UCLouvain

École polytechnique de Louvain

Comparison between Row Generation, Column Generation, and Column-and-Row Generation for Computing Convex Hull Prices in Day-Ahead Electricity Markets

Author: **Maxime Paquet**
Supervisors: **Mehdi Madani, Anthony Papavasiliou**
Reader: **Julien Hendrickx**
Academic year 2020–2021
Master [120] in Mathematical Engineering

Abstract

Due to the non-convexities in the day-ahead electricity markets, it can be shown that there are no uniform prices, i.e. there will be side payments for the participants to compensate losses and missed opportunities. The pricing policy minimizing those side-payments aims at computing the convex hull prices. To compute such prices one requires the convex hull of the feasibility constraints describing each generator involved in the electricity market. The convex hull is modelled through an extended formulation introducing more variables and constraints than the initial mixed-integer optimization program modelling the market. Therefore the extended formulation is a large linear optimization program. Instead of simply solving this large optimization program, different improved methods: row generation, column generation, and column-and-row generation. They can be used to provide CHP and improve the initial time performances. The purpose of this master thesis is to compare those improved methods. Each of the following methods, the extended formulation, row generation, column generation and column-and-row generation, is separately described to understand its basic principles and its time performances on one real-world example. Once each method is understood, they are compared based on four different data sets with different inputs, i.e. the number of generators and the length of the time horizon.

Acknowledgements

I would like to express my gratitude to my supervisors Anthony Papavasiliou and Mehdi Madani. They both supported me and guided me all along the thesis. They also provide valuable advices through the year in order to improve my work.

I also would like to thank particularly Julien Hendrickx for agreeing to be a reader of this thesis jury.

As this thesis marks the end of my academic years and studies, I would like to thank my family and my friends for the support, all the moments of joy, the good time partying and the laughs I have been through those last five years.

Contents

Introduction

Pricing policy for energy markets brings current interesting challenges. Those energy markets represent a huge amount of money. To have an idea, the total consumption in Belgium in 2019 was 83.73 [TWh],[25], with an average price of 270.5[€/MWh],[8]. For the year 2019 it involved on average a total amount of money that is 2.26e10[€]. The markets need to be flexible because they have to manage uncertainties, especially coming from renewable energies. Hence the generators must be precisely described by their technical constraints to switch from on-status to off-status in order to meet the demand.

The day-ahead electricity market can be formulated as an optimization program called the **Unit Commitment**. This market considers non-convexities and may incur losses for some participants on the market, therefore side payments must compensate those losses. The **Convex Hull Prices** minimize those side payments,[10]. Several US Independent System Operators (ISOs) consider convex hull prices,[1], but they only compute approximations because of the challenging computability.

The initial framework to compute such prices is given by an extended formulation. However the extended formulation can not be considered as the general framework to compute Convex Hull Prices for real-world day-ahead electricity markets. The provided optimization program is too large to be efficiently solved. Several optimization methods can be considered to solve this large scale optimization program. Three of those methods from the literature are investigated in this thesis, namely row generation [16], column generation [1], and column-and-row generation [24].

This manuscript aims at comparing the different methods to compute Convex Hull Prices. Therefore three parts are clearly distinguished, among which different chapters are detailed.

Part I The first part provides an intuitive description of the pricing policy that is Convex Hull Prices. It provides the Unit Commitment problem modelling the day-ahead electricity market, and defines what are exactly the Convex Hull Prices.

Part II The second part is about the understanding of each method. It presents separately each method in order to give the mathematical background supporting the method, the complete optimization programs used in the algorithm related to the method, and an intuition about the method. The order in which the methods are investigated is first the extended formulation, second row generation, third column generation and finally, fourth, column-and-row generation.

Part III The third part compares the four methods computing the Convex Hull Prices on different real-world electricity markets to point out the most efficient method.

In the last part of the thesis, the conclusion discusses which method is improving the most the extended formulation and should be considered as the solution to provide the Convex Hull Prices.

Nomenclature

Sets

$g \in \mathcal{G}$ Set of generators

$t \in [T]$ Set of the time horizon

Parameters

T Last time of the time horizon, card $([T]) = T$

$VOLL$ Values Of Lost Load

L_t Total amount of demand at time instance t

$\bar{P}^g$ Maximum run capacity of generator $g \in \mathcal{G}$

$\underline{P}^g$ Minimum run capacity of generator $g \in \mathcal{G}$

RU^g Ramp up limit of generator $g \in \mathcal{G}$

RD^g Ramp down limit of generator $g \in \mathcal{G}$

SU^g Start-up level of generator $g \in \mathcal{G}$

SD^g Shut-down level of generator $g \in \mathcal{G}$

UT^g Minimum up time of generator $g \in \mathcal{G}$

DT^g Minimum down time of generator $g \in \mathcal{G}$

NLC^g No load cost of generator $g \in \mathcal{G}$

C^g Marginal cost of generator $g \in \mathcal{G}$

F^g Fixed cost of generator $g \in \mathcal{G}$

Variables

p_g^t Power output of generator g at time instance t

$\bar{p}_g^t$ Maximum power output of generator g at time $t \in [T]$

u_g^t Commitment decision of generator $g \in \mathcal{G}$ at time instance $t \in [T]$

v_g^t Start-up decision of generator $g \in \mathcal{G}$ at time instance $t \in [T]$

u_g^t Shutdown decision of generator $g \in \mathcal{G}$ at time instance $t \in [T]$

l_t Demand meet by the market at time instance $t \in [T]$

Acronyms

CHP Convex Hull Prices

EF Extended Formulation

UC Unit Commitment problem

RG Row Generation method

CG Column Generation method

CRG Column-and-row Generation method

VOLL Value of Lost Load

Part I

Description of the Problem

Chapter 1

Real-world Day-ahead Electricity Market

The market considered in this manuscript is an electricity market. The basic concept of such a market is to find which generators should produce electricity in order to meet the demand for the time horizon $[T]$. There are two kinds of participants.

The first kind of participant are the producers. It is the set $\mathcal{G}$ of generators on the market. Those generators have technical constraints defined by the minimum running capacity ($\underline{P}$), the maximum running capacity ($\bar{P}$), the maximum ramp-up rate (RU), the maximum ramp-down rates (RD), the minimum up time (UT), the minimum down time (DT), the start-up level (SU) and the shut-down level (SD). They have three types of costs :

- Start-up cost : fixed cost of the generator whenever it starts to produce. It does not depend on its production.
- No load cost : fixed cost of the generator when it is producing. It does not depend of its production.
- Marginal cost : cost of the generator for its production. It depends on what amount of energy the generator is producing.

The second kind of participant is the consumer. There is one big consumer having an inelastic demand L_t. The electricity market considered in this manuscript does not model the electrical network. All the electricity produced by the generators goes directly to the consumer assuming a theoretical electrical line with an infinite load and no loss.

1.1 Unit Commitment Problem

The market can be described as an optimization program. This optimization program dispatches the production between all the generators while optimizing the objective function that is the total welfare. The optimization can either be a maximization of utilities, or a minimization of costs, for each participant. The market clearing constraint is also called the balance constraint, and it ensures the considered demand meets the total production of all producers.

The dispatch optimization program is known as the **Unit Commitment** problem (UC). It considers the set of generators $\mathcal{G}$ and the demand over the time horizon $[T]$.

Definition 1.1. *A polyhedron P is a formulation for set X if $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$, where P is the intersection of a finite number of half-spaces, i.e. it is such that $P = \left\{x \in \mathbb{R}_+^{n+p} : Bx \geq b\right\}$.*

The unit commitment problem is a Mixed Integer Linear Programming (MILP) problem of the general form F 1.1, where constraint $Dx \geq d$ represents a "complicate constraint" and set X represents a tractable set, i.e. $\min\{cx : x \in X\}$ can be solved rapidly in practice. Set X is of the form $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$, with $P = \left\{x \in \mathbb{R}_+^{n+p} : Bx \geq b\right\}$ that is a polyhedron from definition 1.1.

$$F \equiv \textbf{min } cx \tag{1.1a}$$
$$\text{s.t. } Dx \geq d \tag{1.1b}$$
$$x \in X \tag{1.1c}$$

The n first variables of vector x define the binary variables (u, v, w) and last p variables of vector x define the continuous variables $(p, \bar{p}, l)$. Constraint $Dx \geq d$ defines the balance constraint. The set X defines the technical feasible region Π_{3-bin}^g for all generator $g \in \mathcal{G}$, and the bounds of the demand l_t considered by the market at each time instance. The unit commitment problem MILP problem F is given by UC 1.2.

Unit Commitment Problem (UC)

$$UC \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g \left(p_g, u_g, v_g, w_g \right) \right] - VOLL \sum_t l_t \right\} \tag{1.2a}$$

$$\text{s.t.} \quad \left(\sum_{g \in \mathcal{G}} p_g^t \right) - l_t = 0 \qquad \forall t \in [T] \tag{1.2b}$$

$$(p_g, \bar{p}_g, u_g, v_g, w_g) \in \Pi^g_{3-bin} \qquad \forall g \in \mathcal{G} \tag{1.2c}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{1.2d}$$

Constraint 2.5b is the balance constraint ensuring that the total production meets the demand.

The demand for each time instance is inelastic and it is defined by the variable l_t that belongs to the interval $[0, L_t]$. It can be cut if the producers are not able to meet all the demand L_t. The demand that is not meet penalizes the total welfare of the market by the Value of Lost Load (VOLL). VOLL is assumed to be 3000 [€/MWh], [23].

The technical constraints of each generator are defined by the **three-bin** feasible set Π^g_{3-bin}. It uses technical constants defined above $\underline{P}$, $\bar{P}$, RU, RD, UT, DT, SU and SD. The state of a generator g is defined using five different variables $\left(p_g^t, \bar{p}_g^t, u_g^t, v_g^t, w_g^t\right)$.

1. u_g^t is a binary variable. It indicates whether generator g is producing $u_g^t = 1$ or not $u_g^t = 0$. It is the commitment status of generator g at time t.

2. v_g^t is a binary variable. It indicates whether generator g was started up at time t $v_g^t = 1$ or not $v_g^t = 0$. It is the start-up decision of generator g at time t.

3. w_g^t is a binary variable. It indicates whether generator g was shut down $w_g^t = 1$ at time t or not $w_g^t = 0$. It is the shut-down decision of generator g at time t.

4. p_g^t is a continuous variable. It indicates the power output of generator g at time t.

5. $\bar{p}_g^t$ is a continuous variable. It indicates the maximum power output of generator g at time t.

Maximum power output $\bar{p}_g^t$ is usually involved in the following constraint $\sum_{g\in\mathcal{G}} \bar{p}_g^t \geq L_t + R_t \ , \forall t \in [T]$, where R_t is the reserve required at time instance t. Reserves are not represented in the considered market, therefore the previous constraint is not implemented. For the sake of generality, the maximum power output $\bar{p}_g^t$ is still represented in the set Π_{3-bin}^g.

For a given generator g, the technical feasible region is the three-bin feasible region Π_{3-bin}^g. It is defined by the expressions 1.3.

$$\begin{cases} u_g^t - u_g^{t-1} = v_g^t - w_g^t & \forall t \in [T] \quad (1.3a) \\ \sum_{i=t-UT+1}^{t} v_g^i \leq u_g^t & \forall t \in [UT^g, T] \quad (1.3b) \\ \sum_{i=t-DT+1}^{t} w_g^i \leq 1 - u_g^t & \forall t \in [DT^g, T] \quad (1.3c) \\ \underline{P}^g u_g^t \leq p_g^t \leq \bar{p}_g^t \leq \bar{P}^g u_g^t & \forall t \in [T] \quad (1.3d) \\ \bar{p}_g^t - p_g^{t-1} \leq RU^g u_g^{t-1} + SU^g v_g^t & \forall t \in [T] \quad (1.3e) \\ \bar{p}_g^{t-1} - p_g^t \leq RD^g u_g^t + SD^g w_g^t & \forall t \in [T] \quad (1.3f) \\ (u_g, v_g, w_g) \in \{0,1\}^{3T} & \quad (1.3g) \end{cases}$$

Logical constraint

Constraint 1.3a is the logical constraint combining the binary variables u_g^t, v_g^t and w_g^t. There are three possible scenarios.

1. If generator g was committed off at time $t-1$ $u_g^{t-1} = 0$, and on at time t $u_g^t = 1$, then it starts up at time t $v_g^t = 1$.

$$v_g^t = \begin{cases} 1 & \text{if } u_g^{t-1} = 0 \text{ and } u_g^t = 1 \\ 0 & \text{otherwise} \end{cases}$$

2. If generator g was committed on at time $t-1$ $u_g^{t-1} = 1$, and off at time t $u_g^t = 0$, then it shuts down at time t $w_g^t = 1$.

$$w_g^t = \begin{cases} 1 & \text{if } u_g^{t-1} = 1 \text{ and } u_g^t = 0 \\ 0 & \text{otherwise} \end{cases}$$

3. If generator g stays in the same commitment state as previous time. In that case $u_g^t = u_g^{t-1}$ and $w_g^t = 0 = v_g^t$.

For the first time instance of the time horizon, $t = 1$, there might be some initial condition imposed to specify which generators were committed before the time horizon, and which were not for a horizon $[T_0]$ prior to the time horizon $[T]$, [22].

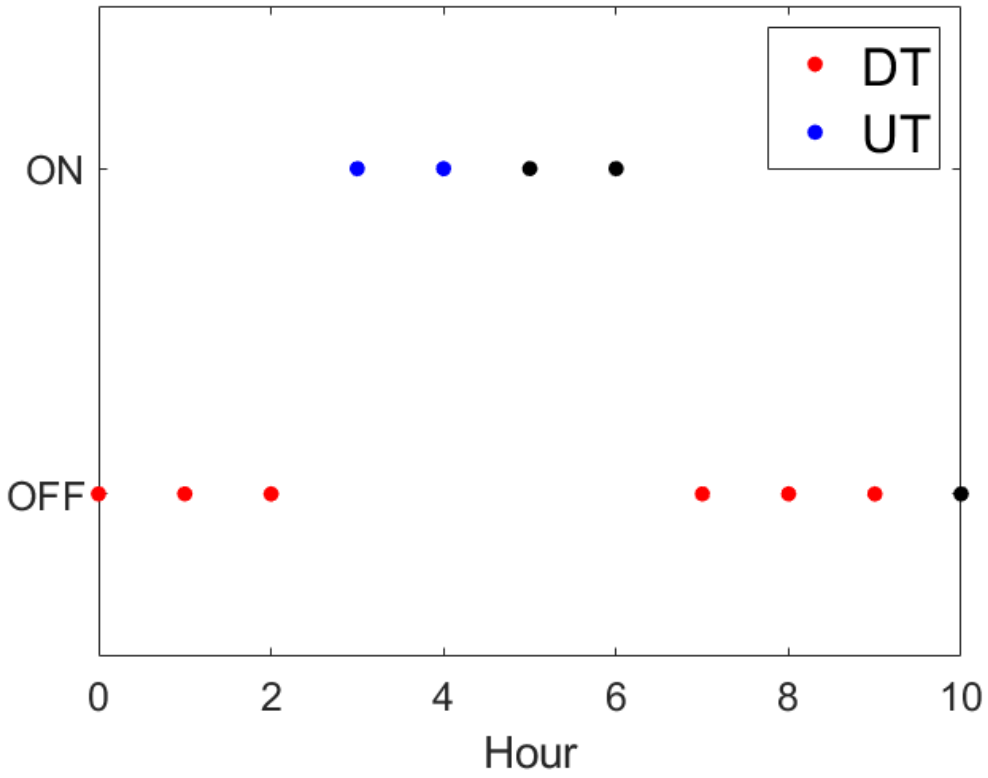


Figure 1.1: Illustration of example 1.1.

Minimum up and down times

Constraints 1.3b and 1.3c are respectively the minimum up time and down time constraints. Constraint 1.3b ensures the generator is committed on a minimum up-time UT before it can be shut down. Constraint 1.3c ensures generator is committed off a minimum down-time DT before being start-up.

Example 1.1. *Consider a generator such that $UT = 2$[hour] and $DT = 3$[hour]. When the generator starts up, it must stay at least 2 hours on. It is represented by the blue points. When it shuts down, it must stay off at least 3 hours. It is represented by red points. Figure 1.1 provides an illustration of the example.*

Maximum and minimum power output

Constraint 1.3d gives upper and lower bounds on the power output and the maximum power output. If the generator is on, then power output p_g^t and maximum

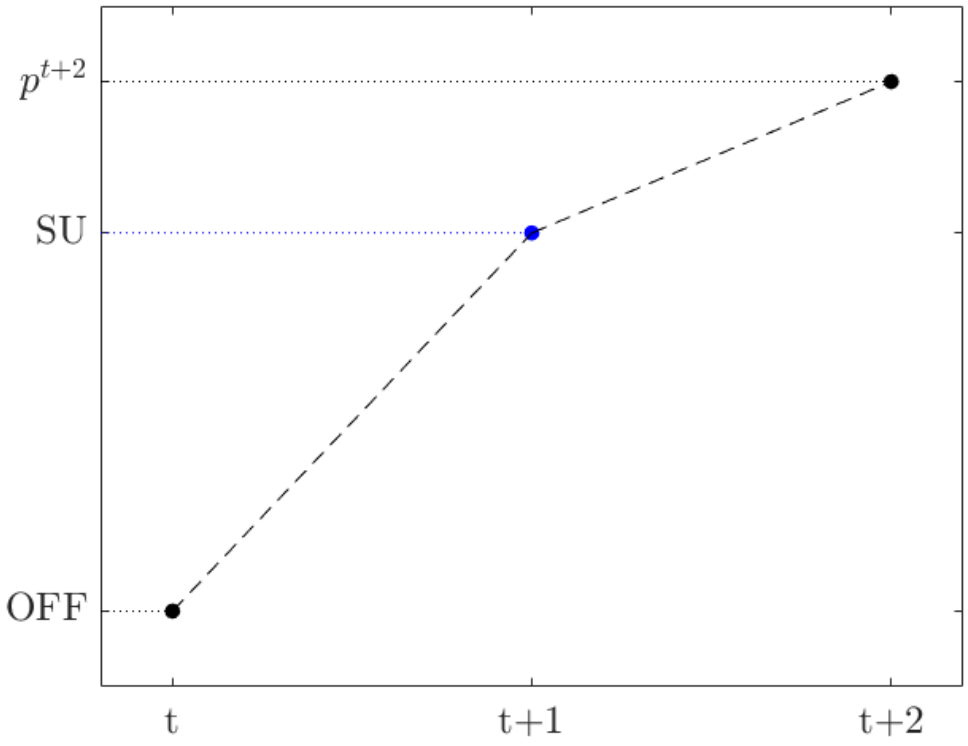


Figure 1.2: Illustrative representation of the start-up level SU for a generator.

power output $\bar{p}_g^t$ must be in the interval $[\underline{P}^g, \bar{P}^g]$. If the generator is off, then the power output is 0 since there is no production.

Ramping up rate and start-up level

Constraint 1.3e provides an upper bound on the power output using the ramp-up rate RU. It also forces the power output to be lower or equal than the start-up level SU^g when the generator starts at time t.
Consider that the generator starts at time t. It was not committed at time $t-1$, which implies that $u_g^{t-1} = 0$ and $p_g^{t-1} = 0$. Constraint 1.3e can be rewritten $p_g^t \leq SU^g$. Constraints 1.3e and 1.3d provide a range of production $\underline{P}^g \leq p_g^t \leq SU^g$, when the generator starts producing as described on figure 1.2.
Consider now a generator that is already producing at time $t-1$. It results that $v_g^t = 0$ and $u_g^{t-1} = 1$. Constraint 1.3e can be rewritten $p_g^t - p_g^{t-1} \leq RU^g$. The generator can not increase its production more than RU. Figure 1.3 gives an illustrative example.

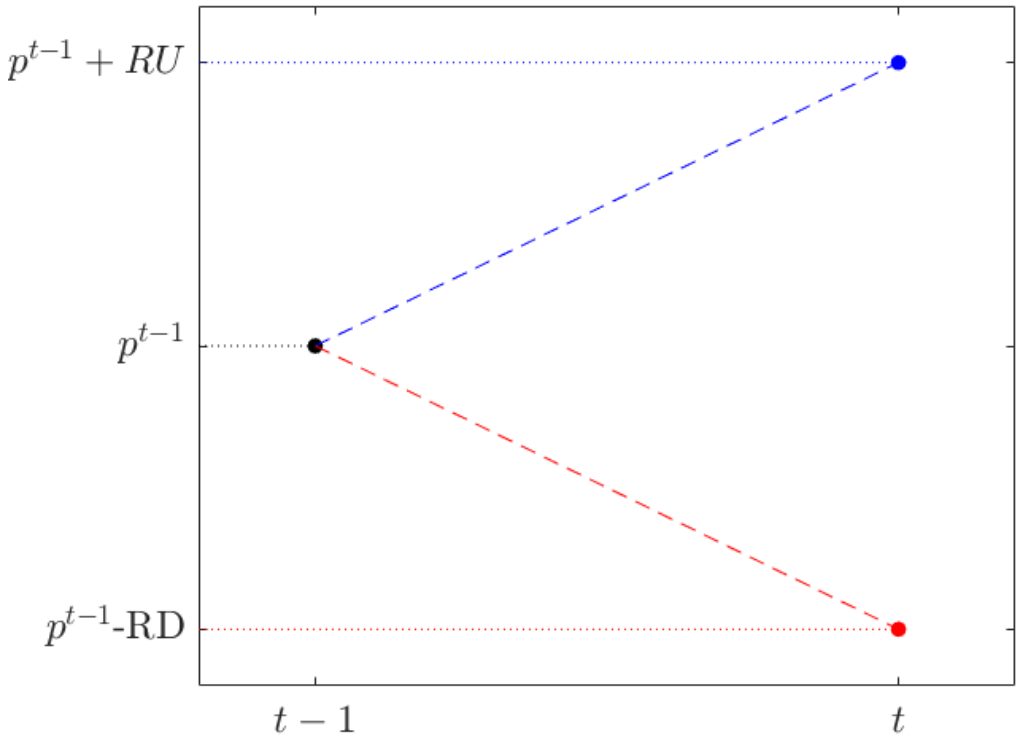


Figure 1.3: Illustration of the range of production after one timestep.

