denote the fact that the function has a piecewise nature, that is, it is finite-valued only in $v + 1$ intervals $[m_0, m_1], [m_1, m_2] ... [m_v, m_{v+1}]$, with $l^k \leq m_0, m_{v+1} \leq u^k$ and $v \leq 2(k-h)$, in which

$$z_{hk}(\bar{p}) = z^i(\bar{p}) \quad \text{if } \bar{p} \in [m_i, m_{i+1}] \text{ for } i = 0, 1, \ldots , v$$

where each function $z^i$ is the sum of at most $k-h+1$ functions $f^t$ for $t \in \{h, h+1, \ldots , k\}$ (and therefore it is convex). In the following few paragraphs we will demonstrate some of the key...
elements of the validity of the above statements. The full description of the proves is found in
the paper mentioned above and the interested reader is referred there for more specific details.

The prove of the above statements will be achieved by induction, and the corresponding piecewise
representations of the functions \(z_{hk}, z_{h(h+1)}, \ldots z_{hk}\) will be constructed as well. On top of this
at every time step \(k\) there will be also computed \(p_{hk}^* = \text{argmin}\{ z_{hk}(p) : p \in [l^k, u^k]\}\), which
refers to the \(k\)-th component of the optimal solution of \((ED_{hk})\) where constraint (3.2.2.4) is
relaxed.

The first step for the overall induction process refers to the case where \(k = h\). In this case
things are pretty obvious and simple since we have that \(z_{hk}(\bar{p}) = f^h(\bar{p})\) for all \(\bar{p} \in [\bar{p}_h, \bar{p}_h^0]\). So
we have that exist \(v + 1 = 1\) intervals and \(v + 1 = 1\) functions with the required properties
(i.e., \(0 = v \leq 2(h - h) = 0\)). Moreover we have that \(p_{hk}^*\) is equal to \(p_h^*\) as computed with
formulas (3.2.1.1) and (3.2.1.2), but with its upperbound being \(p_h^*\). Now we also need to examine
the case where \(h = 1\) and we have that the unit is online, in this scenario we have \(z_{11}(\bar{p})\) defined
as follows. We have that (3.2.2.3) is not applied, but we need to include the ramping constrain
ts and \(p_0\). The above lead to \(\bar{p} \in [\max\{l^1, p_0 - \Delta_1^+\}, \min\{u^1, p_0 + \Delta_1^+\}\}].

Now we proceed with assuming that the above conditions have been proved and hold for some
value of \(k\), and also we assume that the equivalent set of intervals and functions has already
been computed, and we proceed with the prove of the conditions hold for \(k + 1\), too. For this
we will denote \(\hat{m}_i\) as the extremes of the intervals for \(z_h(k+1), z^i\) the corresponding functions
and \(\bar{v} + 1\) their number. It can easily understood that for any fixed value \(\bar{p} \in [\bar{p}_{k+1}, \bar{p}_{k+1}]\) the
constraints (3.2.2.5) and (3.2.2.6), written for \(p_{k+1} = \bar{p}\) and \(p_k\), result in \(\bar{p} - \Delta_k^+ \leq p_k \leq \bar{p} + \Delta_k^-\).

Since \(z_{hk}\) is infinite-valued for \(p_k\) outside \([m_0, m_{v+1}]\), one has to set \(\bar{m}_0 = \max\{p_{k+1}^*, m_0 - \Delta_k^-\}\)
and \(\bar{m}_{v+1} = \min\{p_{k+1}^*, m_{v+1} + \Delta_k^+\}\). In fact, \(z_h(k+1)\) is clearly infinite-valued outside this
interval, and finite-valued inside it. Let us note as well that if \(\bar{m}_0 > \bar{m}_{k+1}\) infeasibility of the (1UC)
problem has been detected. On this point since we have already computed the piecewise representation of \(z_{hk}\), we can exploit this in order to compute the optimal solution of
\((ED_{h(k+1)}(\bar{p}))\). As a result we can rewrite the problem as:

\[z_{h(k+1)}(\bar{p}) = f^{k+1}(\bar{p}) + \min\{ z_{hk}(p_k) : m_0 \leq p_k \leq m_{v+1} , \bar{p} - \Delta_k^+ \leq p_k \leq \bar{p} + \Delta_k^- \} \].

The above implies that the optimal solution of atleast the \(k\)-th component of \((ED_{h(k+1)}(\bar{p}))\) is
found as the constrained minimum of \(z_{hk}\) in the intersection of the intervals \([\bar{p} - \Delta_k^+, \bar{p} + \Delta_k^-]\)
and \([m_0, m_{v+1}]\). Let us denote this minimum as \(p_k^*(\bar{p})\). For us to be able to define and compute
\(z_{h(k+1)}(\bar{p})\), we need to be able to know the behavior of \(p_k^*(\bar{p})\) to the changes of \(\bar{p}\). This is
something we can easily understand taking into account the following things.

To do this we start from the assumed to be already computed constrained minimum \(p_{hk}^*\) of \(z_{hk}\)
over \([m_0, m_{v+1}]\). Since \(z_{hk}\) is convex, \(p_k^*(\bar{p})\) is simpl its projection over the feasible interval

\[\bar{p} - \Delta_k^+ , \bar{p} + \Delta_k^- \] (cf. (3.2.1.2)), that is,

\[p_k^*(\bar{p}) = \min \{ \bar{p} + \Delta_k^- , \max \{ p_{hk}^*, \bar{p} - \Delta_k^+ \} \} . \] (3.2.3.2)

We have to note that we compute \(p_k^*\) under the assumption that it is unique, however, the
following arguments can be easily extended to cases where \(z_{hk}\) has a (known) non-pointed
interval as set of optimal solutions. We can proceed and describe the overall process as follows, with the help of Figure 3.1. More specifically due to the variation of \( \bar{p} \) from \( \bar{m}_0 \) to \( \bar{m}_{v+1} \), we can have the following three different cases:

(a) The first case refers to the scenario that \( \bar{p} \) is on the left end of the interval \([\bar{m}_0, \bar{m}_{v+1}]\) where \( z_{hk(k+1)} \) is finite-valued, e.g., \( \bar{p} = \bar{m}_0 \). Then we have that \( p_{hk}^* \) is “on the right” of the feasible interval \([3.2.3.1]\) (i.e., \( p_{hk} > \bar{p} + \Delta_k^k \)), and that \( p_k^*(\bar{p}) = \bar{p} + \Delta_k^k \), which means that \( p_k^*(\bar{p}) \) is a linear function of \( \bar{p} \).

(b) Moving on, as \( \bar{p} \) increases, we have that \( p_{hk}^* \) lies inside the feasible interval \([3.2.3.1]\), and as a result we have that \( p_k^*(\bar{p}) \) is equal to \( p_{hk}^* \), and keeps being fixed until \( \bar{p} \) becomes too large.

(c) Finally, the last case to be considered is the one, where \( \bar{p} \) becomes larger than \( p_{hk}^* + \Delta_k^k \). Then we have that \( p_{hk}^* \) doesn’t lie anymore within the feasible interval \([3.2.3.1]\), this time remaining “on the left”. As a result we then have that \( p_k^*(\bar{p}) = \bar{p} - \Delta_k^k \), which implies that in this case as well \( p_k^*(\bar{p}) \) increases linearly as \( \bar{p} \) does.

Figure 3.1: Evolution of \( p_k^*(\bar{p}) \) as \( \bar{p} \) varies

Based on the above, having the explicit description of \( z_{hk} \) in terms of the \( v + 1 \) sub-intervals of \([m_0, m_{v+1}]\) and the associated functions \( z^k \), is straightforward to see how is possible to construct a piecewise representation of \( z_{hk(k+1)} \) with \( \bar{v} + 1 \) intervals where \( \bar{v} \leq v + 2 \).
0) Set \( \tilde{p} = m_0, \tilde{v} = 0 \), and let \( 0 \leq q \leq v \) be the index of the interval to which \( p^*_k(\tilde{p}) = \tilde{p} + \Delta^k_\tilde{p} \) belongs (if it is a breakpoint, choose the interval on the right). Set \( \tilde{p}' = \min\{\tilde{p}_{k+1}, m_{v+1} + \Delta^k_\tilde{p}\} \).

1) If case (a) is not verified goto step 2), otherwise set \( \tilde{z}^q(p) = f^{k+1}(p) + z^q(p + \Delta^k_\tilde{p}) \). Compute the maximum value of \( \tilde{p} \) such that \( p^*_k(\tilde{p}) \) remains in the \( q \)-th interval, \( p^*_h \) remains outside the feasible interval and \( \tilde{p} \) remains feasible, that is, \( \tilde{p} = \min\{m_{q+1} - \Delta^k_\tilde{p}, \Delta^k_\tilde{p} - \Delta^k_q, \tilde{p}'\} \). Set \( \tilde{v} = \tilde{v} + 1, m_{\tilde{v}} = \tilde{p}, \) if \( \tilde{p} = \tilde{p}' \) then goto step 1).

2) If case (b) is not verified goto step 3), otherwise set \( \tilde{z}^q(p) = f^{k+1}(p) + z^q(p^*_h) \). Compute the maximum value of \( \tilde{p} \) such that \( p^*_h \) remains inside the feasible interval and \( \tilde{p} \) remains feasible, that is, \( \tilde{p} = \min\{p^*_h + \Delta^+_h, \tilde{p}'\} \). Set \( \tilde{v} = \tilde{v} + 1, m_{\tilde{v}} = \tilde{p} \) and goto step 3).

3) If \( \tilde{p} = \tilde{p}' \) then terminate, otherwise set \( \tilde{z}^q(p) = f^{k+1}(p) + z^q(p^*_h) \). Compute the maximum value of \( \tilde{p} \) such that \( p^*_h(\tilde{p}) = \tilde{p} - \Delta^h_\tilde{p} \) remains in the \( q \)-th interval and \( \tilde{p} \) remains feasible, that is, \( \tilde{p} = \min\{m_{q+1} + \Delta^+_h, \tilde{p}'\} \). Set \( \tilde{v} = \tilde{v} + 1, m_{\tilde{v}} = \tilde{p}, q = q + 1 \) and repeat step 3).

It is obvious that in total we can have for \( z_{h(k+1)} \) an number of intervals at most equal to that for \( z_{hk} \) plus the two ones corresponding to \( p^*_h \) “entering” and “leaving” the feasible set. And so we have the former interval \( q \) in step 2) being replaced by at most 3 new intervals. Note that the intervals with right extreme less than or equal to \( p^*_h - \Delta^h_\tilde{p} \), if any, correspond to intervals for \( z_{hk} \) “shifted left” by \( \Delta^h_\tilde{p} \), while the intervals with left extreme greater than or equal to \( p^*_h + \Delta^+_h \), if any, correspond to intervals for \( z_{hk} \) “shifted right” by \( \Delta^+_h \). The total number of intervals may well be strictly less than \( v + 3 \). Also, due to the fact that each \( z^i \) is composed of the sum of at most \( k - h + 1 \) original functions \( f^j \), each \( z^j \) is composed of the sum of at most \( k - h + 2 = (k + 1) - h + 1 \) original functions \( f^j \). So, finally summarising the above we complete the proof of the properties of the function that were mentioned in the begin.

Moving on, the reason why the above prove is of great importance and was mentioned here is the fact that throughout the above described process can be used in order to solve the \((ED_{hk})\) problems. This is because throught the process we can easily compute not only \( p^*_h \), but also the optimal solution of

\[
\min\{ z_{h(k+1)}(p) : p \in [p_{k+1}^0, p_{k+1}^1]\},
\]

that is, the last component of the optimal solution of \((ED_{h(k+1)})\), where constraint \((3.2.2.4)\) is imposed. So we end up with an efficient way to tackle all the corresponding ED problems that need to be computed in order to be able to implement an shortest-path algorithm in the original graph of the 1UC where now all the corresponding costs in all the arcs are computed.

As far as the complexity of the overall procedure is concerned, we need to take into account the following things. First of all for a quadratic objective function, we have each step costing \( O(1) \). And so if we take the assumption that we have already computed and tackled \((ED_{h(k-1)})\), we have that for solving \((ED_{hk})\) we have a complexity of \( O(k - h) \). Which means that the complexity for tackling all the problems associated with node \((h, k)\) are set to \( O((k - h)^2) \). As a result this means that for solving all the \( O(T^3) \) (ED) problems that are concerned within the DP-framework of the graph \( G \) described in the previous section one has \( O(T^3) \) complexity.
3.2. **Improving the DP-Procedure for the Non-linear 1UC with Ramp-Constraints**

The above efficient DP-procedure has been further improved in [68], where the following revised changes have been proposed. In this case the original graph $G$ is being altered by proceeding with a transformation of the nodes of the problem. We introduce the new graph $G' = (N', A')$, where one instead of the original set of nodes the following ones:

- $(1, ON), (2, ON), \ldots, (T, ON)$, where the set of these nodes denote the unit being committed in the corresponding time steps.
- $(1, OFF), (2, OFF), \ldots, (T, OFF)$, where the set of these nodes denote the unit being uncommitted in the corresponding time steps.

On top of this one has a start and sink node. And moreover there exist arc connections between the online and offline nodes such that the arc starting from node $(h, ON)$ and arriving to node $(k, OFF)$ denotes that the unit has been committed in time-step $h$ and then decommitted in time-step $k$, thus marking the corresponding $ED_{h,k}$. In the same manner the arc starting from node $(r, OFF)$ and arriving to node $(q, ON)$ denotes that the unit has been decommitted in time-step $r$ and then has been switched on in time-step $q$. And as a result the arc $(r \rightarrow q)$ to the start up of the unit at time step $q$ after having remained offline for a number of time steps $q - r$ and thus the arc is also assigned with the corresponding start up cost. Finally all the arcs are subjected to the min up/down constraints and only valid arc connections are being considered.

In the following we proceed with a small interpretation of the two different graphs in order for their differences to be better understood:

![Graph H](image)

**Figure 3.2: Graph H**

In this figure 3.2 that corresponds to graph $H$ and we have that $\tau^0_0 = 1$ (that is the unit has been uncommitted for one time step prior to the begin of the optimization period) and moreover that $\tau_-= 4$ and $\tau_+ = 3$. Based on this condition the figure presented can be understood as follows. Since originally the unit is uncommitted and is constrained to remain uncommitted for three more time steps, all the arcs from the start to the nodes of the first two time steps are invalid. As a result we see that the only arc from the start is allowed arrive to the third time step and more specifically to the node $(3, -4)$. Moreover apart from the arcs connecting the start node with all the corresponding valid node, we have ofcourse the arc connections that start from the first valid node $(3, -4)$. That is that there are ongoing arcs leaving from that node connecting it with node $(4, ON)$ denoting the choice of the unit to switch to online mode in which case this arc is also associated with the corresponding start up cost that refers to the
unit being offline for 4 consecutive time-steps. And on top of that there is also the arc starting from this node arriving to node $(4, -5)$ denoting the decision of the unit to continue being offline after the covering of its compulsory decommitment period due to $\tau_-$. And then we have all the corresponding possible arc connections for the rest of time steps until the sink node $d$.

In this figure 3.3 that corresponds to graph $G$ and we have that for both graphs that $\tau^0_0 = 1$ (that is the unit has been uncommitted for one time step prior to the begin of the optimization period) and moreover that $\tau_- = 3$ and $\tau_+ = 4$. Based on this condition the figure presented can be understood as follows. Since originally the unit is uncommited and is constrained to remain uncommitted for two more time steps, all the arcs from the start to the nodes of the first two time steps are invalid. As a result we see that the arcs from the start arrive to the third time steps and ofcourse all the rest time steps after it. Note ofcourse that these arcs are associated with the corresponding time dependent start-up costs of the unit. Moreover apart from the arcs connecting the start node with all the rest valid nodes, we have ofcourse the arc connections that start from the third time-step until the last one that denote all the possible valid start-ups and shut downs of the unit within the optimisation period. Finally as mentioned above all the valid nodes are connected as well with the sink node, for instance one could have that the unit is not producing any power at all within the optimization horizon if the final path would include only the arc from the start $s$ to the sink $d$. In any other case the start node would be connected with one of the rest nodes $(h, k)$ that correspond to the unit being online for a time period of $k - h$ time steps and so on.

In the second figure 3.4 that corresponds to graph $G'$ and we have that for both graphs that $\tau^0_0 = 1$ (that is the unit has been uncommitted for one time step prior to the begin of the
optimization period) and moreover that $\tau_- = 4$ and $\tau_+ = 2$. And as a result we have that the unit is initially uncommitted and is imposed to remain offline until the third time step, and thus all the online nodes prior to the third time step are not valid. On top of this also the offline nodes are not valid since the unit is already uncommitted and we have that the arcs starting from the node $s$ will be connected whether with some online node from the third time step onwards or otherwise with the sink node $d$. Moving on from the node $(3, ON)$ and all the rest online nodes we have the arcs connecting them with the offline nodes with respect to $\tau_+$. This means that from node $(3, ON)$ we have arcs going to all the offline nodes from the fifth time step onwards, for the node $(4, ON)$ we have the corresponding arcs starting from the offline node of the sixth time step and so one. Of course all these nodes are connected with the sink node as well, denoting that the unit remains online for all the rest time steps. Then of course we have all the corresponding arcs from the offline nodes to the next online ones with respect to $\tau_-$. Denoting the restart of the unit and the equivalent arc connections with the sink node. And this goes on for all the rest of the nodes until the sink node. Only further thing to be mentioned is that for time steps $t > T - \tau_+$ there are no more outgoing arcs connecting them with the offline nodes but only with the sink and in the same manner for the time steps $t > T - \tau_-$ there are no more outgoing arcs connecting them with the next online nodes but only with the sink node.

As it can be easily understood the above altered graph corresponds to a total of $2T + 2$ nodes and $O(T^2)$ arcs. Here the same algorithmic procedure is followed as well, where one has first to compute all the EDP problems that in this case corresponds to the valid arcs between every $(h, ON)$ and $(k, OFF)$ node and compute as well the corresponding start-up costs of the valid arcs between every $(r, OFF)$ and $(q, ON)$ node. And then clearly by proceeding with a shortest-path algorithm the obtain $s - d$ path will correspond to the feasible power production schedule of the unit.

Finally, as part of our work we have proceeded with implementing the above mentioned DP-approaches that are using graph $G$ and $G'$ respectively via SMS++ and comparing their efficiency, testing them over the instances that were used in [63]. Where three UC problems of 100 thermal units but with different values of $n$: 24, 96, and 168 were solved the problems with the Lagrangian approach of [28], where in all three cases 23 iterations were needed to reach the optimal solution of the Lagrangian dual with respect to demand constraints and spinning-reserve constraints. Thereafter the dual prices of these constraints at four particular iterations, that can be considered a representative sample: 1, 12, 16 and 23, that is, the first one, the last one and two intermediate iterations. And on top of that in order to achieve even bigger instances we proceeded with creating some extra artificial instances by creating lagrangian duals for $n = 336$ and 504 by taking the duals of the instance of $n = 168$ and multiplying them with random numbers in the interval between $[0.5, 2]$. Thus giving raise to a set of $100 \times 4 \times 4 = 1600$ (1UC) problems, each one corresponding to a specific unit, a given time horizon and one of the iterations of the Lagrangian approach. So we then solved all these instances using the DP approach for both graph $G$ and $G'$ on a Quad-Core Intel Core i5-4300U CPU @ 1.90GHz with 12 Gb of RAM. Our code was compiled with g++ version 5.4.1 and optimization option -O3. In [31] is displayed the fact that the longer the size of the instances the more efficiently works the second choice of graph $G'$ sth that was somehow expected.
### Table 3.1: Computational results of the DP algorithms between graphs $G$ versus $G'$

<table>
<thead>
<tr>
<th>instance $n$</th>
<th>iter. $i$</th>
<th>Graph $G$</th>
<th></th>
<th>Graph $G'$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># instances solved faster</td>
<td>avg % of time decrease</td>
<td># instances solved faster</td>
<td>avg % of time decrease</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>65</td>
<td>25.95%</td>
<td>35</td>
</tr>
<tr>
<td>12</td>
<td>68</td>
<td>27.10%</td>
<td>32</td>
<td>16.55%</td>
</tr>
<tr>
<td>16</td>
<td>80</td>
<td>25.54%</td>
<td>20</td>
<td>15.20%</td>
</tr>
<tr>
<td>23</td>
<td>78</td>
<td>24.82%</td>
<td>22</td>
<td>10.47%</td>
</tr>
<tr>
<td>96</td>
<td>1</td>
<td>42</td>
<td>20.97%</td>
<td>58</td>
</tr>
<tr>
<td>12</td>
<td>46</td>
<td>17.72%</td>
<td>54</td>
<td>17.18%</td>
</tr>
<tr>
<td>16</td>
<td>47</td>
<td>18.37%</td>
<td>53</td>
<td>18.16%</td>
</tr>
<tr>
<td>23</td>
<td>45</td>
<td>20.46%</td>
<td>35</td>
<td>18.37%</td>
</tr>
<tr>
<td>168</td>
<td>1</td>
<td>16</td>
<td>12.68%</td>
<td>84</td>
</tr>
<tr>
<td>12</td>
<td>17</td>
<td>11.83%</td>
<td>83</td>
<td>15.53%</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>14.58%</td>
<td>90</td>
<td>15.23%</td>
</tr>
<tr>
<td>23</td>
<td>12</td>
<td>6.38%</td>
<td>87</td>
<td>16.03%</td>
</tr>
<tr>
<td>336</td>
<td>1</td>
<td>18</td>
<td>11.05%</td>
<td>82</td>
</tr>
<tr>
<td>12</td>
<td>20</td>
<td>5.09%</td>
<td>80</td>
<td>11.57%</td>
</tr>
<tr>
<td>16</td>
<td>19</td>
<td>6.77%</td>
<td>81</td>
<td>15.23%</td>
</tr>
<tr>
<td>23</td>
<td>8</td>
<td>5.74%</td>
<td>92</td>
<td>9.78%</td>
</tr>
<tr>
<td>504</td>
<td>1</td>
<td>17</td>
<td>11.10%</td>
<td>83</td>
</tr>
<tr>
<td>12</td>
<td>22</td>
<td>12.05%</td>
<td>78</td>
<td>10.09%</td>
</tr>
<tr>
<td>16</td>
<td>20</td>
<td>13.19%</td>
<td>80</td>
<td>8.18%</td>
</tr>
<tr>
<td>23</td>
<td>15</td>
<td>5.74%</td>
<td>85</td>
<td>12.34%</td>
</tr>
</tbody>
</table>

3.3 New MILP Formulations for 1UC

In this part of this chapter we proceed with combining the above two parts concerning the reformulations of the problem and the corresponding DP methods for tackling it, by presenting two MILP formulation that derive from the above described DP-approaches. The formulation of 3.3.1 was initially proposed in [68] while the one of 3.3.2 has been developed as part of the work of the thesis. All three formulations have been implemented and solved with CPLEX via the use of SMS++, as part of the work of this thesis and in the end of the section are presented the corresponding extensive computational comparison between them.

Before presenting the specific formulations let us first introduce two new set of binary variables that will be used in all three different formulations, that will substitute the classic commitment variables $u$ and derive from the graph $G$:

- $y_{(h,k)}^+$: binary variable that is equal to 1, in the case the arc that starts from node $(h, ON)$ and arrives to node $(k, OFF)$ is used.

- $y_{(k,r)}^-$: binary variable that is equal to 1, in the the arc that starts from node $(k, OFF)$ and arrives to node $(r, ON)$ is used.

Now let us proceed with presenting the different formulations.
3.3. NEW MILP FORMULATIONS FOR 1UC

3.3.1 The \( p_{h,k}^t \) MILP Formulation

In this formulation there are introduced power variables \( p_{h,k}^t \) exhibit a three-indexed nature being associated with each node \((h, k)\) of the network with \( t = h, \cdots, k \). And all the constraints of the formulations hold \( \forall (h, k) \in N \):

\[
c(y) + \sum_{(h,k) \in N} \sum_{t \in T(h,k)} f(p_{h,k}^t)
\]

\[
\text{s.t.}
\]

\[
Ey = b \quad y \geq 0
\]

\[
(p_{h,k}^t)_{+} \leq p_{h,k}^t \leq (p_{h,k}^t)^0 \quad t \in T(h+1, k-1)
\]

\[
(p_{h,k}^t)_{+} \leq p_{h,k}^t \leq (p_{h,k}^t)_{-} \quad t \in T(h, k-1)
\]

We have that (3.3.1.1) denotes the objective function that is composed by the sum of the production costs of all different nodes through-out the optimization horizon, together with the stable production costs and start up costs that are connected with variable \( y^+ \) and \( y^- \) respectively. Then (3.3.1.2) refers to the equations that derive from the graph's node-arcs incidence matrix \( E \) that is multiplied with the vector of arc flow variables \( y \) and are set to be equal to the rhs vector \( b \), which has all of its entities equal to zero besides the first and the last one that refer to the source and sink node and are set to be equal to \( b_s = -1 \) and \( b_d = 1 \) respectively. (3.3.1.3) is set to replace the corresponding \( ED_{(h,k)} \) problems. More specifically for each \( ED_{(h,k)} \) there is one EDP Problem with a corresponding set of variables \( p_{h,k}^t \) for \( t = \{h, \cdots, k\} \), which are subjected to the above power output and ramp up/down constraints.

The above formulation has been proposed in [68] together with a prove that it describes the convex hull of the 1UC if the objective function \( f \) is linear. In the following paragraphs we present the main elements and points that were used and proven by the authors for the formulations. Following the prove of the authors we start with the use of the well-known principle labeled as “Approach no. 4” by Wolsey [169], and after this a lemma will be proven on the composition of polyhedra.

**Proposition 1** For \( A \in \mathbb{R}^{m \times n} \) and \( b \in \mathbb{R}^m \), let

\[
S = \{ x \in \mathbb{R}^n : Ax \leq b \quad x \geq 0 , \quad x_j \in \mathbb{Z} \quad j \in J \subseteq \{1, \ldots, n\} \} .
\]

If \( S \) is bounded, then the inequalities in (3.3.1.4) describe the convex hull of \( S \) if and only if for each vector \( c \in \mathbb{R}^n \) there exists \( \lambda \in \mathbb{R}^m \) such that

\[
\max\{ cx : x \in S \} = \min\{ \lambda b : \lambda A \geq c , \quad \lambda \geq 0 \}.
\]

**Definition 2** Let \( S_i \subset \mathbb{R}^{n_i} \times \mathbb{R} \) be two sets; their 1-sum composition is defined as

\[
S_1 \oplus S_2 = \{ (x^1, x^2, y) \in \mathbb{R}^{n_1+n_2+1} : (x^i, y) \in S_i \quad i = 1, 2 \}.
\]
The following lemma enables us to prove that the linear description of the convex hull of \( S_1 \oplus S_2 \) can be obtained from the linear description of the convex hulls of \( S_1 \) and \( S_2 \).