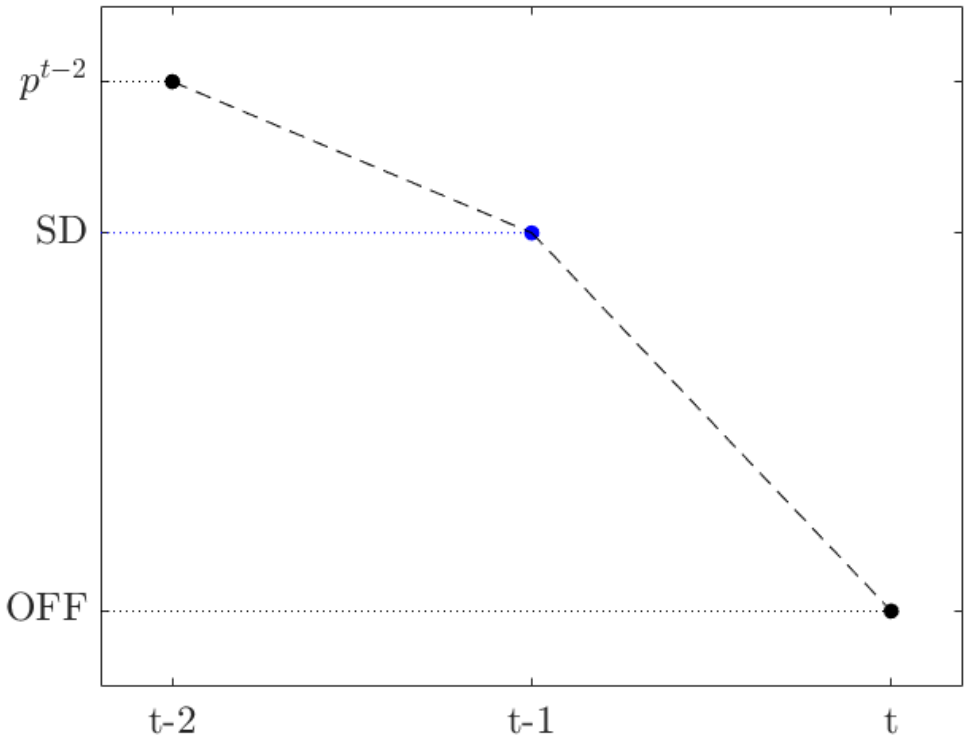


Figure 1.4: Illustrative representation of the shut-down SD level for a generator.

Ramping down rate and shut-down level

Constraint 1.3f provides a lower bound on the power output using the ramp-down rate RD^g. It also forces the power output to be lower or equal than the shut-down level at time $t-1$, before shutting down the generator at time t.
Consider a generator shutting down at time t, then $w_g^t = 1$. Hence constraint 1.3f can be rewritten such that $p_g^{t-1} \leq SD^g$. The production must be smaller or equal than the shut-down level SD before shutting down the generator. Figure 1.4 provides an illustrative example.
Consider now a generator that was already producing at time $t-1$ and that is still producing at time t. Constraint 1.3f can be rewritten such that $p_g^{t-1} - p_g^t \leq RD^g$. Figure 1.3 provides an illustrative representation.

Separable Problem

It is important to notice that the generators can be considered separately. The three-bin feasible region Π^g_{3-bin} is specific to a generator g. Π^g_{3-bin} does not take

into account the technical constraints of other generators. Intuitively it may seem as an obvious remark. However it is important to be noted. Feasible regions can be investigated separately for each generator. The only constraint binding all the generators on the market is the balance constraint 2.5b. This characteristic of the unit commitment problem will be used later in the manuscript.

1.2 Non-convex Bids

The day-ahead electricity market defined in previous section considers non-convexities. They come from non-convex bids made by the generators due to technical constraints and start-up costs involving binary variables. Such constraints are the minimum-maximum power output, the ramping rates and the start-up and shut-down levels. It is not possible to find a market equilibrium supported by uniform prices 1.2 under general conditions [11], [20].

Definition 1.2 (Uniform Prices [20])**.** *Uniform prices λ are prices supporting a market equilibrium such that in the market clearing solution every participant of a same location and time t will pay or receive the same price and no other side payments are considered.*

Having uniform prices means that all the participants have the incentives to commit and produce such that the market clearing from the unit commitment problem solution asks them to do, [20]. Since it does not always exist near-equilibrium prices are computed in practice.

1.2.1 Non Uniform Prices : Example of Missed Opportunities

Example 1.2 shows that participants may miss opportunities when following the market clearing solution of the unit commitment. An illustrative graph is provided on figure 1.5. In the example *marginal pricing* is used, i.e. the price is the maximum marginal price among all the generators involved in the market clearing solution.

Example 1.2. *Consider two generators and a time horizon of one hour.*

- *Since there are 2 generators, then $\mathcal{G} := \{g_1, g_2\}$. The technical constraints are only min-max power outputs. There are respectively $\underline{P} = \begin{pmatrix} 0 & 6 \end{pmatrix}$[MW] and $\bar{P} = \begin{pmatrix} 15 & 10 \end{pmatrix}$[MW].*

  The generators only have the marginal cost, which are $\begin{pmatrix} 10 & 40 \end{pmatrix}$[MW] respectively for g_1 and g_2.

- *There is one demand block of 20[MW], with VOLL 60[€/MW].*

The unit commitment problem applied to this problem is the following one.

$$\begin{aligned}
\textbf{\textit{min}}\ & 10p_1 + 40p_2 - VOLL \cdot l && \text{(1.4a)}\\
s.t.\ & p_1 + p_2 - l = 0 && \text{(1.4b)}\\
& u_1 \cdot 0 \leq p_1 \leq u_1 \cdot 15 && \text{(1.4c)}\\
& u_2 \cdot 6 \leq p_2 \leq u_2 \cdot 10 && \text{(1.4d)}\\
& 0 \leq l \leq 20 && \text{(1.4e)}\\
& u_1 \in \{0,1\} && \text{(1.4f)}\\
& u_2 \in \{0,1\} && \text{(1.4g)}\\
& && \text{(1.4h)}
\end{aligned}$$

The optimal solution provided by the optimization program 1.4 is the following one.

$$\begin{pmatrix} p^{1*} & u^{1*} & p^{2*} & u^{2*} & l^* \end{pmatrix} = \begin{pmatrix} 14 & 1 & 6 & 1 & 20 \end{pmatrix}$$

Equilibrium price is $\lambda^* = 40$[€/MW]. If it is less than 40[€/MW], then g_2 won't produce. If it is more than 40[€/MW], then g_2 will produce at the maximum of its capacity to have a maximum profit.

- *Generator g_1.*
  Given the market clearing solution, g_1 is not producing at its maximum capacity. It could produce 1[MW] more. Hence there is a missed opportunity for g_1 :

$$capacity\ left \cdot (\lambda^* - C_{g_1}) = 1 \cdot (40 - 10) = 30$$

  A side payment of 30[€] is required for g_1 so that it stays at the market clearing solution provided by the unit commitment problem. Otherwise it would produce at full capacity (15 [MW]) and there would be 1 [MW] unconsumed on the market.

- *Generator g_2.*
  The benefit of g_2 following the market clearing solution is $p^{2*}(\lambda - C_2) = 6 \cdot (40 - 40) = 0$[€].
  There are no side payments required for g_2.

There is a total side payments of 30[€] to pay for the participants to follow the market clearing solution.

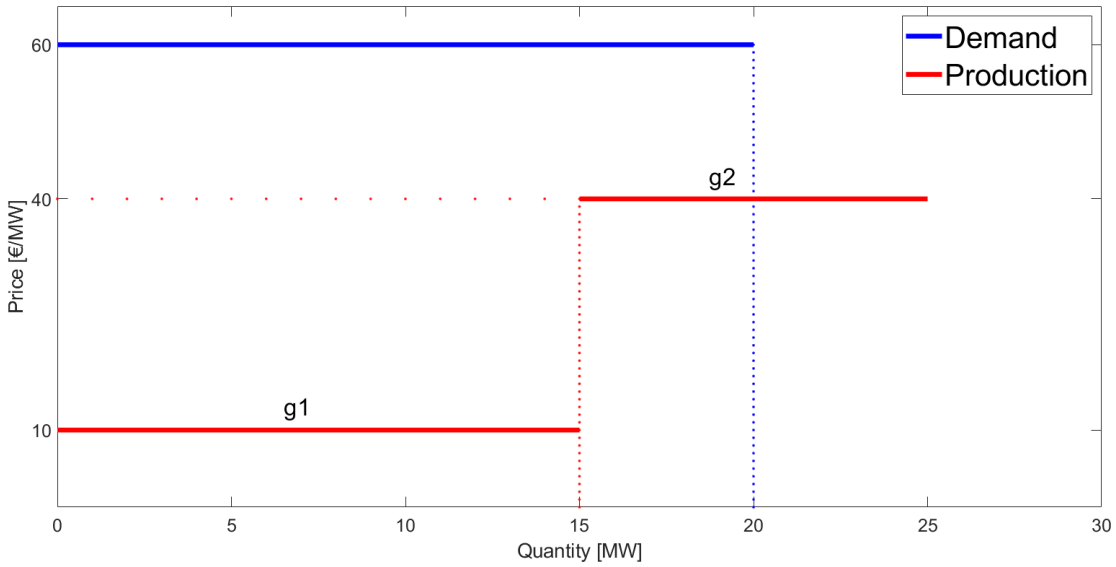


Figure 1.5: Illustration of example 1.2 showing missed opportunities for participants.

1.2.2 Non Uniform Prices : Example of Losses

Example 1.3 shows that participants may face losses considering the market clearing solution of the unit commitment. As previously the example considers two generators over 1 time period. An illustrative graph is provided on figure 1.6.

Example 1.3. *Consider two generators and a time horizon of one hour.*

- *There are 2 generators, therefore* $\mathcal{G} := \{g_1, g_2\}$. *Technical constraints are only min-max power outputs. There are respectively* $\underline{P} = \begin{pmatrix} 0 & 10 \end{pmatrix}$*[MW] and* $\bar{P} = \begin{pmatrix} 10 & 20 \end{pmatrix}$*[MW].*
  Generators have marginal cost $C = \begin{pmatrix} 10 & 30 \end{pmatrix}$ *and fixed cost* $F = \begin{pmatrix} 400 & 0 \end{pmatrix}$.
- *There is one demand block of 25[MW] with VOLL 500[€/MW].*

The unit commitment problem applied to this problem is the following one.

$$
\begin{aligned}
\textbf{\textit{min}}\ & 10p_1 + 400u_1 + 30p_2 + 0u_2 - \mathit{VOLL}\ \cdot l && \text{(1.5a)}\\
s.t.\ & p_1 + p_2 - l = 0 && \text{(1.5b)}\\
& u_1 \cdot 0 \leq p_1 \leq u_1 \cdot 10 && \text{(1.5c)}\\
& u_2 \cdot 10 \leq p_2 \leq u_2 \cdot 20 && \text{(1.5d)}\\
& 0 \leq l \leq 25 && \text{(1.5e)}\\
& u_1 \in \{0,1\} && \text{(1.5f)}\\
& u_2 \in \{0,1\} && \text{(1.5g)}\\
& && \text{(1.5h)}
\end{aligned}
$$

The optimal solution provided by the the optimization program 1.5 is the following one.

$$\begin{pmatrix} p^{1*} & u^{1*} & p^{2*} & u^{2*} & l^* \end{pmatrix} = \begin{pmatrix} 10 & 1 & 15 & 1 & 25 \end{pmatrix}$$

If there is a market equilibrium supported by uniform prices then the price is set by accepting a fraction of g_2. The price is $\lambda^* = 30$[€/MW] because the marginal cost of g_2 is 30[€/MW]. If the price is more than 30[€/MW], then g_2 would produce more than 15[MW]. If it less than 30[€/MW], then g_2 won't produce.

- *Generator g_1.*
  The benefit of generator g_1 is $\lambda^* p^{1*} - C_1 p^{1*} - u^{1*} F_1 = 30 \cdot 10 - 10 \cdot 10 - 400 = -200$[€]
  Generator g_2 has a loss of 200[€]. It needs a side payment of 200[€] to be willing to produce the 10[MW] of the market clearing solution.

- *Generator g_2.*
  The benefit of generator g_2 is $\lambda^* p^{2*} - C_2 p^{2*} - u^{2*} F_2 = 30 \cdot 15 - 30 \cdot 15 - 0 = 0$[€]
  Generator g_2 has no loss and is not missing an opportunity following the market clearing solution.

There is a total amount of side payments of 200[€] to pay, for the participants to follow the market clearing solution.

1.3 Definition of Side Payments : Uplifts

Examples 1.2 and 1.3 show that it is generally not possible to have uniform prices for an electricity market considering non-convex bids. Participants may face losses or miss opportunities following the market clearing solution, also called the

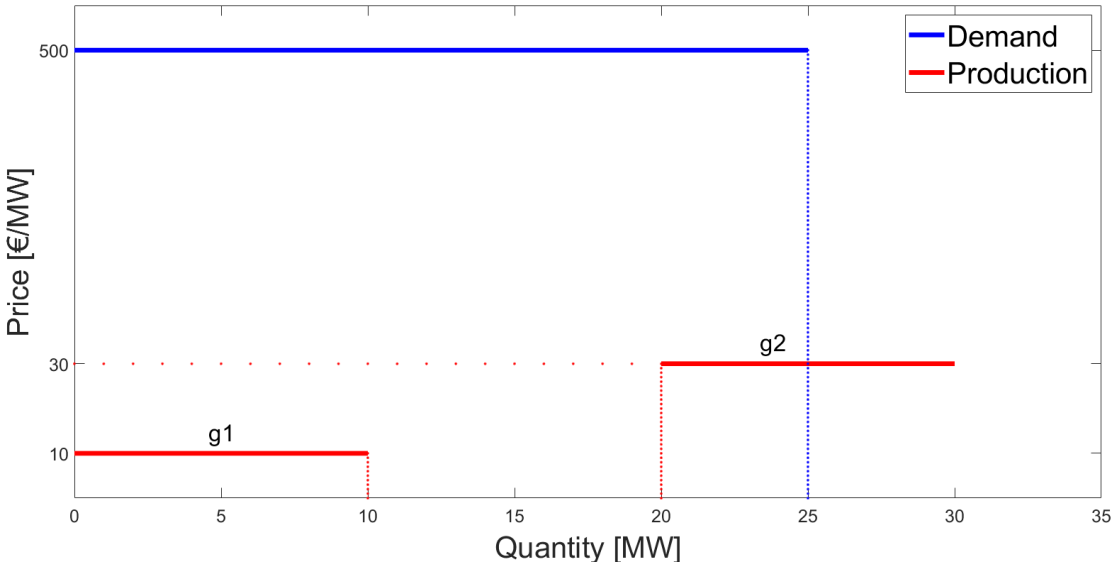


Figure 1.6: Illustration of example 1.3 showing losses for participants.

dispatch solution. The **Uplifts** are the side payments given to the participants in order to compensate their losses and missed opportunities. They depend on the near-equilibrium prices λ computed.

Definition 1.3 (Uplifts)**.** *The uplift for a participant p of the market is the difference between the maximum surplus participant p could extract from the market at a price λ, and the surplus it receives from the dispatch solution, [20].*

The uplifts have a mathematical background leading to the definition 1.3. Source [11] has shown that the sum of uplifts for all participants can be expressed as the difference between the unit commitment problem and its **dual Lagrangian**.

Dual Lagrangian of the general MILP formulation F is $L(\lambda)$ 1.6, where the "complicate" constraint has been dualized with dual multipliers λ.

$$L(\lambda) = \mathbf{min}\ \{cx - \lambda(Dx - d); x \in X\} \tag{1.6a}$$
$$= \mathbf{min}\ \{(c - \lambda D)x; x \in X\} - \lambda d \tag{1.6b}$$

The dual Lagrangian of the unit commitment problem is thus $L(\lambda)$ 1.8, where the balance constraint has been dualized with dual multipliers λ. The term d of equation 1.6 is 0, because the balance constraint has a 0 independent term. It is thus not taken into account.

The sum of the uplifts is the duality gap between the unit commitment and the dual Lagrangian $L(\lambda)$.

$$\sum_{p \in \text{participants}} \text{uplift}\,(\lambda) = UC - L\,(\lambda) \geq 0 \tag{1.7}$$

This duality gap is always positive. Proposition 1.1 states dual Lagrangian $L\,(\lambda)$ is a lower bound on the unit commitment problem UC.

Proposition 1.1. *Any positive price $\lambda \geq 0$ for a dual Lagrangian $L\,(\lambda)$ 1.8 provides a lower bound on the optimal value of UC.*

Proof. See Appendix 9.3.1. □

The dual Lagrangian is a relaxation of the unit commitment problem, from definition 1.4. Condition (i) is trivial since the feasible set of the dual Lagrangian does not consider the balance constraint, that is considered in unit commitment problem UC. Proposition 1.1 shows that condition (ii) of the definition 1.4 is respected.

Definition 1.4 (Relaxation)**.** *A minimization problem Q is said to be a relaxation of the minimization problem P if :*

(i) The feasible set of the relaxation problem Q is at least as big as feasible set of relaxed problem P.

(ii) The objective function of Q is less or equal than the one of P.

Dual Lagrangian ($L\,(\lambda)$)

$$L(\lambda) \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g\,(p_g, u_g, v_g, w_g) \right] - VOLL\left(\sum_t l_t \right) \right. \tag{1.8a}$$

$$\left. - \sum_t \lambda_t \left[\left(\sum_{g \in \mathcal{G}} p_g^t \right) - l_t \right] \right\} \tag{1.8b}$$

$$\text{s.t.} \quad (p_g, \bar{p}^g, u_g, v_g, w_g) \in \Pi^g_{3-bin} \qquad \forall g \in \mathcal{G} \tag{1.8c}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{1.8d}$$

1.3.1 Mathematical Reasoning

Assume the dispatch solution of the unit commitment problem 1.2 is given by $\begin{pmatrix} p_g^* & u_g^* & v_g^* & w_g^* & l^* \end{pmatrix}$. Consider a reformulation of the cost of generator g for convenience.

$$c^{g*} := c^g \left(p_g^*, u_g^*, v_g^*, w_g^*\right) \text{ and } c^g := c^g \left(p_g, u_g, v_g, w_g\right)$$

Firstly, consider the objective function of unit commitment problem UC from expression 1.2. It can be rewritten using the dispatch solution 1.9a. The last term of expression 1.9b does not change the objective value. Indeed the dispatch solution satisfies the balance constraint $\left(\sum_g p_t^g\right) - l_t = 0 \ , \forall t \in [T]$. Expression 1.9c is composed of two terms :

- First term $\left[\sum_t \lambda_t p_g^{t*} - c^{g*}\right]$ is the surplus generator g received from the dispatch solution considering price λ on the market.
- Second term $\sum_t \lambda_t \left(VOLL - l_t^*\right)$ is the surplus consumer received from the dispatch solution considering price λ.

$$UC = \left[\sum_t \left(\sum_g c^{g*}\right)\right] - VOLL \sum_t l_t^* \tag{1.9a}$$

$$= \left[\sum_t \left(\sum_g c^{g*}\right)\right] - VOLL \left(\sum_t l_t^*\right) - \sum_t \lambda_t \left[\left(\sum_g p_g^t\right) - l_t\right] \tag{1.9b}$$

$$= -\left\{\left[\sum_g \left(\sum_t \lambda_t p_g^{t*} - c^{g*}\right)\right] + \sum_t l_t^* \left(VOLL - \lambda_t\right)\right\} \tag{1.9c}$$

Secondly, consider the dual Lagrangian $L\left(\lambda\right)$ 1.8. In expression 1.10b terms are only re-arranged. Expression 1.10c uses the separability characteristic of the unit commitment problem. Indeed the balance constraint is the only constraint binding the producers and the consumer. Since this constraint has been dualized in $L\left(\lambda\right)$, the optimization problems can be performed separately for each participant on the market. Expressions 1.10c is made of two terms :

- The first term $\mathbf{max}\left\{\sum_t \lambda_t p_g^t - c^g\right\}$ is the maximum selfish surplus a generator g can extract from the market considering price λ.
- The second term $\mathbf{max}\left\{\sum_t l_t \left(VOLL - \lambda_t\right)\right\}$ is the maximum selfish surplus a generator g can extract from the market considering price λ.

$$-L(\lambda) = -\mathbf{min}\left\{\sum_t \left[\sum_{g\in\mathcal{G}} c^g\right] - VOLL\sum_t l_t - \sum_t \lambda_t \left[\left(\sum_{g\in\mathcal{G}} p_g^t\right) - l_t\right]\right\} \quad (1.10a)$$

$$= \mathbf{max}\left\{\left[\sum_g \left(\sum_t \lambda_t p_g^t - c^g\right)\right] + \sum_t l_t\left(VOLL - \lambda_t\right)\right\} \quad (1.10b)$$

$$= \left(\sum_g \mathbf{max}\left\{\sum_t \lambda_t p_g^t - c^g\right\}\right) + \mathbf{max}\left\{\sum_t l_t\left(VOLL - \lambda_t\right)\right\} \quad (1.10c)$$

Finally the sum of uplifts can be rewritten using expressions 1.9c and 1.10c, respectively derived from UC and $L(\lambda)$. For both the producers and the consumer, their uplift is the difference between the maximum selfish surplus and the surplus received from the dispatch solution considering a price λ. It gives the mathematical basis supporting the definition of uplifts 1.3.

$$\sum_p \text{uplift}(\lambda) = UC + (-L(\lambda)) \quad (1.11a)$$

$$= (-L(\lambda)) + UC \quad (1.11b)$$

$$= \left(\sum_g \mathbf{max}\left\{\sum_t \lambda_t p_g^t - c^g\right\}\right) + \mathbf{max}\left\{\sum_t l_t\left(VOLL - \lambda_t\right)\right\} \quad (1.11c)$$

$$- \left\{\left[\sum_g \left(\sum_t \lambda_t p_g^{t*} - c^{g*}\right)\right] + \sum_t l_t^*\left(VOLL - \lambda_t\right)\right\} \quad (1.11d)$$

$$= \left[\sum_g \mathbf{max}\left\{\lambda_t p_g^t - c^g\right\} - \left(\sum_t \lambda_t p_g^{t*} - c^{g*}\right)\right] \quad (1.11e)$$

$$+ \left[\mathbf{max}\left\{\sum_t l_t\left(VOLL - \lambda_t\right)\right\} - \sum_t l_t^*\left(VOLL - \lambda_t\right)\right] \quad (1.11f)$$

Uplifts for Example 1.2

Example 1.2 considers three different participants on the market. There are two producers which are generators g_1 and g_2, and there is one consumer with a demand of 20[MW]. The price is $\lambda = 40$[€/MW]. Figure 1.7 shows the amount of maximum selfish surplus, the received surplus and the uplift for each participant of the example.

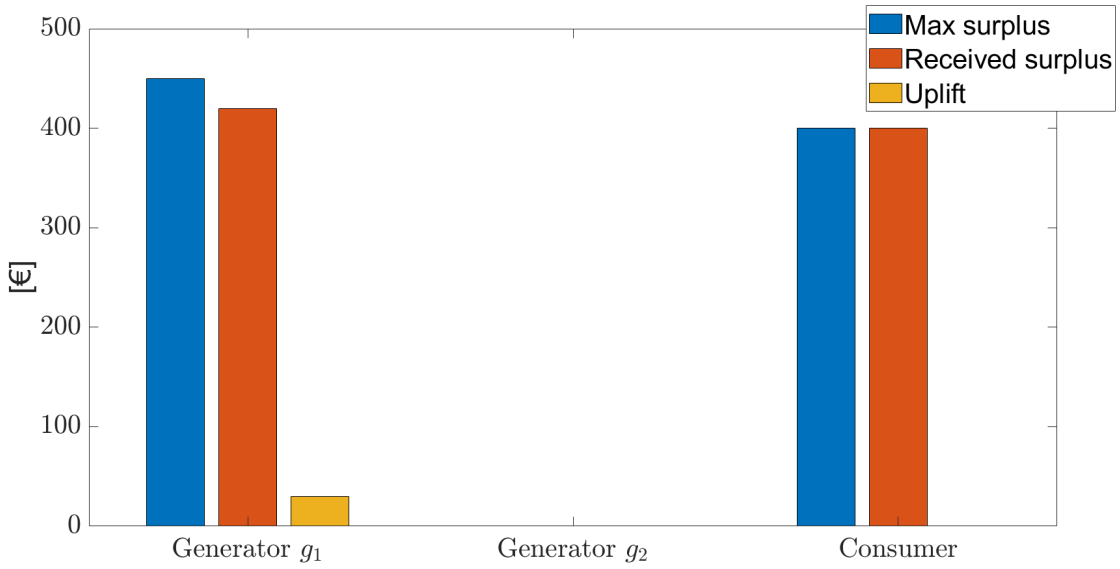


Figure 1.7: The maximum selfish surplus, the surplus received from the dispatch solution and the uplift for each participant of example 1.2

Generator g_1

$$\begin{aligned}
\text{maximum selfish surplus}_{g_1} &= \mathbf{max}\left\{p_1\left(40-10\right) \mid \text{s.t. } 0 \leq p_1 \leq u_1 \cdot 15; u_1 \in \{0,1\}\right\} \\
&= 450[€] \\
\text{received surplus}_{g_1} &= p_1^*\left(40-10\right) = 14\left(40-10\right) = 420[€] \\
\text{uplift}_{g_1} &= 450 - 420 = 30[€]
\end{aligned}$$

Generator g_2

$$\begin{aligned}
\text{maximum selfish surplus}_{g_2} &= \mathbf{max}\left\{p_2\left(40-40\right) \mid \text{s.t. } u_2 \cdot 6 \leq p_2 \leq u_2 \cdot 10; u_2 \in \{0,1\}\right\} \\
&= 0[€] \\
\text{received surplus}_{g_2} &= p_2^*\left(40-40\right) = 0[€] \\
\text{uplift}_{g_2} &= 0[€]
\end{aligned}$$

Consumer l

$$\begin{aligned}
\text{maximum selfish surplus}_{l} &= \mathbf{max}\left\{l\left(60-40\right) \mid \text{s.t. } 0 \leq l \leq 20\right\} = 400[€] \\
\text{received surplus}_{l} &= l^*\left(60-40\right) = 400[€] \\
\text{uplift}_{l} &= 0[€]
\end{aligned}$$

Uplifts for Example 1.3

Example 1.3 considers also two generators g_1 and g_2, and one consumer with a demand of 25[MW]. The price is $\lambda = 20[€/\text{MW}]$. Figure 1.8 shows the amount of selfish surplus maximization, received surplus and uplift for each participant of the example.

Generator g_1

$$\begin{aligned}
\text{maximum selfish surplus}_{g_1} &= \mathbf{max}\left\{p_1\left(30-10\right) - 400u_1 \,|\text{s.t.} 10u_1 \leq p_1 \leq 10u_1; u_1 \in \{0,1\}\right\} \\
&= 0[€] \\
\text{received surplus}_{g_1} &= 20p_1^* - 10p_1^* - 400u_1^* = -200[€] \\
\text{uplift}_{g_1} &= 0 - (-200) = 200[€]
\end{aligned}$$

Generator g_2

$$\begin{aligned}
\text{maximum selfish surplus}_{g_2} &= \mathbf{max}\left\{p_2\left(30-30\right) \,|\text{ s.t. } 10u_2 \leq p_2 \leq 20u_2; u_2 \in \{0,1\}\right\} \\
&= 0[€] \\
\text{received surplus}_{g_2} &= p_2^*(30-30) = 0[€] \\
\text{uplift}_{g_2} &= 0[€]
\end{aligned}$$

Consumer l

$$\begin{aligned}
\text{maximum selfish surplus}_{l} &= \mathbf{max}\left\{l\left(500-30\right) \,|\text{ s.t. } 0 \leq l \leq 25\right\} = 9400[€] \\
\text{received surplus}_{l} &= l^*(500-30) = 9400[€] \\
\text{uplift}_{l} &= 0[€]
\end{aligned}$$

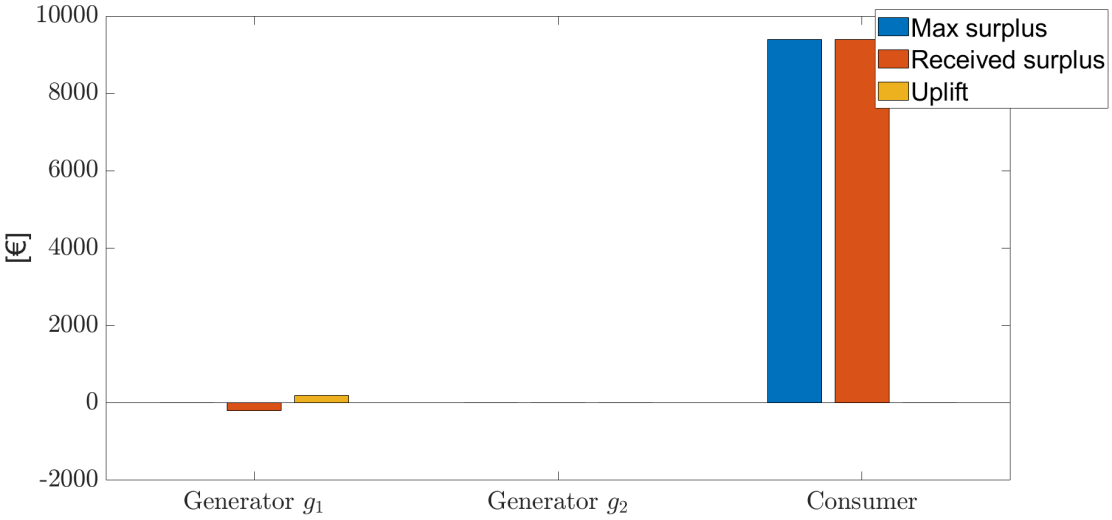


Figure 1.8: The maximum selfish surplus, the surplus received from the dispatch solution and the uplift for each participant of example 1.3.

Chapter 2

Minimum Uplifts and Convex Hull Prices

Due to the non-convexities in the unit commitment problem modelling the day-ahead electricity market, there will be side payments defined as uplifts. The uplifts depend on the near-equilibrium prices λ. The idea is to choose the prices λ in a way to minimize the sum of uplifts to be paid to all participants on the market.

In the following sections the term *uplifts*, in the plural, is a short for *sum of uplifts*.

2.1 Maximization of the Dual Lagrangian

Minimizig the uplifts is equivalent to minimizing the duality gap according to the non-uniform prices λ. Minimizing the duality gap is equivalent to the maximizing the dual Lagrangian $L(\lambda)$.

$$\mathbf{min}_\lambda \left\{ \sum_{p \in \text{participants}} \text{uplift}_p \right\} = \mathbf{min}_\lambda \{\text{duality gap}\} \tag{2.1a}$$

$$= \mathbf{min}_\lambda \{UC - L(\lambda)\} \tag{2.1b}$$

$$= UC - \mathbf{max}_\lambda \{L(\lambda)\} \tag{2.1c}$$

The optimization problem to be considered for the uplifts minimization is the maximization of the dual Lagrangian $\mathbf{max}_\lambda \{L(\lambda)\}$ 2.2. Figures 2.1 and 2.2 show for both examples what price leads to minimum sum of uplifts, and what are the uplifts with the marginal cost pricing.

Using the marginal cost pricing, the computed price of example 1.2 is $\lambda = 40$[€/MW]. This price is the one providing the maximum dual Lagrangian, or equiv-

alently it is the price providing the minimum uplifts. For example 1.3, the marginal cost pricing provides a price of $\lambda = 30$[€/MW]. It provides uplifts($\lambda = 30$) = 200[€]. In this case the marginal cost pricing does not lead to the minimum uplifts. Price $\lambda = 50$[€/MW] is the one providing the minimum uplifts, that is 100[€]. It is obviously less than for price $\lambda = 30$[€/MW].

In general, the marginal cost pricing does not provide the minimum uplifts.

$$\mathbf{max}_{\lambda} \left\{ \begin{array}{c} \mathbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g \left(p_g, u_g, v_g \right) \right] - VOLL \left(\sum_t l_t \right) - \sum_t \lambda_t \left[\left(\sum_{g \in \mathcal{G}} p_g^t \right) - l_t \right] \right\} \\ \text{s.t.} \quad \left(p_g^t, \overline{p}_g^t, u_g^t, v_g^t, w_g^t \right) \in \Pi_{3-bin}^g \ \forall g \in \mathcal{G} \\ 0 \leq l_t \leq L_t \ \forall t \in [T] \end{array} \right\} \tag{2.2}$$

2.2 Convex Hull Optimization Program and Convex Hull Prices

Solving directly the problem 2.2 is not so easy for more elaborated problems than examples 1.2 and 1.3. Next proposition 2.1 allows to find the maximal dual Lagrangian using an equivalent optimization program.

Proposition 2.1. *Maximizing the dual Lagrangian* $\boldsymbol{max}_{\lambda} \{L(\lambda)\}$ *is equivalent to minimize the objective value from UC 1.2 over the feasible region containing :*

- $conv(X)$*, where* X *is made of the non-dualized constraints.*

$$conv(X) = \left\{ \begin{array}{c} \left(p_g^t, \overline{p}_g^t, u_g^t, v_g^t, w_g^t \right) \in conv\left(\Pi_{3-bin}^g \right) \\ l_t \in conv\left([0, L_t] \right) \end{array} \right\}$$

- *the balance constraint.*

Proof. See Appendix 9.3.2. □

Maximizing the dual Lagrangian 2.2 is equivalent to optimize over the set made of the convex hull of the non-dualized constraints, and the balance constraint. Therefore the optimization program must consider the convex hull of the three-bin formulation Π_{3-bin}^g, and the convex hull of the upper and lower bounds on the demand $0 \leq l_t \leq L_t$.

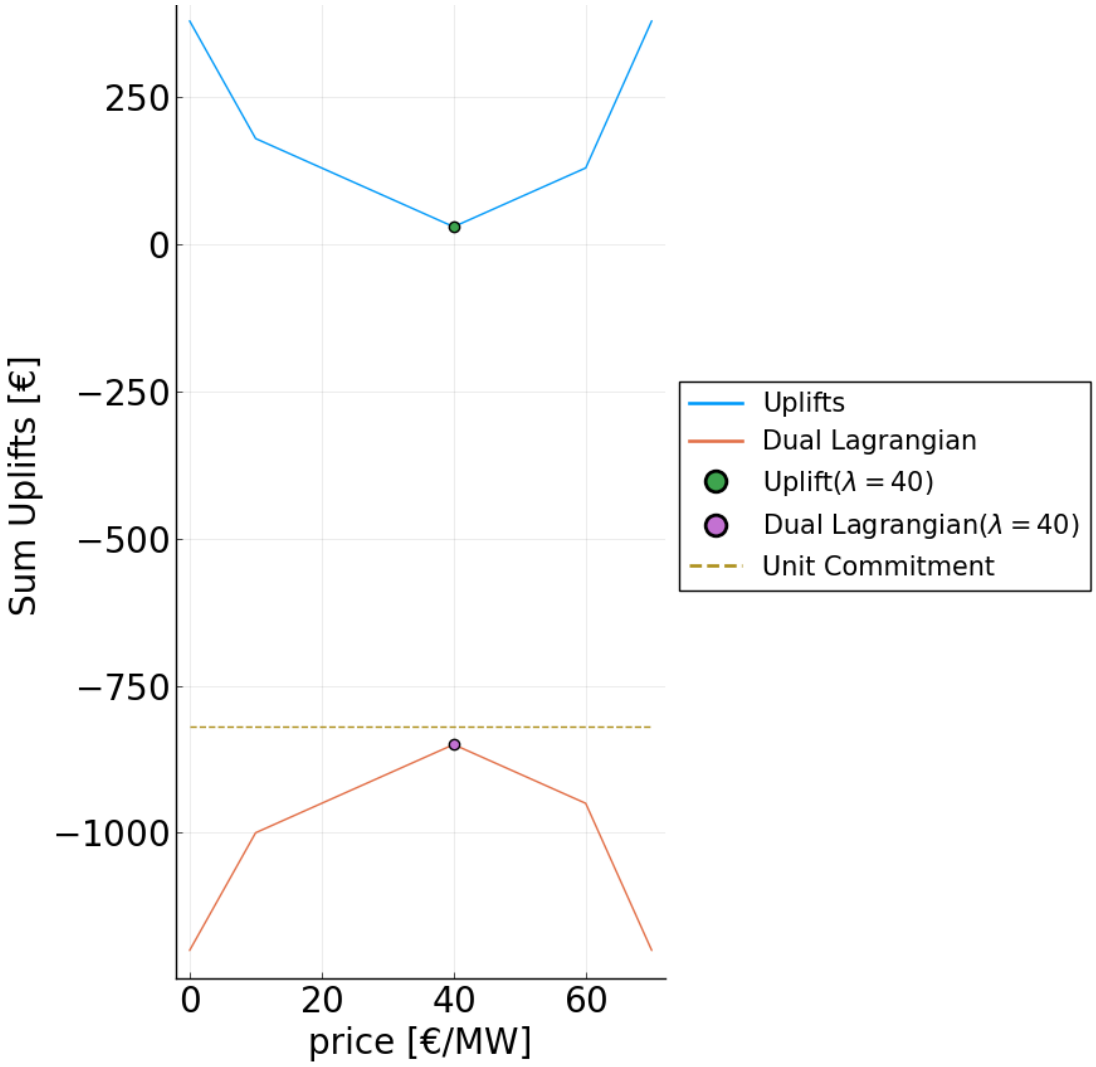


Figure 2.1: Sum of the uplifts and the dual Lagrangian according to prices applied to market for example 1.2.

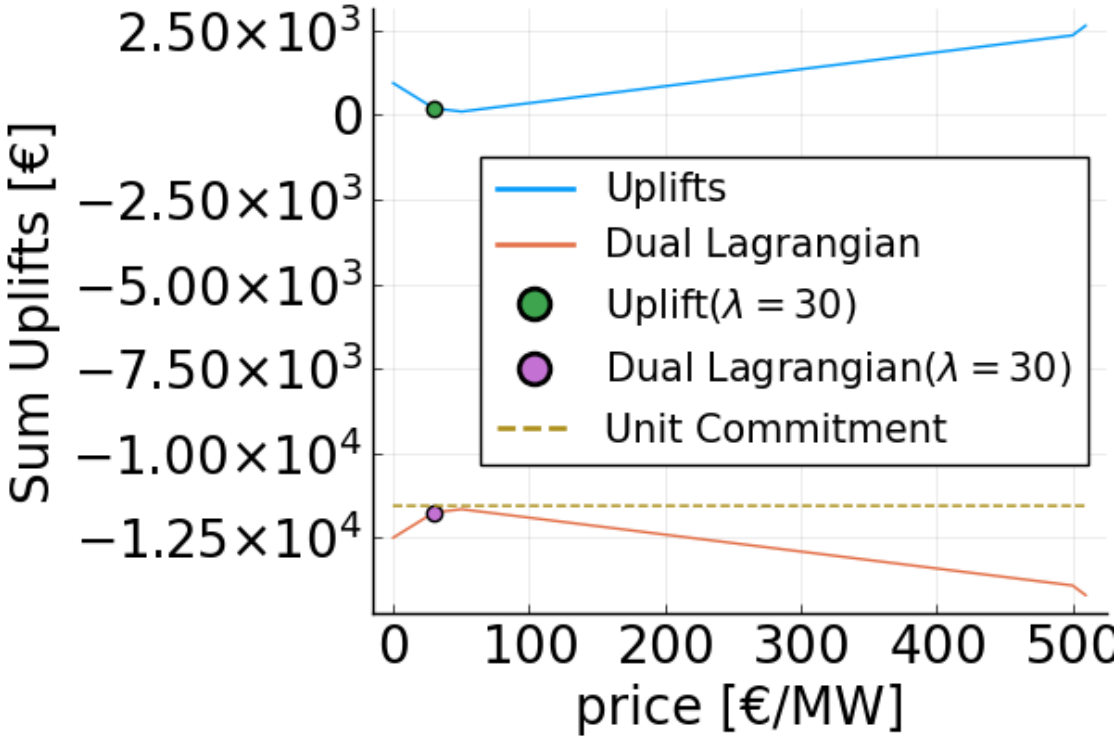


Figure 2.2: Sum of the uplifts and the dual Lagrangian according to prices applied to market for example 1.3.

Definition 2.1. *The convex hull of a set S is equivalently defined as :*

- *The smallest convex set containing the initial set S.*
- *The set of all the convex combinations of the elements of S, enumerated such that $x^1, x^2, ..., x^{|S|}$, assuming there is a finite number of elements.*

$$conv(S) := \left\{ x : \sum_{i=1}^{|S|} \mu_i x^i, \sum_{i=1}^{|S|} \mu_i = 1, \mu_i \geq 0 \right\}$$

It is denoted as $conv(S)$.

Proposition 2.2. *Convex hull of set $[0, L_t]$ is the set $[0, L_t]$.*

Proof. See Appendix 9.3.3. □

Convex hull of the upper and lower bounds on demand is simply the constraint it-self. Hence the convex hull optimization is given by expressions 2.3.

Dual multipliers λ^{CHP} of the balance constraint in CH 2.3 are the ones providing the maximum dual Lagrangian. Indeed due to linear dual optimality $UC = L\left(\lambda^{CHP}\right) \geq L(\lambda)\ \forall\lambda$. Since they come from Convex Hull optimization prices λ are called **Convex Hull Prices** (CHP). The main issue is now to find a formulation for the convex hull of the technical constraints of generator g, i.e. a formulation for conv $\left(\Pi^g_{3-bin}\right)$.

Convex Hull Optimization Program (CH)

$$CH \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g\left(p_g, u_g, v_g\right) \right] - VOLL \sum_t l_t \right\} \tag{2.3a}$$

$$\text{s.t. } \left(\lambda^{CHP}\right) \left(\sum_{g \in \mathcal{G}} p^t_g \right) - l_t = 0 \qquad \forall t \in [T] \tag{2.3b}$$

$$(p_g, \bar{p}^g, u_g, v_g, w_g) \in \text{conv}\left(\Pi^g_{3-bin}\right) \forall g \in \mathcal{G} \tag{2.3c}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{2.3d}$$

Important Propositions regarding Convex Hull Optimization

The following propositions are important for the convex hull optimization. They will be used in later sections. In the considered case the set X is defined by expression 2.4.

$$X = \left\{ \begin{array}{c} (p_g, u_g, v_g, w_g) \in \Pi^g_{3-bin} \ , \forall g \in \mathcal{G} \\ l_t \in [0, L_t] \ , \forall t \in [T] \end{array} \right\} \tag{2.4}$$

Proposition 2.3 states that minimizing over X, or directly over $\text{conv}\,(X)$, is equivalent.