**Lemma 3** For \( i = 1, 2 \), let \( S_i \subset \mathbb{R}^{n_i} \times \mathbb{R} \) be two sets, and suppose that \( P_i = \{ (x^i, y) \in \mathbb{R}^{n_i} \times \mathbb{R} : A^i x^i + a^i y \leq b^i, \ x^i \geq 0, \ y \geq 0 \} \) describes the convex hull of \( S_i \) for \( i = 1, 2 \). If \( (x^i, y) \in S_i \) implies that \( y \in \{0, 1\} \) for \( i = 1, 2 \), then the system

\[
\begin{align*}
A^1 x^1 + a^1 y & \leq b^1 \\
A^2 x^2 & \leq b^2 \\
x^1, y, & \geq 0
\end{align*}
\]  
(3.3.1.5)

describes the convex hull of \( S_1 \oplus S_2 \).

**Proof.** We prove that for every \( (c^1, c^2, d) \in \mathbb{R}^{n_1+n_2+1} \) there exists a dual optimal solution for \( (3.3.1.5) \) whose objective function value is equal to \( z = \max \{ c^1 x^1 + c^2 x^2 + dy : (x^1, y) \in S_i, \ i = 1, 2 \} \), according to Proposition [1]. This proof follows similar steps as the proof by Chvátal [39] for composition of stable set polyhedra by clique-cutsets.

For \( y \in \{0, 1\} \), we define \( z^i_0 = \max \{ c^i x^i + dy : (x^i, y) \in S_i \} \). Note that \( \max \{ c^i x^i + (d + z^i_0 - z^i_1) y : (x^i, y) \in S_i \} = z^i_0 \) for \( i = 1, 2 \). Therefore, by Proposition [1] there exists \( \lambda^i \) such that

\[
\lambda^i A^i \geq c^i, \ \lambda^i a^i \geq d + z^i_0 - z^i_1, \ \lambda^i b^i = z^i_0 .
\]

Let \( z_0 = z^1_0 + z^2_0, z_1 = z^1_1 + z^2_1 - d \), and \( z = \max \{ z_0, z_1 \} \). Clearly \( z - z_0 \geq 0, \max \{ (z - z_0) y : (x^1, y) \in S^1 \} = z - z_0 \), and by Proposition [1] there exists \( \gamma \geq 0 \) such that

\[
\gamma A^1 \geq 0, \ \gamma a^1 \geq z - z_0, \ \gamma b^1 = z - z_0 .
\]

Finally, consider the dual solution of \( (3.3.1.5) \) given by \( \mu = [\mu^1, \mu^2] = [\lambda^1 + \gamma, \lambda^2] \), where \( \mu^1 = \): the following relationships hold

\[
\begin{align*}
\mu^1 A^1 &= \lambda^1 A^1 + \gamma A^1 \geq c^1 + 0 = c^1 \\
\mu^2 A^2 &= \lambda^2 A^2 \geq c^2 \\
\mu^1 a^1 + \mu^2 a^2 &= \lambda a^1 + \gamma a^1 + \lambda^2 a^2 \geq (d + z^1_0 - z^1_1) + (z - z_0) + (d + z^2_0 - z^2_1) \\
&= d + z - z_1 \geq d \\
\mu^1 b^2 + \mu^2 b^2 &= (\lambda^1 + \gamma) b^1 + \lambda^2 b^2 = z^1_0 + (z - z_0) + z^2_0 = z,
\end{align*}
\]

and by Proposition [1] the lemma follows. \( \blacksquare \)

In order to apply Lemma 3 to (1UC) we also need the following result.

**Lemma 4** The formulation \( (3.3.1.3) \) describes the convex hull of the feasible integer solutions.

**Proof.** In \( (3.3.1.3) \) there is only one binary variable: \( y^{ik} \). We can then apply the disjunctive programming principle of lift-and-project [11]: we multiply each row of the system \( (3.3.1.3) \) once
by \( y^{hk} \) and once by \( 1 - y^{hk} \), obtaining

\[
\begin{align*}
l(y^{hk})^2 & \leq p_h^0 y^{hk} \\
l(y^{hk})^2 & \leq p_t^h y^{hk} \leq \bar{P}(y^{hk})^2 \\
l(y^{hk})^2 & \leq p_t^h y^{hk} \leq p_0^0 (y^{hk})^2 \\
p_{t+1}^h y^{hk} & \leq p_t^h y^{hk} + (y^{hk})^2 \Delta^+ \\
p_t^h y^{hk} & \leq p_{t+1}^h y^{hk} + (y^{hk})^2 \Delta^- \\
0 & \leq (y^{hk})^2 \leq y^{hk} 
\end{align*}
\]  

(3.3.1.6)

By applying the standard reduction rules \((y^{hk})^2 = y^{hk}, y^{hk} p_{t+1}^h = p_t^h (\text{for all } t \in T(h,k))\), that are valid because \( y^{hk} \in \{0,1\} \) and \( p_t^h = 0 \) (for all \( t \in T(h,k) \)) when \( y^{hk} = 0 \), we obtain that (3.3.1.6) can be reduced to

\[
\begin{align*}
l y^{hk} & \leq p_h^0 y^{hk} \\
l y^{hk} & \leq p_t^h y^{hk} \leq \bar{P}(y^{hk}) \\
l y^{hk} & \leq p_t^h y^{hk} \leq p_0^0 y^{hk} \\
p_{t+1}^h & \leq p_t^h + y^{hk} \Delta^+ \\
p_t^h & \leq p_{t+1}^h + y^{hk} \Delta^- \\
0 & \leq y^{hk} \\
0 & \leq 1 - y^{hk} 
\end{align*}
\]  

(3.3.1.7)

Because (3.3.1.7) is clearly equivalent to (3.3.1.3), by lift-and-project the formulation (3.3.1.3) describes the convex hull of its integer feasible solutions.

It is now straightforward to prove the desired result.

**Theorem 5** Formulation (3.3.1.2)-(3.3.1.3) describes the convex hull of the feasible solutions for (1UC).

**Proof.** Define \( S_0 \) the set of feasible solutions of the network flow problem (3.3.1.2) associated with the DP graph \( G' \), and \( S^{hk} \) the set of feasible solutions of (3.3.1.3) for each pair \((h,k)\). We can build the set of solutions for the complete problem by iteratively composing the solutions of \( S_0 \) with the sets \( S^{hk} \), e.g., in lexicographic order of the pairs \((h,k) \in N \). By Lemma 3 at the first step, the system obtained by adding to the inequalities of the system (3.3.1.2) plus the inequalities of the system (3.3.1.3) associated with the first feasible pair \((h_1,k_1)\) describes the convex hull of the solutions \( S_1 = S_0 \oplus S^{h_1k_1} \), because the two systems share only the binary variable \( y^{h_1k_1} \). One can then iteratively define \( S_j \) as the set of feasible solutions obtained as \( S_j = S_{j-1} \oplus S^{h_jk_j} \); combining Lemma 3 with Lemma 4 at each step the corresponding system of inequalities describes \( \text{conv}(S_j) \). So, at the end of the composition process we have obtained a description with linear inequalities for the overall set of solutions.
3.3.2 The $p_t$ MILP Formulation

Finally the second formulation that was developed as part of the work of this thesis can be considered for the power variable $p_t$ containing only one index over the time horizon $T$. In this case the following formulation is introduced, with all the constraints of the formulations hold $\forall (h, k) \in N$:

\[
\begin{align*}
    c(y) + \sum_{t \in T} f(p_t) & \quad (3.3.2.1) \\
    \text{s.t.} & \quad Ey = b \quad y \geq 0 \quad (3.3.2.2) \\
    p_{t+1} - p_t + (p^0 - \Delta^+) \sum_{t=h}^{k-1} y_{h,k}^+ \leq p^0 & \quad \forall t \in T \quad (3.3.2.3) \\
    p_t - p_{t-1} + (p^0 - \Delta^-) \sum_{t=h}^{k-1} y_{h,k}^- \leq p^0 & \quad \forall t \in T \quad (3.3.2.4) \\
    -p \sum_{t=h}^{k} y_{h,k}^+ - p_t \leq 0 & \quad \forall t \in T \quad (3.3.2.5) \\
    p^0 \sum_{t=h}^{k} y_{h,k}^- - p^0 y_{t,k}^+ - p^0 y_{h,t}^- + p_t \leq 0 & \quad \forall t \in T \quad (3.3.2.6)
\end{align*}
\]

Where (3.3.2.1) denotes the objective function that is composed by the sum of the production costs throughout the optimization horizon $T$ together with the stable production costs and start up costs that are connected with variable $y^+$ and $y^-$ respectively. (3.3.2.2) is the network matrix and remains identical with the above formulation. (3.3.2.3) and (3.3.2.4) refer to the ramp up and down constraints respectively. (3.3.2.5) and (3.3.2.6) refers to the minimum and maximum power output constraints.

3.3.3 Computational Results

Let us now proceed with presenting the results for the computational comparisons for the different formulations. Note that in order to see how the formulations compete against the UC Formulations apart from the above two new MILP Formulations we tested also one more UC formulation with the standard $u,p$ variables together with 4 sets series of auxiliary variables $u^0_{0,0}, u^0_{0,1}, u^1_{1,0}, u^1_{1,1}$ denoting the commitment status of the variable in time steps $t - 1$ and $t$, which were used for the min up/down and the start up costs. Let us note that for all the experiments in this subsection the objective functions exhibited a linear nature of the form where the previously diagonal quadratic production cost function are transformed to $f'(p_t) = b \cdot p_t$ due to the fact that the formulation [3.3.1] is proven to be the convex hull with linear objective function. For our experiments the same machine used in [3.2.4] and for the case of the 1UC instances again we used the same data as in, while for the tests in the overall UC instances we used randomly-generated, realistic instances of the ramp-constrained, thermal units that were also used in [65]. As already mentioned all the instances were created and solved via CPLEX.
with the use of SMS++. None that in both tables rel gap 1 refers to the gap between the obtained solution and the root node solution where CPLEX cuts are applied whereas rel gap 2 denotes the gap between the obtained solution and the root node solution without CPLEX cuts, all the results are on average for the number of instances, with the exception of the number of problems solved to optimality and the time is measured in seconds. Finally let us mentioned that for the 1UC instances a time limit of one hour was set and for the UC instances the time limit was extented to 4 hours.

We proceed also to test the different formulations for the 1UC instances with 400 instance with \( t = 24 \) and 400 instances with \( t = 96 \).

The results of 3.2 show that for such small problems all formulations solve the problem in very small time and manage to solve all the instances to optimality. And on top of the \( p_t \) formulation of 3.3.3 seems to be in all the different cases the faster one and the only one capable of solving all instances to optimality. As far as the nodes are concerned this formulation seems again to be dominating in all the cases except of the one of 200 units where \( p_t^{h,k} \) is the winner. Finally as far as the gaps are concerned in both type of gaps for all the cases the the \( p_t^{h,k} \) does obtain smaller gaps.

In conclusion what be said with certainty is that both formulations presented in this section dominate the classic \( u,p \) formulation in all the different instances and all the different fields to be examined.
Chapter 4

The EDF Unit Commitment Problem

Following the above discussion on the UC Problem’s ingredients and its most well known solution methodologies, in this chapter we will focus and demonstrate a large scale, real-life case of UC. That is the real-world Hydro-Thermal Power Unit Commitment Problem that arises in EDF’s day-ahead scheduling process. First the main difficulties that arise in this problem will be shortly discussed and then three different MILP Formulations of the problem will be presented and thoroughly analyzed.

EDF, being the main electrical operator in France and one of the major operators worldwide provides a very challenging and hard to solve case of UC. The first thing that needs to be taken into consideration that adds great difficulty to the problem is the size of the problem itself. EDF manages a very large number of different production units, that can be summarized in nearly 60 nuclear plants, 100 classical thermal plants (coal, fuel, gas, combined cycles), and 500 hydraulic plants dispatched in 50 valleys. Above and beyond these difficulties, the EDF UC has specific load and safety requirements (the latter mainly represented as reserve constraints) that couple the generating units together. Another critical contributor to this difficulty is the fact that in the short-term version of the problem (i.e. the day-ahead scheduling process) one has to face strictly and highly demanding computational limits. More specifically due to the highly challenging operational process one is asked to provide an feasible global solution within 15 minutes.

One more very important factor that makes the problem difficult to solve, is the complicated physical and operational characteristics of the units, that makes the problem non-linear and non-convex nature. Starting with the thermal units some of the main elements of the thermal units that make them difficult to be formulated and solved are the treatment of production levels, that are found inside some minimum and maximum bounds, which can vary in time. For instance, the maximum level is equal to zero during refueling periods. On top of that the production level must be formulated in a discrete manner, since there are only specific values and intervals that are accepted for the unit production and the production variations are subjected to several rules. Moreover the units follow specific start-up and shut-down curves and there are specific daily limitations for the production variations, and the number of start-up and shut-down of the units. On top of the above the thermal production costs are adding an extra complication factor since they include time dependent start-up costs that need to be formulated with care. Finally besides the classic generation units (coal, oil, gas), there exist as well nuclear units that are more difficult to formulate and add extra constrains in the formulation.
The hydro-units as well have a highly complicated, non-linear and non-convex nature. In order to describe these difficulties we start with a short description of the structure of a hydro-plant. A hydro-plant is composed of two water reservoirs and a set of turbines, where the turbines are used to discharge water from the upstream reservoir to the downstream one in order to produce energy. One can have also the reverse process for energy production through the use of pumping units. The energy-production is being computed not for the individual hydro-plants, but it is optimized in a more global way for the whole hydro-valley, in which several individual hydro-plants and reservoirs are interconnected. In general the hydro-valleys energy production is subject to some technical aspects that follow some external regulations or mid-term optimization criteria. As far as the power production of the plants itself is concerned it takes only a finite number of values and its variations are subject to some fixed lower and upper values. Moreover it should be noted that all the different factors concerning the reservoir volumes are formulated via an appropriate water volume conservation function. Finally in the cases where two reservoirs are connected with both turbining and pumping plants. Some additional constraints are introduced in order to make sure that pumping and turbining won’t occur simultaneously, and that there will be a time threshold of at least thirty minutes between successive switching from one state to another.

The physical description of the EDF UC, which was described very shortly above (and will be further explained below), due to its complicated and highly complex nature offers a great variety of modeling approaches to our problem, which we are willing to examine to the greatest extent possible. Let us note that the physical description of the units exhibits such a highly complicated nature that exist some few cases, where the physical and/or operational reasons for some specific properties of the units that imply the addition of some type of constraints, go beyond our knowledge. More specifically a very interesting investigation point in the EDF UC is the treatment of the commitment variables for the thermal units. In most of the cases in the extended literature about MILP-models for the UC the binary commitment variables of the thermal units simply denote if the thermal unit is on or off during each time step of the optimization period, which leads to a binary variable. Yet again the complicated relationships that describe the primary power production (i.e. power produced to satisfy active demand) together with the reserved power production (i.e. power available to be used for different purposes) makes it interesting to consider a commitment variable that will have 3 indexes in order besides the unit and time instance also to take into consideration the different production states as well. On top of this the EDF thermal units require a very specific treatment on the shift of power production, which again makes it necessary to consider the addition of an extra index for the commitment variables that will describe the different states of change in power production. All the above give us a variety of different possible model formulations for the EDF UC based on the treatment of the binary commitment variables. Our first aim is to put these different model formulations under examination, in order to come up with the best possible formulation for our problem.

More specifically, we introduce the following three formulation:

1. A formulation where for the thermal unit a 3-index commitment variable is introduced. Taking into account the different states of production that can occur.

2. A formulation where for the thermal unit a 4-index commitment variable is introduced, taking into account the different states of modulation (technical term used in the Edf documents to refer to the power shifts in the thermal units and thus being further reproduced
in this chapter) that can occur. (i.e. decreasing, increasing, stationary)

3. A second formulation where for the thermal unit a 4-index commitment variable is introduced. In this case the 4th index is extended in order to include also the changes in active power that can occur due to start of some type of reserved power production (more details will be illustrated below).

In general these three models were chosen to be developed in an attempt to discover the tightest possible formulation, and also because through such formulation of the commitment variables through the different states in production states and shifts of active power, it can lead to a potentially compact DP-formulation for the thermal subproblems, that in turn may lead to smaller computational time for the thermal subproblems. For the presentation of the different models, since all of them share a common part (due to the fact that the whole hydro part remains the same) it was decided to proceed with presenting the three models together and for every different part of the formulation present all the different formulation together. Let us note that the formulations presented below have not yet been solved and tested whether via a general purpose solver or any other optimisation technique. Thus no results are yet to be presented for the Edf UC and we focus our work in this thesis in designing MILP formulations such as the ones of this chapter and possible solution approaches such as the one proposed in §.

4.1 Nomenclature

4.1.1 Sets

$T = \{1, \ldots, T\}$, set of time periods in the optimization horizon, where $t = 96$, each time step considered to be half-hour.

$I$, set of available thermal units.

$I_1, I_2$, A partition of the thermal units based on the treatment of secondary reserved power during modulation (see below in section 4.4.4 in the part of Reserved power production for detailed explanation)

$I_N \subseteq I$, set of available nuclear thermal units, which is a subset of the set $I$ of all the available thermal units.

$R$, set of available reservoirs.

$R^C, R^T, R^W$, sets of reservoirs with respectively no storage capacity (like channels), managed exclusively with target values, with non-zero water usage value. We assume that \{\$,$R^C, R^T, R^W$\} is a partition of $R$.

$H$, set of available hydro units.

$H_D, H_C \subset H$, set of available hydro units with discrete and continuous operations respectively.

$H_G, H_P \subset H$, set of hydro units that generate power and that pump water respectively.
H_r^+, H_r^-, set of available hydro units that release, receive water respectively in reservoir r ∈ \(R^T \cup R^W\).

\(J(h,t) = \{1,...,\bar{j}(h,t)\}\), set of operational points of each hydro unit h ∈ H for each time step t, where \(\bar{j}(h,t)\) refers to the last operational point of the production curve.

\(S(i) = \{sa, sb, sc, sd\}\), set of production states of each thermal unit i, where the first denotes the production of active power only (i.e. the power produced to satisfy the global demand), the second denotes the production of active power and primary reserved power (i.e. the power assigned to satisfy the global primary reserved demand), the third one denotes production of active power, primary reserved and secondary reserved power (i.e. the power assigned to satisfy the global secondary reserved demand) and the fourth one denotes the production of active power and maximum primary reserve (i.e. a special case of primary reserved power, where the overall possible power output for both primary and secondary reserve power is assigned altogether to satisfy the global primary reserved demand, this applies only for nuclear units). Moreover for simplicity reasons we introduce the subset \(S'(i) = \{sb, sc, sd\}\) used to describe the states of reserved power only.

\(N_2(i) = \{na, nb, nc\}\), set of states of changes in active power production for the second type of formulations: a unit i at time step t can perform a decrease in its active power (na), keep it constant (nb), or increase it (nc).

\(N'_2(i) = \{na, nc\}\), subset used for simplicity reasons in order to describe ongoing modulations only.

\(N_3(i) = \{na, nb, nc, nd, ne\}\), set of states of changes in production of active power output referring to the third type of formulation. Where a unit i at time step t can perform a decreasing or increasing modulation (nb and nd respectively), decrease or increase its active power due to start of some reserved power production (na and ne respectively) and finally keep its active power stable (nc).

\(N'_3(i) = \{nb, nd\}\), subset used for simplicity reasons in order to describe ongoing modulations only.

\(N(i) = N_2(i) \cup N_3(i)\), set that describes all possible different states over the two sets \(N_2(i)\) and \(N_3(i)\), and is used for notational convenience for presenting the different formulations constraints.

\(N'(i) = N'_2(i) \cup N'_3(i)\), subset used to define the modulation states for both different formulations (second and third one respectively).

\(S_N(i) = S(i) \times N(i)\), the set of all feasible states concerning set \(S(i)\) and set \(N(i)\). Note that for the set \(N_3(i)\) the states \{(sa, na), (sa, ne)\} are excluded as they are not feasible, because states na and ne rely on the existence of some sort of reserved power.

4.1.2 Constants

\(D_t, D_t^1, D_t^2 [\text{MW}]\), the amount of active, primary reserved and secondary reserved power respectively, that needs to be satisfied at each time step \(t \in T\).
4.1. NOMENCLATURE

c_i^1, c_i^2, c_i^3 [€, €, minutes], fixed costs used for the calculation of the start up cost of each unit
\( i \in I \). The first two refer to some fixed amount of cost given by the data and the latter
one is a time constant given by the data that needs to be taken into consideration for the
function that defines how the time the unit has been offline is affecting the start up costs.

c_{i,t}^p, c_{i,t}^f [€/MW-half-hour, €/half-hour], the proportional and fixed cost respectively for the
produced power of each thermal unit \( i \in I \) for every time step \( t \in T \).

c_{i,t}^{\text{d}} [€], the cost assigned to each online thermal unit \( i \) if it’s performing a decreasing modulation
at time step \( t \).

\( \tau_{i,\text{sb}}^b, \tau_{i,\text{sc}}^c, \tau_{i,\text{sd}}^d \) [time steps], the fixed number of time steps that a unit has to produce primary,
secondary and max primary reserved production respectively after starting producing (the
definition of the different types of reserve power is given in full detail in ).

c_{i,t}^{\text{d}} [€], the cost assigned to each online thermal unit \( i \) if it’s starting a deep decrease at time
step \( t \).

\( \omega_{r,t} [€/m^3] \), water value of reservoir \( r (\forall r \in R^W) \) at time step \( t \).

\( \bar{p}_i, \underline{p}_i [MW] \), the minimum and maximum allowed produced power for each online thermal unit
\( i \in I \)

\( \bar{p}_i^1, \bar{p}_i^2 [MW] \), the fixed amount of power for the primary and secondary reserved power respec-
tively for each thermal unit \( i \in I \).

\( \tau_i^-, \tau_i^+ \) [time steps], the minimum amount of time that each thermal unit \( i \in I \) has to remain
offline once it’s shut down and online once it’s turned on respectively.

\( V_i \) [time steps] the maximum number of starts for each thermal unit \( i \in I \).

\( \tau_h^-, \tau_h^+ \) [time steps], the time-delay of water to flow through hydro unit \( h \in H^+ \) to release
reservoir \( r_h^+ \) and respectively from source reservoir \( r_h^- \) through hydro unit \( h \in H^- \).

\( \Delta_i^-, \Delta_i^+ [MW/s] \), the maximum decrease and increase respectively in the produced amount of
power between two time periods for each thermal unit \( i \in I \).

\( \Psi_{r,0} [m^3] \), initial volume of each reservoir \( r \in R^T \cup R^W \).

\( \Psi_{r,t}, \Psi_{r,t}^r [m^3] \), bounds of the values of volume of each reservoir \( r \in R^T \cup R^W \) for every time
step \( t \).

\( \Psi_r^r, \Psi_r^r, \Psi_r^r \Psi_r^r [m^3] \), mid-horizon and final minimum and maximum volume of each reservoir
\( r \in R^T \cup R^W \) respectively.

\( E_{r,t} [m^3/s] \) external water inflow in each reservoir \( r \) for every time step \( t \).
$Y_{r,k} \text{ [m}^3\text{]}, a subdivision of the water levels of the reservoir } r \in R^W, \text{ where } k = \{0, ..., K_r\}. \text{ } \omega_{r,k},$

the water value at the end of time period $T$ of every compartment $[Y_{r-1,k}, Y_{r,k})$, note that the water values follow a decreasing order starting from bottom up (i.e. $\omega_{r,k-1} > \omega_{r,k} \geq 0$).

$\mathbb{N}_h \text{ [m}^3\text{/s]}, \text{ upper bound of values of spillage at each hydro unit } h \in H.$

$\Delta P_{h,j,t}, \Delta P_{h,j,t}^1, \Delta P_{h,j,t}^2 \text{ [MW], active power, primary reserved and secondary reserved respectively power generated by each operational point } j \text{ of each hydro unit } h \in H \text{ at each time step } t.$

$\Delta Q_{h,j,t} \text{ [m}^3\text{/s]}, \text{ value of water discharge by each operational point } j \text{ of each hydro unit } h \in H \text{ at every time step } t.$

$\Delta Q_h, \Delta Q_h \text{ [m}^3\text{/s], maximum water flow ramp down and ramp up respectively between two consecutive time periods at each hydro unit } h \text{ during a time step.}$

$\Delta T_s \text{ [s], duration of a time period/step measured in seconds.}$

$\Delta T_m \text{ [min], duration of a time period/step measured in minutes.}$

$A_i \text{ [integer nonnegative number], the maximum amount of starts of deep decrease for a nuclear unit } i \text{ during a day.}$

$\Delta \tilde{p}_i \text{ [MW/s], the gradient of deep decrease for nuclear unit } i \in I_N.$

$\tilde{p}_i \text{ [MW], The deep decrease power threshold of nuclear unit } i \in I_N.$

$B_i \text{ [time steps], the number of time steps each unit } i \in I \text{ needs to remain in stability mode after finishing a modulation.}$

$C_i \text{ [integer nonnegative number], the maximum amount of starts of modulation for each unit } i \in I \text{ during a day.}$

$M_i, \text{ the maximum no. of time steps that a modulation can last for thermal unit } i \text{ (} M_i = \max\{ \left\lceil \frac{\tilde{p}_i - p_i}{\Delta p_i} \right\rceil, \left\lceil \frac{\tilde{p}_i - p_i}{\Delta p_i} \right\rceil \}).$
4.1. NOMENCLATURE

\( \psi_{r,t} \quad [\text{m}^3], \) the amount of water volume in each reservoir \( r \in R^T \cup R^W \) at time step \( t \).

\( q_{h,t} \quad [\text{m}^3/\text{s}], \) the amount of water flow released through each hydro unit \( h \in H \) at time step \( t \).

\( \sigma_{h,t} \quad [\text{m}^3/\text{s}], \) the amount of water spillage released through each hydro unit \( h \in H \) at time step \( t \).

\( g_{r,k} \quad [\text{m}^3], \) for each reservoir \( r \in R^W \) the amount of water in compartment \([Y_{r,k-1}, Y_{r,k})\).

\( g_{r,k}^0, \quad g_{r,k}^t \quad [\text{m}^3], \) for each reservoir \( r \in R^W \) the amount of water in compartment \([Y_{r,k-1}, Y_{r,k})\) at the begin and the end of the optimization period respectively.

4.1.4 Discrete and binary variables

\( u_{s,i,t}^s, \) the commitment state of thermal unit \( i \in I \) for time step \( t \in T \) at production state \( s \in S \).

These variables are used only in the first type of formulation.

\( u_{s,n,i,t}^s, \) the commitment state of thermal unit \( i \in I \) for time step \( t \in T \) at production state \( s \in S \) and state of power change \( n \in N \). These variables are used only in the second and third type of formulation.