Proposition 2.3. *Let $X \subset \mathbb{R}^n$ and $c \in \mathbb{R}^n$.*
Then $\boldsymbol{min}\{cx : x \in X\} = \boldsymbol{min}\{cx : x \in conv(X)\}$.

Proof. See Appendix 9.3.4. □

Proposition 2.4 states that for a mixed integer linear set X, its convex hull $\text{conv}\,(X)$ can be described with linear inequalities $\{x : A'x \geq a\}$. Hence, for any linear objective function cx, the mixed integer program $\mathbf{min}_x \{cx : x \in X\}$ and the linear program $\mathbf{min}_x \{cx : A'x \leq a'\}$ are equivalent.

Proposition 2.4. *Assume a set of the form $X := \left\{x \in \mathbb{Z}^n_+ \times \mathbb{R}^p_+ : Ax \leq a\right\}$, then $conv(X)$ is a set of the form $\{x : A'x \leq a'\}$, where A', a' have rational entries.*

Proof. See Appendix 9.3.5. □

Considering an integer set $X \subset (\mathbb{Z}^n \times \mathbb{R}^p)$, the MILP program F 1.1 can be reduced to a linear program providing binary values $(u, v, w) \in \mathbb{Z}^n$ from UC problem 1.2.

Proposition 2.5 states that the infimum of cx is attained over X if and only if it is attained over $\text{conv}\,(X)$. It implies that the integer program $\mathbf{min}\,\{cx : x \in X\}$ admits an optimal solution whenever the linear program $\mathbf{min}\,\{cx : x \in \text{conv}\,(X)\}$ is feasible and bounded, and conversely, [6].

Proposition 2.5. *Consider two optimization programs*
$inf\{cx : x \in X\}$ and $inf\{cx : x \in conv(X)\}$.
From proposition 2.3 $inf\{cx : x \in X\} = inf\{cx : x \in conv(X)\}$.
The infimum of cx is attained over X if and only if it is attained over $conv(X)$.

Proof. See Appendix 9.3.6. □

2.3 Continuous Linear Relaxation of the three-bin Formulation

The continuous linear program LP 2.5 considers the continuous linear relaxation of the technical constraints ${}^{\mathcal{R}}\Pi^{g}_{3-bin}$ for each of the generators as feasible region. It has thus continuous variables $\left(u^t_g, v^t_g, w^t_g\right) \in [0,1]^3$. When only considering the minimum and maximum run capacities, and the ramping rates, the continuous linear relaxation of three-bin formulation ${}^{\mathcal{R}}\Pi^{g}_{3-bin}$ provides a convex hull for the technical constraints $\text{conv}\left(\Pi^{g}_{3-bin}\right)$, [20]. In such cases the continuous linear program provides a convex hull optimization program, and maximzes the dual Lagrangian. Therefore in that case, the dual multipliers of the balance constraints λ^{LP} are prices leading to minimum uplifts.

Continuous Linear Program (LP)

$$LP \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g \left(p_g, u_g, v_g\right) \right] - VOLL \sum_t l_t \right\} \tag{2.5a}$$

$$\text{s.t.} \ \left(\lambda^{LP}\right) \left(\sum_{g \in \mathcal{G}} p^g_t \right) - l_t = 0 \qquad \forall t \in [T] \tag{2.5b}$$

$$\left(p^g_t, u^g_t, v^g_t, w^g_t\right) \in {}^{\mathcal{R}}\Pi^{g}_{3-bin} \qquad \forall g \in \mathcal{G} \tag{2.5c}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{2.5d}$$

Under general conditions, the continuous linear relaxation of the three-bin formulation ${}^{\mathcal{R}}\Pi^{g}_{3-bin}$ is a superset of the convex hull $\text{conv}\left(\Pi^{g}_{3-bin}\right)$. Therefore the continuous linear program is a relaxation of the convex hull optimization program, and it does not provide the maximum dual Lagrangian. Under general conditions, the dual multipliers of the balance constraint λ^{LP} do not lead to minimum uplifts. It can be summarised by the expressions 2.6 and 2.7.

$$LP \leq \quad CH \leq \quad UC \tag{2.6}$$

$${}^{\mathcal{R}}\Pi^{g}_{3-bin} \supseteq \ \text{conv}\left(\Pi^{g}_{3-bin}\right) \supseteq \ \Pi^{g}_{3-bin} \tag{2.7}$$

Part II

Computing Convex Hull Prices

Chapter 3

Extended Formulation

Knueven's papers [16], [18], provide a tight extended formulation Q of the convex hull $\text{conv}\left(\Pi^g_{3-bin}\right)$, in the sense of definition 3.1. The extended formulation for $\text{conv}\left(\Pi^g_{3-bin}\right)$ is also a compact formulation, in the sense of definition 3.2.

This extended formulation is based on three main components. The first main component is a **feasible dispatch polytope** $D^{[a,b]}$ considering the time interval $[a, b]$ and assuming generator g can only produce during this time interval $[a, b]$. A polytope $D^{[a,b]}$ is described for each of the intervals $[a, b] \in \mathcal{T}$ respecting minimum up time of the generator g.

The second main component is the **indicator variable** $\gamma_{[a,b]} \in \{0, 1\}$. It indicates whether the polytope related to time interval $[a, b]$ is considered in the packing dispatch polytopes or not. The indicator variables are defined by a restriction Γ.

The third main component is a **linear projection** that enables to perform $X = \text{proj}_x(Q) \cap \mathbb{Z}^{n+p}$, i.e. to project the introduced variable of the extended formulation in the initial space of UC problem 1.2 $\left(p^t_g, u^t_g, v^t_g, w^t_g, l_t\right)$.

Definition 3.1 (Tight Extended Formulation [14])**.** *An extended formulation $Q \subseteq \mathbb{R}^{n+p}$ for an integer set $X \subseteq \mathbb{Z}^n \cap \mathbb{R}^p$ is tight if $proj_x(Q) = conv(X)$.*

Definition 3.2 (Compact Extended Formulation [14])**.** *An extended formulation $Q \subseteq \mathbb{R}^{n+p}$ for an integer $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$ is compact if the length of the description of Q is polynomial in the length of the description of X.*

3.1 Extended formulation : Basic Principles and Definitions

Definition 3.3 details the general term *extended formulation* for a general polyhedron P. Indeed convex hull the optimization program 2.3 needs a formulation for the sets conv $\left(\Pi^g_{3-bin}\right)$ $\forall g \in \mathcal{G}$ where Π^g_{3-bin} is an integer set $\forall g \in \mathcal{G}$.

Definition 3.3 (Extended Formulation)**.** *An extended formulation for a polyhedron $P \subseteq \mathbb{R}^n$ is a polyhedron $Q = \left\{z \in \mathbb{R}^e_+ : Hz \geq h\right\}$ such that $P = proj_x(Q)$.*

Definition 3.4 (Extended Formulation for IP set)**.** *Assume a linear transformation F such that $F : z \in \mathbb{R}^e_+ \longrightarrow x = Tz \in \mathbb{R}^{n+p}_+$.*
An extended IP-formulation for a set $X \subseteq (\mathbb{N}^n \times \mathbb{R}^p)$ is a set $Z = Q \cap \mathbb{Z}^e_+$, with a polyhedron $Q = \left\{z \in \mathbb{R}^e_+ : Hz \geq h\right\}$ such that

$$X = proj_x(Z) = \left\{x \in \mathbb{R}^{n+p}_+ : x = Tz; Hz \geq h, z \in \mathbb{Z}^e_+\right\}$$

Recall the general MILP is $F = \textbf{min}\ \{cx : Dx \geq d, x \in X\}$, where $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$ with polyhedron $P = \left\{x \in \mathbb{R}^{n+p}_+ : Bx \geq b\right\}$. Constraint $Dx \geq d$ represents "complicate" constraint and set X represents a tractable set.

The general extended formulation of an optimization program is given by R 3.1, and its linear relaxation is R_{LP} 3.2. For a tight extended formulation both R and R_{LP} are equivalent. As consequence, an optimization program over the extended formulation of an integer set can be reduced to a linear program, that is R_{LP} 3.2, under the assumption that the convex hull has a tight extended formulation.

$$\begin{aligned} R \equiv \textbf{min}\ & cTz && (3.1a) \\ \text{s.t. } & DTz \geq d && (3.1b) \\ & Hz \geq h && (3.1c) \\ & z \in \mathbb{Z}^e_+ && (3.1d) \end{aligned} \qquad \begin{aligned} R_{LP} \equiv \textbf{min}\ & cTz && (3.2a) \\ \text{s.t. } & DTz \geq d && (3.2b) \\ & Hz \geq h && (3.2c) \\ & z \in \mathbb{R}^e_+ && (3.2d) \end{aligned}$$

3.2 First Main Component : Feasible Dispatch Polytope

The feasible dispatch polytope $D^{[a,b]}$ represents the set of power output $p_t^{[a,b]}$ and maximum power output $\bar{p}_t^{[a,b]}$ considering time interval $[a, b]$ at time t. Both the power output and the maximum power output are vectors such that $\left(p_t^{[a,b]}, \bar{p}_t^{[a,b]}\right) \in$

$\mathbb{R}_+^{2T}$. $D^{[a,b]}$ is defined by the set of linear equations 3.4. It uses technical constraints related to ramping rates, start-up and shut-down levels, minimum and maximum power outputs related to the given generator g. It is a bounded polytope. $D^{[a,b]}$ can be represented such that

$$D^{[a,b]} = \left\{ (p^{[a,b]}, \bar{p}^{[a,b]}) \in \mathbb{R}_+^{2T} \,\middle|\, A^{[a,b]} p^{[a,b]} + \bar{A}^{[a,b]} \bar{p}^{[a,b]} \leq \bar{b}^{[a,b]} \right\} \tag{3.3}$$

Constraints 3.4a and 3.4b specify that the polytope only considers the power output and the maximum power output in the time interval $[a, b]$. Constraints 3.4c, 3.4d and 3.4e specify the upper and lower bounds of the power output and the maximum power output. Constraint 3.4f specifies that generator g can not have a maximum power output higher than the power level it could ramp up by time time t. Constraint 3.4g specifies that generator g can not have a maximum power output than what it could ramp down by time t. Constraint 3.4h specifies that the power jump between two consecutive time steps must be less than the maximum ramping up rate. Constraint 3.4j specifies that this power jump must be less than the maximum ramping down rate. Constraint 3.4i specifies that generator g must be able to ramp down by the remaining time and constraint 3.4k gives an upper bound on what it can ramp up by the remaining time.

$$p_t^{[a,b]} \leq 0 \qquad \forall t < a \text{ and } t > b \tag{3.4a}$$

$$\bar{p}_t^{[a,b]} \leq 0 \qquad \forall t < a \text{ and } t > b \tag{3.4b}$$

$$-p_t^{[a,b]} \leq -\underline{P} \qquad \forall t \in [a, b] \tag{3.4c}$$

$$p_t^{[a,b]} \leq \bar{p}_t^{[a,b]} \qquad \forall t \in [a, b] \tag{3.4d}$$

$$\bar{p}_t^{[a,b]} \leq \bar{P} \qquad \forall t \in [a, b] \tag{3.4e}$$

$$\bar{p}_t^{[a,b]} \leq SU + (t - a)RU \qquad \forall t \in [a, b] \tag{3.4f}$$

$$\bar{p}_t^{[a,b]} \leq SD + (b - t)RD \qquad \forall t \in [a, b] \tag{3.4g}$$

$$\bar{p}_t^{[a,b]} - p_{t-1}^{[a,b]} \leq RU \qquad \forall t \in [a + 1, b] \tag{3.4h}$$

$$\bar{p}_t^{[a,b]} - p_{t-1}^{[a,b]} \leq SD + (b - t)RD - \underline{P} \qquad \forall t \in [a + 1, b] \tag{3.4i}$$

$$\bar{p}_{t-1}^{[a,b]} - p_t^{[a,b]} \leq RD \qquad \forall t \in [a + 1, b] \tag{3.4j}$$

$$\bar{p}_{t-1}^{[a,b]} - p_t^{[a,b]} \leq SU + (t - a)RU - \underline{P} \qquad \forall t \in [a + 1, b] \tag{3.4k}$$

The feasible dispatch polytope $D^{[a,b]}$ is considered for a given time interval $[a, b]$.

Let $\mathcal{T}$ be the set of all possible time intervals $[a, b]$ respecting the minimum up time UT of generator g. Definition of $\mathcal{T}$ is given by expression 3.5.

$$\mathcal{T} := \{ [a, b] \,|\, 1 \leq a \leq a + (UT - 1) \leq b \leq T \} \tag{3.5}$$

For a given generator g all feasible dispatch polytopes $D^{[a,b]}$ such that $[a,b] \in \mathcal{T}$ are considered. Since all possible time intervals are considered, some of them overlap. Hence the time intervals need to be chosen and variables $(p_t^{[a,b]}, \bar{p}_t^{[a,b]})$ are described by the feasible dispatch polytope $D^{[a,b]}$ for those intervals.

One may note that all technical constants have been taken into account, except the minimum down time. It is taking into account in the next section.

3.3 Second Main Component : Indicator Variables to build Packing Dispatch Polytopes

The feasible dispatch polytopes related to the chosen time intervals construct the packing dispatch polytopes. Two overlapping time intervals, in the sense of definition 3.5, can not be together in the same packing. Variable $\gamma_{[a,b]} \in \{0,1\}$ indicates whether a feasible dispatch polytope $D^{[a,b]}$ is in the packing, i.e. $\gamma_{[a,b]} = 1$, or not, i.e. $\gamma_{[a,b]} = 0$.

Definition 3.5 (Overlapping Time Intervals [16])**.** *Two intervals* $[a,b]$ *and* $[c,d]$ *are overlapping intervals if* $[a, b+DT] \cap [c, d+DT] \neq \emptyset$.

The problem of choosing non-overlapping time intervals can be reduced to the cliques inequalities in a graph. The undirected graph is $G = (V, E)$, where the set of vertices is $V = \mathcal{T}$. An edge connects two vertices ($[a,b]$) and ($[c,d]$) if their respective time intervals overlap each other.

Definition 3.6 (Clique [15])**.** *A clique is a subset of vertices of an undirected graph such that every 2 distinct vertices in a clique are adjacent.*

The basic clique inequality is given by equation 3.6, where C is the clique and x_e is a variable related to an edge e of the graph, [14]. For a graph G a clique represents the time intervals that overlap each other at a given time instance t, by definition of the graph. Therefore the clique inequality for the considered graph becomes 3.7.

$$\sum_{e \in C} x_e \leq 1 \tag{3.6}$$

$$\sum_{\{[a,b] \in \mathcal{T} | t \in [a, b+DT]\}} \gamma_{[a,b]} \leq 1 \ \forall t \in [T] \tag{3.7}$$

Based on clique inequalities, Γ 3.8 defines a restriction on indicator variables. Polyhedron $\Gamma \subseteq \mathbb{R}_+$ is a non-empty and non-negative polyhedron, as it is discussed in proof of proposition 3.2. This restriction Γ enforces to not choosing overlapping time intervals in the packing.

$$\Gamma = \begin{cases} \sum_{\{[a,b]\in\mathcal{T}|t\in[a,b+DT]\}} \gamma_{[a,b]} \leq 1 & \forall t \in [T] \\ \gamma_{[a,b]} \geq 0 & \forall [a,b] \in \mathcal{T} \end{cases} \tag{3.8}$$

Proposition 3.1. *Restriction* Γ *3.8 provides a convex hull description of the vertex packing problem, using non-negative variables* $\gamma_{[a,b]}$.

Proof. See Appendix 9.3.7. □

From proposition 3.1, optimizing a linear objective function in terms of $\gamma_{[a,b]}$ over Γ 3.8 provides binary values for $\gamma_{[a,b]}$.

The set $\mathcal{T}$ deals with the minimum up time when building all $[a,b]$ intervals $1 \leq a \leq a + (UT - 1) \leq b \leq T$. The clique inequalities deal with the minimum down time. All the other constraints are used to describe the dispatch polytope $D^{[a,b]}$. Hence all technical constraint are taken into account.

In the appendix, the section **Illustrative Representation of Restriction** Γ 9.2 provides an illustrative representation of the restriction Γ.

3.4 Third Main Component : Linear Transformation to write the Extended Formulation of the Three-bin Formulation

The extended formulation of the three-bin formulation is given by equation 3.9. It uses the convex hull description of the packing problem provided by the combination of the clique inequalities and the feasible dispatch polytopes $D^{[a,b]}$. The indicator variable can consider, or not, a time interval in its packing.

- if the $\gamma_{[a,b]} = 0$, then the power output and the maximum power output for that time interval $[a,b]$ are null production
- if $\gamma_{[a,b]} = 1$, then the power output and the maximum power output are specified by the dispatch polytope using equation $A^{[a,b]}p^{[a,b]} + \bar{A}^{[a,b]}\bar{p}^{[a,b]} \leq \bar{b}^{[a,b]}$, i.e. $\left(p^{[a,b]}, \bar{p}^{[a,b]}\right) \in D^{[a,b]}$

Hence the packing polytope is described by equations 3.9.

$$D = \begin{cases} A^{[a,b]}p^{[a,b]} + \bar{A}^{[a,b]}\bar{p}^{[a,b]} \leq \gamma_{[a,b]}\bar{b}^{[a,b]} & \forall [a,b] \in \mathcal{T} \\ \left(p^{[a,b]}, \bar{p}^{[a,b]}\right) \in \mathbb{R}_+^{2T} & \forall [a,b] \in \mathcal{T} \\ \sum_{\{[a,b]\in\mathcal{T} | t\in[a,b+DT]\}} \gamma_{[a,b]} \leq 1 & \forall t \in [T] \\ \gamma_{[a,b]} \geq 0 & \forall [a,b] \in \mathcal{T} \end{cases} \tag{3.9}$$

Proposition 3.2. *Polytope D is a compact extended formulation, in the sense of definition 3.2, for a constrained generator g described by Π^g_{3-bin}. Vertices of D have integer γ, i.e. it is a tight extended formulation in the sense of definition 3.1.*

Proof. See Appendix 9.3.8. □

The extended formulation of the three-bin formulation can be written using polytope D^g for each generator $g \in \mathcal{G}$.

$$\textbf{min} \left\{ \sum_{[a,b]\in\mathcal{T}} \left[\sum_{g\in\mathcal{G}} c^g \left(p_g^{[a,b]}, \gamma_{[a,b]}^g\right) \right] - VOLL \sum_t l_t \right\} \tag{3.10a}$$

$$\text{s.t} \sum_g \sum_{[a,b]} p_g^{[a,b],t} - l_t = 0 \qquad \forall t \in [T] \tag{3.10b}$$

$$A^{[a,b]}p^{[a,b]} + \bar{A}^{[a,b]}\bar{p}^{[a,b]} \leq \gamma_{[a,b]}\bar{b}^{[a,b]} \qquad \forall [a,b] \in \mathcal{T} \tag{3.10c}$$

$$\left(p^{[a,b]}, \bar{p}^{[a,b]}\right) \in \mathbb{R}_+^{2T} \qquad \forall [a,b] \in \mathcal{T} \tag{3.10d}$$

$$\sum_{\{[a,b]\in\mathcal{T} | t\in[a,b+DT]\}} \gamma_{[a,b]} \leq 1 \qquad \forall t \in [T] \tag{3.10e}$$

$$\gamma_{[a,b]} \geq 0 \qquad \forall [a,b] \in \mathcal{T} \tag{3.10f}$$

$$\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]}\right) \in \mathbb{R}_+^{2T} \qquad \forall g \in \mathcal{G} \tag{3.10g}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{3.10h}$$

Three variables $\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]}, \gamma_{[a,b]}^g\right)$ define the state of a generator g over a time interval $[a,b]$.

Linear transformation $F : z \in \mathbb{R}_+^e \longrightarrow x = Tz \in \mathbb{R}^{n+p}$ is described by equations 3.11, where $x = (p_g, \bar{p}_g, u_g, v_g, w_g)$ and $z = (p_g^{[a,b]}, \bar{p}_g^{[a,b]}, \gamma_{[a,b]}^g)$. F provides

a way to project the packing polytope space D^g into the three-bin space.

$$p_g = \sum_{\{[a,b]\in\mathcal{T}\}} p_g^{[a,b]} \tag{3.11a}$$

$$\bar{p}_g = \sum_{\{[a,b]\in\mathcal{T}\}} \bar{p}_g^{[a,b]} \tag{3.11b}$$

$$u_g^t = \sum_{\{[a,b]\in\mathcal{T}|t\in[a,b]\}} \gamma_{[a,b]} \qquad \forall t \in [T] \tag{3.11c}$$

$$v_g^t = \sum_{\{[a,b]\in\mathcal{T}|t=a\}} \gamma_{[a,b]} \qquad \forall t \in [T] \tag{3.11d}$$

$$w_g^t = \sum_{\{[a,b]\in\mathcal{T}|t=b+1\}} \gamma_{[a,b]} \qquad \forall t \in [T] \tag{3.11e}$$

If the linear transformation is defined in the optimization problem, one can use the variables $\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]}, \gamma_{[a,b]}^g\right)$ to describe the convex hull, and the variables $(p, \bar{p}, u, v, w)$ to describe the objective function and the balance constraint. Therefore the extended formulation is provided by expressions 3.12.

Extended Formulation Program (EF)

$$EF \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g \left(p_g, u_g, v_g \right) \right] - VOLL \sum_t l_t \right\} \tag{3.12a}$$

$$\text{s.t } (\lambda_t) \left(\sum_g p_g^t \right) - l_t = 0 \qquad \forall t \in [T] \tag{3.12b}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{3.12c}$$

$$A^{[a,b]} p_g^{[a,b]} + \bar{A}^{[a,b]} \bar{p}_g^{[a,b]} \leq \gamma_{[a,b]} \bar{b}_g^{[a,b]} \qquad \forall [a,b] \in \mathcal{T}, \forall g \in \mathcal{G} \tag{3.12d}$$

$$\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]} \right) \in \mathbb{R}_+^{2T} \qquad \forall [a,b] \in \mathcal{T}, \forall g \in \mathcal{G} \tag{3.12e}$$

$$\sum_{\{[a,b] \in \mathcal{T} | t \in [a, b+DT]\}} \gamma_{[a,b]}^g \leq 1 \qquad \forall t \in [T], \forall g \in \mathcal{G} \tag{3.12f}$$

$$\gamma_{[a,b]}^g \geq 0 \qquad \forall [a,b] \in \mathcal{T}, \forall g \in \mathcal{G} \tag{3.12g}$$

$$\sum_{\{[a,b] \in \mathcal{T}\}} p_g^{[a,b]} = p_g \qquad \forall g \in \mathcal{G} \tag{3.12h}$$

$$\sum_{\{[a,b] \in \mathcal{T}\}} \bar{p}_g^{[a,b]} = \bar{p}_g \qquad \forall g \in \mathcal{G} \tag{3.12i}$$

$$\sum_{\{[a,b] \in \mathcal{T} | t \in [a,b]\}} \gamma_{[a,b]}^g = u_g^t \qquad \forall t \in [T], \forall g \in \mathcal{G} \tag{3.12j}$$

$$\sum_{\{[a,b] \in \mathcal{T} | t = a\}} \gamma_{[a,b]}^g = v_g^t \qquad \forall t \in [T], \forall g \in \mathcal{G} \tag{3.12k}$$

$$\sum_{\{[a,b] \in \mathcal{T} | t = b+1\}} \gamma_{[a,b]}^g = w_g^t \qquad \forall t \in [T], \forall g \in \mathcal{G} \tag{3.12l}$$

Constraint 3.12b is the balance constraint. Its dual multipliers λ_t are the CHP. Constraint 3.12c specifies that the amount of demand meets on the market is between 0 and the actual total demand L_t. Constraints 3.12d, 3.12e, 3.12f and 3.12g define the packing polytope, given by expressions 3.9. The remaining constraint 3.12h, 3.12i, 3.12j, 3.12k and 3.12l define the linear transformation F 3.11.

Proposition 3.3. *EF is a relaxation of UC.*

Proof. See Appendix 9.3.9. □

Proposition 3.4. *EF solves $\boldsymbol{max}\{L(\lambda)\}$.*

Proof. The extended formulation EF considers the polytope D that defines a compact and tight extended formulation for conv $\left(\Pi_{3-bin}^g\right)$ from proposition 3.2, i.e.

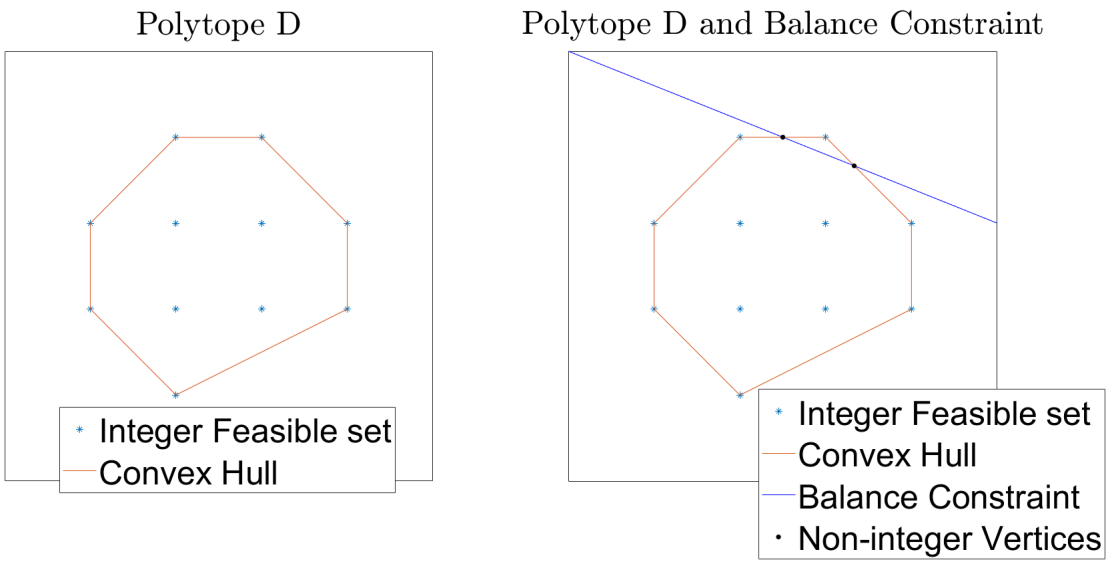


Figure 3.1: Illustrative example of the intersection between polytope D and balance constraint.

$\text{proj}_{p_g,\bar{p}_g,u_g,v_g,w_g}(D) = \text{conv}\left(\Pi^g_{3-bin}\right)$. Expressions 3.11 provides a linear transformation $F : z \in \mathbb{R}^e_+ \longrightarrow x = Tz \in \mathbb{R}^{n+p}$. It provides a way to get back from the packing polytope space to the three-bin space.

From proposition 2.1, solving $\mathbf{max}\{L(\lambda)\}$ is equivalent to optimize over the set S that is

$$S = \left\{ \begin{array}{c} (p^g_t, u^g_t, v^g_t, w^g_t) \in \text{conv}\left(\Pi^g_{3-bin}\right) \\ l_t \in \text{conv}([0, L_t]) \\ \left(\sum_{g\in\mathcal{G}} p^t_g\right) - l_t = 0 \ \forall t \end{array} \right\}$$

Note that from proposition 2.2 $\text{conv}([0, L_t])$ is $[0, L_t]$. □

From proposition 3.4 there is now a solution to solve $\mathbf{max}\{L(\lambda)\}$ 2.2. The dual multipliers of balance constraint provide CHP.

One thing to note is that even if packing polytope D 3.9 provides a convex hull, the balance constraint 3.12b cuts the polytope. The optimal solution may be on a corner of this intersection. It that case the optimal solution may not have binary values for u, v and w. Figure 3.1 provides an illustrative example.

3.5 Examples

Now that the extended formulation 3.12 is completely describes, it can be tested on two examples. Up to now the cost of generator g was represented such that $\sum_t \left[c^g\left(p_g, u_g, v_g\right)\right]$. This cost takes into account the three different kinds of cost.

1. Marginal Cost C [€/MWh] : the cost of production of 1 [MWh].
2. Start Up Cost F [€] : the fixed cost whenever the generator starts producing.
3. No Load Cost NLC [MW] : fixed cost of the generator when it is producing.

The total cost for generator g is given by expression 3.13.

$$\sum_t \left[c^g\left(p_g, u_g, v_g\right)\right] = \sum_t \left[NLC_g C_g u_g^t + F_g v_g^t + C_g p_g^t\right] \tag{3.13}$$

For those two examples and the following ones, *total Running Time* RT and *Solving Time* ST are two performances indicator. RT indicates the total amount of time the program has been running. It takes into account the time required to model the optimization problem and the time required for the built model to be solved. ST only indicates the time required to solve the optimization program without the modelling part.

The number of variables and constraints are also used as performance indicators for the two examples. They are directly taken from the optimization model `Gurobi` builds.

Small Example : One Generator on Four Hours

Example 3.1 (One Generator Example : Extended Formulation)**.** *This first example is a small example considering only one generator g_1 over a time horizon of four hours, i.e. $[T] = \{1, 2, 3, 4\}$. The technical constants of generator g_1 are provided in table 3.1. The demand of the consumer is provided in table 3.2.*

Table 3.3 summarises the uplifts and performances of UC, LP and EF programs. UC and EF have the same objective values however it is not an observation that can be generalized. Indeed EF is a relaxation of UC problem, from proposition 3.3. Thinking about the primal approach of LP, and knowing its feasible region is under general conditions not a convex hull for Π^g_{3-bin}, it is clear that this program can achieve better results in terms of objective value than EF. It is represented by a smaller objective value, since it is a minimization. However dual multipliers related to the balance constraint from LP do not lead to minimum amount of uplifts as it is the case for EF.

	$\underline{P}_{\min}$	$\bar{P}^{\max}$	RU	RD	SU	SD	UT	DT	No Load Consumption	Marginal Cost	Fixed Cost
g_1	6	16	5	5	6	6	1	1	10	53	30

Table 3.1: Producers description for example 3.1.

L	VOLL
$\begin{pmatrix} 6 & 11 & 16 & 11 \end{pmatrix}$	3000[€/MWh]

Table 3.2: Consumer description for example 3.1.

Table 3.4 shows prices from both EF and LP programs. CHP from EF lead to the maximal dual Lagrangian $L\left(\lambda^{CHP}\right)$ that is -66 161[€], and the minimum sum of uplifts that is 0[€]. LP prices lead to a dual Lagrangian that is -66 317.25[€], and a sum of uplifts that is 77.187[€]. Analyse the dual Lagrangian function and the uplifts according to price of hour 1 $\lambda_{t=1}$ and price of hour 2 $\lambda_{t=2}$. Prices of hour 3 and 4 are $3000[€/MWh] = \lambda_{t=3} = \lambda_{t=4}$. One can see on figure 3.2 that prices from EF indeed lead to maximal value of $L(\lambda)$ and prices from LP do not. Figure 3.3 represents the uplifts for both prices for EF and LP. Prices from EF provide the minimum uplifts, i.e. 0[€], and prices from LP do not lead to minimum uplifts, i.e. 77.187€.

Table 3.3 reports the uplifts and performances for both EF and LP methods. For this small example the time performances are not that representative because the current example does not represent real-world market. One may see a small difference for RT and ST between LP and EF methods. This comes from the fact that EF considers more variables and constraints, i.e. 164 variables and 444 constraints, than LP, i.e. 34 variables and 97 constraints.

Real-World Example

Example 3.2 (Winter-WE : Extended Formulation)**.** *This second example is bigger than the previous one since it is a real world example. The current example considers 68 generators from Belgian electricity network and a demand over 24 hours. Belgian generators are fully detailed in the Appendix in tables 9.5 and 9.6. One may note that the start-up and shutdown levels, SU and SD, are not given for Belgian generators. Hence assume they are both equal to the minimum run capacity $\underline{P}$ of the generator.*

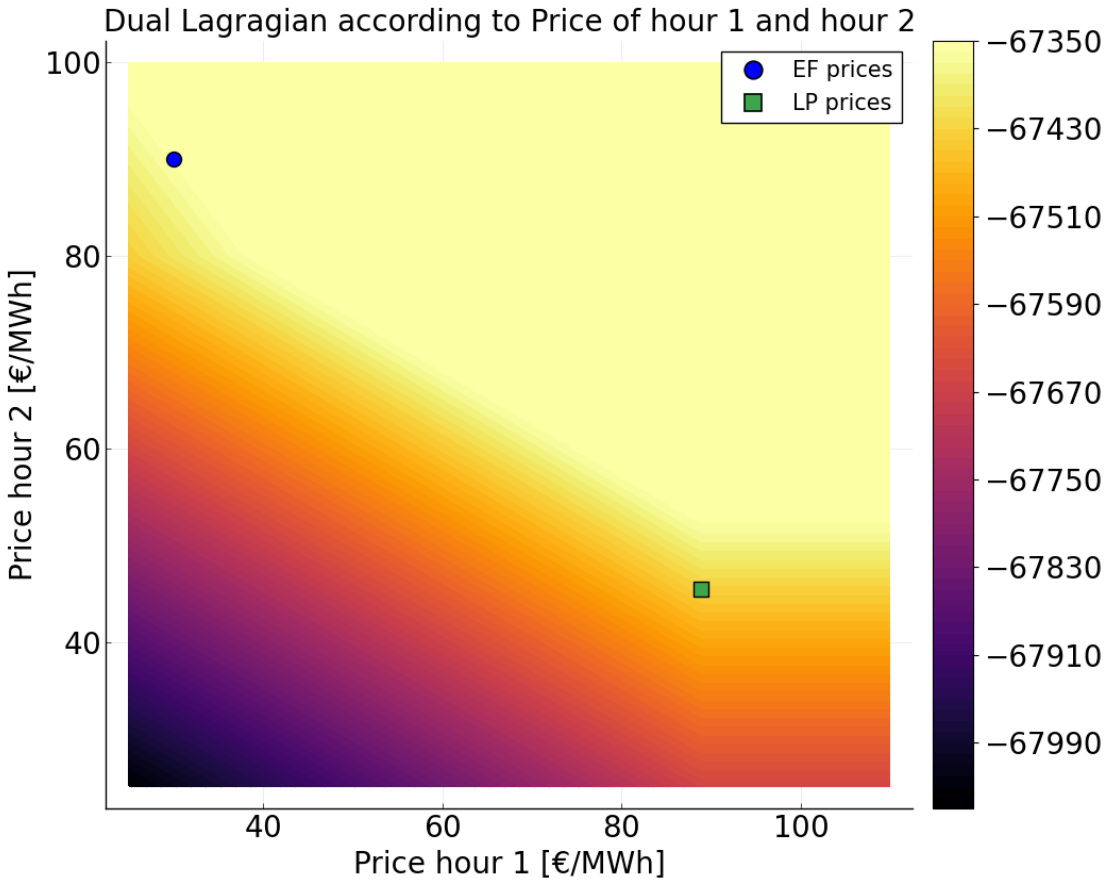


Figure 3.2: Dual Lagrangian function according to price of hour 1 and 2 for example 3.1.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-67,357	/	0.013	0.001	-
LP	-67,434.187	77.187	0.013	0.001	-
EF	-67,357	0	0.016	0.0038	-

Table 3.3: Description of uplifts and their performances using LP and EF methods to compute prices for example 3.1.
RT : total Running Time of the program.
ST : Solving Time of the program.

Method	CHP ?	Prices [€/MWh]
LP	No	$\begin{pmatrix} 88.8333 & 45.4375 & 3000 & 3000 \end{pmatrix}$
EF	Yes	$\begin{pmatrix} 30 & 90 & 3000 & 3000 \end{pmatrix}$

Table 3.4: Prices from different methods for example 3.1.

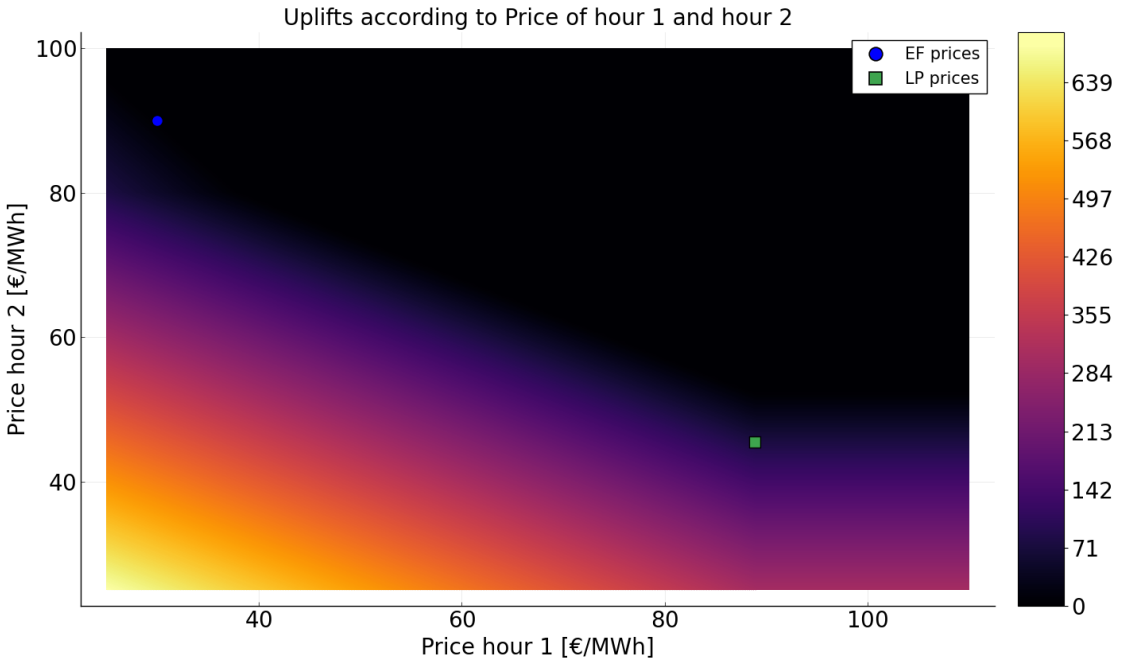


Figure 3.3: Uplifts according to price of hour 1 and 2 for example 3.1.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-607,224,305	/	1.128	0.699	-
LP	-607,435,576	35,550	1.940	0.205	-
EF	-607,245,477	21,171	242.347	197.293	-

Table 3.5: Description of uplifts and their performances using LP and EF methods to compute prices for example 3.2.
RT : total Running Time of the program.
ST : Solving Time of the program.

Tables 3.5 reports the uplifts and performances for UC, LP and EF methods. As expected LP and EF provides smaller objective value than UC since there are relaxation programs. CHP from EF give a 0[€] uplift for the consumer and a total amount of uplifts that is 21 171[€] for the generators. LP does not provide CHP. Uplift for the consumer is 0[€] and total amount of uplifts for generators is 35 550[€]. Both uplifts of generators for CHP from EF and prices from LP are compared on figure 3.4.

To obtain CHP one need to consider the 607,446 variables and 2,080,438 constraints required to build the EF optimization program. In comparison, LP considers 8,864 variables and 26,697 constraints. Such number of variables and constraints impacts RT and ST. Both RT and ST from EF are bigger than the ones of LP.

3.6 Size of the Extended Formulation

Extended formulation from paper [16] is a compact formulation. However even if polytope D 3.9 has a polynomial size in T, there is a considerable amount of variables and constraints. Figures 3.5 and 3.6 represent the amount of variables needed to describe polytope D for one generator only, considering different values of UT.

Besides the amount of variables and constraints required to describe D, the balance constraint and the linear transformation F also need to be specified. For a real-world problem, one needs to consider more than one generator. Therefore, the number of variables and constraints can be very high for real-world problems.

Considering the Belgian system made of 68 generators and considering a vector $[T]$ with a length 24, the amount of variables and constraints required to model and

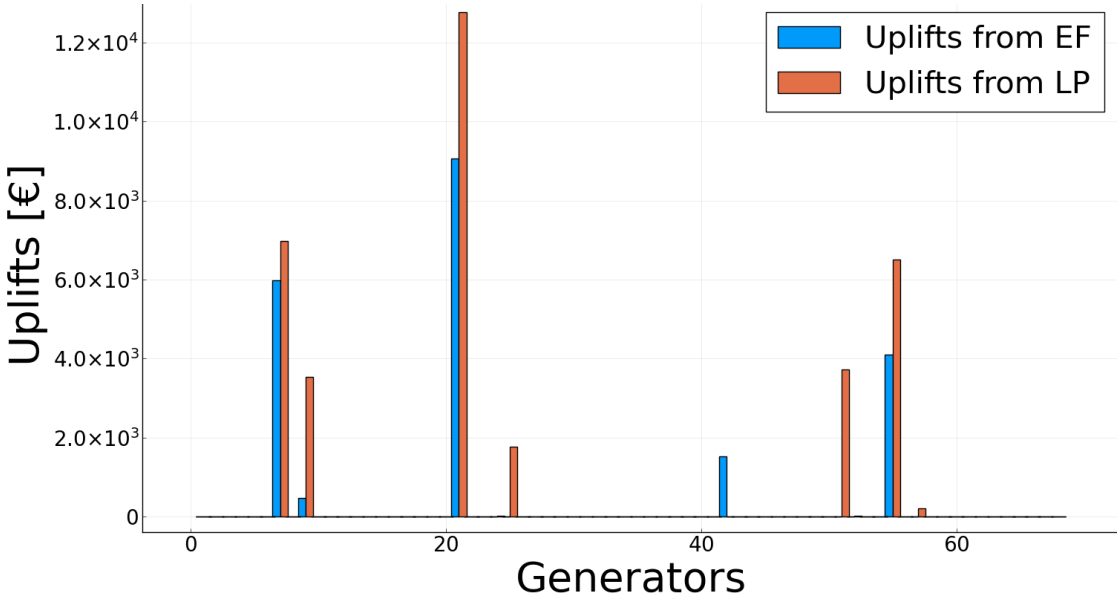


Figure 3.4: Uplift for each generator with CHP from EF and prices from LP.

solve EF are respectively 607 446 and 2 080 438. The demand can be modelled using different timesteps than 1 hour. One can use 15-minutes timestep. Hence the length of vector $[T]$ changes due to the timestep. For a timestep of 5 minutes $[T]$ as a length of 96. With such a time horizon $[T]$ the extended formulation that considers the Belgian generators can even not be modelled on an usual laptop.

To tackle the problem of size of the extended formulation three methods have been explored. The first method is a row generation algorithm and more precisely it is a Bender decomposition. The second method is a column generation. The third method is a column-and-row generation.

3.7 Prior and Posterior Production

In this section terms, *prior* and *posterior* respectively refer to *prior to time horizon* $[T]$ and *posterior to time horizon* $[T]$.