\( v_{i,t}, w_{i,t}, \) denote the start and the shut down respectively of thermal unit \( i \in I \) at time step \( t \).

\( v_{i,t}^s, w_{i,t}^s, \) denote the start and the finish respectively of production state \( s \) for thermal unit \( i \in I \) at time step \( t \).

\( v_{i,t}^n, w_{i,t}^n, \) denote the start and the finish respectively of state of power change \( n \) for thermal unit \( i \in I \) at time step \( t \).

\( a_{i,t}^-, a_{i,t}^+ \), denote if unit \( i \in I \) is performing an decreasing or an increasing modulation respectively in time period \( t \). These variables are used only for the first type of formulation.

\( v_{i,t}^-, v_{i,t}^+, \) denote the start of a decreasing or increasing modulation respectively during time step \( t \). These variables are eligible only for the first type of formulation.

\( w_{i,t}^-, w_{i,t}^+, \) denotes the finish of a decreasing or increasing modulation respectively during time step \( t \). These variables are used only in the first type of formulation.

\( m_{i,t}^-, m_{i,t}^+, \) denotes the shift down and up respectively of active power of unit \( i \) during time step \( t \). These variables are used only for the first and second type of formulation.

\( f_{i,t}^-, f_{i,t}^+, \) denotes the cases for which unit \( i \in I \) is obliged to perform an decreasing or an increasing modulation in its active power, due to the start of some kind of reserved power respectively in time period \( t \). These variables are used only for the first and second type of formulation.

\( d_{i,t}, \) denotes for each nuclear unit \( i \in I_N \), if unit is performing a deep decrease at time step \( t \), the state of deep decrease is further explained in 4.4.8.
\(d'_{i,t}\) denotes for each nuclear unit \(i \in I_N\), if unit is producing active power below or equal \(\tilde{p}\) at time step \(t\).

\(d''_{i,t}\) denotes for each nuclear unit \(i \in I_N\), if unit is dropping its power with a rate greater or equal \(\Delta \tilde{p}_i\) at time step \(t\).

\(x_{i,j,t}\) denotes if operational point \(j\) is used for discrete hydro units \(h \in H_D\), or are used as auxiliary variables for the continuous-hydro units \(h \in H_C\) during every time period \(t\).

\(z_{h,t}\) denotes the spillage of water for hydro unit \(h \in H\) at time step \(t\).

### 4.2 Objective Function

- The EDF UC is considered to be a minimization problem, where the several costs of the thermal units must be minimized, together with the target to keep the usage of water as low as possible are taken into account as well. The above are described in the following objective function, which is presented for all the three different formulations as follows:

\[
\min_{u,p,a} \sum_{i \in I} \sum_{t \in T} \left( c^0_{i,t} + c^p_{i,t} \cdot p_{i,t} + c^f_{i,t} \cdot \sum_{s \in S(i)} u^s_{i,t} + a_{i,t} \cdot c^a_{i,t} \right) + \sum_{i \in I_N} \sum_{t \in T} d'_{i,t} \cdot c^d_{i,t} - \sum_{r \in R_W} \sum_{k=1}^{K_r} \omega_{r,k} \cdot (g^0_{r,k} - g^f_{r,k}) \] (4.2.0.1a)

\[
\min_{u,p,d} \sum_{i \in I} \sum_{t \in T} \left( c^0_{i,t} + c^p_{i,t} \cdot p_{i,t} + c^f_{i,t} \cdot \sum_{(s,n) \in S_N(i)} u^{s,n}_{i,t} + \sum_{s \in S(i)} u^{s,na}_{i,t} \cdot c^a_{i,t} \right) + \sum_{i \in I_N} \sum_{t \in T} d'_{i,t} \cdot c^d_{i,t} - \sum_{r \in R_W} \sum_{k=1}^{K_r} \omega_{r,k} \cdot (g^0_{r,k} - g^f_{r,k}) \] (4.2.0.1b)

\[
\min_{u,p,d} \sum_{i \in I} \sum_{t \in T} \left( c^0_{i,t} + c^p_{i,t} \cdot p_{i,t} + c^f_{i,t} \cdot \sum_{(s,n) \in S_N(i)} u^{s,n}_{i,t} + \sum_{s \in S(i)} u^{s,nb}_{i,t} \cdot c^a_{i,t} \right) + \sum_{i \in I_N} \sum_{t \in T} d'_{i,t} \cdot c^d_{i,t} - \sum_{r \in R_W} \sum_{k=1}^{K_r} \omega_{r,k} \cdot (g^0_{r,k} - g^f_{r,k}) \] (4.2.0.1c)
4.3 Global Constraints

The three equations above refer to the objective functions for the three different formulations respectively. The second and third sum are identical to all three formulations and refer to the start up costs of the deep decrease and to the expected water values that are volume dependent respectively. In the first sum we have first the the start up costs, then the proportional production costs where again these two are identical for all three formulations. Thereafter we have the fixed production costs where in all different models the constant value is multiplied with all the different commitment variables that can arise in the different models that are expressed with the corresponding sum and finally we also have the modulation costs (for when the unit is in a decreasing modulation), where the constant value is multiplied with the variables that denote a unit during a decreasing modulation modulation.

4.3 Global Constraints

- In the EDF UC in every time step $t$ the thermal and hydro units need to satisfy an active demand $D_t$ and in addition to this a primary and a secondary spinning reserve demand $D^1_t$ and $D^2_t$. This leads us to the following three global constraints. Since the first one is the same for all the three different formulation it is written once, while for the other two constraints two different descriptions are introduced, with the latter one in both cases referring to both the second and third formulation respectively. Note that all the constraints below are formulated for $t \in T$:

\[
\sum_{i \in I} p_{i,t} + \sum_{h \in H} p_{h,t} = D_t \quad (4.3.0.1)
\]

\[
\sum_{i \in I} u_{i,t}^{sa} \cdot \tilde{p}_i^1 + \sum_{i \in I} u_{i,t}^{sd} \cdot (\tilde{p}_i^1 + \tilde{p}_i^2) + \sum_{h \in H} p_{h,t}^1 \geq D^1_t \quad (4.3.0.2a)
\]

\[
\sum_{i \in I} \sum_{n \in N(i)} u_{i,t}^{sh,n} \cdot \tilde{p}_i^1 + \sum_{i \in I} \sum_{n \in N(i)} u_{i,t}^{sd,n} \cdot (\tilde{p}_i^1 + \tilde{p}_i^2) + \sum_{h \in H} p_{h,t}^1 \geq D^1_t \quad (4.3.0.2b)
\]

\[
\sum_{i \in I} u_{i,t}^{sc} \cdot \tilde{p}_i^2 + \sum_{h \in H} p_{h,t}^2 \geq D^2_t \quad (4.3.0.3a)
\]

\[
\sum_{i \in I} \sum_{n \in N(i)} u_{i,t}^{sc,n} \cdot \tilde{p}_i^2 + \sum_{h \in H} p_{h,t}^2 \geq D^2_t \quad (4.3.0.3b)
\]

We have (4.3.0.1) denoting the satisfaction of global demand and this constraint is identical for all different formulations. Moving on we have (4.3.0.2a) and (4.3.0.2b) referring to the satisfaction of primary reserve demand, with (4.3.0.2a) referring to the first formulation, and (4.3.0.2b) corresponding to both the second and third formulations respectively. Where in both constraints we have for the thermal units two products where are multiplied the constants for the primary reserved power production with the corresponding commitment variables that refer
to the case of primary reserved production. Finally in a similar fashion (4.3.0.3a) and (4.3.0.3b) refer to the satisfaction of secondary reserve demand, where here again (4.3.0.3a) refers to the first formulation, and (4.3.0.3b) to both the second and third formulations respectively. And again in both constraints are used for the thermal units products where are multiplied the constants for the secondary reserved power production with the corresponding commitment variables that refer to the case of secondary reserved production.

4.4 Thermal Constraints

4.4.1 Production States

- The set of power production states $S(i) = \{sa, sb, sc, sd\}$ of the thermal units can be considered to have the following four states:

1. Production of active power only, denoted for the different formulations with the following variable:
   - $u_{sa,i,t}$
   - $\sum_{n \in N(i)} u_{sa,n}^{i,t}$

2. Production of active power and primary reserved power, denoted for the different formulations with the following variable:
   - $u_{sb,i,t}$
   - $\sum_{n \in N(i)} u_{sb,n}^{i,t}$

3. Production of active power, primary and secondary reserved power, denoted for the different formulations with the following variable:
   - $u_{sc,i,t}$
   - $\sum_{n \in N(i)} u_{sc,n}^{i,t}$

4. Production of active power, maximum primary reserved power -this case only occurs for nuclear units-, denoted for the different formulations with the following variable:
   - $u_{sd,i,t}$
   - $\sum_{n \in N(i)} u_{sd,n}^{i,t}$

- In order for the above variables to be consistent we introduce the following constraints that exist for $i \in I, t \in T$:

$$\sum_{s \in S(i)} u_{s,i,t}^{i} \leq 1 \quad (4.4.1.0.1a)$$

$$\sum_{(s,n) \in S_N(i)} u_{s,n}^{i,t} \leq 1 \quad (4.4.1.0.1b)$$
4.4. THERMAL CONSTRAINTS

\[
\sum_{s \in S(i)} u_{s,i,t}^{s} - \sum_{s \in S(i)} u_{s,i,t-1}^{s} = v_{i,t} - w_{i,t} (4.4.1.0.2a)
\]

\[
\sum_{(s,n) \in S_N(i)} u_{s,n,i,t}^{s,n} - \sum_{(s,n) \in S_N(i)} u_{s,n,i,t-1}^{s,n} = v_{i,t} - w_{i,t} (4.4.1.0.2b)
\]

\[
u_{s,i,t}^{s} - u_{s,i,t-1}^{s} = v_{i,t} - w_{i,t} (4.4.1.0.3a)
\]

\[
u_{s,n,i,t}^{s,n} - u_{s,n,i,t-1}^{s,n} = v_{i,t} - w_{i,t} (4.4.1.0.3b)
\]

\[
\sum_{s \in S(i)} u_{s,n,i,t}^{s,n} - u_{s,n,i,t-1}^{s,n} = v_{n,i,t} - w_{n,i,t} (4.4.1.0.4b)
\]

\[
\sum_{s \in S(i)} u_{s,na,i,t}^{s,na} + u_{s,ne,i,t}^{s,ne} \leq \sum_{s \in S(i)} v_{i,t}^{s} (4.4.1.0.5c)
\]

Constraints (4.4.1.0.1a)-(4.4.1.0.1b) are applied in all different modeling formulations ensuring that in each type step \(t\) only one of the above mentioned production states can occur. In addition to this (4.4.1.0.2a, 4.4.1.0.2b), which as well are applied in all different modeling formulations, make sure that variables \(v_{i,t}\) and \(w_{i,t}\) will take their appropriate values based on the commitment state of the thermal units. Moreover (4.4.1.0.3a)-(4.4.1.0.4b) being also applied in all different modeling formulations are ensuring that \(v_{i,t}\) and \(w_{i,t}\) will only take the appropriate integer values based on the status of the units, for all the different production states. Also (4.4.1.0.3a)-(4.4.1.0.3b) are valid for all \(s \in S(i)\). Moreover (4.4.1.0.4b) that applies to the second and third model only is valid for all \(n \in N(i)\) and is ensuring that variables \(v_{n,i,t}\), \(w_{n,i,t}\) and will only take their appropriate integer values, based in the equivalent modulation status of the units. Finally (4.4.1.0.5c) that is formulated only for the third model, is ensuring that in case in time step \(t\) unit has not started producing some type of reserved power then \(u_{s,na,i,t}^{s,na}\) and \(u_{s,ne,i,t}^{s,ne}\) will be forced to be set to zero, as they can only occur in case unit have started producing some type of reserved power at time step \(t\).

4.4.2 Interval of primary power values

- For each thermal unit \(i \in I\) the active power is divided in the following three power intervals:
  - **Low Interval**: \([p_i; p_i + p_i^1 + p_i^2]\)
  - **Intermediate Interval**: \((p_i + p_i^1 + p_i^2; p_i - p_i^1 - p_i^2)\)
  - **High Interval**: \([p_i - p_i^1 - p_i^2; p_i]\)
• Naturally, primary and secondary reserved power are produced in order to cover ancillary services that help in case of loss of units. That is primary spinning reserves are automatically “used” whenever the frequency deviates from the nominal value. Secondary is there to restore primary. In the case of Edf the primary and secondary reserved power are considered to take three different fixed values one for each different power interval. In order to simplify the model we are allowed to consider \( p_1 \) and \( p_2 \) to take only one fixed value for all the different power intervals. As a result \( p_1 \) and \( p_2 \) are set to the minimum from the three different possible values, in order to avoid any infeasibilities. This actually means that based on the data input we make the following conversion to simplify things, where for all thermal units \( i \in I \):

\[
-p_1^i := \min\{p_1^\text{low}, p_1^\text{inter}, p_1^\text{high}\}, \text{ where } p_1^\text{low}, p_1^\text{inter}, p_1^\text{high}\text{ are the three different input values for primary reserved power for each power interval.}
\]

\[
-p_2^i := \min\{p_2^\text{low}, p_2^\text{inter}, p_2^\text{high}\}, \text{ where } p_2^\text{low}, p_2^\text{inter}, p_2^\text{high}\text{ are the three different input values for secondary reserved power for each power interval.}
\]

• In general the output of active production of each thermal unit \( i \in I \) is determined by the following factors:

1. Its values must lie inside the interval \([p_i, \bar{p}_i]\).

2. All produced reserved power when subtracted from active power must be greater or equal than \( p_i \).

3. The sum of active power together with all the produced reserved power types should be smaller or equal \( p_i \).

• Consider the following numerical example, where \( p_i = 300 \text{MW}, \bar{p}_i = 900 \text{MW}, \bar{p}_1^i = 30 \text{MW}, \bar{p}_2^i = 20 \text{MW} \):

<table>
<thead>
<tr>
<th>Reserve status</th>
<th>( p_{i,t} )</th>
<th>( p_{1,t} )</th>
<th>( p_{2,t} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reserve power</td>
<td>[300; 900]</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Primary reserve production</td>
<td>[330; 870]</td>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>Primary &amp; secondary reserve production</td>
<td>[350; 850]</td>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>Max primary reserve production</td>
<td>[350; 850]</td>
<td>50</td>
<td>0</td>
</tr>
</tbody>
</table>

• The above restriction can be modeled with the following constraints that are valid for \( i \in I, t \in T \):

\[
p_{i,t} \geq p_i \cdot \sum_{s \in S(i)} u_{i,s,t} + \bar{p}_1^i \cdot \sum_{s \in S'(i)} u_{i,s,t} + \bar{p}_2^i \cdot (u_{sc,i,t} + u_{sd,i,t}) \tag{4.4.2.0.1a}
\]

\[
p_{i,t} \geq p_i \cdot \sum_{(s,n) \in S_N(i)} u_{i,s,n} + \bar{p}_1^i \cdot \sum_{s \in S'(i)} \sum_{n \in N(i)} u_{i,s,n} + \bar{p}_2^i \cdot \sum_{n \in N(i)} (u_{sc,n,i,t} + u_{sd,n,i,t}) \tag{4.4.2.0.1b}
\]
4.4. THERMAL CONSTRAINTS

\[
p_{i,t} \leq \bar{p}_i \cdot \sum_{s \in S(i)} u_{i,t}^s - \bar{p}_i^1 \cdot \sum_{s \in S'(i)} u_{i,t}^s - \bar{p}_i^2 \cdot (u_{i,t}^{sc} + u_{i,t}^{sd}) \quad (4.4.2.0a)
\]

\[
p_{i,t} \leq \bar{p}_i \cdot \sum_{(s,n) \in S_N(i)} u_{i,t}^{s,n} - \bar{p}_i \cdot \sum_{s \in S'(i)} \sum_{n \in N(i)} u_{i,t}^{s,n} - \bar{p}_i^2 \cdot \sum_{n \in N(i)} (u_{i,t}^{sc,n} + u_{i,t}^{sd,n}) \quad (4.4.2.0b)
\]

Where (4.4.2.0.1a)-(4.4.2.0.1b) & (4.4.2.0.2a)-(4.4.2.0.2b) are applied for all three versions of the model. More specifically combined they are ensuring that when unit \( i \) is turned off, then both active and reserved production are set to zero. Moreover if unit \( i \) is turned on and there is no reserved production then Where (4.4.2.0.1a)-(4.4.2.0.1b) & (4.4.2.0.2a)-(4.4.2.0.2b) are restricting active production to take values inside the interval \([\bar{p}_i, \bar{p}_i] \). Finally in the case that unit \( i \) is turned on and there is reserved production too, then (4.4.2.0.1a)-(4.4.2.0.1b) are ensuring that the subtraction of reserved power production from active power will remain greater or equal \( p_{i} \) and (4.4.2.0.2a)-(4.4.2.0.2b) are ensuring that the sum of primary power together with all the produced reserved power types will remain smaller or equal \( p_{i} \).

### 4.4.3 Power changes of primary power values

- There are two cases in which the active power output of a thermal unit is allowed to change its value:
  
  - For the unit to be in **modulation state**, where the unit increase/decrease rate of active power is set to the maximum possible level, that is the change of the power output should be equal to the equivalent maximum gradient of increase/decrease for all the time steps of the modulation except from the last one. The above restrictions are shown in the following example where unit \( i \) wants to increase its production from 790 MW to 860 MW having \( \Delta_i^{+} = 30 \text{ MW/time step} \):

    | State \( p_{i,t} \) | Modulation \( k \) | Modulation \( k + 1 \) | Modulation \( k + 2 \) | Modulation \( k + 3 \) |
    |-------------------|-------------------|-------------------|-------------------|-------------------|
    | \( \Delta p_i = p_k - p_{k-1} \) | 790              | 820              | 850              | 860              |
    | \( \Delta p_i = p_k - p_{k-1} \) | -                | 30 (= \( \Delta_i^{+} \)) | 30 (= \( \Delta_i^{+} \)) | 10                |

  - The unit starts producing some type of reserved power and in order not to violate the power restrictions mentioned in (4.1) -that is: \( p_{i,t} - p_{i,t}^{1} - p_{i,t}^{2} \geq \bar{p}_i \) and \( p_{i,t} - p_{i,t}^{1} - p_{i,t}^{2} \leq \bar{p}_i \)- it is forced to change its active power production. Consider the following example where again as in (4.1) \( p_i = 300 \text{ MW}, \bar{p}_i = 900 \text{ MW}, \bar{p}_i^1 = 30 \text{ MW}, \bar{p}_i^2 = 20 \text{ MW} \):

    | \( t \) | \( k \) | \( k + 1 \) | \( k + 2 \) | \( k + 3 \) |
    |--------|--------|--------|--------|--------|
    | \( p_{i,t} \) | 310    | 330    | 350    | 350    |
    | \( \Delta p_i = p_k - p_{k-1} \) | -      | \( \leq 20 \) | \( \leq 20 \) | 0      |
    | \( \Delta p_i = p_k - p_{k-1} \) | -      | 30     | 30     | 0      |
    | \( \Delta p_i = p_k - p_{k-1} \) | -      | 0      | 30     | 0      |
In time steps \( k + 1 \) and \( k + 2 \) the primary power is shifted in order for the above mentioned constraint not to be violated. In general there are 4 different cases, in which the active power should change its value because of reserved power production that are presented below together with their modeling proposal:

1. When active power output of unit \( i \) in time step \( t - 1 \) lies inside the interval of \([p_i, p_i + p_1] \) and unit is producing primary reserved power in time step \( t \).

2. When active power output of unit \( i \) in time step \( t - 1 \) lies inside the interval of \([p_i, p_i + p_1 + p_2] \) and unit is producing primary and secondary reserved or maximum primary power in time step \( t \).

3. When active power output of unit \( i \) in time step \( t - 1 \) lies inside the interval of \((p_i - p_1, p_i] \) and unit is producing primary reserved power in time step \( t \).

4. When active power output of unit \( i \) in time step \( t - 1 \) lies inside the interval of \((p_i - p_1 - p_2, p_i] \), and unit is producing primary and secondary reserved or maximum primary power in time step \( t \).

In order to ensure that \( m_{i,t}^+ \) and \( m_{i,t}^- \) will be zero in any other case the following set of constraints that are valid for \( i \in I, t \in T \) together with variables \( f_{i,t}^+ \) and \( f_{i,t}^- \) are introduced:

\[
\begin{align*}
 f_{i,t}^- & \leq \sum_{s \in S'(i)} v_{i,t}^s \quad \text{(4.4.3.0.1a)} \\
v_{i,t}^{sh,na} & \leq \frac{p_{i,t-1}}{(p_i - p_1) \cdot (1 + \delta)} \quad \text{(4.4.3.0.1b)} \\
f_{i,t}^- & \leq \frac{p_{i,t-1}}{(p_i - p_1)^2} \cdot (1 + \delta) + v_{i,t}^{sc} + v_{i,t}^{sd} \quad \text{(4.4.3.0.2a)} \\
u_{i,t}^{sc,na} + u_{i,t}^{sd,na} & \leq \frac{p_{i,t-1}}{(p_i - p_1 - p_2) \cdot (1 + \delta)} \quad \text{(4.4.3.0.2b)} \\
f_{i,t}^- & \leq \frac{p_{i,t-1}}{(p_i - p_1 - p_2) \cdot (1 + \delta)} \quad \text{(4.4.3.0.3a)}
\end{align*}
\]

As it is obvious from the above constraints the formulation treatment for the first two models is identical, while in the third case things vary. More specifically for the two first models we have the above constraints ensuring that in case that the unit \( i \) should not drop its active power due to the start of some type of reserved power (i.e. case 3 or 4 from the above described cases) at time step \( t \), then \( f_{i,t}^- \) is forced to be zero. More specifically (4.4.3.0.1a) ensure that if at time step \( t \) unit is not starting producing some type of reserved production, then \( f_{i,t}^- \) is forced to be zero. In addition to this in (4.4.3.0.2a) the fraction of the rhs in case \( p_{i,t-1} \in (p_i - p_1, p_i] \) will take a value greater or equal 1, while in case that \( p_{i,t-1} \notin (p_i - p_1, p_i] \), then the fraction will take a value between \([0, 1)\). Finally \( v_{i,t}^{sc} \) and \( v_{i,t}^{sd} \) are added to the rhs to make the constraint inactive, because in case unit starts producing secondary or max. primary
reserved power we are interested to examine if \( p_{i,t} \in [p_i - p_i^1, p_i] \) and as a result (4.4.3.0.2a) should stay inactive, while (4.4.3.0.3a) will impose \( f_{i,t}^+ \) to be zero in case \( p_{i,t} \not\in [p_i - p_i^1, p_i] \) in the same fashion as in (4.4.3.0.2a). Let us note that in both (4.4.3.0.2a) and (4.4.3.0.3a) the denominator with \((1 + \delta)\), in order to exclude from the case where \( p_{i,t-1} \) is equal to the lower bound of the examined intervals.

In a similar fashion for the third model the above constraints are ensuring that in case that the unit \( i \) should not drop its active power due to the start of some type of reserved power (i.e. case 3 or 4 from the above described cases) at time step \( t \), then \( u_{i,t}^{s,na} \) are forced to be zero. More specifically in (4.4.3.0.1b) the fraction of the rhs in case \( p_{i,t-1} \not\in [p_i - p_i^1, p_i] \) will take a value greater or equal 1, while in case that \( p_{i,t-1} \not\in [p_i - p_i^1, p_i] \), then the fraction will take a value between \((0,1)\) and as a result \( u_{i,t}^{s,na} \) will be forced to be zero. Finally (4.4.3.0.2b) will impose \( u_{i,t}^{sc,na} \) and \( u_{i,t}^{sd,na} \) to be zero in case \( p_{i,t} \not\in [p_i - p_i^1, p_i^2] \) in the same fashion as in (4.4.3.0.1b). Let us note that in both (4.4.3.0.1b) and (4.4.3.0.2b) the denominator with \((1 + \delta)\), in order to exclude from the case where \( p_{i,t-1} \) is equal to the lower bound of the examined intervals.

\[
\begin{align*}
f_{i,t}^+ &\leq \sum_{s \in S'(i)} v_{i,t}^s \tag{4.4.3.0.4a} \\
u_{i,t}^{sh,ne} &\leq 1 - \frac{(p_{i,t-1} + \delta) - (p_i + p_i^1)}{p_i} \tag{4.4.3.0.4b} \\
f_{i,t}^+ &\leq 1 - \frac{(p_{i,t-1} + \delta) - (p_i + p_i^1)}{p_i} + v_{i,t}^{sc} + u_{i,t}^{sd} \tag{4.4.3.0.5a} \\
u_{i,t}^{sc,ne} + v_{i,t}^{sd,ne} &\leq 1 - \frac{(p_{i,t-1} + \delta) - (p_i + p_i^1 + p_i^2)}{p_i} \tag{4.4.3.0.5b} \\
f_{i,t}^+ &\leq 1 - \frac{(p_{i,t-1} + \delta) - (p_i + p_i^1 + p_i^2)}{p_i} \tag{4.4.3.0.6a}
\end{align*}
\]

Here as well the formulation treatment for the first two models is identical, while in the third case things vary. More specifically for the two first models we have the above constraints are ensuring that in case the unit \( i \) should not drop its active power due to the start of some type of reserved power (i.e. case 3 or 4 from the above described cases) at time step \( t \), then \( f_{i,t}^+ \) is forced to be zero. More specifically (4.4.3.0.4a) ensures that if at time step \( t \) unit is not starting producing some type of reserved production, then \( f_{i,t}^+ \) is forced to be zero. In addition to this in (4.4.3.0.5a) the fraction of the rhs in case \( p_{i,t-1} \in [p_i, p_i^1] \) will take a value between \((-1,0)\) and as a result when this value will be subtracted from 1 we end up with a rhs greater or equal to 1, in case that \( p_{i,t-1} \not\in [p_i - p_i^1, p_i] \), then the fraction will take a value between \((0,1)\) and as a result when this value will be subtracted from 1 we end up with a rhs smaller to 1. Finally \( v_{i,t}^{sh} \) and \( v_{i,t}^{sd} \) are added to the rhs to make the constraint inactive, because in case unit starts producing secondaty or max. primary reserved power we are interested to examine if \( p_{i,t} \in [p_i^1, p_i^1 + p_i^2] \) and as a result
CHAPTER 4. THE EDF UNIT COMMITMENT PROBLEM

(4.4.3.0.4a) should stay inactive. Let us note that in both (4.4.3.0.5a) and (4.4.3.0.6a) in the numerator with we add \( \delta \) to the value of \( p_{i,t-1} \), in order to exclude from the case where \( p_{i,t-1} \) is equal to the upper bound of the examined intervals.