One may note CHP from EF for example 3.1 have values 3000[€/MWh] for hour 3 and hour 4. It is quite a high price regarding the costs of generator g_1. Actually generator g_1 is not able to meet all demand at hour 3 and hour 4, therefore $\lambda_{t=3} = \lambda_{t=4} = VOLL = 3000[€/MWh]$.

The maximum run capacity $\bar{P}$ is 16 [MW]. The demand is 16[MWh] for hour 3 and 11[MWh] for hour 4. Hence the generator g_1 should be able to meet all demand

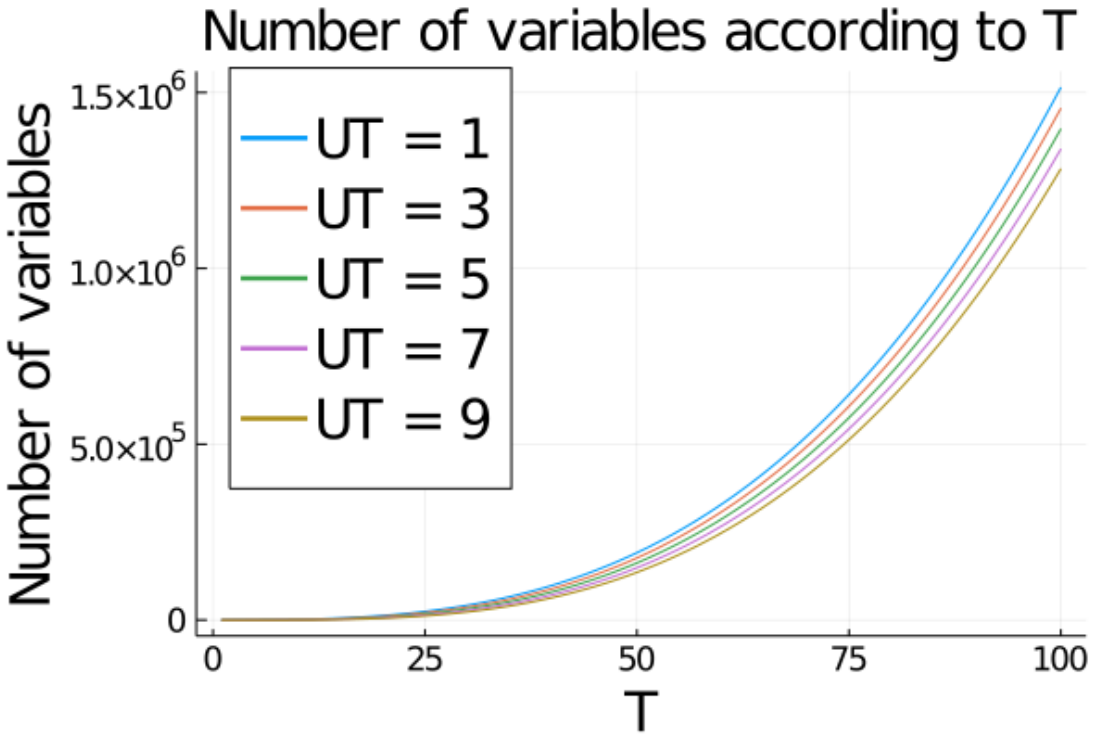


Figure 3.5: Number of variables of polytope D according to T considering different value of UT.

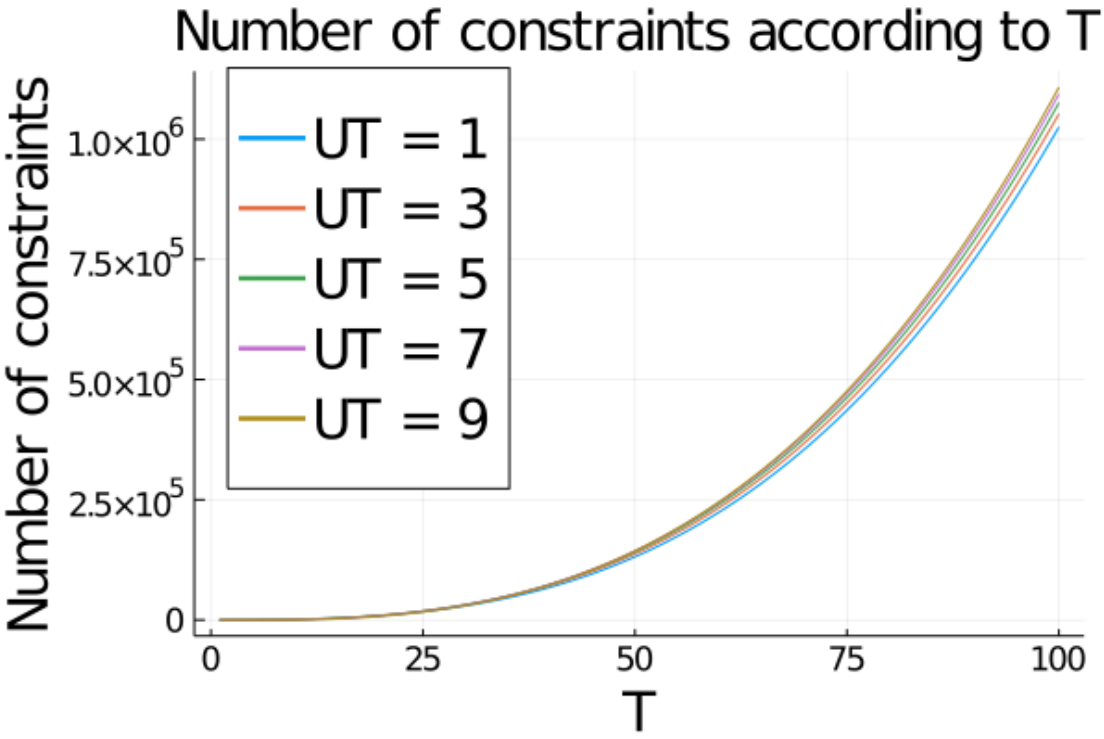


Figure 3.6: Number of constraints of polytope D according to T considering different value of UT.

at hour 3 and 4. However with the extended formulation prior and posterior productions are set to 0. Next lines explain why it is the case and why it is a problem.

Intervals $[a, b]$, used by the extended formulation to build feasible dispatch polytope $D^{[a,b]}$, consider that production is starting at $t = a$ and ending at $t = b$. Hence production outside a given interval $[a, b]$ is a 0[MW] production. To consider prior and posterior production, i.e. outside time horizon $[T]$, one needs to consider intervals outside time horizon $[T]$, i.e. intervals $[a, b]$ such that $a < 1$ for prior intervals $[a, b]$ and intervals such that $b > T$ for posterior intervals.
Sources [16], [18] propose the following solution :

- Consider intervals $[0, b]$ for intervals starting before $t = 1$, i.e. $[0, b]$ is a prior interval.
- Consider intervals $[a, T+1]$ for intervals shutting down after $t = T$, i.e. $[a, T+1]$ is a posterior interval.

It would enable prior and posterior productions in the extended formulation. As consequence, it would be possible to consider committed generators prior time to horizon $[T]$, and to provide committed generators for the next time horizon.

However this solution is not always sufficient. Assume generator is shutted down before $t = 1$, i.e. there are no prior intervals. Consider interval $[a, b = T + 1]$ to have production posterior to time $[T]$. $b = T + 1$ should be sufficiently large such that constraint 3.4g does not force the production to be less than what it could be in $t = T$.

Case 1 Consider the generator provided in table 3.1. Consider a demand of $L = \begin{pmatrix} 10 & 10 \end{pmatrix}$[MWh]. Hence the time horizon is $[T] = \{1, 2\}$ and $T = 2$.
To meet the demand at $t = 2$, the production should be 10[MW] at $t = 2$. Note that generator can ramp up to 10[MW]. Therefore compute the term b in the time interval $[a = 1, b]$ such that there is a production of 10[MW] at $t = 2$.

From constraint 3.4g :

$$p_{t=2}^{[a=1,b]} = 10 \leq SD + (b - t)RD = 6 + (b - 2)5 \iff b \geq \frac{10 - 6}{5} + 2 = 2.8$$

It leads to $b \geq 2.8 \implies b = 3 = T + 1$. The proposed solution works in this case.

Case 2 Consider same generator provided in table 3.1. Consider now a demand $L = \begin{pmatrix} 20 & 20 & 20 \end{pmatrix}$. Time horizon is $[T] = \{1, 2, 3\}$.

In this case production starts at $t = a = 1$ with start-up level that is 6[MW], i.e. $p_{t=1}^{[a=1,b]} = 6$[MW].
At $t = 2$ production increases at most of ramp-up RU, $p_{t=2}^{[a=1,b]} = p_{t=1}^{[a=1,b]} + RU = 6 + 5 = 11$[MW].
At $t = 3$ production increases again at most of ramp-up RU, i.e. $p_{t=3}^{[a=1,b]} = p_{t=2}^{[a=1,b]} + RU = 11 + 5 = 16$[MW].
As previously computed the term b in the time interval $[a = 1, b]$ such that there is a production of 16[MW] at $t = 3$.

From constraint 3.4g :

$$p_{t=3}^{[a=1,b]} = 60 \leq SD + (b - t)RD = 6 + (b - 3)5 \iff b \geq \frac{16 - 6}{5} + 3 = 5$$

It leads to $b \geq 5 \implies b = 5 \neq T + 1 = 4$. The proposed solution does not work in this case.

Same reasoning can be made to show that considering prior intervals $[a = 0, b]$ is not always sufficient to represent prior production.

The proposed solution can not represent any prior or posterior productions. Therefore it is imposed for all methods in the next chapters and for the extended formulation, that prior and posterior productions to time horizon $[T]$ are a null production. There are neither committed generator at the beginning of time horizon $[T]$, nor committed generators provided to a possible next time horizon.

At the beginning of the time horizon generators need to be started. The production increases step by step. At the end of the time horizon generators need to be shutted down, and before being shutted down they must decrease step by step. This is why at the beginning and at the end of the time horizon prices may be high, i.e. equal to $VOLL$, because all demand can not be meet.

Chapter 4

Row Generation based on Extended Formulation

The first improved method providing CHP is a **Row Generation** (RG) method. More precisely it is a Bender decomposition. The master program is the linear relaxation of the three-bin formulation. Slave programs are built for each generator. They model the convex hull of the three-bin formulation Π^g_{3-bin}, related to the considered generator. Required cuts are added using slave programs based on a master solution until convergence of the method, i.e. for a given master solution no cuts have to be added.

4.1 Bender Decomposition : Basic Principles

Bender master problem is denoted M^{RG}_{LP}. It considers a relaxation of Π^g_{3-bin} as feasible region. This relaxation is denoted $\mathcal{R}\left(\Pi^g_{3-bin}\right)$. Assume this relaxation is more compact than $\text{conv}\left(\Pi^g_{3-bin}\right)$ and is less tight than $\text{conv}\left(\Pi^g_{3-bin}\right)$. It can be certified that a proposed master solution $(p^*, \bar{p}^*, u^*, v^*, w^*)$ is such that $(p^*, \bar{p}^*, u^*, v^*, w^*) \in \text{conv}\left(\Pi^g_{3-bin}\right)$, in one of the three ways,[18].

(i) For generator g, $\mathcal{R}\left(\Pi^g_{3-bin}\right) = \text{conv}\left(\Pi^g_{3-bin}\right)$.
In such a case it is not needed to have slave Bender programs since the relaxation is already the convex hull of Π^g_{3-bin}. As detailed in section **Continuous Linear Relaxation of the three-bin Formulation** 2.3, under general conditions relaxation $\mathcal{R}\left(\Pi^g_{3-bin}\right) =^{\mathcal{R}} \Pi^g_{3-bin}$ does not provide the convex hull $\text{conv}\left(\Pi^g_{3-bin}\right)$.

(ii) For generator g, it may be that (u^*, v^*, w^*) are binary values in the master programs solution.

Because $\mathcal{R}\left(\Pi^g_{3-bin}\right)$ and $\text{conv}\left(\Pi^g_{3-bin}\right)$ must agree when (u^*, v^*, w^*) are binary values, this certifies $(p^*, \bar{p}^*, u^*, v^*, w^*) \in \text{conv}\left(\Pi^g_{3-bin}\right)$. Again slave Bender programs are not needed.

(iii) If (i) and (ii) are false for generator g, then the master solution must be tested $(p^*, \bar{p}^*, u^*, v^*, w^*) \in \text{conv}\left(\Pi^g_{3-bin}\right)$ through a Bender slave program. If the master solution is not in $\text{conv}\left(\Pi^g_{3-bin}\right)$, then the slave program generates a valid cut of the form

$$(\epsilon^*)^T p_g + (\mu^*)^T \bar{p}_g + (\xi^*)^T u_g + (\alpha^*)^T v_g + (\sigma^*)^T w_g \leq \nu$$

The slave programs use the master solutions as fixed parameters and minimize the distance to feasibility considering the feasible region $\text{conv}\left(\Pi^g_{3-bin}\right)$, at each iteration. If it is not feasible, i.e. the distance to feasibility is strictly positive, then it eliminates the point $(p^*, \bar{p}^*, u^*, v^*, w^*)$ from $\mathcal{R}\left(\Pi^g_{3-bin}\right)$, using the dual information from the slave program related to generator g to build a linear cut. The linear cut is added to the master program M^{RG}_{LP} in order to not choose the same point again.

4.2 Master Program, Slave Programs and Row Generation

The big issue when using the extended formulation for the three-bin formulation is that it needs a huge number of variables and constraints at once. Row generation is an iterative algorithm converging towards the maximum dual Lagrangian. It reduces the number of variables and constraints used to model the optimization programs to be solved. All variables and constraints are not simultaneously used at once. It also speeds up the process of computing CHP.

4.2.1 Description of Master Program

The master program M^{RG}_{LP} is the continuous Linear Program LP 2.5. As described in a previous section the linear relaxation of the three-bin formulation ${}^{\mathcal{R}}\Pi_{3-bin}$ is not the convex hull of the three-bin formulation, but it is a superset of it. Hence optimal solution provided by master program without cuts does not generally provides the CHP.

Master program has one main advantage; it is more compact than the extended formulation. There are much less variables and constraints than the extended formulation. Continuous linear program has a number of variables and constraints

linear in T, from proposition 4.1. The extended formulation uses polytope D, that is compact but with a polynomial size in T.

Considering the Belgian system of 68 generators, a time horizon over 24 hours with a timestep of 1 hour. On the one hand, EF needs the 607,446 variables and the 2,080,438 constraints, describe in table 3.5, to be modelled. On the other hand, the linear relaxation of the three-bin formulation ${}^{\mathcal{R}}\Pi_{3-bin}$ needs 8,864 variables and 26,697 constraints. The difference of constraints and variables between the two optimization programs is explicit.

Proposition 4.1. *Continuous linear program 2.5 has a number of variables and a number of constraints linear in size of T.*

Proof. See Appendix 9.3.10. □

The master program of Bender decomposition is defined by expressions 4.1.

Master Program - Row Generation (M_{LP}^{RG})

$$M_{LP}^{RG} \equiv \textbf{min} \left\{ \sum_t \left[\sum_{g \in \mathcal{G}} c^g \left(p_g, u_g, v_g, w_g \right) \right] - VOLL \sum_t l_t \right\} \tag{4.1a}$$

$$\text{s.t } (\lambda_t) \sum_g p_g^t - l_t = 0 \qquad \forall t \in [T] \tag{4.1b}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{4.1c}$$

$$(p, \bar{p}, u, v, w) \in \Pi_{3-bin}^g \qquad \forall g \in \mathcal{G} \tag{4.1d}$$

$$(u_g, v_g, w_g) \in [0,1]^{3T} \qquad \forall g \in \mathcal{G} \tag{4.1e}$$

$$(u_g, v_g, w_g, p_g, \bar{p}_g) \in \mathcal{C}^g \qquad \forall g \in \mathcal{G} \tag{4.1f}$$

Let v^{EF}, $v^{M_{LP}^{RG}}$ and v^{LP} respectively represent the objective function of the extended formulation, the master of Bender decomposition and the linear relaxation of Π_{3-bin}^g. Because the extended formulation is at least as tight as the three-bin formulation, then $v^{EF} \geq v^{RG}$. Adding cuts $\mathcal{C}^g$ tights the feasible region of LP and it is clear that $v^{EF} \geq v^{RG} \geq v^{LP}$.

After adding all required cuts, the master program M_{LP}^{RG} converges towards EF. After convergence of Bender decomposition algorithm then $EF = M_{LP}^{RG}$.

4.2.2 Description of Slave Program

The slave program related to a given generator g is a **Cut-Generating Linear Program** ($CGLP$). Those programs are proceed independently for each generator. It verifies whether the master solution $(p^*, \bar{p}^*, u^*, v^*, w^*)$ is feasible or not in the convex hull, by computing its distance to feasibility in the extended formulation. If the distance is strictly positive, then the slave program generates a cut to the feasible region of the master program to restrict it.

$CGLP$ is defined by expressions 4.2. Variable z represents the distance to feasibility of the master solution to the extended formulation. Constraints 4.2c and 4.2d represent the convex hull description given by polytope D. Constraints 4.2e, 4.2f, 4.2g, 4.2h and 4.2i define linear transformation F described in previous section.

Slave Program : Cut Generating Linear Program ($CGLP$) - Row Generation

$$CGLP \equiv \quad \textbf{min } z \tag{4.2a}$$

s.t. (4.2b)

$$\pi \quad A[a,b]p^{[a,b]} + \bar{A}^{[a,b]}\bar{p}^{[a,b]} \leq \gamma_{[a,b]}\bar{b}^{[a,b]} + ze \quad \forall [a,b] \in \mathcal{T} \tag{4.2c}$$

$$\delta \quad \sum_{\{[a,b]\in\mathcal{T} | t\in[a,b+DT^g]\}} \gamma_{[a,b]} \leq 1 + z \quad \forall t \in [T] \tag{4.2d}$$

$$\epsilon \quad \sum_{\{[a,b]\in\mathcal{T}\}} p^{[a,b]} = p^* \tag{4.2e}$$

$$\mu \quad \sum_{\{[a,b]\in\mathcal{T}\}} \bar{p}^{[a,b]} = \bar{p}^* \tag{4.2f}$$

$$\xi \quad \sum_{\{[a,b]\in\mathcal{T} | t\in[a,b]\}} \gamma_{[a,b]} = u_t^* \quad \forall t \in [T] \tag{4.2g}$$

$$\alpha \quad \sum_{\{[a,b]\in\mathcal{T} | t=a\}} \gamma_{[a,b]} = v_t^* \quad \forall t \in [T] \tag{4.2h}$$

$$\sigma \quad \sum_{\{[a,b]\in\mathcal{T} | t=b+1\}} \gamma_{[a,b]} = w_t^* \quad \forall t \in [T] \tag{4.2i}$$

$$z \in \mathbb{R}_+; p^{[a,b]}, \bar{p}^{[a,b]} \in \mathbb{R}_+^T, \gamma_{[a,b]} \in \mathbb{R}_+ \quad \forall [a,b] \in \mathcal{T} \tag{4.2j}$$

Variables π, δ, ϵ, μ, ξ, α and σ are dual variables of corresponding constraints and e is a vector full of ones of the appropriated size. There are two possible cases for a master solution processed by $CGLP$:

- If $z^* = (\delta^*)^T e + (\epsilon^*)^T p^* + (\mu^*)^T \bar{p}^* + (\xi^*)^T u^* + (\alpha^*)^T v^* + (\sigma^*)^T w^* = 0$ then $(p^*, \bar{p}^*, u^*, v^*, w^*)$ is feasible in the extended formulation.
  Nothing has to be done. No cut is needed for the considered generator.

- Else $z^* = (\delta^*)^T e + (\epsilon^*)^T p^* + (\mu^*)^T \bar{p}^* + (\xi^*)^T u^* + (\alpha^*)^T v^* + (\sigma^*)^T w^* > 0$. A cut is needed for solution $(p^*, \bar{p}^*, u^*, v^*, w^*)$.
  This cut has to be applied on the feasible region of the master program. The cut is given by

$$\text{cut}^g : (\delta^*)^T e + (\epsilon^*)^T p_g + (\mu^*)^T \bar{p}_g + (\xi^*)^T u_g + (\alpha^*)^T v_g + (\sigma^*)^T w_g \leq 0 \quad (4.3)$$

4.2.3 Bender Decomposition Algorithm

Bender decomposition algorithm repeatedly computes a master solution, and then it adds cuts if needed. It is an outer method of primal EF. It starts with a bigger feasible region than EF. The cuts reduce the feasible region of the master program to restrict to the convex hull of the explored region by the solver. Hence adding cuts increases, not necessarily strictly, the objective value of the master program since it is a minimization.

At the beginning of Bender decomposition set of cuts $\mathcal{C}^g := \emptyset$ since there are no cuts computed yet. At each iteration of Bender decomposition, the master program computes a master solution in feasible set set ${}^R\Pi^g_{3-bin} \cap \mathcal{C}^g$, which is simply ${}^R\Pi^g_{3-bin} \cap \emptyset = {}^R\ \Pi^g_{3-bin}$ at first iteration. This first step provides a solution that might or not be feasible in the extended formulation. Hence the second step uses slave programs for one generator at a time to check whether a cut needs to be built or not. If a cut needs to be built then $\mathcal{C}^g$ is updated such that $\mathcal{C}^g := \mathcal{C}^g \cap \text{cut}^g$. Once master solution is checked for all generator then it goes to next iteration. If the master solution did not add a cut to a generator, then the master solution is feasible for the extended formulation and Bender decomposition has converged. Illustrative representation of the algorithm is provided on figure 4.1.