Likewise for the third model the above constraints are ensuring that in case that the unit \( i \) should not drop its active power due to the start of some type of reserved power (i.e. case 3 or 4 from the above described cases) at time step \( t \), then \( u_{i,t}^{na} \) are forced to be zero. More specifically in (4.4.3.0.4b) the fraction of the rhs in case \( p_{i,t-1} \in [\underline{p}_i - \underline{p}_i, \overline{p}_i] \) will take a value greater or equal 1, while in case that \( p_{i,t-1} \notin (\underline{p}_i - \underline{p}_i, \overline{p}_i] \), then the fraction will take a value between \([0, 1]\) and as a result \( u_{i,t}^{na} \) will be forced to be zero. Finally (4.4.3.0.5b) will impose \( u_{i,t}^{sc,na} \) and \( u_{i,t}^{sd,na} \) to be zero in case \( p_{i,t} \notin (\underline{p}_i - \underline{p}_i - \Delta_{i,k}, \overline{p}_i] \) in the same fashion as in (4.4.3.0.4b). Let us note that in both (4.4.3.0.4b) and (4.4.3.0.5b) the denominator with \((1 + \delta)\) in order to exclude from the case where \( p_{i,t-1} \) is equal to the lower bound of the examined intervals.

To complete the modeling requirements for the first two formulations constraints (4.4.3.0.7a) - (4.4.3.0.8b) are introduced, imposing \( m_{i,t}^- \) and \( m_{i,t}^+ \) to be able to set its value to one only if there is a need of active power change due to start of reserved production or there is an ongoing modulation, while the first time step of the modulation is excluded. That is because the first change in active power output during a modulation, will only occur from the second time step onwards (as we define \( \Delta p_i = p_k - p_{k-1} \)).

- Moving on we present the ramp constraints for the three different formulations, where
for notational convenience we denote $\Delta^+_i = \Delta T_m \cdot \Delta p^+_i$ and $\Delta^-_i = \Delta T_m \cdot \Delta p^-_i$:

$$\Delta^+_i \cdot (a^+_i - v^+_i - w^+_i) - \Delta^-_i \cdot (m^-_i - v^-_i) - w_i \cdot p_i \leq p_{i,t} - p_{i,t-1} \leq \Delta^-_i \cdot (m^+_i - v^+_i) - \Delta^+_i \cdot (a^-_i - v^-_i - w^-_i) + v_i \cdot p_i$$

\[(4.4.3.0.9a)\]

$$-\Delta^-_i \cdot (m^-_i - v'^{na}_i) + \Delta^+_i \cdot (\sum_{s \in S(i)} u^{'na}_{i,t} - v^{nc}_{i,t} - w^{nc}_{i,t+1}) - w_i \cdot p_i \leq p_{i,t} - p_{i,t-1} \leq \Delta^-_i \cdot (m^+_i - v'^{na}_i) - \Delta^+_i \cdot (\sum_{s \in S(i)} u^{'na}_{i,t} - v^{nc}_{i,t} - w^{nc}_{i,t+1}) + v_i \cdot p_i$$

\[(4.4.3.0.9b)\]

$$-\Delta^-_i \cdot (\sum_{s \in S(i)} u^{'nd}_{i,t} + u^{'nb}_{i,t} + v^{'nb}_{i,t}) + \Delta^+_i \cdot (\sum_{s \in S(i)} u^{'nd}_{i,t} - v^{'nd}_{i,t} - w^{'nd}_{i,t+1}) - w_i \cdot p_i \leq p_{i,t} - p_{i,t-1} \leq \Delta^-_i \cdot (\sum_{s \in S(i)} u^{'nd}_{i,t} + u^{'nb}_{i,t} - v^{'nb}_{i,t}) - \Delta^+_i \cdot (\sum_{s \in S(i)} u^{'nd}_{i,t} - v^{'nb}_{i,t} - w^{'nd}_{i,t+1}) + v_i \cdot p_i$$

\[(4.4.3.0.9c)\]

Where \[(4.4.3.0.9a)-(4.4.3.0.9c)\] are introduced so that for all the different formulations the units are allowed to change their active power production only if there is an ongoing modulation or primary production is forced to change its value in order not to violate constraints \[(4.4.2.0.1a) - (4.4.2.0.2b)\] due to start of reserved power. More specifically:

* The first products of both lhs and rhs of all constraints are making sure that in case $m^+_i$ or the corresponding $u^{'na}_{i,t}$ or $u^{'nb}_{i,t}$ are equal to 1 then unit will be allowed to decrease its active power at any level smaller or equal the maximum the maximum decrease $\Delta p^-_i$. In the same fashion in case $m^-_i$ is equal to 1 or the corresponding $u^{'nd}_{i,t}$ or $u^{'ne}_{i,t}$, then the unit will be allowed to increase its active power at any level smaller or equal the maximum increase $\Delta p^+_i$.

* The second products of both lhs and rhs of all three different constraints ensure that in case there is an ongoing decreasing modulation, then for all the time steps except the first and the last one $p_{i,t} - p_{i,t-1} = -\Delta T_m \cdot \Delta p^-_i$ and the same applies for an ongoing increasing modulation, where again for all the time steps except the first and the last one $p_{i,t} - p_{i,t-1} = -\Delta T_m \cdot \Delta p^+_i$.

* Finally the last products of both lhs and rhs of all different constraints are added in order to allow any thermal unit that is turned on or off in time step $t$ to change equivalently its active power production, where in all the other cases both the lhs and the rhs are set to zero and the unit is not allowed to perform any change in its power output.
The thermal units are only allowed to perform a maximum number of modulation per day (i.e. half of the optimization period), which can be modeled for the different formulations in the following way for \( i \in I \) and \( \tau = \{0, \lfloor \frac{T}{2} \rfloor \} \):

\[
\sum_{t=\tau}^{\tau+\lfloor \frac{T}{2} \rfloor} u_{i,t}^+ + v_{i,t}^- \leq C_i \tag{4.4.3.0.10a}
\]

\[
\sum_{t=\tau}^{\tau+\lfloor \frac{T}{2} \rfloor} \sum_{n \in N'(i)} v_{i,t}^0 \leq C_i \tag{4.4.3.0.10b}
\]

We use \( \frac{T}{2} \) in order to express the time step referring to the middle of the optimization horizon, in order for the modeling to be compatible with any time-unit we choose for the time steps (hour, half-hour, quarter, minute, sec, etc).

The thermal units after every modulation are restricted not to perform another one for a certain number of time steps. This is referred as Stability state and can be modeled for the different formulations as follows that are valid for \( i \in I, t \in [B', \bar{t}] \):

\[
\sum_{k=t-B'+1}^{t} w_{i,t}^+ + w_{i,t}^- \leq 1 - a_{i,t}^+ - a_{i,t}^- \tag{4.4.3.0.11a}
\]

\[
\sum_{k=t-B'+1}^{t} \sum_{n \in N'(i)} w_{i,t}^n \leq 1 - \sum_{s \in S(i)} \sum_{n \in N'(i)} v_{i,t}^{s,n} \tag{4.4.3.0.11b}
\]

Since due to the EDF restrictions each time a unit finishes a modulation it must enter stability mode for a certain number of time steps, this gives us the chance to use \( w \) variables for all different models (i.e. \( w_{i,t}^+ \& w_{i,t}^- \) for the first and \( w_{i,t}^{nc} \& w_{i,t}^{nc} \) for the second and third formulation) as both the variable denoting the finish of a modulation and in the same time variable denoting the first time step of stability mode. As a result of this we model the stability constraint using the above mentioned \( w \) variables for the beginning of stability mode in all three different models.

In order for the modulation to be modeled properly in the first formulation the following logical constraints need to be introduced as well for \( i \in I, t \in T \):

\[
a_{i,t}^+ + a_{i,t}^- \leq \sum_{s \in S(i)} u_{i,t}^s \tag{4.4.3.0.12}
\]

\[
a_{i,t}^+ - a_{i,t-1} = v_{i,t}^+ - w_{i,t}^+ \tag{4.4.3.0.13}
\]

\[
a_{i,t}^- - a_{i,t-1} = v_{i,t}^- - w_{i,t}^- \tag{4.4.3.0.14}
\]

Equation \([4.4.3.0.12]\) ensures that a thermal unit can perform a modulation only if it is already online and producing active power. Moreover \([4.4.3.0.13]\) and \([4.4.3.0.14]\) ensure that variables \( v_{i,t}^+, w_{i,t}^+ \) and \( v_{i,t}^-, w_{i,t}^- \) will only take their appropriate integer values, based on the equivalent modulation status of the units.
4.4.4 Reserved power production

- Every time a thermal unit starts producing any type of reserved power, the unit is imposed to continue producing the same type of reserved power for a minimum number of time steps. In order to model this condition the following modeling approach is followed for all the different models that are valid for \( i \in I, s \in S'(i), h \in \{0, \tau^i_s - 1\}, t = \{\tau^i_s - 1, \cdots, \bar{t}\} \):

\[
v^s_{i,t-h} \leq u^s_{i,t} + \sum_{k=t-h}^{t} (v^-_{i,k} + v^+_{i,k}) \tag{4.4.4.0.1a}
\]

\[
v^s_{i,t-h} \leq \sum_{n \in N(i)} u^{s,n}_{i,t} + \sum_{k=t-h}^{t} \sum_{n \in N'(i)} v^n_{i,k} \tag{4.4.4.0.1b}
\]

Constraints (4.4.4.0.1a)-(4.4.4.0.1b) ensure that a unit that has just started producing reserved power to keep doing so for the next \( \tau^s \) steps, unless a modulation occurs when the constraint becomes inactive. In case in the current time step \( t \) there has been a start of production of the equivalent (i.e. \( v^s_{i,t-h} = 1 \)) reserved production inside the time interval of \( \tau^s \), then the lhs becomes equal to 1 forcing \( u^s_{i,t} \) to become equal to 1 as well, this enforcement is not valid in case between the time step \( t \) a modulation has started, in which case the sum of the rhs will become 1.

- In addition to the above, reserved power follows some specific rules when the thermal units are performing a modulation:

  - Primary reserve power output during modulation should be equal to the minimum of its value when starting the modulation and the value at its end.

  - Secondary reserve power output can take one of the following two modes specified by the data input:

    1. Equal to the minimum of its value when starting the modulation and the value at its end. This applies to the subset \( I_1 \) of thermal units.

    2. Equal to zero to all the time steps of the modulation except the starting one. This applies to the subset \( I_2 \) of thermal units.

- The above mentioned restrictions would imply that actually in case of modulation the all the time steps except of the first and the last one are forced to have the same reserved production state, which can lead to one of the following four cases:

- In order to model the above restrictions properly the following constraints are introduced,
CHAPTER 4. THE EDF UNIT COMMITMENT PROBLEM

<table>
<thead>
<tr>
<th>Case</th>
<th>Reserved Status</th>
<th>begin of modulation</th>
<th>Ongoing modulation</th>
<th>end of modulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>No reserved power at any time step</td>
<td>$u_{i,t}^s$</td>
<td>0</td>
<td>0, ..., 0</td>
<td>0</td>
</tr>
<tr>
<td>Reserved power only at begin</td>
<td>$u_{i,t}^s$</td>
<td>1</td>
<td>0, ..., 0</td>
<td>0</td>
</tr>
<tr>
<td>Reserved power only at end</td>
<td>$u_{i,t}^s$</td>
<td>0</td>
<td>0, ..., 0</td>
<td>1</td>
</tr>
<tr>
<td>Reserved power at all time steps</td>
<td>$u_{i,t}^s$</td>
<td>1</td>
<td>1, ..., 1</td>
<td>1</td>
</tr>
</tbody>
</table>

for each different state of reserved power, that are valid for $s \in S'(i), i \in I, t \in T$:

\[
-1 + a_{i,t}^- + a_{i,t}^+ - v_{i,t}^- - v_{i,t}^+ - v_{i,t,1}^- - v_{i,t,1}^+ - w_{i,t,1}^- - w_{i,t,1}^+ \\
\leq u_{i,t}^s - u_{i,t-1}^s \
1 - a_{i,t}^- + a_{i,t}^+ + v_{i,t}^- + v_{i,t}^+ + v_{i,t,1}^- + v_{i,t,1}^+ + w_{i,t,1}^- + w_{i,t,1}^+
\]

Constraint (4.4.4.0.2a) - (4.4.4.0.2b)

Constraints (4.4.4.0.2a)- (4.4.4.0.2b) ensure that for all different formulations that for all the time steps of the modulation except of the first and the last one, that the primary reserved production will remain unchanged. For all the different cases the constraints follow the same logic, where in the middle we have the subtraction between each two different consecutive time steps for the integer variables that denote the production of primary energy. And by adding in both left and right hand side the integer variables that denote an increasing and decreasing modulation and the switch on and off of the units. And when the unit is modulating then both left and right hand side of the constraints become equal to zero, except of the case that unit is being switched on or off, in which case lhs becomes equal to $-1$ and rhs equal to $1$.

\[
u_{i,t}^s \leq u_{i,t+2}^s - w_{i,t+2}^s \\
\sum_{n \in N'(i)} u_{i,t}^n \leq \sum_{n \in N'(i)} u_{i,t}^n + 1 - \sum_{n \in N'(i)} w_{i,t+2}^n \\
u_{i,t}^s \leq u_{i,t-1}^s + 1 - v_{i,t-1}^- - v_{i,t-1}^+ \\
\sum_{n \in N'(i)} u_{i,t-1}^n \leq \sum_{n \in N'(i)} u_{i,t-1}^n + 1 - \sum_{n \in N'(i)} v_{i,t-1}^n
\]
Constraints (4.4.4.0.3a) - (4.4.4.0.4b) ensure for all the different formulations that throughout the modulation the status of reserved power will only take its appropriate values. More specifically as mentioned above due to (4.4.4.0.2a) - (4.4.4.0.2b) the status of reserved power throughout the modulation is not allowed to change except the first and the last time step in all different formulations. As a result in case that the status of reserved power is equal to 1 for these time steps then (4.4.4.0.3a) - (4.4.4.0.3b) impose the status of reserved power at the last time step of the modulation to be equal to 1 in all different models. In a similar fashion (4.4.4.0.4a) - (4.4.4.0.4b) impose the status of reserved power at the first time step of the modulation to be equal to 0 in all formulations. On top of these (4.4.4.0.3a) - (4.4.4.0.4b) ensure that in case the reserved power status of the first or last time steps is equal to zero then all the other time steps of the modulation are forced to take this value too.

\[
  u_{s,i,t} \leq (1 - v_{i,h}^- - v_{i,h}^+ + \sum_{k=h+B_i}^{t-1} v_{i,k}^- + v_{i,k}^+) + (1 - u_{s,i,h}) + u_{s,i,h+1} + (1 - w_{i,t+1}^- - w_{i,t+1}^+) \quad (4.4.4.0.5a)
\]

\[
  \sum_{n \in N'(i)} u_{s,i,t}^{n+} \leq 1 - \sum_{n \in N'(i)} v_{i,h}^+ + \sum_{k=h+B_i}^{t-1} \sum_{n \in N'(i)} v_{i,k}^+ + (1 - u_{s,i,h}) + \sum_{n \in N'(i)} u_{s,i,h+1} + (1 - \sum_{n \in N'(i)} w_{i,t+1}^+) \quad (4.4.4.0.5b)
\]

Finally (4.4.4.0.5a) - (4.4.4.0.5b) that are both valid for \( h = \{t - M_i, \ldots, t - 1\} \) ensure for all different models that in case both the first and the last time-steps of the modulation are producing reserved power then all the intermediate time steps will be forced to do so as well, which is achieved in the following way. In both constraints the first parenthesis will be equal to zero only for the time-step \( h \) that refers to the first time step of the current modulation. Here we need to take in consideration the case that \( M_i \) (where \( M_i \) refers to the maximum duration of a modulation, and thus we examine all the previous time steps until \( t - M_i \) larger or equal than \( B_i \), which means that in the examined time-interval we may have more than one modulation. In order to overcome the issue in both constraints are added the sums of \( k = \{h + B_i, \ldots, t - 1\} \), and thus in case there is a new start of modulation in this time interval the value of this sum in both constraints will be equal to 1, making the overall value of the parenthesis equal to 1 in case in time-step \( h \). Thus we are sure that the first parenthesis will be equal to zero only in case time-step \( h \) refers to the start of a new modulation and within the possible time-steps of the modulations no other modulation can take place. Now in case we are in such a scenario then in case time step \( t \) refers to the last time step of the modulation that starts in time step \( h \) then in both constraints the last parenthesis will be equal to zero (since the corresponding variable \( w_{i,t+1} \) will be equal to 1). If in this scenario in the first time step of the modulation (which is time step \( h \)) reserved power is produced, then the second parenthesis in both constraints will be also set to zero. Finally in case that also in the last time step of the modulation (which is time step \( t \)) reserved power is produced then the lhs will be set to 1 and as a result the second time step of the modulation (i.e. \( u_{s,i,h+1} \)) will be forced to be set to 1.
too and then via \((4.4.4.0.2a)\) \((4.4.4.0.2b)\) for all the other intermediate time-steps of the modulation \(u^s\) will be forced to be equal to 1.

- Now for the case of secondary reserved power of subset \(I_2\) we have the following constraints that are valid for \(i \in I_2, t \in T:\)

\[
\begin{align*}
    u^s_{i,t} & \leq 1 - a_{i,t}^- - a_{i,t}^+ + v^+_{i,t} + v^-_{i,t} \\
    \sum_{n \in N(i)} u^s_{i,t,n} & \leq \sum_{n \in N(i)} v^-_{i,t,n}
\end{align*}
\]

\((4.4.4.0.6a)\) \((4.4.4.0.6b)\)

Where in all different formulations the lhs denoting the commitment variable for the secondary reserved production is forced to be set to zero every time we have an ongoing modulation except from the first time step of the modulation.

### 4.4.5 Start Up Costs

- Every time a thermal unit is turned on, there is a specific start up cost assigned to the unit, which is calculated as a function of the time duration the unit has previously stayed offline. The following constraints are used to model the start up cost, which are valid for \(i \in I, t \in T\) and for \(j \in \{1, \cdots, t-1\} :\)

\[
\begin{align*}
    c^0_{it} & \geq \left[ c_{i,1} + c_{i,2} \cdot \ln \left( \frac{(t-j) \cdot \Delta T_m}{c_{i,3}} \right) \right] \cdot \left[ \sum_{s \in S(i)} u^s_{i,t} - \sum_{\tau = j}^{t-1} \sum_{s \in S(i)} u^s_{i,\tau} \right] \\
    c^0_{it} & \geq \left[ c_{i,1} + c_{i,2} \cdot \ln \left( \frac{(t-j) \cdot \Delta T_m}{c_{i,3}} \right) \right] \cdot \left[ \sum_{s \in S(i)} \sum_{n \in N(i)} u^s_{i,t,n} - \sum_{\tau = j}^{t-1} \sum_{s \in S(i)} \sum_{n \in N(i)} u^s_{i,\tau,n} \right]
\end{align*}
\]

\((4.4.5.0.1a)\) \((4.4.5.0.1b)\)

\(c^0_{it} \geq 0\)

\((4.4.5.0.2)\)

Where for all different formulations \((4.4.5.0.1a)\) \((4.4.5.0.1b)\) we have the first factor of the rhs is calculating the value of the start-up cost function for all the different possible values of the offline duration based on the current time step \(t\). Within the first factor of the multiplication in both cases the fraction within the natural logarithm is the part that adds the time dependency in the start up costs, more specifically since both \(c_{i,3}\) and \(\Delta T_m\) are constants and \((t-j)\) are in the numerator this means that the bigger their difference (i.e. the more time steps unit has been offline) the bigger the resulting cost. While the second factor is equal to 1 if unit has only been online in time step \(t\) between the time period \(\{j, \cdots, t\}\) (i.e. case of the unit been started in time period \(t\), while been offline in all \(j\) previous time steps). For all the different cases the second factor is zero or negative and as a result the values of the rhs of \((4.4.5.0.1a)\) \((4.4.5.0.1b)\) will be zero or negative and will be excluded due to \((4.4.5.0.2)\) -which is common for all different
4.4. THERMAL CONSTRAINTS

formulations that impose $c_{i,t}^0$ to be non-negative. Now it’s obvious that since $c_{i,t}^0$ is set in the minimization obj. function it will be forced to be equal to the greatest positive value of the rhs (4.4.5.0.1a) & (4.4.5.0.1b), which in case the unit has not been just turned on in time step $t$ will be zero, and in case it has been started in time step $t$ the greatest value will be the value of the start up cost function for the duration that the unit has been previously been offline.

4.4.6 Minimum on/off duration

- Every time thermal units are turned on they are required to stay online for a certain amount of time steps (minimum up time), and in the same fashion every time thermal units are turned off, they are to remain offline for a certain amount of time step (minimum down time). The following inequalities model these restrictions and are valid for all $i \in I$:

$$\sum_{k=t-\tau_i^+ + 1}^{t} v_{i,k} \leq \sum_{(s,n) \in S_N(i)} u_{s,n}^t$$  \hspace{1cm} (4.4.6.0.1)

$$\sum_{k=t-\tau_i^- + 1}^{t} w_{i,k} \leq 1 - \sum_{(s,n) \in S_N(i)} u_{s,n}^t$$  \hspace{1cm} (4.4.6.0.2)

For all different formulations (4.4.6.0.1) ensures that the minimum up duration of the thermal units is valid for all $t \in [\tau_i^+, t]$. And (4.4.6.0.2) ensures the minimum down duration of each thermal unit respectively is valid for all $t \in [\tau_i^-, t]$.

4.4.7 Maximum number of starts

- The thermal units are allowed to perform a maximum number of starts throughout the time horizon. In case of the nuclear units the maximum number of starts for the units applies for each day (half of the optimization horizon). The following constraints are introduced, Where we have that (4.4.7.0.1) refers to the thermal units other than nuclear ones (i.e. $i \in I \setminus I_N$) and (4.4.7.0.2) refers to the nuclear units (i.e. is valid for $i \in I_N$):

$$\sum_{t=\in T} v_{i,t} \leq V_i$$  \hspace{1cm} (4.4.7.0.1)

$$\sum_{t=\tau + \left\lfloor \frac{T}{2} \right\rfloor}^{t} v_{i,t} \leq V_i$$  \hspace{1cm} (4.4.7.0.2)

4.4.8 Deep decrease

- The state of deep decrease describes the situation where a nuclear unit is dropping its active power production below a certain threshold with a rate that is bigger than a certain gradient. In addition to the above there is a maximum number times that a unit is allowed
to start a deep decrease per day. The following constraints are introduced, that are valid for $i \in I_N, t \in T$:

\[
\begin{align*}
\tilde{d}_{i,t} & \geq \frac{\Delta \tilde{p}_i - (p_{i,t} - p_{i,t-1})}{\tilde{p}_i} & (4.4.8.0.1) \\
\hat{d}_{i,t} & \geq \frac{\hat{p}_i - p_{i,t}}{\hat{p}_i} & (4.4.8.0.2) \\
\hat{d}_{i,t} & \geq \hat{d}_{i,t} + \hat{d}_{i,t} - 1 & (4.4.8.0.3) \\
\sum_{t=\tau}^{T_\text{d}} d_{i,t} & \leq A_i & (4.4.8.0.4)
\end{align*}
\]

The above constraints are shared among all the different formulation making sure that for each different model the deep decrease condition is met. More specifically Constraint (4.4.8.0.1) ensures that $\hat{d}_{i,t}$ will be forced to be equal to one in the case that the unit drops its power put with a rate equal or greater than $\Delta \tilde{p}_i$ and (4.4.8.0.2) ensures respectively that $\hat{d}_{i,t}$ will be forced to be equal to one in the case that the power output of the unit will be smaller or equal $\tilde{p}_i$. Constraint (4.4.8.0.3) ensures that $d_{i,t}$ will be forced to be equal to 1, in the case that both $\hat{d}_{i,t}$ and $\hat{d}_{i,t}$ are equal to 1 (i.e. the case where we have the start of a deep decrease). Finally (4.4.8.0.4) which is constructed for $\tau = \{0, \bar{t}/2\}$ restricts the number of starts of a deep decrease to be smaller or equal $A_i$ in a daily basis.

### 4.4.9 Initial Conditions

- For the thermal units the following things need to be taken into consideration regarding the initial behavior of the units, for the different formulations:
  - The initial status of production states that is formulated for the different models as follows:
    1. * $u_{i,0}^s$, initial status of production states $s \in S(i)$ of unit $i$. 
    * $a_{i,0}^+, a_{i,0}^-$, initial status of increasing and decreasing modulations of unit $i$
    2. * $u_{i,0}^n$, initial status of production states $s \in S(i)$ and state of power change $n \in N(i)$ of unit $i$.
  - $\tau_{i,0}^1, \tau_{i,0}^2, \tau_{i,0}^3$, number of time steps unit $i$ has been producing primary, secondary, or maximum primary reserved power respectively prior the begin of optimization period, that is formulated for the different models as follows:
    1. $\tau_{i,0}^{sh}, \tau_{i,0}^{sc}, \tau_{i,0}^{ed}$
    2. $\tau_{i,0}^1, \tau_{i,0}^2, \tau_{i,0}^3$
    3. $\tau_{i,0}^1, \tau_{i,0}^2, \tau_{i,0}^3$
  - Number of time steps unit $i$ has been on stability mode prior the begin of optimization period, which is identical for all different formulation and is denoted as $B_{i,0}$.
  - Number of time steps unit $i$ has been offline and online respectively prior the begin of optimization period, which is identical for all different formulations and is denoted as $\tau_{i,0}^{1}, \tau_{i,0}^{1}$. 

## 4.4. THERMAL CONSTRAINTS

### Initial Reserved Power

- As stated above the units have to continue producing reserved power for a fixed amount of time steps after they started doing so. Based on the above we calculate the number of time steps that a unit \( i \) needs to initially produce any type of reserved power as follows for all different formulations, which are valid for \( i \in I, s \in S'(i) \):

\[
\bar{\tau}_i^s = \max\{0, (\tau_{i,0}^s - \tau_i^s) \cdot (u_{i,0}^s - a_{i,0}^- - a_{i,0}^+)\} \quad (4.4.9.0.1a)
\]

\[
\bar{\tau}_i^s = \max\{0, (\tau_{i,0}^s - \tau_i^s) \cdot u_{i,0}^{s,nb}\} \quad (4.4.9.0.1b)
\]

\[
\bar{\tau}_i^s = \max\{0, (\tau_{i,0}^s - \tau_i^s) \cdot \sum_{n \in N_2(i) \setminus N'_2(i)} u_{i,n}^{s,n}\} \quad (4.4.9.0.1c)
\]

- If the above calculation leads to a positive number then, we have for all \( i \in I, h \in \{1, \bar{\tau}_i^s - 1\}, s \in S'(i) \):

\[
u_{i,h}^s \geq 1 - \sum_{k=1}^{h} (v_{i,k}^- - u_{i,k}^+)^{\text{a}} \quad (4.4.9.0.2a)
\]

\[
u_{i,h}^{s,nb} \geq 1 - \sum_{k=1}^{h} \sum_{n \in N_2(i)} v_{i,k}^{n}\]

\[
\sum_{n \in N_2(i) \setminus N'_2(i)} u_{i,n}^{s,n} \geq 1 - \sum_{k=1}^{h} \sum_{n \in N'_2(i)} v_{i,k}^{n}\]

Note that in both constraints in all different formulations we take into account any ongoing modulation, as in case an ongoing modulation is occurring then these constraints are inactive.