Remark 4.1 (Separable Problem)**.** *Bender decomposition algorithm uses separability characteristic of UC. It considers the convex hull description of one generator at a time to check whether master solution is feasible or not. It enables to have one slave program per generator and not to consider all generators at a same time.*

One may notice that the total number of variables and constraints does not get smaller than for the extended formulation because all generators need a corresponding slave program. However what gets smaller is the number of variables and constraints involved simultaneously in an optimization program. Polytope D that is providing a convex hull description for a given generator does not need to

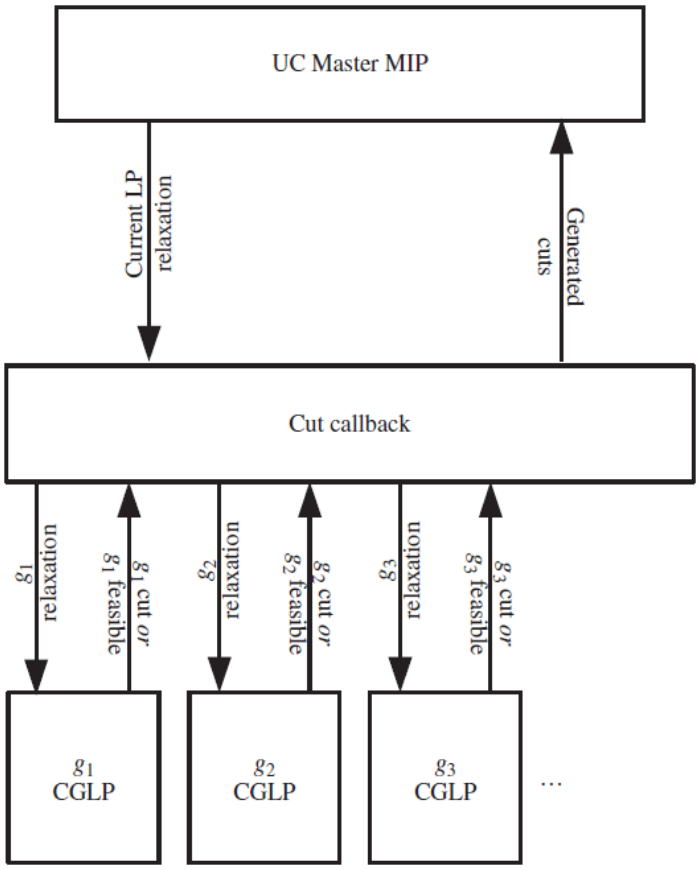


Figure 4.1: Illustrative representation of Bender decomposition,[16].

$\mathcal{C}^g = \emptyset \quad \forall g \in \mathcal{G}$;
for *iter in 1..iterMax* **do**
  Solve RG : $(p^*, \bar{p}^*, u^*, v^*, w^*) \longleftarrow M_{LP}^{RG}(C^g)$;
  for *g in* $\mathcal{G}$ **do**
    Compute the cut for generator g :
    $(z^*, \delta^*, \epsilon^*, \mu^*, \xi^*, \alpha^*, \sigma^*) \longleftarrow CGLP\left(p_g^*, \bar{p}_g^*, u_g^*, v_g^*, w_g^*\right)$;
    if $z^* > 0$ **then**
      Build cut cut^g :
      $(\delta^*)^T e + (\epsilon^*)^T p + (\mu^*)^T \bar{p} + (\xi^*)^T u + (\alpha^*)^T v + (\sigma^*)^T w \leq 0$;
      Add cut : $\mathcal{C}^g := \mathcal{C}^g \cap \text{cut}^g$;
    end
  end
  if *No cut added for any generator* **then**
    Optimal solution found : Stop;
  end
end

Algorithm 1: Bender Decomposition

consider other generator. Slave programs can be processed independently of each other. They are processed one at a time. Much less constraints and variables are involved in each iteration of row generation.

Regarding the generation of slave programs there is a choice to make.

- First approach is to build all slave programs only once and store them into memory. In that case even if less variables and constraints are involved in each iteration of row generation than for EF, all constraints and variables are still stored in memory. Depending on size of $\mathcal{G}$ and $[T]$ this solution might not be always possible.
- Second approach is to build a slave program each time it is needed. In that way it is possible to solve problems considering bigger $\mathcal{G}$ and $[T]$. However it wastes time because same problems may be build several times.

What has been done is to apply first approach, store slave problems in memory, when possible. When it is not possible second approach is applied

4.3 Examples

For iterative methods performances indicator RT and ST accumulate time at each iteration.

First example is a small example to illustrate how row generation add cuts and converge towards maximal dual Lagrangian. Second example is a bigger one and gives an idea of the performances of the different methods.

Small Example : One Generator on Four Hours

Example 4.1 (One Generator Example : Row Generation)**.** *Consider the same market as the one defined in tables 3.1 and 3.2, i.e. one generator and four hours. Table 4.1 shows the iterates of row generation.*
At iteration 1, as expected row generation has the exact same objective value and prices as LP of previous chapter, since no cut has been added yet. The master solution from M_{LP}^{RG} provides the following optimal solution to CGLP.

$$\begin{aligned} p^* &= \begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix} \\ \bar{p}^* &= \begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix} \\ u^* &= \begin{pmatrix} 1 & 0.6875 & 1 & 0 \end{pmatrix} \\ v^* &= \begin{pmatrix} 1 & 0 & 0.3125 & 0 \end{pmatrix} \\ w^* &= \begin{pmatrix} 0 & 0.3125 & 0 & 1 \end{pmatrix} \end{aligned}$$

Slave program CGLP adds the following cut to M_{LP}^{RG}.

$$-1.1u_{g_1}^{t=1} + 0.1\bar{p}_{g_1}^{t=2} - 1.6v_{g_1}^{t=2} + 1.1w_{g_1}^{t=2} - 0.5u_{g_1}^{t=4} \leq 0$$

At iteration 2, M_{LP}^{RG} provides the following solution to CGLP. For this master solution no cut needs to added. Row generation has converged to maximal dual Lagrangian. Objective value of RG is the same as EF.

$$\begin{aligned} p^* &= \begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix} \\ \bar{p}^* &= \begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix} \\ u^* &= \begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix} \\ v^* &= \begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix} \\ w^* &= \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Prices of LP, EF and RG for this example are provided in table 4.2. CHP are not unique. Prices from EF and RG are not exactly the same but they both lead to maximal dual Lagrangian.

Figure 4.2 shows the dual Lagrangian function according to prices of hour 1 and hour 2. Prices from EF or RG provide higher dual Lagrangian value than LP. Price iterates in table 4.1 are represented on the dual Lagrangian function. One can see iterates converge towards $\boldsymbol{max}_{\lambda} L(\lambda)$.

Iteration	Objective value of Row Generation method	Prices λ
1	-67,434.1875 [€]	$\begin{pmatrix} 88.8333 & 45.4375 & 3000 & 3000 \end{pmatrix}$
2	-67,357 [€]	$\begin{pmatrix} 98.7121 & 52.4545 & 3000 & 3000 \end{pmatrix}$

Table 4.1: Iterates of row generation method for example 4.1.

Figure 4.3 shows the uplifts according to price of hour 1 and hour 2. The two CHP from EF and RG provide 0[€] uplifts and prices from LP provide 77.187[€] uplifts. The prices iterates from row generation converge towards minimum uplifts, that is 0[€] at iteration 2 in this case.

Method	CHP ?	Prices [€/MWh]
LP	No	$\begin{pmatrix} 88.8333 & 45.4375 & 3000 & 3000 \end{pmatrix}$
EF	Yes	$\begin{pmatrix} 30 & 90 & 3000 & 3000 \end{pmatrix}$
RG	Yes	$\begin{pmatrix} 98.7121 & 52.4545 & 3000 & 3000 \end{pmatrix}$

Table 4.2: Prices from different methods for example 4.1.

Performances are described in table 4.3. For this small example EF is faster than RG to compute CHP, but it is not really representative of a real-world electricity market.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-67,357	/	0.013	0.001	/
LP	-67,434.187	77.187	0.013	0.001	/
EF	-67,357	0	0.016	0.0038	/
RG	-67,357	0	4.052	0.015	2

Table 4.3: Description of uplifts and their performances using LP and EF methods to compute prices for example 4.1.
RT : total Running Time of the program.
ST : Solving Time of the program.

Real-World Example

Example 4.2 (Winter-WE : Row Generation)**.** *Consider Belgian generators described in tables 9.5, 9.6 and the demand $Winter - WE$ on a time horizon of 24*

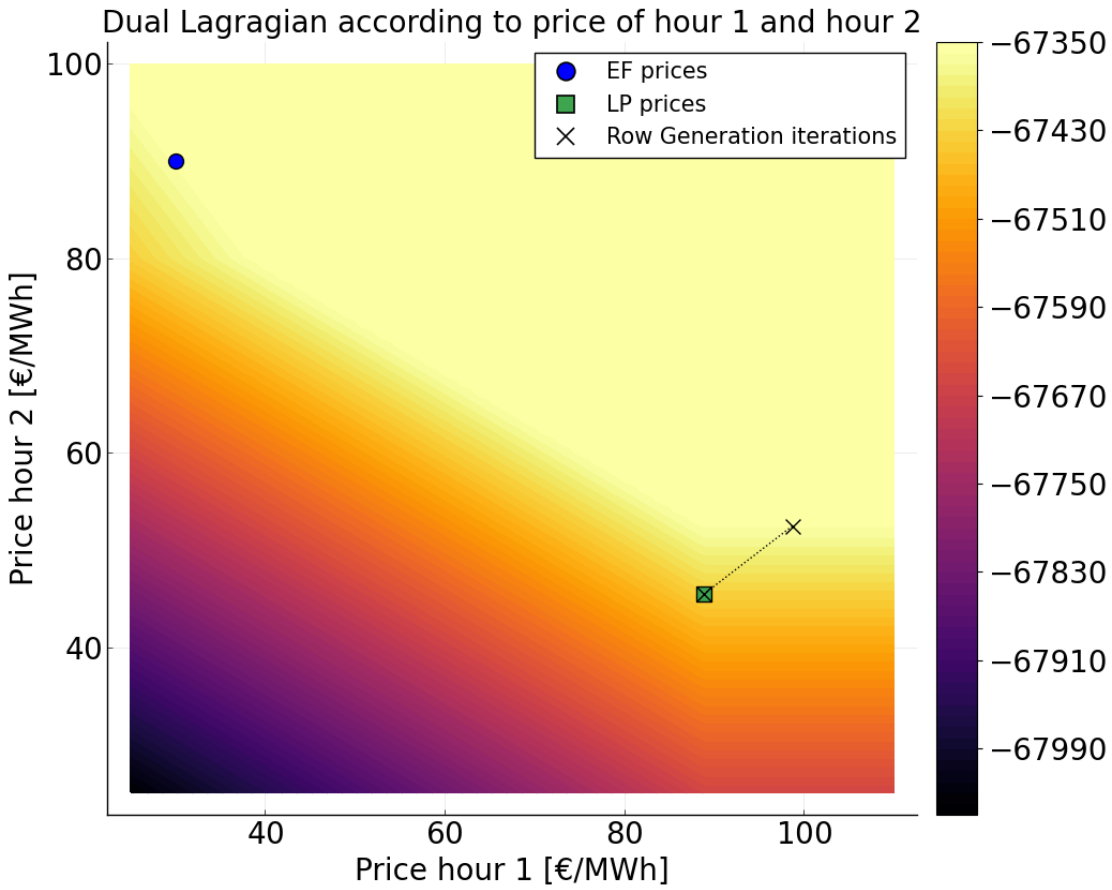


Figure 4.2: Dual Lagrangian function according to price hour 1 and hour 2 for example 4.1.

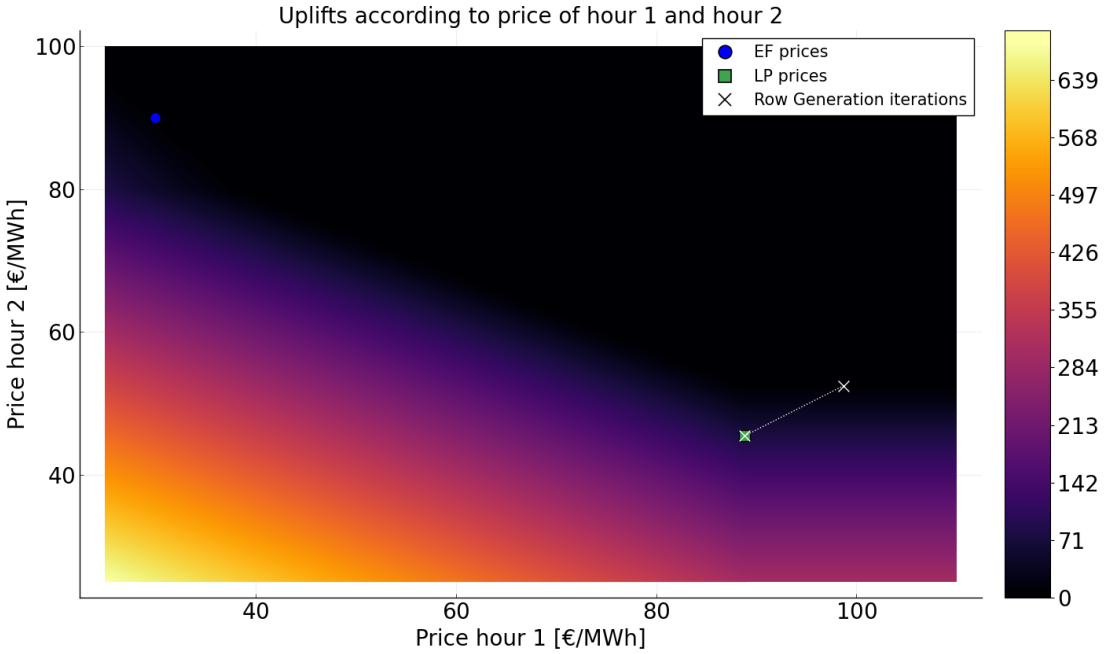


Figure 4.3: Uplifts according to price of hour 1 and hour 2 for example 4.1.

hours with a timestep of 1 hour, i.e. $[T]$ is of length 24.

For this real-worl electricity market, it is possible to build all slave programs CGLP and keep them in memory. Row generation converges after 31 iterations, as one can see on figure 4.5. RT and ST are respectively 123.585[s] and 77.746[s], and are reported in table 4.4. There is an improvement from row generation of 50% for RT and 60% for ST, based on the extended formulation. Row generation method is already having interesting results for the data sample of this example.

Row generation with a rebuilding of slave programs has also been performed. It provides the same results for the objective value and the uplifts, but not with the same performances. As expected rebuilding slave programs takes more time and it is visible in both RT and ST. This rebuilding impacts more RT than ST. Indeed row generation with rebuilding is 8.84 times slower than without rebuilding, in terms of RT, and it is 4.16 times slower than without rebuilding in terms of ST.

Figure 4.4 shows the convergence of RG towards the maximum dual Lagrangian, iteration after iteration. It also shows the enhancement each iteration provides to the objective value of M_{LP}^{RG} in order to reach the maximum dual Lagrangian value. Figure 4.5 shows the uplift for each generator according to CHP from EF and RG, and price from LP. It is clear that total amount of uplifts is lower for EF and RG than for LP.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-607,224,305	/	1.128	0.699	/
LP	-607,435,576	35,550	1.940	0.205	/
EF	-607,245,477	21,171	242.347	197.293	/
RG	-607,245,477	21,171	123.585	77.746	31
RG re-build	-607,245,477	21,171	1092.884	325.807	31

Table 4.4: Uplifts and performances using LP, EF and RG methods to compute prices for example 4.2.
RT : total Running Time of the program.
ST : Solving Time of the program.

4.4 Hyper-Parameter of RG

In the row generation algorithm a cut is added if $z^* > 0$. Due to floating point errors, or because one is not interested to find the exact CHP but is satisfied with approximate prices, the condition to add a cut might be formulated such that $z^* > \tau_{RG}$. Hyper-parameter τ_{RG} has an impact on the behaviour of RG and on the optimal solution of row generation.

Figure 4.6 shows the objective value and uplifts after converge of row generation for different values of τ_{RG}. While τ_{RG} is increasing the objective value is decreasing and the sum of uplifts is increasing. Being less strict when adding cut does not exactly describe convex hull. Feasible region is bigger than convex hull and objective value can thus reach a smaller objective value than the maximal dual Lagrangian, causing uplifts to be higher. For $\tau_{RG} \geq 0.23$ the objective value of row generation v^{RG} is simply the one of the linear relaxation v^{LP}. One can see on figure 4.8 for $\tau_{RG} \geq 0.23$ that only one iteration is performed. No cuts are added to master program.

Figure 4.7 shows the total running time RT and solving time ST needed for row generation to converge. Both RT and ST are correlated to the number of iterations on figure 4.8 to converge. The more hyper-parameter τ_{RG} is strict, i.e. close to 0, the more cuts will be added, the more RT and ST will be big. Inversely, the more τ_{RG} is loose, i.e. close to 1, the less cuts are added, the more RT and ST are small. For $\tau_{RG} \geq 0.23$ no cuts are added and thus RT and St stay quite constant. For such τ_{RG}, it converges in 1 iteration.

The objective of this master thesis is to compute exactly CHP therefore $\tau_{RG} = 0$, therefore a cut is added if $z^* > 0$. The reader is aware that the behaviour of RG might change with this hyper-parameter τ_{RG}.

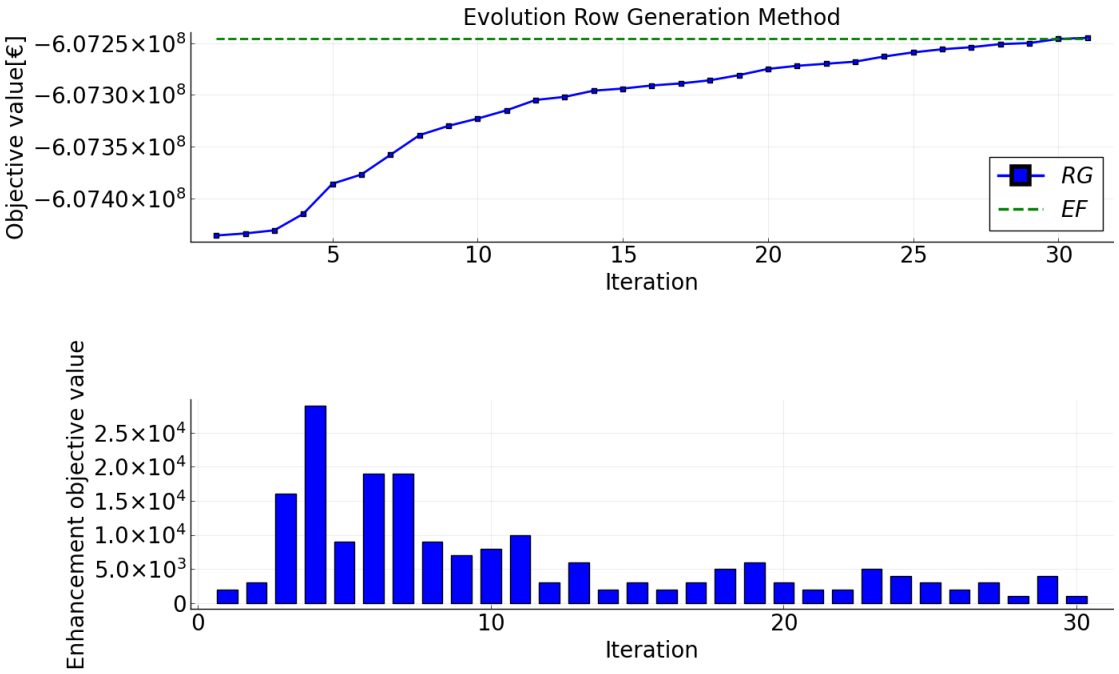


Figure 4.4: Evolution of row generation method and the impact of each iteration on the objective value.

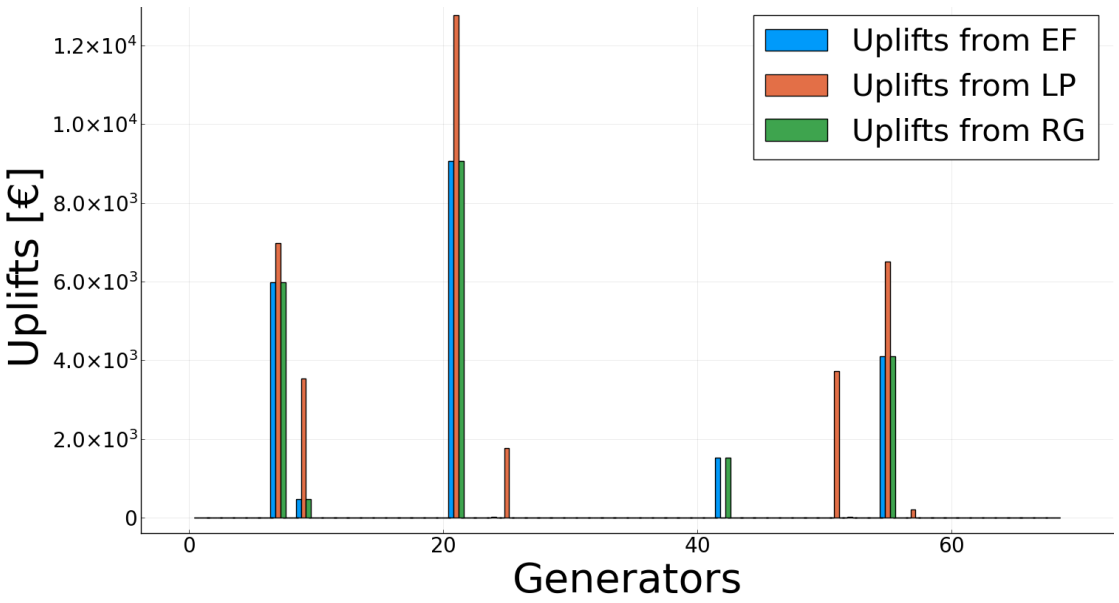


Figure 4.5: Uplift for each generator with CHP from EF and RG and prices from LP.