### Initial Reserved Power in Modulation State

- In order to deal with the reserved production properly in case that unit has initially being modulating the following constraints are introduced:

- In order the unit has been initially modulating then we introduce the following fixed variables \( \phi_{i,0}^s, \forall s \in S'(i) \), which will be set equal to 1 in case the unit in the first time-step of the ongoing modulation (that has started prior to the optimization period) was also producing the corresponding reserve power \( s \), otherwise their value is set to zero. Then we have the following constraints that are valid for \( h = \{2, \cdots, M_i - 1\} \) and \( s \in S'(i) \):

\[
\sum_{n \in N_2(i) \setminus N'_2(i)} u_{i,n}^{s,n} \geq 1 - \sum_{k=1}^{h} \sum_{n \in N'_2(i)} v_{i,k}^{n}\]
\[ \phi_{i,0}^s + (u_{i,h}^s - 1) + w_{i,h+1}^-- w_{i,h+1}^+ - 1 - \sum_{k=1}^{h} w_{i,k+1}^- + w_{i,k+1}^+ \leq u_{i,1}^s \leq \phi_{i,0}^s \]  

(4.4.9.0.3a)

\[ \phi_{i,0}^s + \left( \sum_{n \in N'(i)} u_{i,n}^s - 1 \right) + \sum_{n \in N'(i)} w_{i,n,h+1}^-- w_{i,n,h+1}^+ - 1 - \sum_{k=1}^{h} \sum_{n \in N'(i)} w_{i,k+1}^- + w_{i,k+1}^+ \leq \sum_{n \in N'(i)} u_{i,1}^s \leq \phi_{i,0}^s \]  

(4.4.9.0.3b)

\[ u_{i,1}^s \leq u_{i,h}^s + w_{i,h+1}^- + w_{i,h+1}^+ + \sum_{k=1}^{h} w_{i,k+1}^- + w_{i,k+1}^+ \]  

(4.4.9.0.4a)

\[ \sum_{n \in N'(i)} u_{i,n}^s \leq 1 - \sum_{n \in N'(i)} w_{i,n,h+1}^- + \sum_{k=1}^{h} \sum_{n \in N'(i)} w_{i,k+1}^- + w_{i,k+1}^+ \]  

(4.4.9.0.4b)

Note that in both constraints in all different formulations we take into account any ongoing modulation, as in case an ongoing modulation is occurring then these constraints are inactive.

**Initial minimum on/off duration**

- The number of hours during which thermal unit \( i \) must be initially on (\( \bar{\tau}_i^+ \)) or off (\( \bar{\tau}_i^- \)) is calculated for the different formulations as follows for \( i \in I \):

\[ \bar{\tau}_i^+ = \max \{0, (\tau_{0,+}^i - \tau_+^i) \cdot \sum_{s \in S(i)} u_{i,0}^s \} \]  

(4.4.9.0.5a)

\[ \bar{\tau}_i^+ = \max \{0, (\tau_{0,+}^i - \tau_+^i) \cdot \sum_{(s,n) \in S_N(i)} u_{i,0}^{s,n} \} \]  

(4.4.9.0.5b)

\[ \bar{\tau}_i^- = \max \{0, (\tau_{0,-}^i - \tau_-^i) \cdot (1 - \sum_{s \in S(i)} u_{i,0}^s) \} \]  

(4.4.9.0.6a)

\[ \bar{\tau}_i^- = \max \{0, (\tau_{0,-}^i - \tau_-^i) \cdot (1 - \sum_{(s,n) \in S_N(i)} u_{i,0}^{s,n}) \} \]  

(4.4.9.0.6b)

- If \( (\bar{\tau}_i^+ + \bar{\tau}_i^-) \geq 1 \) then for \( i \in I, t \in [1, \bar{\tau}_i^+ + \bar{\tau}_i^-] \) we have that:

\[ \sum_{s \in S(i)} u_{i,t}^s = \sum_{s \in S(i)} u_{i,0}^s \]  

(4.4.9.0.7a)

\[ \sum_{(s,n) \in S_N(i)} u_{i,t}^{s,n} = \sum_{(s,n) \in S_N(i)} u_{i,0}^{s,n} \]  

(4.4.9.0.7b)
4.5 HYDRO CONSTRAINTS

Initial startup cost

- In order to calculate properly the startup cost of units taking into consideration the initial times the units have been previously offline we need to add the following constraints in the corresponding formulations for \( i \in I, t \in T \):

\[
c_{0it} \geq \left[ c_{i,1} + c_{i,2} \cdot \ln\left(\frac{(t + s^t_{0i}) - 1}{c_{i,3}} \cdot \Delta T m\right)\right] \cdot \left[ v_{it} - \sum_{\tau=1}^{t-1} v_{i,\tau}\right] \quad (4.4.9.0.8a)
\]

These works in the similar fashion as (4.4.5.0.1a)-(4.4.5.0.1b), with the difference that in this case the second factor of the rhs can be equal to 1 only for the first time that the unit is been turned on, which is the only case where we should take into consideration the initial offline duration of the unit.

4.5 Hydro Constraints

Due to the fact that the hydro part of all the formulations is common, the formulation will be presented only once.

4.5.1 Power production

- Hydro units that are considered to be discrete (i.e. \( H_D \)) can take only a set of discrete values of their power and flow selected by operators, whereas hydro units that are considered continuous (i.e. \( H_C \)) have as well a set of different values, with the difference that in their case operations are allowed in between these points and are approximated by affine segments. The set of the available values for both different types of hydro plants are expressed through the so called operational points (i.e. \( J(h,t) \)). In order to formulate the flow and the power production the following equations are proposed that are valid for all \( t \in T \):

\[
p_{ht} = \begin{cases} 
\sum_{j \in J(h,t)} x_{h,j,t} \cdot \delta P_{h,j,t} & \forall h \in H_D, \\
\sum_{j \in J(h,t)} y_{h,j,t} \cdot \delta P_{h,j,t} & \forall h \in H_C.
\end{cases} \quad (4.5.1.0.1)
\]

\[
p^{1}_{ht} = \begin{cases} 
\sum_{j \in J(h,t)} x_{h,j,t} \cdot \delta P^1_{h,j,t} & \forall h \in H_D, \\
\sum_{j \in J(h,t)} y_{h,j,t} \cdot \delta P^1_{h,j,t} & \forall h \in H_C.
\end{cases} \quad (4.5.1.0.2)
\]

\[
p^{2}_{ht} = \begin{cases} 
\sum_{j \in J(h,t)} x_{h,j,t} \cdot \delta P^2_{h,j,t} & \forall h \in H_D, \\
\sum_{j \in J(h,t)} y_{h,j,t} \cdot \delta P^2_{h,j,t} & \forall h \in H_C.
\end{cases} \quad (4.5.1.0.3)
\]
\[ q_{ht} = \begin{cases} \sum_{j \in J(h,t)} x_{h,j,t} \cdot \delta Q_{h,j,t} & \forall i \in H_D, \\ \sum_{j \in J(h,t)} y_{h,j,t} \cdot \delta Q_{h,j,t} & \forall i \in H_C. \end{cases} \tag{4.5.1.0.4} \]

\[ x_{h,j+1,t} \leq x_{h,j,t} \tag{4.5.1.0.5} \]
\[ x_{h,j,t} \leq y_{h,j,t} \tag{4.5.1.0.6} \]
\[ y_{h,j+1,t} \leq x_{h,j,t} \tag{4.5.1.0.7} \]

Where for the determination of the level of the different types of production and of the flow in the hydro units we use the decision variables \( x_{h,j,t}, y_{h,j,t} \) that denote if the equivalent operation points of the discrete and the continuous curve respectively (note that \( y_{h,j,t} \) can take any value in the interval \([0, 1]\)), then in order to define the amount of production or flow we multiply with the equivalent point of each curve (i.e. \( \delta P_{h,j,t}, \delta P^1_{h,j,t}, \delta P^2_{h,j,t}, \delta Q_{h,j,t} \)). Moreover in order to safeguard that the activation of the operational points is being done properly by avoiding the activation of any operational point without all of its predecessors to have been activated we use constraint \((4.5.1.0.5)\) for \( h \in H_D \) is imposing the operational point \( j + 1 \) for the discrete hydro units to be able to be equal to one, only if operational point \( j \) is also equal to one. In the same fashion \((4.5.1.0.6)\) and \((4.5.1.0.7)\) are imposing for \( h \in H_C \) the operational point \( j + 1 \) of the continuous hydro units to be able to take non-zero values only if operational point of the continuous power curve \( j \) has been reached (i.e. \( y_{h,j,t} = 1 \)). Finally let us note that the above modeling approach lies heavily in the fact that the operational points of the units are not following necessarily a concave function and also power level are not necessarily expected to follow an increasing or decreasing order for the generating and pumping unit respectively, since in the first one we can expect to find also negative power increments and in the latter one we can expect to find also positive power increments.

4.5.2 Water volume conservation

- The water volume conservation ensures that for all the different types of reservoirs the flow of the water is calculated properly based on the various factors (inflows, spillage, discharge etc) and is valid for \( t \in T \).

\[ \psi_{r,t} - \psi_{r,t-1} = \Delta T_s \cdot \left[ E_{r,t} + \sum_{h \in H^+(r) \cap H_G} (q_{h,t} - r^+ + \sigma_{h,t} - r^+) \right] + \sum_{h \in H^-(r) \cap H_P} q_{h,t} \tag{4.5.2.0.1} \]

\[ E_{r,t} + \sum_{h \in H^+(r) \cap H_G} (q_{h,t} - r^+ + \sigma_{h,t} - r^+) \tag{4.5.2.0.2} \]

\[ - \sum_{h \in H^-(r) \cap H_G} (q_{h,t} + \sigma_{h,t}) - \sum_{h \in H^+(r) \cap H_P} q_{h,t} = 0 \]
In (4.5.2.0.1) we refer to the case where the reservoirs \((r \in R_T \cup R_W)\) have storage capacity and in that case the difference of water volume between two consecutive periods should be equal to the sum of external inflows and the inflows from the upstream and downstream reservoirs (taking into account the time lapse needed for water to reach the reservoir), minus the inflows to the downstream reservoir and to the upstream reservoir, and also note that \(\psi_{r,0} = \Psi_{r,0}, r \in R_T \cup R_W\). In (4.5.2.0.2) we refer to the case where the reservoirs \((r \in R_C)\) have no storage capacity where we have a simplified version of the first constraint, where the volume variables are not included and the parameter \(\delta T\) can be dropped out too.

### 4.5.3 Volume

- The volume of the reservoirs is restricted to stay inside a specific interval of values. In addition to this the volume of the reservoirs is imposed to be inside a specific interval of values in the middle and in the end of the optimization period respectively. The above mentioned restrictions in the values of the volume of the reservoirs can be modelled as follows that are valid for \(r \in R_T \cup R_W\):

\[
\begin{align*}
\Psi_r & \leq \psi_{r,t} \leq \Psi_r & (4.5.3.0.1) \\
\Psi'_r & \leq \psi_{r,t-2} \leq \Psi'_r & (4.5.3.0.2) \\
\Psi''_r & \leq \psi_{r,t-4} \leq \Psi''_r & (4.5.3.0.3)
\end{align*}
\]

Here the minimum and maximum values that the reservoir volume can take during the optimization period are imposed through (4.5.3.0.1), which holds for \(t \in T\). And in similar fashion (4.5.3.0.2) and (4.5.3.0.3) are imposing the volume of the reservoir to stay between some certain boundaries at the middle and at the end of the optimization period.

### 4.5.4 Spillage

- The values of spillage are restricted to stay inside a specified interval and apart from this in order for a hydro unit to be allowed to spill water, it must have already reached its production pick (i.e. if the production reached the last operational point of the corresponding production curve.). The above conditions are valid for \(h \in H_G, t \in T\) and can be modelled as follows:

\[
\begin{align*}
s_{h,t} & \leq z_{h,t} \cdot S_h & (4.5.4.0.1) \\
z_{h,t} & \leq x_{h,j(i),t} & (4.5.4.0.2)
\end{align*}
\]

Where we introduce binary variable \(z_{h,t}\) that due to constrain (4.5.4.0.2) can only be non-zero in case the equivalent unit is producing until the end of the corresponding power curve (i.e. the point \(j(i)\)). Finally in constraint (4.5.4.0.1) the amount of spillage is bounded according to its boundaries in case of course the unit is allowed to spill any water (case where \(z_{h,t}\) is non-zero).
4.5.5 Flow

- The amount of increase or decrease of the water flow between two consecutive time periods for the hydro units is restricted to stay inside some certain bounds. More specifically, the following constraints are valid for \( h \in H, t \in T \):

\[
(q_{h,t} + s_{h,t}) - (q_{h,t-1} + s_{h,t-1}) \leq \Delta Q_h \cdot \Delta T_m \tag{4.5.5.0.1}
\]
\[
(q_{h,t} + s_{h,t}) - (q_{h,t-1} + s_{h,t-1}) \geq -\Delta Q_h \cdot \Delta T_m \tag{4.5.5.0.2}
\]

\( \Rightarrow \) Where constraint (4.5.5.0.1) is imposing the rate of increase to be smaller of equal than \( \Delta Q_h \), and constraint (4.5.5.0.2) is imposing the rate of decrease to be smaller of equal than \( \Delta Q_h \).

4.5.6 Pumping

- In order not to allow simultaneously pumping and turbining following constraints are introduced, that are valid for \( t \in T \) and for variables that have at least one operational point (i.e. \( \bar{j}(h_P) \geq 1 \bar{j}(h_G) \geq 1 \)):

\[
\begin{align*}
x_{h_G,1,t} + x_{h,P,1,t} &\leq 1 \\
x_{h_G,1,t} + x_{h,P,1,t-1} &\leq 1 \\
x_{h_G,1,t-1} + x_{h,P,1,t} &\leq 1
\end{align*} \tag{4.5.6.0.1}
\]

The above constraints are referring to the set of production and pumping units and assure that there won’t be any “illegal” activities, such as for a unit to produce power when the equivalent pumping unit is active or the other way around.

4.5.7 Monotonicity

- Monotonicity constraints are ensuring that there are no \( \gamma \) shapes in the power production of the hydro units. That means that for every 3 time-step period the hydro unit can not change its status more than once (i.e. no 1-0-1 or 0-1-0). This constraint ensures smooth operations to maintain a sustainable level of stress and wear for the hydro-units. There has been also taken into account the above mentioned fact (in 5.1) that two equivalent time steps may refer to different operational set of points, as a unit can have more than one set of operational points through the whole optimization horizon. So one has the following constraints that are valid for \( h \in H_D, j \in J(h,t), t \in T \):

\[
\begin{align*}
x_{h,j,t} &\geq x_{h,j-1,t-1} + x_{h,j+1,t+1} - 1 \\
x_{h,j,t} &\leq x_{h,j-1,t-1} + x_{h,j+1,t+1} - 1
\end{align*} \tag{4.5.7.0.1}
\]

Where \( (j^-, j^+) \in J(h, t-1) \times J(h, t+1) \): \( \sum_{k=1}^{j^-} \delta Q_{h,k,t-1} = \sum_{k=1}^{j} \delta Q_{h,k,t} = \sum_{k=1}^{j^+} \delta Q_{h,k,t+1} \) and where we replace \( x_{h,j,0} \) with the given operational point at the beginning of the time horizon.
4.5.8 Volume dependent water values

- In order to avoid the maximum use of the water during the time horizon, the following formulation where water values are assigned for the different possible final reservoir levels is proposed, that is valid for \( r \in R^W \):

\[
\sum_{k=1}^{K_r} g_{r,k} = Y_{r,i} - Y_{r,0} \tag{4.5.8.0.1}
\]

\[
0 \leq g_{r,k} \leq Y_{r,k} - Y_{r,k-1} \tag{4.5.8.0.2}
\]

where the volume of each reservoir is divided in an increasing order into \( K_r \) intervals and the auxiliary variable \( g_{r,k} \) is introduced that denotes the amount of water in each compartment \( [Y_{r,k} - Y_{r,k-1}) \), finally for each compartment there is an equivalent water value \( \omega_{r,k} \), its values following a decreasing order. Through the above two constraints we guarantee that the upper inequality in \( (4.5.8.0.2) \) is satisfied as an equality as long as possible, which is straightforward due to \( (4.5.8.0.1) \) and because of the fact as we want to maximize the value of the final water level and the lower compartment have the highest water values. Also in \( (4.5.8.0.2) \) we have that \( k = 1, ..., K_r \). Finally let us note that the water values are -the only part of the hydros that- are also included in the objective function constituting the last summation that is included in all three formulations.

4.6 Comparing the EDF and Academic UCs

Before we close this chapter it would be useful to present some considerations about the main differences between the Academic and EDF UC problems, that make the latter much harder to be solve.

The first thing to be noted here is the fact that proposed MILP-formulations of the EDF UC are significantly bigger in size when compared with the academic and much more simplified versions of the problem. In order for this to be clearly demonstrated the following table is introduced where the several different number of variables and constraints for the cases of the three different proposed formulations and a basic Academic UC are presented for the case of a thermal problem with 60 nuclear units, 100 other units and 96 time steps:

<table>
<thead>
<tr>
<th></th>
<th>Total Variables</th>
<th>Continuous Variables</th>
<th>Integer Variables</th>
<th>Unit Commitment Variables</th>
<th>Auxiliary Integer Variables</th>
<th>Total Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-index Model</td>
<td>401280</td>
<td>30720</td>
<td>370560</td>
<td>46080</td>
<td>324480</td>
<td>1176820</td>
</tr>
<tr>
<td>4-index Model</td>
<td>530520</td>
<td>30720</td>
<td>508800</td>
<td>184320</td>
<td>324480</td>
<td>1566600</td>
</tr>
<tr>
<td>4-index Model (expanded)</td>
<td>662400</td>
<td>30720</td>
<td>631680</td>
<td>307200</td>
<td>324480</td>
<td>1129910</td>
</tr>
<tr>
<td>Academic UC</td>
<td>46080</td>
<td>30720</td>
<td>15360</td>
<td>15360</td>
<td>0</td>
<td>291840</td>
</tr>
</tbody>
</table>

As it becomes obvious from the above numbers on the table, the size of the EDF-UC is significantly larger between 9-14 times in variable size and around 4 times in constraint size depending on the different MILP-Formulation.
An other factor that needs to be taken into consideration is the fact that all these complicating constraints for the power output and the power shifts of the thermal units do make the problem much more difficult to be tackled by a DP. Without going into too any real analysis on this important subject, the construction of a graph that refers to the Edf Thermal units with a first look seems to be much more complicating and with a much bigger size of arcs and nodes and also the proper handling of the arc connections due to the thermal constraints seems to be a really delicate task. Here let us note though that the above constructed models were designed keeping in mind the DP-algorithms and thus it can be argued that the adoption of the third and fourth index for the commitment variables could help make thins a bit easier. Yet again since no real research and investigation has been made so far on this aspects we can not further extend the above considerations for the time being.

Moreover an other crucial factor is the fact that the more complicated nature of the units make it impossible to directly adopt and implement the lagrangian heuristics that have been successfully developed and applied for the academic versions. As it will be discussed in more detail in 5.5.2, the already existing approaches proceed with assumptions that here clearly are not valid (such as the fact that the hydro units are continuous, or that the thermal units have only at maximum a set of gradient and min-up/down time constraints).

In order to conclude, what is clear is the fact that bigger size and the more complicated nature of the EDF UC makes it not possible to directly implement the already existing approaches for the Academic UC versions. This situation doesn’t make it impossible for such methods to be applied, but first a careful and in depth research and analysis needs to be done that can lead to the design to some new approaches that adopt the challenging characteristics of the EDF UC.
Chapter 5

Lagrangian Approaches for UC

As already briefly discussed already in the first chapter the UC is considered to be a difficult to solve problem, requiring dedicated optimisation tools and techniques in order to manage to obtain primal feasible solutions within reasonable time frameworks. In our work we consider to tackle the Edf UC via Lagrangian Decomposition and relax the linking constraints of demand and spinning reserve which were presented for the different formulations of the problem in 4.3. Since these constraints refer to the case of a single bus model as it was discussed in 2.1.5 their relaxation is not a complicated process and thus makes this choice even more attractive. This will lead to a series of independent subproblems one for each unit (simple thermal, nuclear, hyrdo) that need to be tackled down (via DPs and/or general purpose solvers) and upon their solution we consider to obtain primal feasibility by implementing lagrangian heuristics. Here it let us stress out again the fact that the Lagrangian Decomposition together with the Lagrangian Heuristics that will be further presented in 5.5.2 have not yet been implemented and tested and thus there are no results to be presented. Thus in this chapter we will proceed with presenting first the basic mathematical framework that is connected with the Lagrangian Decomposition, after this we will give a brief overview of the existing literature of lagrangian heuristics successfully applied in academic versions of UC and finally we will go on with presenting a proposed approach for a lagrangian heuristic for the complicated Edf UC.

5.1 Integer Linear Programming

The first thing to be mentioned in this chapter is class of Integer Linear Problems (ILP). ILPs in general are considered to be one of the most relevant classes of hard Combinatorial Optimization problems and can be informally described as the class of finite-dimensional problems where both the objective function and the constraints are linear, and (a subset of) the variables are required to assume only integer values. This class has been very extensively researched in the last decades, resulting in the fact that a clever combination of several algorithmic techniques has permitted the development of general-purpose solvers that can nowadays routinely solve many instances of mid- to large-scale (although even at small-scale, particularly hard instances still can be found that elude solution by these approaches). In our development we will be mainly interested in problems with the form

\[
(P) \quad \min \{ cx : Ax = b , \; Ex = d , \; x \in \mathbb{N}^n \}
\]
where \( Ax = b \) is a set of “difficult” (complicating) constraints and \( Ex = d \) is a set of “easy” constraints, in the sense that \((P)\) would be an easy optimization problem (or, at least, very substantially easier to solve in practice) if the constraints \( Ax = b \) were not there. A very typical example is the case where \( Ax = b \) is a set of linking constraints and \( Ex = d \) is a set of separable constraints; that is, \( x \) can be partitioned in sub-vectors \( x^1, x^2, \ldots, x^k \) and thus we define and will use in the rest of the chapter \( X \) as:

\[
X = \{ x \in \mathbb{N}^n : Ex = d \} = X^1 \times X^2 \times \ldots \times X^k
\]

so that \((P)\) takes the form

\[
(P') \quad \min \{ \sum_{i=1}^k c^i x^i : \sum_{i=1}^k A^i x^i = b, x^i \in X^i \quad i = 1, \ldots, k \}.
\]

In this case, it can be easily seen that if the constraints \( Ax = b \) could be somehow disregarded, then \((P)\) would decompose into a set of independent smaller subproblems, one for each \( X^i \). These, just because they are much smaller, can be expected to be much easier to solve than the original problem; besides, one can use parallel computing techniques in order to solve them in a much smaller computational time. A more detailed presentation for decomposable sets and its applications will be given in §5.3

One of the fundamental reasons why effective techniques have been developed for ILP (at least, as much as this is permitted by their \( \mathcal{NP} \)-completeness) is to be traced to the fact that the continuous relaxation

\[
(\bar{P}) \quad \min \{ cx : Ax = b, Ex = d, x \geq 0 \}
\]

is a Linear Program (LP) and can therefore in general be solved very efficiently, for sizes of millions of variables and constraints, with either simplex-type methods or interior-point ones like the one proposed in [32, 77, 171]. This is useful for at least two reasons:

1. if the optimal solution \( x^* \) to \((\bar{P})\) happens to be integer \( (x^* \in \mathbb{N}^n) \), then it can be easily proven to be optimal for \((P)\);
2. more in general, \((\bar{P})\) provides a valid lower estimate of the optimal value of the original problem, i.e., \( z(\bar{P}) \leq z(P) \) (here and in the following \( z(\cdot) \) denotes the optimal value of an optimization problem).

In conclusion the continuous relaxation can provide some basic means for calculation of feasible solutions, and at least to estimate the quality of solutions that have been otherwise produced. This is in particular the main ingredient that is required to construct enumerative techniques for the solution of \((P)\). The fact that an LP can be efficiently solved in practice irrespective of the structure of the matrices \( A \) and \( E \) (and in particular for these coming from real applications, that are typically sparse) is, at heart, what makes it possible to develop general-purpose ILP (and MILP) solvers. Unfortunately, it is often the case that the estimate is poor, i.e., \( z(\bar{P}) \ll z(P) \), this is known to be very detrimental to the performances of enumerative approaches. In fact, one of the main challenges in algorithmic approaches when solving these problems, is that of obtaining more accurate lower bounds than these afforded by the standard continuous relaxation. There are different ways to achieve that, among which particularly powerful ones are the addition of valid inequalities based on polyhedral theory, or reformulation techniques
that help in developing different formulations of \((P)\) (different data \(A, E, b\) and \(d\), often even in a different variables space) whose continuous relaxation is tighter like the one proposed and developed in [23, 91, 165]. One of the most general reformulation techniques will assume a central role in this Thesis, and is therefore described in details below.

5.2 Lagrangian Relaxation

As already mentioned in the chapters above, one of the most applied techniques capable of exploiting the structure of problem \((P)\) above (such as in the case of the UC) is forming the relaxation of \((P)\):

\[
\begin{align*}
(P_\lambda) & \quad \min \{ cx + \lambda (b - Ax) : x \in X \} \\
(D) & \quad \max \{ f(\lambda) = z(P_\lambda) : \lambda \in \mathbb{R}^m \}
\end{align*}
\]

where the complicating constraints \(Ax = b\) are relaxed and a penalty term is added to the objective function to (partly) represent them. This is based on the use of a fixed \(m\)-dimensional vector \(\lambda\) (under the assumption that the number of complicating constraints is \(m\)), called the Lagrangian multipliers. It is straightforward to show that for any choice of \(\lambda \in \mathbb{R}^m\) we will always have that \(f(P_\lambda) \leq f(P)\); as a result, it makes sense to search for the tightest possible Lagrangian Relaxation of the original problem \((P)\), which leads to the Lagrangian Dual

\[
(D) \quad \max \{ f(\lambda) = z(P_\lambda) : \lambda \in \mathbb{R}^m \}.
\]

It is important to remark that, as firstly shown in [73], solving \((D)\) is equivalent of solving the “convexified relaxation” of \((P)\):

\[
(\tilde{P}) \quad \min \{ cx : Ax = b, x \in \text{conv}(X) \}
\]

where \(\text{conv}(\cdot)\) denotes the convex hull of a set. Indeed, one can easily see (cf. e.g. [61] and §5.3) that not only \(z(\tilde{P}) = z(D)\), but \((D)\) and \((\tilde{P})\) are the linear dual of each other. This has the following useful consequences:

- Since the feasible region of \(\tilde{P}\) contains the feasible region of \((\tilde{P})\), while both problems have the same objective function, \((\tilde{P})\) is a better relaxation (not worse) of the original problem \((P)\) than \(\tilde{P}\), i.e., \(z(D) \geq z(\tilde{P})\).

- In the case where the “easy” set \(X\) has the integrality property (that is that all the vertices of \(X\) are integral, which means that \(P_\lambda\) is as easy as an LP), then \(z(D) = z(\tilde{P}) = z(\tilde{P})\).

So, in order for the Lagrangian Dual to provide a better bound there should be a set \(X\) that lack the integrality property, which implies that \((P_\lambda)\) is “difficult”.

For a more detailed analysis of the fundamentals of Lagrangian Relaxations the interested reader is referred e.g. to [61, 104]; a few of the very many examples of the successfully development and implementation of Lagrangian methods in large scale difficult optimization problem are [84, 71, 38, 31].

5.3 Cutting Plane methods, the Dantzig-Wolfe decomposition approach, and Column Generation

Lagrangian Relaxation methods can be quite efficient in order to tackle optimization problems, but at least some approximate solution of \((D)\) is needed. In principle we intent to tackle \((D)\) via
the use of Bundle methods, but before present Bundle it is thought to proceed with discussing
the most fundamental techniques that have been developed for tackling \((D)\) and it can be
considered that in a way inspired the creation of Bundle methods. Let us start with assuming
that \(X\) is discrete, nonempty and compact, which means that \(f(\lambda)\) is finite everywhere; moreover
\(f\) is concave and more precisely polyhedral, i.e., its epigraph \(\text{Epi}(f) = \{ (v, \lambda) : v \leq f(\lambda) \}\)
is a polyhedron. What this means and is of significant importance here, is that for each solution
\(\bar{x}\) to \(P_\lambda\) for a given point \((\lambda)\) we have a supporting hyperplane \(b - A\bar{x}\) to \(\text{Epi}(f)\), which is called
the subgradient of \(f\) at \(\lambda\) (technically a super gradient, but we avoid this distinction). As a
result, we can rewrite \((D)\) as a (very-large scale) LP of the following form
\[
(D) \quad \max_{v, \lambda \in \mathbb{R}^m} \{ v : v \leq cx + \lambda(b - Ax) \}, \forall x \in X
\]
where it should be noted that in this case \(X\) is basically the index set of the constraints. While
\((D)\) can therefore have an enormous number of constraints, in principle in order to solve it one
would only need \(m + 1\) of then, the ones that correspond to an optimal base. This suggests to
consider the (Dual) Master Problem
\[
(D_B) \quad \max_{v, \lambda \in \mathbb{R}^m} \{ v : v \leq cx + \lambda(b - Ax) \}, \forall x \in \mathcal{B}
\]
where \(\mathcal{B}\) is a “small” subset of \(X\), which makes \((D_B)\) possible to solve in practice. Note that this
means solving
\[
\max_{\lambda \in \mathbb{R}^m} \{ f_B(\lambda) = \min\{ cx + \lambda(b - Ax) : \lambda \in \mathbb{R}^m, x \in \mathcal{B} \}\},
\]
where \(f_B\), called the cutting plane model of \(f\), is a concave polyhedral over-approximation; this
means that \((D_B)\) is a relaxation of \((D)\), i.e., \(z(D_B) \geq z(D)\). In order to check if the optimal
solution \(\bar{\lambda}\) of \((D_B)\) is optimal for \((D)\) as well, one just has compute \(f(\bar{\lambda})\), which amounts at
producing a new \(\bar{x} \in X\); if \(\bar{v} > f(\bar{\lambda})\) then a new constraint (corresponding to \(\bar{x}\)) is added to
\(\mathcal{B}\), otherwise we end up with an optimal solution of \((D)\) with \(\bar{v} = f(\bar{\lambda}) = f(D)\). This leads to
Kelley’s Cutting Plane (CP) algorithm \([93]\), illustrated in the following pseudo-code, which can be
easily proven to finitely solve \((D)\) if \(X\) is discrete (but also works for any convex function,
albeit asymptotically).

```
initialize \(\mathcal{B}\);
do
    solve \((D_B)\) for \((\bar{v}, \bar{\lambda})\);
solve \((P_\lambda)\) for \(\bar{x}\), let \(f(\bar{\lambda}) = cx + \bar{\lambda}(b - A\bar{x})\);
    \(\mathcal{B} = \mathcal{B} \cup \{\bar{x}\}\)
while(\(\bar{v} > f(\bar{\lambda})\))
```

An observation that will be relevant in the following is that the above approach is closely linked
(indeed, totally equivalent) to the widely known Danzig-Wolfe Decomposition Approach.
In fact, this is based again on the convexified relaxation \((\hat{P})\), where the convex hull is explicitly
represented as
\[
\text{conv}(X) = \{ x = \sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} : \sum_{\bar{x} \in X} \theta_{\bar{x}} = 1, \theta_{\bar{x}} \geq 0, \bar{x} \in X \}
\]
note again as \(X\) here is an index set, this time of the convex multipliers \(\theta\). This leads to the
explicit reformulation of \((\hat{P})\)
\[
(P_{DW}) \quad \min_{\theta} \left\{ c\left( \sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) : A\left( \sum_{\bar{x} \in X} \bar{x}\theta_{\bar{x}} \right) = b, \theta \in \Theta \right\}
\]
where $\Theta = \{ \theta \geq 0 : \sum_{\tilde{x} \in X} \theta_{\tilde{x}} = 1 \}$ is the unit simplex of proper dimension. Again, one can think of restricting this exponentially large problem by considering a $B \subset X$: the corresponding Primal Master Problem

$$(P_B) \quad \min_{\theta} \left\{ c \left( \sum_{\tilde{x} \in B} \tilde{x}\theta_{\tilde{x}} \right) : A \left( \sum_{\tilde{x} \in B} \tilde{x}\theta_{\tilde{x}} \right) = b, \theta \in \Theta \right\}$$

can be easily seen to be the linear dual of $(D_B)$. Based on this remark the Cutting Plane Algorithm seen in the Primal Space can be considered equivalent as a Column Generation (CG) method [9, 78, 113] that is tackling $(P_B)$, where in every iteration a primal optimal solution $\theta^*$ is obtained that composes a feasible solution $\tilde{x} = \sum_{\tilde{x} \in B} \tilde{x}\theta^*_{\tilde{x}}$ and a dual solution $\tilde{y}$ corresponding to constraints $Ax = b$ as well. The solution of $f(\lambda)$, referred as pricing problem in this context, determines the column $\tilde{x} \in X$ with the smallest possible reduced cost $(c - \lambda A)x$; if this is non-negative then $\tilde{x} = x^*$ is optimal for $(\tilde{P})$, otherwise the column $\tilde{x}$ is added to $B$ with the prospect of constructing a better primal solution in the next iteration. The described process is in fact the same procedure with the CP algorithm described above, except for the the explicit computation of $\tilde{x}$ (that is not strictly required for algorithmic purposes, but will be relevant later on). In other words, the CP algorithm for solving $(D)$ is exactly the standard CG method for solving the “explicit form” of $(\tilde{P})$ as a LP with many columns.

Before moving to the next subsection, one more feature that should be mentioned is the case where the set $X$ has a decomposable structure, i.e., $X = X^1 \times X^2 \times \ldots \times X^k$. This means that any optimal solution $\tilde{x}$ of the Lagrangian relaxation has the form $[\tilde{x}^1, \tilde{x}^2, \ldots, \tilde{x}^k]$, with $\tilde{x}^i$ denoting the optimal solution of the $i$-th subproblem. In other words, the Lagrangian function has the form

$$f(\lambda) = \lambda b + \sum_{i=1}^{k} f^i(\lambda), \quad \text{where} \quad f^i(\lambda) = \{ (c^i - \lambda A^i)x^i : x^i \in X^i \}$$

and one can easily see that $-A^i\tilde{x}^i$ is a subgradient of $f^i(\lambda)$. This suggests to consider the disaggregated Dual Master Problem

$$(D_B) \quad \max_{v, \lambda \in \mathbb{R}^m} \left\{ \lambda b + \sum_{i=1}^{k} v^i : v^i \leq (c^i - \lambda A^i)x^i, \quad x \in B^i \quad i = 1, \ldots, k \right\}$$

which can be rewritten as

$$(D_B) \quad \max_{v, \lambda \in \mathbb{R}^m} \left\{ \lambda b + \sum_{i=1}^{k} f^i_B(\lambda) \right\}$$

where $f^i_B$ is the individual cutting plane model of $f^i$. The sum of the $k$ cutting plane models is a better approximation of $f$ than the aggregated cutting plane model $f_B$, which leads to faster convergence (cf. e.g. [9, 67]); the only drawback of the disaggregated approach is that the disaggregated Master Problem is, roughly, $k$ times bigger than the aggregated one, and hence more costly to solve. Yet, the longer computation time needed for each iteration can be amply compensated by the strong reduction of the number of iterations required to converge to optimality.
5.4 Better versions of the Cutting Plane algorithm: Bundle methods

The CP algorithm presented in the previous section is one of the fundamental algorithms in convex optimization, but is not particularly efficient. The main observation is that in the CP algorithm, the next iterate $\tilde{\lambda}_{k+1}$ can be very far from the previous one $\tilde{\lambda}_k$. This makes the approach “unstable”, which means it has not desirable locality properties whereby the model is more and more accurate at $\tilde{\lambda}_{k+1}$ as the iterations increase, and therefore the convergence speed does not improve when we are near the optimum. That is, a good estimate of dual optimum is unfortunately useless, and the generated constraints can be of “bad quality”; all of this results in slow convergence.

Hence, a crucial need in order to be able to have an efficient algorithm that solves $(D)$ through the use of a model is a way to add some “stability” to the method; i.e., the introduction of a stabilization technique. This is the main idea behind the bundle methods, that provide an efficient way to stabilize the CP approach; and, as these are one of the main tools that will be used throughout our research, it is important to present the key points of the method. We refer any interested reader to [44, 60, 95, 105] for a more detailed and analytical presentation of bundle methods. It should be mentioned, however, that in the literature there are other stabilization techniques such as the Analytic Center Cutting Plane Method (ACCPM), [75] the Primal-Dual Column Generation Method (PDCGM) [78, 79], and the In-Out approach [17].

Bundle methods are trying to deal with this instability of the CP algorithm by imposing some sort of localization approach. This is done with the use of an appropriate stabilization center $\hat{\lambda}$, usually a point $\tilde{\lambda}_k$ that we would like to be near to. In the CP approach the approximation of $D$ may be too optimistic, i.e. $\tilde{v} >> f(\tilde{\lambda})$, which has as a result to end up with points $\tilde{\lambda}$ that are far away from the desired point $\hat{\lambda}$. Yet again even for cases where the function values are reasonably close to it, still the step can be very large. As a result in order to try to work around this situation, it is introduced a quadratic stabilization term $\|\lambda - \hat{\lambda}\|^2$, where this term has the role of a trust region where large steps are penalized, since the model cannot be trusted far away from the current point. Moreover, bundle methods have a stabilization parameter that is determining the effect of the stabilization centre, and which is really important to be tuned properly. In the literature there are two main Bundle approaches, that are divided in the way this stabilizing parameter is handled and used inside the problem.

In the first case the stabilization parameter $l$ of the method is set into the constraint. This approach is called the Proximal Level Bundle Method (PLBM). For a thorough review of the method we refer the interested reader to [97, 105]. In the Proximal Bundle Method (PBM), instead, the stabilization parameter (in this case denoted as $t$) is set into the objective function, leading to a stabilized Master Problem of the form

$$ (D_{B,t}) \quad \max_{v,\lambda \in \mathbb{R}^n} \{ v - \|\lambda - \hat{\lambda}\|^2/2t : v \leq cx + \lambda(b - Ax), x \in B, \ t > 0 \} $$

So far, experience tells that PBM has been proven to be very efficient in dealing with unconstrained problems. As one can understand, the handling of the stabilization parameter plays a crucial role in the convergence and the overall efficiency of the method and thus an appropriate tuning strategy has to be followed, which can lead to updates of $t$ even at each iteration; some update strategies for $t$ can be found [96, 108].

In either case, after the master problem is solved we end up with a hopefully “well stabilized”
point $\tilde{\lambda}$, together with $\tilde{v} = f_B(\tilde{\lambda})$. In this point $\delta = f(\tilde{\lambda}) - \tilde{v}$ can be computed; if $\delta$ is smaller than a prescribed accuracy we can stop the procedure, having obtained an approximated optimum, otherwise the procedure continues with the computation of $f(\tilde{\lambda})$ and the corresponding subgradient that will be added to $B$. The last thing that needs to be done is the upgrade of the stability center, where we choose if we keep the current point we have (which is denotes as a Null Step) or we upgrade the point passing to it the newly obtained $\tilde{\lambda}$; this can be decided by checking the actual decrease as compared to $\delta$, i.e., $f(\tilde{\lambda}) - \tilde{v} \leq \kappa \delta$, where $\kappa \in (0,1)$ is a fixed tolerance. The approach is summarized in the following pseudocode.

**Step 1.** Initialize $\delta > 0$, $\kappa \in (0,1)$, $t$ or $l$, $\lambda$;

**Step 2.** solve $(\hat{D}_{B,t})$ or $(\hat{D}_{B,l})$ for $(\hat{v}, \hat{\lambda})$;

**Step 3.** if $\delta = f(\hat{\lambda}) - \hat{v} \leq \delta$ then STOP;

**Step 4.** solve $(P_\lambda)$ for $\hat{x}$, let $f(\hat{\lambda}) = c\hat{x} + \hat{\lambda}(b - A\hat{x})$;

**Step 5.** If $f(\hat{\lambda}) - \hat{v} \leq \kappa \delta$ then Serious Step (SS): $\hat{\lambda} = \tilde{\lambda}$; else Null Step (NS): $\hat{\lambda}$ unchanged;

**Step 6.** Set $B = B \cup \{\hat{x}\}$ and go to **Step 2.**

Bundle methods have been repeatedly proven to be more effective than the CP Method, see e.g. [30]. Like the CP method, Bundle methods can be even more effective if a disaggregated CP model is used as described in [5,3] see e.g. [9, 67].

## 5.5 Lagrangian Heuristics for UC

One important issue when using the above mentioned approaches is the achievement of good primal feasible solutions. In fact, as we have seen Lagrangian techniques solve a continuous relaxation of $(P)$, albeit (possibly) one of better quality than $(\bar{P})$; hence, they do not “naturally” provide integer feasible solutions. Integer solutions are indeed generated at each iteration, but they are typically not feasible w.r.t. the relaxed constraints $Ax = b$. Hence, Lagrangian methods typically provide multiple solutions that are infeasible for different reasons, but should not be expected to provide feasible ones. Such is the case in various different academic and real-life UC problems both, which when solved via the use of Lagrangian Decomposition don’t provide primal feasible solutions and as a result further steps need to be taken into action in order to regain primal feasibility. One way in order to overcome this issue is the use of Lagrangian Heuristics, in the following paragraphs we will proceed with presenting some successfully implemented Lagrangian Heuristics for academic UCs from the existing Literature and thereafter we will present a proposal for a Lagrangian Heuristic for the Edf UC.

### 5.5.1 Existing Lagrangian Heuristics

The starting point when constructing Lagrangian heuristics is the fact that the lagrangian relaxations as already mentioned in 5.2 provide some vital information that can be exploited and used as guidance such as in [10, 27, 55], where good primal feasible solutions are obtained. Interpreting things based on the UC formulation of the first chapter that refers to the basic academic UC version, a first very important remark is the fact that when the Dual Problem is solved by the Bundle, we have that at every iteration $k$ we obtain nonnegative multipliers $\theta$ at no extra cost such that:
CHAPTER 5. LAGRANGIAN APPROACHES FOR UC

\[
\begin{align*}
\sum_{b \in B} \theta^b = 1, & \quad \sum_{b \in B} \theta^h = 1 \\
u^i_t = \sum_{b \in B} \theta^b \cdot u^b_{i,t}, & \quad s^i_t = \sum_{b \in B} \theta^b \cdot s^b_{i,t} \\
\tilde{q}^j_t = \sum_{b \in B} \theta^h \cdot q^b_{j,t}
\end{align*}
\] (5.5.1.0.1)

Where we have that (5.5.1.0.1) describes the convexified solution which pretty rapidly becomes almost feasible with respect to the relaxed constraints. More specifically, the algorithms that were presented above form a set \( B \subset X \), and in every iteration produce multipliers \( \theta^*_x \) for every \( x \in B \) which define a point \( \hat{x} \in \text{conv}(X) \). Note that \( \hat{x} \) may or may not be feasible \( (A\hat{x} = b) \) depending on the algorithm chosen, but as general rule we can say that the violation of \( \|A\hat{x} - b\| \) is vanishing quickly, especially compared with the equivalent Lagrangian solutions. Using \( \hat{x} \) and multipliers \( \theta^*_x \) as source of primal information for our problems can provide us with some useful tools to quickly obtain good primal solutions. As a result is pretty straightforward to see that at every iteration of the bundle one has:

- \( \hat{x} \) current optimal value of Lang. multiplier
- \([\bar{p}, \bar{u}]\) optimal solution of LR \( \phi(\hat{x}) \) integer but primal infeasible
- \([\tilde{p}, \tilde{u}]\) convexified solution that can be obtained from the bundle information that is primal feasible but continuous

With the use of the above mentioned information there have been a number of different well tested ideas in the literature for constructing and implementing heuristic procedures that manage to obtain primal feasible solutions. Such as in [163] where rounding techniques in the continuous (almost) feasible solution \( \hat{x} \) are proposed in. Or in [56] where the use of multipliers \( \theta^*_x \) as probability distribution on the elements of the set \( B \) in order to combine the solutions \( x \in B \) and obtain primal feasible solutions is proposed. A case with big interest for Lagrangian heuristics is when the relaxation has a disaggregated structure and can as a result of this be decomposed in subproblems with different structure with only some of the components of \( \hat{x} \) to correspond to integer variables, a number of efficient Lagrangian heuristics for such case was proposed in [1][27], the main ideas and algorithms of which we will try to present in the following paragraphs.

Starting with the above as an input, the proposed procedure can be briefly summarized in the following three main steps followed by the authors in order to obtain good primal feasible solution \( \hat{u}, \hat{p}, \hat{q} \):

1. Consider the convexified hydro production \( \tilde{q}^j_t \) as fixed and decrease accordingly total power demand.
2. Greedy heuristic to set the commitmenent status \( \hat{u} \) of thermal units assuring satisfaction of remaining power demand.
3. Calculation of actual power production for the obtained feasible production schedule.

The greedy heuristic in order to produce feasible schedules can be summarized as follows. First thing to be done is the calculation of the residual demand for every time step \( t \), where one has to subtract from the original demand \( d_t \) the overall power production from the hydro units for
5.5. LAGRANGIAN HEURISTICS FOR UC

each time step \( t \) and thus remain with the specific residual demand \( \tilde{d}_t \). Then it needs to
be checked if the computed residual demand can be satisfied by the integer solution \( \tilde{u} \). For this
one has to check if \( \tilde{d}_t \in [\tilde{u}_t^-, \tilde{u}_t^+] \), where :

\[
\tilde{u}_t^- = \sum_{i \in I} p_{\min}^i \cdot \tilde{u}_t^i \quad \tilde{u}_t^+ = \sum_{i \in I} p_{\max}^i \cdot \tilde{u}_t^i
\]

In the case \( \tilde{d}_t < \tilde{u}_t^- \) then the time step is considered to be undercommitted, while in the case that
\( \tilde{d}_t > \tilde{u}_t^+ \) then the time step is considered to be overcommitted. Thus if any of these two cases
applies, then one has to change accordingly the schedule of \( \tilde{u} \) at the corresponding time step.

For this to be done a priority list is constructed to deal with the corresponding modifications
needed, constructed to guide the heuristic turning on/off the most "promising" units. The
priority list itself is concerned there are two different approaches proposed by the authors. In
the first one the priority list follows the convexified commitment status \( \tilde{u}_t^i \) (being considered
as the "probability" of unit \( i \) to be on during \( t \)). In the second one the ratio between \( \tilde{u}_t^i \) and
c\( e^t_i(p_t^i, \tilde{u}_t^i) \) is used as the probability for unit \( i \) to be on during \( t \).

Yet again in the above method the gradient constraints are not taken into account, potentially
resulting to infeasible schedules. As a result in order to avoid this, it is proposed to proceed
with the following modification of \( \tilde{u}_t^- \) and \( \tilde{u}_t^+ \), where for the first time step (i.e. \( t = 1 \)) things
are more straightforward due to the fact that the initial power output \( \hat{p}_0 \) is given by the UC
Data and thus we can proceed with:

\[
\begin{cases}
\tilde{u}_t^- = \sum_{i \in I} \max(p_{\min}^i, \hat{p}_{t-1} - \Delta_t^-) \cdot \tilde{u}_t^i \\
\tilde{u}_t^+ = \sum_{i \in I_{t-1}^+} \min(p_{\max}^i, \hat{p}_{t-1} + \Delta_t^+) \cdot \tilde{u}_t^i + \sum_{i \in I_{t-1}} \hat{p}_t \tilde{u}_t^i \\
where \ I_{t-1}^+ = \{ i \in I | \tilde{u}_t^i = 1 \}, I_{t-1}^- = \{ i \in I | \tilde{u}_t^i = 0 \}
\end{cases}
\]

Yet again because for the rest of the time step of the optimization horizon this is not possible,
due to the fact that the power outputs \( \hat{p}_{t-1} \), which are needed for the above formula to be
applied, are available only after the heuristic has been applied and after fixing the commitment
variables (for which \( \tilde{u}_t^- \) and \( \tilde{u}_t^+ \) are needed). As a result of this one has to come up with a way
to estimate \( \hat{p}_{t-1} \) for \( t > 1 \), and this can be achieved by solving an optimization problem of the
following form on for each \( t = \{1, \cdots, T\} \) and using the results obtain to apply the formula
above:

\[
\begin{cases}
\min \sum_{i \in \mathcal{P}_t^+} c_i^t(p_t^i) \\
\max(p_{\min}, \hat{p}_{t-1} - \Delta_t^-) \leq p_t^i \leq \min(p_{\max}, \hat{p}_{t-1} + \Delta_t^+), i \in I_{t}^+ \cap I_{t-1}^+ \\
p_{\min} \leq p_t^i \leq \hat{l}_i, \quad i \in I_{t}^+ \cap I_{t-1}^+ \\
\sum_{i \in \mathcal{P}_t^+} p_t^i = \tilde{d}_t
\end{cases}
\]

5.5.2 Expanding the Lagrangian Heuristics

However, all the above suggested heuristic approaches have been tested and proved to be effi-
cient in obtaining good primal feasible solutions on the simplified academic versions of the UC
problem, where the thermal units have relatively simple operational constraints (if compared
with those of some of the most complex units in EDF UC, and the hydro units are completely continuous. The latter in particular is crucial for the heuristic, because one can then directly use the "convexified" solution of hydro units to define a residual energy demand to be satisfied by the thermal units; these have much less inter-temporal binding constraints (as opposed to those of hydro units related to the total amount of water in the reservoir), and this greatly simplifies the task of developing heuristics, as these only have to take into account local information.

Unfortunately, extending these ideas from the classical UC formulations (with no nuclear units and continuous hydro units) to the much more complicated structure of the EDF case is quite challenging. In particular, the following issues clearly arise:

- Hydro units being no longer continuous, one cannot directly use the convexified hydro solution and simply remove them from the picture (least re-solving an Economic Dispatch once the commitment is found to perform that little last bit of optimization).

- Thermal units being much more complicated and varied than in the "simplified" academic version, the greedy heuristic should be adapted to each of the different unit types, which would likely make it rather complex in certain cases;

- Even if one would be willing to do the latter, each minor change in the model of the units might in principle require complex changes in the logic of the combinatorial heuristic.

To address all these issues, we plan to exploit the already-mentioned fact that MILP-based heuristics have recently become particularly successful in quickly finding very good-quality solutions to UC formulations (e.g. [65]). In particular, the idea is then to generalize the Lagrangian heuristic [27, 1] by finding ways to exploit the availability of both the convexified and the integer solutions that do not require unit-specific approach, but rather hinge solely on the availability of a MILP model for each unit and on some characteristic of the problem that is common to all units, e.g., the fact that decisions are spread along the same temporal horizon. Based on this we plan to develop a strategy based on the idea of dividing the problem in time, instead of solving one multi-period problem many smaller subproblems are solved, each for one or more time periods. Three different approaches are considered:

- **Forward approach**, where the subproblems are solved from the initial time period to the end, using the solution of the previous subproblem as input to the current one.

- **Backward approach**, where the subproblems are solved from the final time period to the beginning, in other words, it requires the backward window to extend all the way down to first time instant.

- **Mixed approach**, where some subproblems are solved in the forward approach and then the remaining ones are solve by the backward approach (in this case the backward window extends up until the last period solved by the forward approach).
5.5. LAGRANGIAN HEURISTICS FOR UC

Figure 5.1: Rolling horizon approach

In the above figure we display the form of the approach where it can be understood that the subproblems are composed from the amount of time steps that define their main window, from the amount of the time steps that define their forward window, and from the amount of time steps that define their backward window. Basically, the strategy works like a sliding window, with defined number of periods based on the main, backward and forward windows respectively. In this framework the solution is obtained by the the main and backward windows, with the first one setting the vector of integer variables $u$ (i.e. the integer part of the solution) and the latter one taking care of the power production output (i.e. the continuous part), since the commitment variables are set to be fixed. Moreover the forward window is used to check how myopic the obtained decisions are.

Moving on we proceed with presenting the way all the subproblems are formulating, by displaying their corresponding objective function:

$$\min \alpha \cdot \sum_{n \in W} \left( \frac{f^n(p^n)}{v^n} \right) + (1 - \alpha) \cdot \left[ \beta \cdot \left( \gamma \cdot \left\| \frac{p^n - \bar{p}^n}{p_{max} - p_{min}} \right\|_t + (1 - \gamma) \cdot \|u^n - \tilde{u}^n\|_t \right) \right]$$

Where (5.5.2.0.1) can be understood as a way to impose for the solution to remain near the convexified and L.R. integer infeasible solution and also it has to be noted that the values of all parts of obj. function are normalized to simplify the tuning. Finally let us briefly give an explanation of the above displayed terms:

- $\alpha, \beta, \gamma \in \{0, 1\}$: parameters for setting the level of the distance.
- $t \in \{1, 2\}$: define linear or quadratic proximal term
- $W$: set of all nodes in backward, main, forward window
- $v^n$: value of LR obj. function for node $n$
Chapter 6

A Structured Modeling System

In this chapter we will proceed with describing a new software tool, the Structured Modeling System++ (SMS++), which aims at closing this gap by providing well-structured modeling system capable of giving to the user the ability of efficiently experimenting and implementing "generic decomposition" approaches. That is providing general implementations of the base decomposition methods and reformulations, that can be easily applied to many different models (with the required structure) by just implementing the required Blocks. The modeling system has been developed from scratch as part of the work of the thesis in C++, and in the following chapter we will proceed with presenting its main elements that refer to a series of C++ Classes and we will also provide a demonstration of the implementation of SMS++ for modeling versions of the Academic UC that were discussed in chapter 2. Finally let us note that even though the main elements of SMS++ has already been developed and thus an implementation of the system exists and has been applied to test several Academic UC instances. Yet again our implementation of SMS++ for the Academic UC has not been compared with other modeling systems to test its overall efficiency and thus no results of performance comparison against other modeling systems will be presented below.

6.1 Motivations and Design

The development in both hardware and software has provided to researchers powerful general-purpose solvers for optimization problems; for instance, in the very well studied area of MILP the solution process has become around 100 million times faster over the last 20 years [134]. However, general-purpose solvers mostly ignore any existing structure in the model; therefore, in many applications specialized solvers exist that are better than general-purpose ones. These can be used e.g. in the context of decomposition methods, which need be used for tackling very-large-scale, hard optimization models such as the ones arising in energy optimization. Unfortunately, current modeling systems are mostly geared towards general-purpose solvers, and offer little support for developing decomposition approaches.

Moreover the advancement in the programming and mathematical framework progresses allow users to tackle efficiently problems that in the past were considered to difficult to be solved. This has allowed more realistic descriptions in the models but increased significantly the level of difficulty. Take for example the case of UC that as mentioned in 2 its academic simplified version
can be solved efficiently with the current existing tools. This gave raise to the construction of much more complicated and difficult to solve versions of the problem, which are much closer to the operational and physical reality of the problem. Such as the case of the EDF UC, which was described thoroughly in [2].

As a result within the current scheme of things one has the choice of using some general purpose solver that will be able to provide results without a lot of programming efforts but which could be potentially much poorer than some specialised tools developed to tackle the problem. On the other hand if following the choice of developing some specialised decompositions tools that exploit the structure of a specific problem, except from the fact of the overall extra programming and design effort, if these tools are so problem specific they probably will not be applicable to other variations of the same problem. For example in the case one proceeds with developing some specific decomposition and solution method for tackling one specific UC problem, in case the nature of some units is being somewhat changed, it can very well result to the case that an entirely new specialized subproblem solver and heuristics have to be developed from almost scratch. And in a more general context for many applications there exist specialized solvers that are better than general-purpose ones. Yet, because they are limited to smaller classes of models, these are often much less developed, and therefore less robust and user-friendly, than general-purpose ones. In addition, their efficiency may crucially depend on the appropriate setting of some obscure algorithmic parameters, which require an understanding of their inner workings that cannot be expected outside a small circle of specialists. On top of all this, models most often exhibit several structures simultaneously; not only it is not a fortiori clear which of them is computationally more relevant, but also the most efficient approach may require exploiting them all. This would call for integrating several different specialized approaches, a task most often bordering the impossible in the current state of affairs.

The above described situation makes it clear that a modeling system capable of giving to the user the ability of experimenting with different decomposition methods and reformulations (like Lagrangian relaxation [5], Benders decomposition [18], Dantzig-Wolfe Decomposition [5.5], etc.) would be a great tool in the hands of any modeller and researcher. In addition to this, the idea that the user after deciding and implementing the appropriate decomposition should be easily able to whether pass this model to a specialized solver (like a Bundle method, for instance) or to a general purpose solver in order to tackle the problem. The construction of such a modeling system would have as a result the combination of both the compactness and efficiency that a general purpose solver has to offer, together with all the elegant tools of decomposition that exploit as much as possible the specific problem’s structure. Under the above mentioned motivations we proceeded with designing and constructing our Structured Modeling System (SMS++).

Our primal targets with the development of SMS++ is to manage to provide a modeling language/system which will be able to support the developement of any kind of nested structure that can arise within mathematical models. That will allow to abstract away from details when developing algorithmic approaches (mostly decomposition ones), that is to separate "semantic" information (a unit) from "syntactic" one (list of constraints/variables). And also on top of the above being able to provide a a general interface that will efficiently cover the needs of decomposition methods (such as dynamic changes in problems, reoptimization). In order to develop such a challenging modeling system the first question that needs to be answered concerns the choice of the programming language. And in our case even-though in today’s programming reality exist well structured and efficient high-level languages (such as Python, Julia among
6.2 THE CORE ELEMENTS OF SMS++

others) that have been proven to perform pretty well and offer a reasonable alternative even against C++ and C, our programming language choice has been set to the very well known and efficient statically and very strongly typed C++. We should mention beforehand that it passed from our consideration and we even did examine the possibility to develop our system with a more higher-level language and most specifically some initial efforts have been made towards the direction of choosing Julia. But finally the fact that in order to make Julia compatible to work with C++ code was very difficult to be achieved made our decision final towards the construction of SMS++ under the framework of C++. Moreover we felt that the structure of C++ language itself that is a strongly typed class based language would fit perfectly our block-structured interface with its base and derived ingredients. Moreover it goes without saying that in the field of Combinatorial Optimisation the vast majority of Solvers have been developed and are still written in C++. And also on top of this the latest versions of the language (i.e. C++ 14/17) are equipped with a vast set of very useful and efficient tools that help significantly in the development of large-scale projects. And also there are some other significantly important tools for the languages developed out there (such as the very well known boost-libraries) that give ready for use within C++ almost all the programming tools of the higher-level languages and some of them are also included in our modeling system and will be further mentioned below.

6.2 The Core Elements of SMS++

Before going into the specific discussion of each single fundamental elements of SMS++ let us first proceed with a short description of the overall framework. So to begin with trying to break down what is displayed in 6.1 one has to start with the most fundamental concept of SMS++ is the representation of a mathematical model in a block-structured form that is composed by a set of basic generic abstract C++ classes. A model with a specific structure can be expressed via the Block class, which is displayed in the right side of the picture. Block is a base class for representing the general concept of "a part of a mathematical model with a well-understood identity"; in other words, it is expected that it will be used to define derived classes, each of which represents a model with a specific structure (say, a Knapsack, a Traveling Salesman Problem, a SemiDefinite program, ...). be represented as a Block. Thus, each Block will have information and data structures specific for the (part of) mathematical model it specifies; however, it also has an "abstract view" in which it describes the model through its Variables and Constraints, in the picture displayed these are $V^{C}, D_{C}, \ldots$ and $S^{V}, D_{V}, \ldots$ respectively which can be arranged in any number of "sets", or "groups", each of which can be a multi-dimensional array with (in principle) an arbitrary number of dimensions. Also each block can has a corresponding objective function (OF in the picture displayed. to have Each Block can be attached to any number of different general purpose or ad-hoc solvers (Solver), providing the interface between the current mathematical model and the corresponding algorithm. The Block supports modifications of all its components: each time a modification occurs, an appropriate Modification object is dispatched to all Solvers "interested" in the Block like the. Hence, the next time they are required to solve the Block they will know which modifications have occurred since the last time they have solved it (if any) and be able to react accordingly, hopefully re-optimizing to improve the efficiency of the solution process. Finally let us note for the picture displayed that in the left side we have the boxed representing the $C++Classes$ to which refer the different parts of the Block (such as Variables, Constraints etc) and the horizontal arrows just point out the specified elements of the Block of the right hand side to which they refer. Whereas the vertical
arrows display the relations that arise between the different elements of the Block.

![Diagram](image)

Figure 6.1: SMS++ Overview

In the following paragraphs we will give a compact description of the overall functionalities and tools of all the above mentioned fundamental ingredients of SMS++

### 6.2.1 Block

The abstract class Block as mentioned above is considered the cornerstone of SMS++ and is representing the basic concept of a "block" in a block-structured mathematical model. A Block contains some Variables, some Constraints, and some (inner) Blocks. Furthermore, it is (logically) contained into a father Block. Variables and Constraints can either be static (i.e., that never change along the life of the model) or dynamic (i.e., that may appear and disappear during the life of the model). The inner Blocks can individually appear and disappear (although the actual need for this feature is debated). A Block can be attached to any number of Solvers, that can then be used to solve the corresponding mathematical model. Moreover Block has a few initial parameters such as the level of verbosity, a parameter denoting if there are any Solvers attached to and also a feasibility tolerance that should be used as the numerical threshold that the Constraints of the Block use to declare whether or not they are satisfied. Also the interaction between the father and children Blocks is being made via a pointer that exists in each Block that is assigned with the address of its father Block if any.

Variables and Constraints in a Block can be arranged in any number of "sets", or "groups",...
6.2. THE CORE ELEMENTS OF SMS++

each of which can be a multi-dimensional array with (in principle) an arbitrary number of dimensions. The idea is that a model with a specific structure (say, a Knapsack, a Traveling Salesman Problem, a SemiDefinite program, ...) be represented as a specific derived class of Block. Hence, its Variables and Constraints will be organized into appropriate, "natural" (multi-dimensional) sparse or dense arrays, and will be accessed as such by specialized solvers that can exploit the specific structure of the Block. The Variables and Constraints can be represented implicitly by storing and providing the data that characterizes them (i.e. the weights and costs of the item in a knapsack, an annotated graph, the size of a square semidefinite matrix, ...), and specialized solvers will only need access to that data (characterizing the instance of the problem) to be able to solve the Block. We call this the "physical representation" of the Block.

However, Blocks can also be attached to general-purpose solvers that only need the Variables and Constraints to be of some specific type (say, single real numbers and linear functions ...). Hence, the base Block class provides a mechanism whereby, upon request, the Block "exhibits" its Constraints as an "unstructured" list of (multi-dimensional arrays of) Constraints: we call this the "abstract representation" of the Block. Note that, conversely, Variables need always be definite, as they are a crucial part of the interface between the user of the Block and the Solvers (the latter write their solutions into Variables, so that the former can read them). However, Block supports the notion that both (clearly defined sets of) Variables and Constraints can be dynamically generated (in the sense for Variables and Constraints to be generated or deleted after the begin of the optimization process where the Block has been passed to a Solver), in order to be able to cope with "very large descriptions of the model". Also note that there is no direct association between the different Variables of the Blocks.

A Block supports "lazy" modifications of all its components: each time a modification occurs, an appropriate Modification object is dispatched to all Solvers "interested" in the Block, i.e., either directly attached to the Block or attached to one of its ancestors. Hence, the next time they are required to solve the Block they will know which modifications have occurred since the last time they have solved it (if any) and be able to react accordingly, hopefully exploiting any warm/hotstarts to improve the efficiency of the solution process. Each Solver is only interested in the Modifications that occurred after it was (indirectly) attached to the Block and since the last time it solved the Block (if any), but it has the responsibility of cleaning up its list of Modifications.

On this point after having given a proper description of the Block and its main components we should proceed with making a reference in some important elements that concern the way things are stored and arranged within the Block. More specifically as already mentioned within each Block we expect to find one objective function, a series of Constraints and Variables and some nested Blocks as well. From all the above ingredients the one that need the more dedicated and significant care in order to be stored properly are the Constraints and Variables. As already discussed a Block can have different groups of Variables and Constraints, which themselves have one more categorization concerning whether or not they are static or dynamic (i.e. if they are expected to change their size or not with some new appearing or some of the being deleted). The distinction between dynamic and static Constraints and Variables has also some corresponding consequences on the way they are constructed in regards with their organisation inside the Block. More specifically the way the Block is able to address and map all its internal element is by the use of their (unique) memory address as their corresponding name. As a result every time a new Class of Constraints/Variables is constructed it has its unique memory address as its unique name for mapping it. In this point let us mention
that this is of great importance for the arrangement of things inside the Block and there is a very significant restriction that concerns this, that is that the user is not allow to copy and kind of Variables/Constraints but only passing them as pointers. This is pretty significant characteristic of the Block and does indeed affects the construction of its ingredients.

More specifically the first important programming choice concerning the above has to do with the way Static and Dynamic Variables and Constraints are constructed. Due to the fact that one of the main design decisions in SMS++ is that the “name” of a Variable or Constraint is just its memory address. This means that, once constructed, they cannot be moved to a different memory location. Hence, the difference between static and dynamic stuff is that only the former can live into std::vectors, provided that the arrays are never shortened or lengthened (for doing so may cause the memory to be re-allocated, which would change the address violating the basic assumption).

Moreover another significant thing regarding their construction is the fact that the use of objects or std::lists of objects stored within std::vectors limits the constructed groups of constraints/Variables until two-index objects. And as already seen is not rare to end up having formulations with 3-index variables (take for instance the case of EDF-UC described in chapter 4 where we had a series of 3- or even 4-index variables. In order to deal with this problem we proceed with using the efficient tools of boost::libraries and more specifically allocating such objects via boost::multi_array that allows to the user to create and handle multi-dimensional arrays of objects.

In order to find a proper way to manage to store and handle these potentially constructed elements. Our approach was based in the use of an other element found within the boost libraries, that is the boost::any, which allows one to store within any kind of different type of elements. So in our case we proceeded with constructing std::vectors of boost::any where inside are stored the different types of derived Variables/Constraint and thus we ended up with the four vectors for the different dynamic and static Constraints and Variables equivalently. The last thing that needed to be adressed and taken care is the way one could process and uncast these different type of elements within the above mentioned vectors. In order to deal with this some template functions for both the static and dynamic vector of constraints and variables have been created in order to automated the process of receiving, processing and uncasting the elements of these vectors. Yet again for the case of Dynamic Constraints/Variables the same does not apply, since within these classes we expect elements to be deleted or added, simply generating and storing them within std::vectors would seriously affect their address names. Conversely, the latter can only live into std::lists of objects and thereafter storing them in std::vectors, so that elements can be added or removed without causing other elements to be reallocated.

6.2.2 Variable

The base class Variable concerns all the possible type of Variables that a Block can support. Very few assumptions are made about what form the Variable actually has, this being requested from derived classes. In general we could say for a Variable that it belongs to one Block, on top of that it does influence a set of Constraints (those that are satisfied or not depending on the value of that Variable). But on this we have to note that the base class makes no provisions about how this set is stored in order to leave more freedom to derived classes to implement
it in specialized ways. Obviously a \texttt{Variable} has a value, but the methods for reading it must necessarily depend on the type of value it can take (i.e. binary, integer, continuous), and therefore must be provided by derived classes. Moreover the few functionalities that are attached within the base class concern the handling of the set of \texttt{Constraints} the \texttt{Variable} is affecting. More specifically having methods for adding/deleting \texttt{Constraints} attached to the \texttt{Variable}, as well with some more information about the fact if the \texttt{Variable} is still active in the attached \texttt{Constraints}, the number of them etc. On top of that a \texttt{Variable} can be (temporarily) fixed, i.e., becoming from a \texttt{Variable} a constant. Finally the \texttt{Variable} has a corresponding Modification class where is denoted the fixing of its value. Moving on now we proceed with its derived class of \texttt{ColVariable} intended as the base class for all the \texttt{Variable} that are single real values, possibly restricted to some subset (e.g., the integers). In a linear program this would correspond to a column in the coefficient matrix, whence the name. This class extends \texttt{Variable} to support the following further facts:

- A \texttt{ColVariable} has a value, which is a real number. More specifically, a type \texttt{VarValue} is defined, which is bound by default to doubles, to hold the type of the \texttt{Variable}. Changing this type here is possible, but this changes it to the whole SMS++ hierarchy, so this does not look too reasonable; if one really needs a different return value than double can rather re-define a similar class to this.

- Being a real value, a \texttt{ColVariable} can be restricted to live into an extended interval (have upper and lower bounds, possibly infinite);

- A \texttt{ColVariable} can be restricted to live into some "interesting" subsets of the reals, such as integers and binaries.

- Because a \texttt{ColVariable} has bounds, it is "both a \texttt{Variable} and a \texttt{Constraint}". This means that for many classes of problems, a \texttt{ColVariable} may have naturally attached "dual" information, which typically has the form of Lagrangian multipliers, i.e., two real values attached to the upper and the lower bound. Since typically at optimality at least one of the two is zero, this information is (as customary) condensed into a single* real value, which is the difference between the two, and that is usually known as the "reduced cost" of the \texttt{Variable}. This may be redundant in some cases, but if this really is a problem then a similar derived class from \texttt{Variable} lacking support for the dual information can be easily defined.

- There exist the derived class \texttt{ColVariableModification} that adds to the already existing modifications that are found in \texttt{VariableModification} the changes in the upper and lower bound on the feasible value of the \texttt{ColVariable}. And also include the case a \texttt{ColVariable} is able to change the subset of the reals in that the value is allowed to assume.

### 6.2.3 Constraint

The class of \texttt{Constraint} is a quite general base class of all the possible type of constraints that a \texttt{Block} can support. Very few assumptions are made about what form the constraint actually has, this being provided by derived classes. As a result the base class only supports a few fundamental facts:

- A \texttt{Constraint} belong to one \texttt{Block} (thus providing a pointer to the Block it belongs).
• A Constraint is influenced by a set of "active" Variables, i.e., these which contribute to it being satisfied (or not). Thus the base class provides some basic functionalities in order to track down the "active" Variables. Note that the base class makes no provisions about how this set is stored in order to leave more freedom to derived classes to implement it in specialized ways.

• A Constraint can be (temporarily) relaxed, i.e., not really constraining the set of solutions of the Block.

• A Constraint can be either satisfied or not, depending on the values that its "active" Variables currently have.

• A Constraint has a basic ConstraintModification class to communicate to its attached block and Solver the two above mentioned changes.

The first derived class from Constraint is the RowConstraint, which is intended as the base class for all the Constraints that are have a "row form", that is:

\[ \text{LHS} \leq (\text{some function from Variables to reals}) \leq \text{RHS} \]

where LHS and RHS are two extended reals, at least one of which is finite. More specifically, a type "RHSValue" is defined, which is bound by default to doubles, to hold the type of the LHS and RHS. Changing this type here is possible, but this changes it to the whole SMS++ hierarchy, so this does not look too reasonable; if one really needs a different return value than double can rather re-define a similar class to this. The above form encodes all possible kind of equality, inequality and ranged constraints. In this base class, no assumption is done upon the form of the function*: typical examples are linear functions, quadratic functions etc., but this is dealt with in derived classes. For many classes of problems this kind of Constraint naturally has "dual" information attached, which typically has the form of a Lagrangian multiplier, that is a single real value (in this case, again taken to be of type RHSValue). Support to this case is therefore offered by the class. This may be redundant in some cases, but if this really is a problem then a similar derived class from Constraint lacking the dual value can be easily defined. On top of this the basic ConstraintModification, other modifications are possible for this kind of Constraint, namely changing the LHS/RHS.

Finally the last derived class refers to the class LinearConstraint which derives from RowConstraint and implements a linear row constraints, that is, of the form

\[ \text{LHS} \leq (\text{Linear function from ColVariables to reals}) \leq \text{RHS} \]

where LHS and RHS are two extended reals, at least one of which is finite. The class has some extended features in order to efficiently express the linear function of the constraint that is subjected to the LHS and RHS. More specifically there is a corresponding std::vector of pair, with each element consisting from a pointer to a ColVariable Object and its corresponding coefficient (thus a double). The vector is kept sorted in an ascending order based on the pointers of the ColVariables. Then there are a set of methods for accessing, adding, deleting and modifying elements from the vector doing so in an as efficient way as possible (by having to do the minimum computational efforts to find the proper ColVariables and elements due to the fact that the vector is sorted). Finally the class has its corresponding LinearConstraintModification class to communicate the corresponding modification that can arise to its linear function.
6.2.4 Objective Function

The class `ObjectiveFunction` is the base class of all possible objective functions that a Block can support. This base class only supports a few fundamental facts:

- An `ObjectiveFunction` belongs to one Block (thus providing a pointer to the Block it belongs).
- An `ObjectiveFunction` is in general set to refer to a maximization or minimization problem.
- An `ObjectiveFunction` is influenced by a set of "active" Variables, i.e., those which contribute to it setting its value;
- An `ObjectiveFunction` can be evaluated to compute its value. Note that obviously an `ObjectiveFunction` has a value, which in general can be expected to be a single real value. However, this is not defined in this base class (e.g., in order to support multi-objective problems). Therefore, the methods for reading it must be supported by derived classes.
- Within a Block that contains other Blocks

- The corresponding class `ObjFunModification` is responsible for communicating the only possible modification of this base class, that is the modification of the definition of the objective function between a minimization and maximization one.
- Finally following the same fundamental design decision of Variables and Constraints, the unique "name" of an `ObjectiveFunction` is its own memory address. This means that in this case as well moving an object of `ObjectiveFunction` is not possible, since copying an `ObjectiveFunction` to a different memory location makes a distinct `ObjectiveFunction`. As a result the different `ObjectiveFunction` objects can only be passed with pointers.

Moreover there is the derived class `LinearObjectiveFunction` that is responsible for an objective linear function. The structure of the linear objective function is similar to the one of the linear function of the Linear Constraint. In this case as well we have that the corresponding std::vector of pair, with each element consisting from a pointer to a `ColVariable` Object and its corresponding coefficient (thus a double). the vector is kept sorted in an ascending order based on the pointers of the `ColVariables`. Then there are a set of methods for accessing, adding, deleting and modifying elements from the vector doing so in a much efficient way as possible (by having to do the minimum computational efforts to find the proper `ColVariables` and elements due to the fact that the vector is sorted). Moreover there is a corresponding method in order to evaluate the value of the objective value that is derived from the values of the corresponding vector of pairs. The class has its corresponding Modification class to communicate the corresponding modification that can arise to its linear objective function.

As far as the handling of the objective function is concerned when a Block holds also inner Block, the global objective function is considered to be substituted by the sum of all different objective function objects of all different Blocks, with the father Block being its starting point. Take for example the case of a Block that inside contains a series of inner Blocks of Units and a Network, the objective function is composed by the sum of the objective function of the inner unit Blocks (since the Network doesn't contribute to the objective function and thus has an
empty objective function). After this the only way that all the different parts of the objective functions can be accessed, evaluated and so on is by the father Block that contains them, with the inner Blocks being responsible only for their individual parts of the objective function.

Finally there the derived class \texttt{DQObjectiveFunction} that is responsible for a diagonal quadratic objective linear function that is an extension of the \texttt{LinearObjectiveFunction}. In this case there is an added std::vector of pairs in order to describe the diagonal quadratic part of the objective function. That is that for each quadratic element there is pointer to the corresponding \texttt{ColVariable} Object and its equivalent coefficient (thus a double). Likewise the previous vectors, this vector is kept sorted in an ascending order based on the pointers of the \texttt{ColVariables}. And of course there are a set of methods for adding, deleting and modifying elements from the vector doing so in an as much efficient way as possible (by having to do the minimum computational efforts to find the proper \texttt{ColVariables} and elements due to the fact that the vector is sorted). Moreover in this case the corresponding method in order to evaluate the value of the objective function takes into consideration both the linear vector of pairs and the quadratic one. Finally the class has its corresponding Modification class to communicate the corresponding modification that can arise to its quadratic part of the objective function.

### 6.2.5 Solver

The abstract \texttt{Solver} class is meant to represent the basic interface between a \texttt{Block}, representing (possibly together with its recursively nested sub-Blocks) a (block-structured mathematical) model, and any algorithm capable of "solving" it. This means (trying to) obtain the optimal solution. The \texttt{Solver} class is thought primarily (although not exclusively) for mathematical models in the form of an optimization problem; hence, the solutions should be either generically "good" ones, or, better, provably (approximately) optimal ones. This is provided an optimal solution exists at all: if the mathematical model is unbounded (below for a minimization problem, above for a maximization one), then the \texttt{Solver} should be able to detect it and, hopefully, "prove" it. Let us note here each Block that is passed to a \texttt{Solver} holds within all the proper information needed and the Solver only needs to provide a mechanism reading them. All Variables or Constraints associated with a Block have the necessary methods even if they are part of an other Block, they provide atleast the means to be accessed by any Block which they are defining. Take for example the case of a Block whose constraints and/or objective function are defined by variables outside the Block. Constraints and Objective Function as described in \ref{sec:6.2} they store a pointer of each Variable associated with them in a corresponding std::vector, thus if a Solver has access only to the specific inner Blocks is able to receive any information of any Variable that belongs to an other Block but is yet found within the Block under examination by this vector.

Let us know present the very general notions that the base \texttt{Solver} class supports:

- \textbf{A Solver} is attached to a \texttt{Block}, from which it picks all the data characterizing the model and that it uses to output any solution it finds (in the Variables of the \texttt{Block}).

- \textbf{A Solver} can (in fact, must) be specialized to work on a specific class of models, represented by one (or more) specific class(es) derived from \texttt{Block}.

- \textbf{A Solver} can (in fact, must) react to changes in the data of the \texttt{Block}, of which it is made aware by appropriate Modifications.
6.2. THE CORE ELEMENTS OF SMS++

- A **Solver** can (attempt at) solve the model, i.e., obtain at least one feasible solution (to within the required tolerance for each **Block**), that is also provably optimal to within the required accuracy. Upon termination it will provide a full account about how the solution attempt went.

- A **Solver** may be able to provide multiple solutions to the problem for each call of `solve()` be them feasible or unfeasible ones, in an appropriate order (from the "more interesting" to the "less interesting").

- A **Solver** can return some upper and lower bound of the obtained solutions, based on the solution status, via the corresponding methods `get_ub()` and `get_lb()`.

- A **Solver** can set the min/max objective value of the solution (for the unbounded case), via the corresponding method `set_unbounded_threshold()`.

In order to manage to fully describe the functionalities and purposes of the class we need to look more into some of the main methods and elements of the class. First thing is the method `solve()` that is expected to be used every time the user wants to solve the optimization problem described in the constructed **Block**. The method is expected to return one of the following information:

- optimal solution found
- the model is provably unbounded
- the model is provably infeasible
- some solution found but not provably optimal
- stopped because of time limit
- stopped because of iteration limit
- stopped because of unrecoverable error

On top of this the user when solving the **Block** via this function is able to set the following parametric choices for the optimization process:

- set maximum time for the next call to `solve()`
- set maximum number of iterations for the next call to `solve()`
- set "verbosity" of the log
- set relative accuracy for declaring a solution optimal
- set absolute accuracy for declaring a solution optimal
- set upper cutoff for stopping the algorithm
- set lower cutoff for stopping the algorithm
- set maximum number of different solutions to report
- set maximum relative error in any reported solution
- set maximum absolute error in any reported solution
- set maximum constraint violation in any reported solution

It should be mentioned that the Solver class is in general equipped in order to provide to the user the ability to be able to produce other new solutions, in case this is possible. This is achieved via firstly using the method `new_var_solution()`, which more specifically is able to signal whether a solution that is different from the previously reported one (if any) is available. This method returns a boolean variable, where in the case it returns false it means that the Solver has no means of producing any (other) solution for good, so it does not make sense to call it again unless something has changed in the problem encoded in the Block or, in case the last call to `solve()` has stopped due to time or iteration limitation and now `solve()` is called again. For a typical algorithm that only produces one solution, for instance, this method should return true right after a call to `solve()` that returned an optimal solution, and false after each subsequent call (prior to any modification to the problem that makes it possible to call sensibly `solve()` again). This method should in general return false if `solve()` has not been called yet, although some Solvers may be able to easily generate solutions even before having done almost anything. This method must surely return false if no Block is attached to this Solver.

Each time `new_var_solution()` is called, the Solver must return true only if a solution that is different from the previous one (if any) is available; of course, for the first call this only mean that a solution is available. The user can restrict the set of generated solutions, both in number and in terms of their quality or feasibility. In particular, note that the Solver may be instructed to also return infeasible solutions if it has produced any (which may be the case if, for instance, the models is infeasible, and therefore no feasible solution can ever be found). This means in particular that it is in principle sensible to call this method even if `solve()` has returned a value not necessarily implying that a solution has been found, and even that it is guaranteed that no solution exists.

After a call to `new_var_solution()` has returned true, then the user can call the method `get_var_solution()` in order to have the solution actually written in the Variables of the Block. The previous value in the Variables is lost for good, and the Solver class does not provide any mechanism to retrieve it at a later time: only one solution at a time can be stored in the Variables. Note that the base Solver class makes no assumption about how the solution is stored, this being dependent on what the Variables actually are. It is an error to call this method if `new_var_solution()` has not been called and has returned true (which means, in particular, if no Block is attached to this Solver). Also, note that, while the largest burden of producing a solution is typically bore by `solve()` and/or `new_var_solution()`, it is still possible that "decoding" the internal information of the Solver in order to produce one in the format that the Variables of the Block require may be somewhat costly.

Moreover, all in all despite trying to be quite general, the class is clearly somewhat bent towards single-objective, real-valued optimization problems. Hence, although not strictly necessary, it is somewhat intended that the ObjectiveFunction of the model actually is a RealObjectiveFunction. Steps are taken so that problems without an objective function should not find this choice "too intrusive"; basically, every method having to do with the (real) objective function value is given here a safe default implementation so that it can be basically ignored by classes not having that concept (say, because they have no objective function) at all.
6.2.6 Modification

The class Modification is the top of the hierarchy of the different modifications (derived classes) that describe possible changes in the various pieces of a SMS++ model (in the classes derived from Block, Constraint, Variable, ...), that a Solver needs to be made aware of. It is almost an empty base class with only common aspect to all Modifications is that all can be printed, since everything else is being taken care in the derived classes (Which have already being mentioned above). Each change to any of the ingredients of the SMS++ (Constraint, Variable, etc) produces a Modification and from that point onwards a smart pointer to an object of (a derived class of) class Modification will be created by some object (Block, Constraint, Variable, ObjectiveFunction, ...) and it is passed to its father Block for being dispatched to all Solvers attached to it, as well as to all Solvers attached to its ancestors (the father, the father of the father, ...). This being a shared pointer, the original Modification object is automatically destroyed after that all Solvers have deleted it.

Thereafter the Solvers that receive them are supposed to delete each Modification object it receives as soon as it has finished to process the information they contain. Because the pointer is a shared one, this will result in the actual Modification object to be deleted when all the Solvers concerned with the same Block (these either attached to the Block or to some of its ancestors) have finished processing it. Modification objects can have some nontrivial amount of information attached to them that only gets deleted then. Hence, it is crucial that Solvers do actually delete the Modifications. Each Solver has the responsibility of cleaning up its list of Modifications, and since we have to do with smart pointers, there is a certainty that in the end after all the Solvers have processed the Modification the memory will finally be released.

6.3 An SMS++ Application for the UC

In the rest of the chapter we will try to describe and briefly demonstrate the way we implemented SMS++, whose main elements we briefly discussed above, in order to construct and solve the UC Academic Versions that were described in 3.2 and 3.3. Let us start with displaying a graphical representation of the overall process:
As it can be seen in the figure 6.2, the SMS++ implementation for the UC can be described as follows. The main class that represents the overall academic UC and can be considered as the "father" Block is **UCBlock**. Within this class is found the time-horizon of the problem $T$ and a series of inner Blocks that describe the units and the network. The network of the Academic UC is described via the class **BusNetworkBlock** and within the class can be found the vectors of size $T$ that refer to the demand ($d[T]$) and the spinning reserve ($r[T]$) together with the corresponding constraints. Note that these constraints are also associated with the variable $U[T]$ and $P[T]$ of the Thermal units. Moreover, it should be mentioned that the class derives from **NetworkBlock** that is a more general class from which can potentially derive other type of networks for different versions of the problem as well.

The units of the Academic UC are described via the class **AcadThermalBlock** where within the class are stored the corresponding $U[T]$ and $P[T]$ and on top of this within are stored all the necessary information needed in order to create the constraints, objective function and variables that refer to all the different formulations that were described in 3.3. There exists also a parameter for the user to choose the formulation of its choice. The class derives from **UnitBlock**, that is a more general class holding only information for the variables $U[T]$ and $P[T]$ and thus can be used to derive different type of thermal units, such as Edf thermal units, nuclear units and so on. **AcadThermalBlock** is attached to three different solvers, where 1UCDPsolver$_1$ and 1UCDPsolver$_2$ refer to the the DP-approaches described in 3.2.2 & 3.2.4 and MILPSolver is a class designed to receive any block and pass it to CPLEX and then write the retrieved solution back to the corresponding Blocks.

Finally the overall Academic UC that is expressed by the inner blocks described above of **UCBlock** is also solved via CPLEX through the use of the class **MILPSolver**. On this point in order to make the overall implementation process more clear to the interested reader, let us proceed with
presenting some actual blocks of codes, that refer to some specific methods and parts of the implementation code used for the construction of the UCBlock. Let us start with demonstrating the main.cpp, where is demonstrated how one can use the UCBlock and MILPSolver class functionalities in order to construct and solve an Academic UC within a few lines of code:

```cpp
#include "UCBlock.h" // header file of class UCBlock
#include "MILPSolver.h" // header file of class MILPSolver
...
int main()
{
    ifstream inStream (...); // initialize and pass
    ... // input data of UC
    UCBlock AcadUCBlock; // initialize UCBlock object
    AcadUCBlock.instance(inStream);// passing instance to UCBlock
    for(auto inner_block : AcadUCBlock.get_nested_Blocks())
        inner_block->generate_static_constraints();
    // generate static constraints for all different Units and the Network of the UC

    MILPSolver CPLEXSolver;
    //initialize the object of the MILPSolver that in this case is CPLEX
    CPLEXSolver.set_Block( &AcadUCBlock);
    //pass the Academic UC Problem to CPLEX
    CPLEXSolver.set_par(Solver::kMaxTime,3600.0);
    // set the time limit equal to 1 hour and the LP Solver for root node
    CPLEXSolver.solve();
    // set CPLEX to solve the attached UC
}
```

Figure 6.3: main.cpp

In Figure 6.3 the main.cpp demonstrates how the Academic UC is initialized and constructed via SMS++ and then solved via CPLEX again via the use of SMS++ classes. More specifically after the header files of the SMS++ classes that will be used have been added, the first thing to be done is to load the input data files of the corresponding UC (lines 6 & 7). Then a UCBlock object is created (i.e. AcadUCBlock) and thereafter the input data are loaded to the object with the use of method instance(). Within this method the UCBlock manages to initialize all inner Blocks for all different Units and the Network of the examined problem and store within them the corresponding Variables $U$, $P$ and the rest of the problem’s data. After the successful call of method instance() of the UCBlock, all the constructed Unit and Network Blocks are initialized and stored within the corresponding std::vector of the class UCBlock that is responsible for holding the inner blocks (named v_Block).

Thereafter since the problem is expected to be solved by CPLEX the different Blocks of Units and of the Network are expected to create the constraints that characterize them, so that CPLEX can read all the Constraints and Variables and create the corresponding problem. This is achieved by calling the method generate_static_constraints() for all different inner Blocks of UCBlock in lines 10 & 11 (i.e. constraints of the thermal units and global constraints of the network). Within this method all corresponding Units and Network Block are expected to proceed with the needed processes in order to created the equivalent Constraint objects and store them within their structures so that the Solver can read them. Note that in this case we know that all Constraints and Variables are expected not to change in size during the optimisation problems (i.e. we don’t expect new constraints or variables to be added to the UC)
and thus all Variables and Constraints within the classes are static.

After UCBlock has been successfully initialized together with all its inner Blocks the MILPSolver class that represents CPLEX Solver is initialized and there after the UCBlock is passed to the Solver via the method set_Block(). This method is responsible to read all the data of the attached Block and translate them to the corresponding CPLEX structure. Moreover some parameters of the Solver regarding the maximum time limit and the selection of the LP Solver for the root node are tuned with the use of method set_par(). The method can be used to tune any CPLEX parameter. Finally the problem is tackled via the use of method solve(), which is responsible for solving the problem attached, also writing the corresponding CPLEX-version of the problem to an LP file (i.e. the coefficient matrix, the objective function and the bound of the Variables as these have been translated from the Block structures to CPLEX notation) and on top of that storing the status of the solution and writing back to Block the obtained solution (i.e. setting the Variables to the obtained values and the objective function as well), in case one exists.

Let us now proceed with demonstrating how the UCBlock class is constructing all its inner Blocks and elements via the method instance():

```cpp
#include "UCBlock.h"  //include of header files of needed derived classes
#include "UnitBlock.h"
#include "NetWorkBlock.h"
... //other includes and methods of class

void UCBlock::instance(std::istream& inStream) {

std::string thermal, network;

inStream >> t >> units_size >> thermal >> network;

v_Block.resize( units_size + 1 );
v_Units.resize( units_size );

for( int i = 0 ; i < units_size ; i++) {
    v_Units[ i ] = UnitBlock::U_factory()[thermal](this);
    // set the corresponding object via factory
    v_Units[ i ]->load( inStream );
    // pass the data to the corresponding thermal units
    v_Block[ i ] = v_Units[ i ];
    // add each thermal unit Block to the vector of nested Blocks
}

Network = NetWorkBlock::f_factory()[network](this);
// set the corresponding object via factory
Network->load( inStream );
// pass the data to the corresponding Network Block
v_Block[ units_size ] = Network;
// add the Network to the vector of nested Blocks
UC_of.set_type(ObjectiveFunction::eMin);
// set type of objective function
set_objective_function(UC_of);
// add the objective function to the UCBlock
}
```
The Figure 6.4 demonstrates the method instance() of UCBlock class that is responsible of initializing the elements of the class based on the UC data given. First thing to be done is passing to the variable of the class \( t \) the value of the data that refers to the number of the time steps of the optimization horizon, thereafter the variable of the class unit_size is assigned with the corresponding value of the data. Moreover in the local strings thermal and network are passed the names of the type of the network and the units (i.e. in this case BusNetwork and Academic ThermalUnit) that will be used below. Thereafter the vector of inner Blocks \( \text{v\_Block} \) is resized in order to be able to store all the different Blocks of the UC (i.e. thermal units and network) and on top of the vector of the class \( \text{v\_Units} \) that is referring to the Units of the UC and is of type \( \text{std::vector<UnitBlock \*}> \) is also resized accordingly.

Moving on, in lines 15-22 all Units of the UC are initialized and stored in the corresponding vector \( \text{v\_Units} \), which holds all the inner blocks of UCBlock. In line 16 the units are initialized via a factory. More specifically the factory is defined as static map where are stored all the different possible derived classes that are linked with a unique string. In this way the factory works as a registration mechanism, where the user for each new different type of Unit that he wants to create constructing a corresponding class deriving from UnitBlock needs to register it to the factory together with a unique name passed as a string. The unique names are expected to be found in the input data and then stored in the local string thermal that will be used from the factory in order to construct the derived UnitBlock class that corresponds to the name stored in string thermal. Note that for the Network a Factory that is defined and works in an identical way is used as well.

For our Academic UC implementation it has been constructed the class AcadThermalUnitBlock, which inside holds all the data and structures needed in order to create the different MILP formulations of the Academic Units that were discussed in ???. Thus, after each AcadThermalUnitBlock is initialized via the factory, in line 18 the method load() is used, which is responsible to retrieve the corresponding UC Data and populate all the fields of the class accordingly. Finally in line 27 the object is stored in the last position of \( \text{v\_Block} \) (since in all the previous positions lie the objects of the Units).

Moving on a similar process is followed for Network, which is defined as a pointer of a NetworkBlock object within the class of UCBlock. Thus in line 23 a factory that exhibits the same characteristics with the one described above for the units is used. In the case of the Academic UC of 3.3 the network refers to the simple case of a Bus Network and thus via the factory the corresponding BusNetworkBlock is initialized. Moreover in line 25 the method load() is used, which is responsible to retrieve the corresponding UC Data and populate all the fields of the class accordingly. And thereafter the object is stored in \( \text{v\_Block} \).

Last thing to be done is the initialization of the objective function, where we have \( \text{UC\_of} \) to be a field of UCBlock defined as an object of type ObjectiveFunction. And we set its type to a minimization problem in line 30 via the use of the method set_type() and thereafter add the objective function to the block via the method set_objective_function(). Let us note here that the actual type and form of the objective function will be defined by the corresponding inner blocks of the Units (since the Network doesn’t contribute to the objective function). And thus in our case the AcadThermalUnitBlocks are assigned with their own objective function that takes whether a linear or diagonal quadratic form based on the problem one has to solved. The latter choice is made by an appropriate parameter that can be set accordingly by the user. Finally it should be mentioned that the current arrangement of the code implies, that any user interested in using the UCBlock in order to define a UC Problem and then derive some new
classes from UnitBlock and NetworkBlock in order to describe the specific UC Problem that wants to tackle. It needs to provide the UC data input in an appropriate form as shown above, that is provide the time horizon, the number of the units and the unique names of the types of the units and the network (with this specific order). Otherwise it is necessary to proceed with constructing a derived class where the method for reading the input data will be designed accordingly.

After describing how UCBlock is arranging its elements and inner blocks, let us proceed with demonstrating how the above process continues in the constructor UnitBlock, where each unit’s commitment and power variables are stored:

```cpp
UnitBlock::UnitBlock(UCBlock * fbblock) : Block(fbblock) {
    UCBlock * f_UC_Block = dynamic_cast<UCBlock*>(get_f_Block());
    if (f_UC_Block) {
        U.resize(f_UC_Block->get_t());
        P.resize(f_UC_Block->get_t());

        for (int i = 0; i < f_UC_Block->get_t(); i++) {
            U[i].set_Block(this);
            P[i].set_Block(this);
            // set pointer of the father Block
            U[i].set_lb(0.0);
            U[i].set_ub(1.0);
            P[i].set_lb(0.0);
            P[i].set_ub(Inf<double>(0));
            // set lower and upper bounds
            U[i].set_type(ColVariable::binary);
            P[i].set_type(ColVariable::continuous);
            // set type of variables
        }
    }
}
```

Figure 6.5: UnitBlock Constructor

In Figure 6.5 it is shown the constructor of UnitBlock, where is demonstrated the initialization and addition to UCBlock of the commitment and production variables of each Unit. More specifically the first thing to be done is to dynamically cast the pointer of the father block that refers to the UCBlock, since in its block the pointer to its father block is of the base class type Block. Moreover after checking that the pointer is not empty the two std::vectors of type ColVariable U[T] and P[T], that are both fields of the class and refer to the commitment and production variables are being resized to the number of time steps of the time horizon of the UCBlock, that is stored in the field of the UCBlock t and can be accessed outside its class by the corresponding method get_t().
Thereafter for the \texttt{ColVariable} Objects of vectors $U[T]$ and $P[T]$ we proceed with setting the current Block as their father Block (via the use of \texttt{set\_Block(this)}), setting their corresponding upper and lower bounds (via the methods \texttt{set\_ub()} and \texttt{set\_lb()}), and also setting their type via the corresponding method \texttt{set\_type()}. Finally after all their corresponding information have been set properly, we proceed with adding them to the corresponding vector of static variables of \texttt{UnitBlock}. As last part of our demonstration of actual part of code of SMS++ we proceed with presenting the \texttt{generate\_static\_constraints()}, where the global demand and the spinning reserve constraint as described for the Academic UC of need to be generated and stored in the block:

```cpp
void BusNetworkBlock::generate_static_constraints() {
    UCBlock * f_UC_Block = dynamic_cast<UCBlock*>(get_f_Block());
    if (f_UC_Block){
        Demand_Const.resize( f_UC_Block->get_t() );
        Reserve_Const.resize( f_UC_Block->get_t() );

        for( int t = 0 ; t < f_UC_Block->get_t() ; t++ ) {
            // initialize the vector of pairs of the variables and coefficients
            // of the corresponding constraints
            LinearConstraint::v_coeff_pair *v_demand_pair =
                new LinearConstraint::v_coeff_pair( f_UC_Block->get_units_size() );
            LinearConstraint::v_coeff_pair *v_reserve_pair =
                new LinearConstraint::v_coeff_pair( f_UC_Block->get_units_size() );
            LinearConstraint::v_coeff_pair::iterator it;

            for( int i = 0 ; i < f_UC_Block->get_units_size() ; i++ ) {
                it = v_demand_pair->begin() + i;
                *it = LinearConstraint::coeff_pair( f_UC_Block->get_units()[i]->get_P(t), 1.0);
                // store p_it of the Sum
                AcadThermalUnitBlock * Acad_Thermal_Unit =
                    dynamic_cast<AcadThermalUnitBlock*>(f_UC_Block->get_units()[i]);
                it = v_reserve_pair->begin() + i;
                *it = LinearConstraint::coeff_pair( f_UC_Block->get_units()[i]->get_U(t),
                    Acad_Thermal_Unit->get_fMaxPower() );
                // store p_it of the Sum
            }

            Demand_Const[t].add_variables(v_demand_pair , false );
            Demand_Const[t].set_Block(this);
            Demand_Const[t].set_lhs(d[t]);
            Demand_Const[t].set_rhs(d[t]);

            Reserve_Const[t].add_variables(v_reserve_pair , false );
            Reserve_Const[t].set_Block(this);
            Reserve_Const[t].set_lhs(r[t]);
            Reserve_Const[t].set_rhs(Inf<double>());
        }
        // Adding them to the Static Constraints Vector
        add_static_constraint( Demand_Const );
        add_static_constraint( Reserve_Const );
    }
}
```
Here as well the first thing to be done is to dynamically cast the pointer of the father block that refers to the UCBlock and proceed with the corresponding check if the pointer is empty. Moreover the two std::vectors of type LinearConstraint Demand_Const and Reserve_Const that are fields of the class and refer to Demand and Spinning Reserve Constraint respectively are resized to the size of the time-horizon of the UCBlock. The next thing to be done is to fill properly for each Constraint the std::vector<std::pair< Colvariable *,double >> which stores the linear function of each constraints (and was described in 6.2.3) Note here that LinearConstraint::v_coeff_pair refers to the type of vector of pair and also LinearConstraint::coeff_pair refers to the pair of Colvariable and double and both have been defined properly in the corresponding header file of LinearConstraint class.

In lines 12-16 we allocate to the heap —since they will be used outside this local method— vectors of pairs v_demand_pair and v_reserv_pair for each one of the Demand and Spinning Reserve Constraints a respectively, setting their size equal to the size of the units. After the successful creation of these vectors we proceed with filling them within the for-loop of lines 18-30. More specifically in order to access each specific needed u,p variables, we first use the method get_units()[ i ] of the UCBlock (via the corresponding pointer f_UC_Block of the UCBlock), which returns the i-th element of the vector v_Units, where all the Units of the problem are stored. Once we get the i-th Unit of the UCBlock vector, we need for each of them to use the methods get_U(t) and get_P(t), which return the corresponding commitment and power variable that is in the position t in the std::vectors of type ColVariable U and P respectively (which were presented in Figure 6.5). Thus to conclude calling the above methods f_UC_Block->get_units()[ i ]->get_U(t) and f_UC_Block->get_units()[ i ]->get_P(t) returns the variable u,i,t and p,i,t respectively. As a result for each demand and reserve constraint t we create for each unit i one std::pair<u,i,t> in lines 20-21 and one std::pair<p,i,t> in lines 26-28 respectively. Note that in order to manage to retrieve the value of p,i, which is stored in the derived class AcadThermalUnitBlock of each unit, we need to use f_UC_Block->get_units()[ i ] in order to get the i-th Unit and then we need to dynamically cast it to the derived class of AcadThermalUnitBlock since the v_Units vector stores units of UnitBlock class. Then in order to access the corresponding p,i we need to call the method get_fMaxPower() which returns the value of maximum power output of the unit i (i.e. p,i).

Upon the completion of the iterative process of constructing and filling for each time step t the corresponding vector of pairs v_demand_pair and v_reserv_pair, we then proceed with adding them to the corresponding constraints Demand_Const[t] and Reserve_Const[t] via the use of method add_variables which receives them as input together with a true or false boolean to denote if the corresponding vectors are stored in an ascending order based on their memory address, which information is needed since the constraints need to keep their vectors ordered in an ascending order. Thereafter the Constraints set as father Block the current BusNetworkBlock with the use of method set_Block and finally set the values of the lhs and rhs, where d[T] and r[T] as also shown and discussed in 6.3 hold the values for the demand and the spinning reserve respectively. Finally after the Constraints have been properly filled for all different time-steps are then in lines 42 and 43 they are added to the corresponding vector of static variables of BusNetworkBlock.
Chapter 7

Conclusions

In this thesis we tried to proceed with examining and producing efficient tools and mechanisms that lie within the area of Decomposition Techniques for Combinatorial Optimization in order to tackle real-life large scale energy problems. As a result we started with Chapter 2 where we tried to give an overall representation of the main ingredients of the problem under examination (that is the UC of course) with providing the reader a basic overview of the existing literature on the problem itself. And on top of this with providing also an overview of all the main solution techniques that have been used throughout the year in order to tackle the UC.

After this point and while hopefully the reader have taken a basic understanding of the problem itself and the existing methods out there for tackling it. We proceeded with Chapter 3 where a throughout examination of the thermal unit commitment problem has been made. More particular in the first part of the chapter we focused in presenting the main work that has been done the last years on the field of polyhedral analysis and in creating as compact and tight formulations as possible. Something that has clearly brought an improvement of the overall solution process of the examined problem. Moreover and we also proceeded with discussing some already existing in the literature efficient ways in order to tackle the thermal UC subproblems. Two of these DP algorithms, namely the one presented in chapter 3.2.2 and the one presented 3.2.4 have been implemented and tested in this thesis with the use SMS++ with the latter one based on the results presented being potentially more efficient when one has to tackle larger versions of the problem. On top of the above in the last part of the Chapter 3.3 one already existing MILP formulation 3.3.1 for the thermal 1UC inspired by the DP-algorithms which has been proven elsewhere to describe its convex-hull was presented, together with one more similar MILP formulation that was developed as part of our work in 3.3.2 where again both were implemented and tested for their efficiency between each other and against some standard UC formulation (with $u, p$ variables), and in all the cases proved to be better than the standard UC formulation.

In Chapter 4 we proceeded with presenting and examining a real-life industrial version of the Unit Commitment Problem, the EDF UC. This is something of particular interest since in the best of our knowledge this has been the first time that a MILP formulation has been discussed and put into existence for both the hydro and thermal units of the EDF. As it was displayed in the introduction of the chapter this realistic version of the UC is significantly way larger than the simplified academic UC, for which as presented in the two previous chapters already exist a lot of studies in the literature, which have managed to produce efficient results within reasonable time-frameworks. Yet again the EDF UC is a rather challenging problem, for which not so much previous study has been made, especially in respect with its thermal part. What has been of
particular interest and importance for us was the introduction of the third and fourth index in the commitment variables and the corresponding different proposed formulations that give us more confidence that we will succeed in modeling the problem in an as much efficient way as possible.

Thereafter we proceeded with Chapter 5 where in the first part we went on with a brief discussion of the main theoretical elements that concern the methods we are using for tackling the problems under examination. This discussion is somewhat important in itself not only because it gives a proper understanding of the nature and the context of the methods used. But also because some of the elements discussed provide the groundfloor for the Lagrangian Heuristics that are discussed in the second part of the chapter in 5.5. The Lagrangian Heuristics are of great importance for our work since they constitute the efficient means that produces primal feasible solutions to our examined problems. The most interesting and significant part is the proposed Lagrangian Heuristic in [5.5.2] which extents the already existing and efficiently used Lagrangian Heuristics that exist in the literature.

Moving on in Chapter 6 we presented SMS++ our Structured Modeling System, which seeks to provide an efficient tool in the hand of the interested users in order to construct and work around with general decomposition tools and mechanisms. We want to believe that the discussion of the main ingredients of SMS++ in 6.2 gave an insight of the capabilities of our modeling system and a basic idea of the possible ways it could be further extended to cover a very vast majority of well-known decomposition schemes of Combinatorial Optimization. Furthermore in 6.3 we proceeded with presenting a basic implementation of the SMS++ for the academic version of UC, which we hope to have managed to provided a sufficient example of how the above described tools can actually be used to efficiently describe and solve hard to solve optimization problems. Finally since we feel that our above presented work of this thesis has managed to present some interesting work which was described in the chapters above, but in the same time opened a lot of new and interesting fields of further examination and research. We want to finish and conclude this final chapter with some directions for future work and research that derive from our work:

1. The different UC-formulations for the EDF UC presented in chapter 4 should be put in intensive computational examinations to test their efficiency and also to explore further more tight formulations on them.

2. Looking closer to the thermal part of the proposed formulations for the EDF UC we feel that potentially some work could be done in order to derive with some efficiently DP-approaches for tackling the independent thermal sub-problems. The 3 and 4 index of the commitment variables offer a promising groundfloor for discretizing things and come up with some efficient state-spaces. Note that the approaches used in 3.2.2 and 3.2.4 might work as some initial point of thought for being further somehow extented to fit the EDF thermal-subproblems.

3. The proposed Lagrangian Heuristic in 5.5.2 needs as well to be put in extensive computational examination with the EDF UC being the perfect candidate for it.

4. As for SMS++ clearly a lot of future work and research is considered to be in front of our modeling system as well. Our research interest currently focuses in constructing some efficient tools in order to include some very well known decomposition methods to our arsenal (such as derived classes of LagrangianSolver and LagrangianHeuristics). But also in more programming aspects such as providing some mechanisms for parallel computing and serialization of the produced instances.
Bibliography