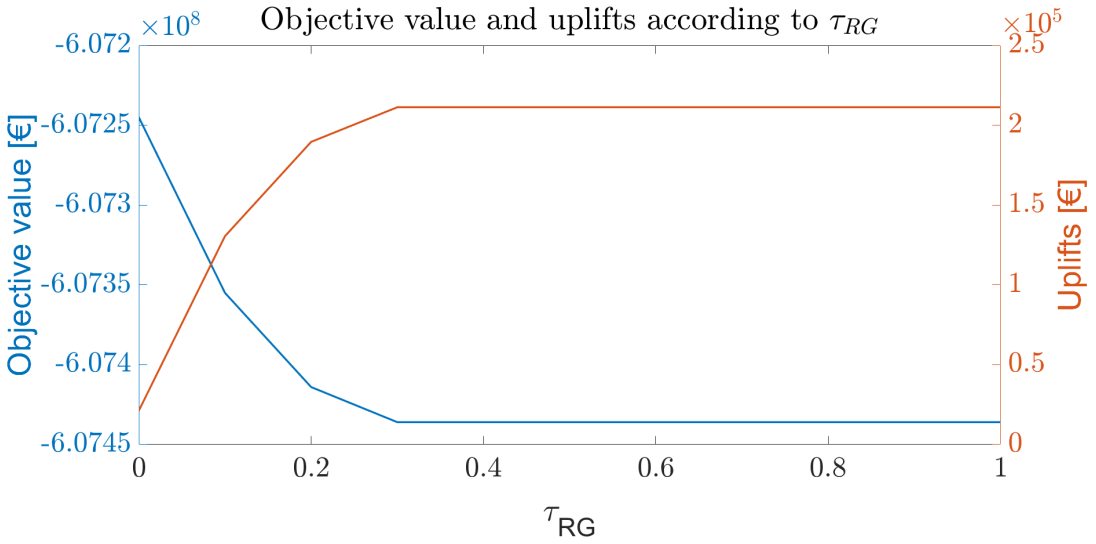


Figure 4.6: Objective value and uplifts for real-world example $Winter - WE$ considering different values of τ_{RG}.

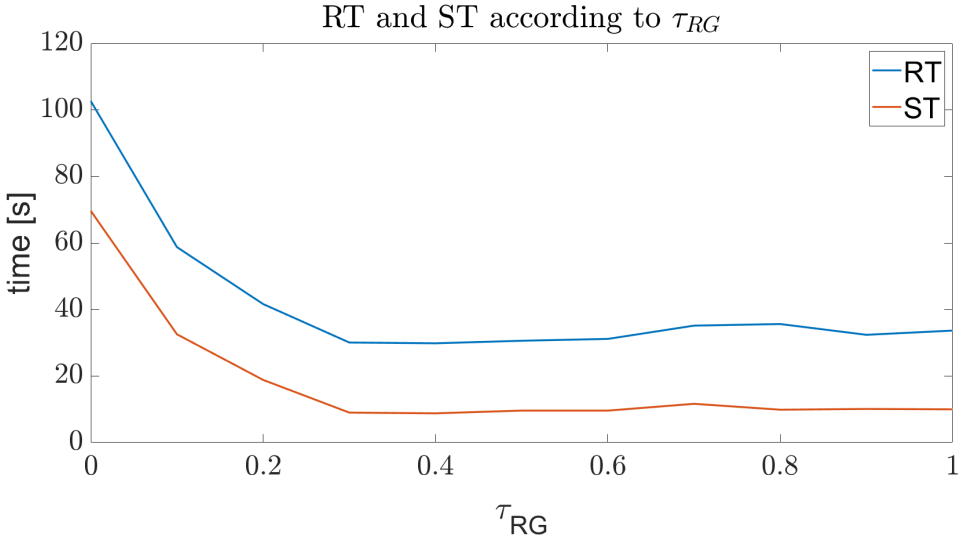


Figure 4.7: RT and ST for real-world example $Winter - WE$ considering different vaues of τ_{RG}.
RT : total Running Time of the program.
ST : Solving Time of the program.

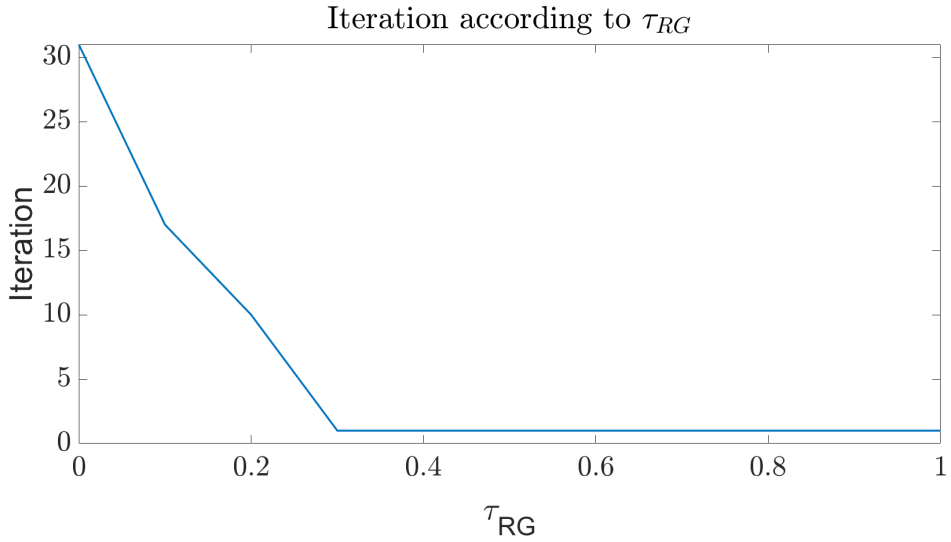


Figure 4.8: Number of iterations row generation needs to converge for real-world example $Winter - WE$ considering different vaues of τ_{RG}.

Chapter 5

Column Generation

The second improved method to compute CHP is a **Column Generation** (CG) method based on Dantzig-Wolfe decomposition, [1]. Promising feasible schedules are added to master program to improve current solution. Those schedules are feasible points of Π^g_{3-bin}. Convex hull of Π^g_{3-bin} is not defined using extended solution but using convex combinations between feasible points.

5.1 Dantzig-Wolfe Decomposition : Basic Principles

Source [26] provides a detailed general Dantzig-Wolfe Decomposition. The basic principles are the following ones.

Recall that the general MILP is $F = \mathbf{min}\ \{cx : Dx \geq d, x \in X\}$, where $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$ with polyhedron $P = \left\{x \in \mathbb{R}^{n+p}_+ : Bx \geq b\right\}$. Constraint $Dx \geq d$ represents a "complicate" constraint and set X represents tractable set.

Assume X is a bounded integer set. Let $\{x^g\}_{g \in G}$ be either extreme points of conv (X) or all points of X. The master program of Dantzig-Wolfe decomposition M for a general optimization program is given by expression 5.1. Its linear relaxation is M_{PL} 5.2. It is equivalent to solve either M 5.1 or F.

$$M \equiv \mathbf{min} \sum_{g \in G} (cx^g)\, z_g \qquad (5.1a)$$
$$\text{s.t. } (\lambda) \sum_{g \in G} (Dx^g)\, z_g \geq d \qquad (5.1b)$$
$$(\nu) \sum_{g \in G} \lambda_t = 1 \qquad (5.1c)$$
$$z \in \{0,1\}^{|G|} \qquad (5.1d)$$

$$M_{LP} \equiv \mathbf{min} \sum_{g \in G} (cx^g)\, z_g \qquad (5.2a)$$
$$\text{s.t. } (\lambda) \sum_{g \in G} (Dx^g)\, z_g \geq d \qquad (5.2b)$$
$$(\nu) \sum_{g \in G} \lambda_t = 1 \qquad (5.2c)$$
$$z \in [0,1]^{|G|} \qquad (5.2d)$$

Observation 5.1. *It is equivalent to solve either M_{LP} or optimization program* $\boldsymbol{min}\{cx : Dx \geq d, x \in conv(X)\}$.

From observation 5.1 one may want to solve directly M_{LP} 5.2. However solving the whole linear program M_{LP} is not a reasonable option since it means to provide an exhaustive enumeration of all x^g. Hence suppose at each iteration t of the simplex algorithm only a subset of the points $\{x^g\}_{g \in G^t}$ available. Optimization program considering only such a subset $G^t \subseteq G$ is the restricted master program $\bar{M}$. Its linear relaxation $\bar{M}_{LP}$ given by expressions 5.3 and its dual D is given by expressions 5.4. Let z' and (λ', ν') represent respectively the primal and dual solutions of the restricted master program.

Observation 5.2 (From source [26])**.** *There are three important observations to build column generation algorithm.*

(i) *Given a current dual solution (λ', ν'), the reduced cost of the column associated to solution x^g is $cx^g - \lambda' Dx^g - \nu'$.*

(ii) *$\xi = min_{g \in G}(cx^g - \lambda' Dx^g) = min_{x \in X}(c - \lambda' D)\, x$. Not needed to examine reduced costs on a huge number of columns, it can be carried out by solving a single integer program over X.*

(iii) *The objective value of the restricted master problem $v^{\bar{M}} = \sum_{g \in G^t}(cx^g) z'_g = \lambda' d + \nu'$ gives an upper bound on v^M. M 5.1 is solved when $\xi - \nu' = 0$, i.e. when there is no column with negative cost.*

$$\bar{M}_{LP} \equiv \mathbf{min} \sum_{g \in G^t} (cx^g)\, z_g \qquad (5.3a)$$
$$\text{s.t. } (\lambda) \sum_{g \in G^t} (Dx^g)\, z_g \geq d \qquad (5.3b)$$
$$(\nu) \sum_{g \in G^t} z_g = 1 \qquad (5.3c)$$
$$z \in [0,1]^{|G^t|} \qquad (5.3d)$$

$$D \equiv \mathbf{max}\ \lambda d + \nu \tag{5.4a}$$
$$\text{s.t. } \lambda D x^g + \nu \leq c x^g \qquad \forall g \in \mathcal{G}^t \tag{5.4b}$$
$$\lambda \geq 0, \nu \in \mathbb{R} \tag{5.4c}$$

5.1.1 Application to Unit Commit Problem

General formulations previously detailed can be adapted to UC problem in order to find $\mathbf{max}_\lambda \{L(\lambda)\}$, that is the initial goal.

Assume the set Π^g_{3-bin} has a finite number of feasible schedules, defined by 5.1, then UC is equivalent to the Set Partitioning Problem (SPP) 5.5, from proposition 5.1. Let $n_g \in \mathcal{N}_g$ be an index to define feasible schedule $(\hat{p}_g^{n_g}, \hat{\bar{p}}_g^{n_g}, \hat{u}_g^{n_g}, \hat{v}_g^{n_g}, \hat{w}_g^{n_g})$ of generator g. Each variable $z_g^{n_g}$ is related to a feasible schedule $\left(\hat{p}_g^{n_g}, \hat{\bar{p}}_g^{n_g}, \hat{u}_g^{n_g}, \hat{v}_g^{n_g}, \hat{w}_g^{n_g}\right)$. For simplicity in let $\hat{c}_g^{n_g} := c\left(\hat{p}_g^{n_g}, \hat{\bar{p}}_g^{n_g}, \hat{u}_g^{n_g}, \hat{v}_g^{n_g}, \hat{w}_g^{n_g}\right)$. Then SPP 5.5 is an integer linear program selecting optimal schedules to minimize the total welfare.
It is important to note that feasible schedules are parameters of the optimization program.

Definition 5.1 (Feasible Schedule)**.** *A feasible schedule for generator g is a set of fixed variables $(\hat{p}_g, \hat{\bar{p}}_g, \hat{u}_g, \hat{v}_g, \hat{w}_g)$ such that $(\hat{p}_g, \hat{\bar{p}}_g, \hat{u}_g, \hat{v}_g, \hat{w}_g) \in \Pi^g_{3-bin}$.*

Constraint 5.5b defines the balance constraint. The total production is the sum of all feasible schedules for all generators, and it has to meet the demand l_t. A given schedule might represent a zero production. Constraint 5.5c requires for each generator to select exactly one schedule. Constraint 5.5d imposes $z_g^{n_g}$ to be a binary variable, i.e. a schedule is chosen or not. Constraint 5.5e defines bounds on the demand l_t that is meet be the market.

Set Partitioning Problem (SPP)

$$\mathbf{min}_{z_g^{n_g}, l_t} \sum_{g \in \mathcal{G}} \left[\sum_{n_g \in \mathcal{N}_g} \left(\hat{c}_g^{n_g} \cdot z_g^{n_g} \right) \right] - VOLL \sum_t l_t \tag{5.5a}$$

$$\text{s.t.} \sum_{g \in \mathcal{G}} \sum_{n_g \in \mathcal{N}_g} \hat{p}_g^{n_g} = l_t \qquad \forall t \in [T] \tag{5.5b}$$

$$\sum_{n_g \in \mathcal{N}_g} z_g^{n_g} = 1 \qquad \forall g \in \mathcal{G} \tag{5.5c}$$

$$z_g^{n_g} \in \{0, 1\} \qquad \forall g \in \mathcal{G}, \forall n_g \in \mathcal{N}_g \tag{5.5d}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{5.5e}$$

Proposition 5.1. *SPP is equivalent to UC.*

Proof. See Appendix 9.3.11. □

From proposition 5.1 SPP and UC are equivalent. UC optimizes over continuous variables and SPP selects the schedules leading to optimality among all possible feasible schedules.

Using SPP to solve UC does not seem reasonable since it would require an exhaustive enumeration of all feasible schedules in order to solve a huge integer linear program that is 5.5. However SPP is not considered as optimization to be solved instead of UC. It is a basis for restricted master program $\bar{M}_{LP}$ of column generation method.

Reminder of definition 2.1; assuming there is a finite number of elements in set S, its convex hull is the set containing convex combinations of elements in S enumerated such that $x^1, x^2, ..., x^{|S|}$. The restricted master program $\bar{M}_{LP}$ of column generation considers continuous linear variables $z_g^{n_g}$ allowing to make convex combinations of feasible schedules. Those feasible schedules are provided by slave programs and are elements of $S := \Pi_{3-bin}^g$. Column generation is building $\text{conv}\left(\Pi_{3-bin}^g\right)$ by convex combinations between the schedules in the direction the solver is looking for a solution.

$$\text{conv}\,(S) = \left\{ x : \sum_{i=1}^{|S|} \mu_i x^i \,\middle|\, \sum_{i=1}^{|S|} \mu_i = 1, \mu_i \geq 0 \right\}$$

5.2 Restricted Master Program, Slave Programs and Column Generation

5.2.1 Restricted Master Program

The master program of column generation is $\bar{M}_{LP}$ 5.6. It considers fixed feasible schedules and can perform convex combinations with the schedules, i.e. $z_g^{n_g}$ is continuous linear variables. To make a parallel with source [26] given Dantzig-Wolfe principles, $z_g^{n_g}$ is equivalent to z_g in expressions 5.3. Feasible schedules are parameters for $\bar{M}_{LP}$, i.e. fixed values, and their equivalent is x^g in expressions 5.3. Promising feasible schedules are provided to $\bar{M}_{LP}$ by the slave programs in order to improve the current optimal solution.

Restricted Master Program - Column Generation

$$\bar{M}_{LP} \equiv \mathbf{min} \sum_{g \in \mathcal{G}} \left[\sum_{n_g \in \mathcal{N}_g} \left(\hat{c}_g^{n_g} \cdot z_g^{n_g} \right) \right] - VOLL \sum_t l_t \tag{5.6a}$$

$$\text{s.t.} \quad (\lambda_t) \sum_{g \in \mathcal{G}} \sum_{n_g \in \mathcal{N}_g} \hat{p}_g^{n_g} = l_t \qquad \forall t \in [T] \tag{5.6b}$$

$$(\nu_g) \sum_{n_g \in \mathcal{N}_g} z_g^{n_g} = 1 \qquad \forall g \in \mathcal{G} \tag{5.6c}$$

$$0 \leq z_g^{n_g} \qquad \forall g \in \mathcal{G}, \forall n_g \in \mathcal{N}_g \tag{5.6d}$$

$$0 \leq l_t \leq L_t \qquad \forall t \in [T] \tag{5.6e}$$

5.2.2 Slave Programs

Reduced cost in Dantzig-Wolfe principles from [26] are defined such that $rc = \mathbf{min}_{x \in X} \{(c - \lambda' D)\, x\} - \nu'$. Hence the reduced cost of a column can be defined for a generator g by expression 5.7.

$$rc_g = rc_g\left(p_g, \bar{p}_g, u_g, v_g, w_g\right) = \mathbf{min} \left\{ c_g\left(p_g, \bar{p}_g, u_g, v_g, w_g\right) - \sum_t \lambda_t p_g^t \right\} - \nu_g \tag{5.7}$$

Slave programs generate new feasible schedules that are actually new columns to be added to $\bar{M}_{LP}$, as long as they have a negative reduced cost $rc_g(p, \bar{p}, u, v, w) < 0$. Slave programs are built for each generator g and they are given by expression 5.8. λ_t and ν_g are constant terms in this program. They are provided from master program $\bar{M}_{LP}$ in which they are dual multipliers of the two constraints 5.6b and 5.6c.

Last term $-\nu_g$ can be removed during the optimization, as it is a constant, but it must remain for computing reduced cost. Reduced cost can be solved as an integer program for each generator from observation 5.2-(ii).

Slave Program - Column Generation

$$\textbf{min } c_g\left(p_g, \bar{p}_g, u_g, v_g, w_g\right) - \sum_t \lambda_t p_g^t - \nu_g \tag{5.8a}$$

$$\text{s.t. } \left(p_g, \bar{p}_g, u_g, v_g, w_g\right) \in \Pi_{3-bin}^g \tag{5.8b}$$

From proposition 5.2 a finite number of feasible schedules, such that their reduced cost is strictly negative, is required to be added to $\bar{M}_{LP}$.

Proposition 5.2 (From source 9.9)**.** *Only a finite number of iterations is required if each feasible solution is improved by introducing into the basis either an extreme point such that reduced cost* $rc_g(p, \bar{p}, u, v, w) < 0$ *or any homogenenous solution from the finite set* Π_{3-bin}^g *such that* $rc_g(p, \bar{p}, u, v, w) < 0$.

Proof. See Appendix 9.3.12. □

5.2.3 Column Generation Algorithm

Column generation is an inner method of primal EF. The column generation algorithm starts with an initialization of the feasible schedules defined by FS^g. A possible initalization is to estimate prices and then compute feasible schedules but the initalization implemented is to consider feasible schedules from the dispatch solution as initialization.

For each iteration, the master program $\bar{M}_{LP}$ is solved. It provides the dual multipliers λ_t and μ_g to the slave programs. Then for each generator g a slave program computes reduced cost rc_g 5.7. If the reduced cost is non-positive then it adds the corresponding feasible schedule $\left(\hat{p}, \hat{\bar{p}}, \hat{u}, \hat{v}, \hat{w}\right)$ as parameter to $\bar{M}_{LP}$. If the reduced costs are positive for all generator $g \in \mathcal{G}$, then the algorithm has converged and it stops, from observation 5.2-(iii). Those columns allow $\bar{M}_{LP}$ to perform convex combinations with points of set Π_{3-bin}^g to build convex hull conv $\left(\Pi_{3-bin}^g\right)$. Iteration after iteration, slave programs provide columns to build the convex hull in the direction $\bar{M}_{LP}$ is looking for a solution.

$FS^g \longleftarrow$ initialization feasible schedules $\forall g \in \mathcal{G}$;
for *iter in 1..iterMax* **do**
  Solve master : $(\lambda_t, \nu_g) \longleftarrow \bar{M}_{LP}\,(FS^g)$;
  for g *in* $\mathcal{G}$ **do**
    Compute reduced cost for generator g : $rc_g \longleftarrow Slave\,(\lambda_t, \nu_g)$;
    if $rc_g < -\tau_{CG}$ **then**
      Add feasible schedule to $\bar{M}_{LP}$: $FS^g := FS^g \cup \left(\hat{p}_g, \hat{\bar{p}}_g, \hat{u}_g, \hat{v}_g, \hat{w}_g\right)$;
    end
  end
  if *No negative reduced cost* **then**
    Optimal solution found : Stop;
  end
end

Algorithm 2: Column Generation : Dantzig-Wolfe Decomposition

$\bar{M}_{LP}$ involves less variables and constraints simultaneously than the initial optimization program EF computing CHP. Slave programs are actually MILP and the master program $\bar{M}_{LP}$ is a linear program. The expensive step in the algorithm is solving the slave programs. To avoid loosing time, the slaves programs are kept in memory when it is possible. They are built only once, at the beginning of the algorithm, and then λ_t and ν_g are updated at each iteration. $\bar{M}_{LP}$ is also built only once at the beginning of the algorithm. Then variables and their respective coefficients are added to the two existing constraints.

When it is not possible because the problem considered it too big either in the size of $\mathcal{G}$ or in the size of $[T]$, then slave program are re-built at each iteration. However for all examples presented through the manuscript, it was always possible to store the slave programs and $\bar{M}_{LP}$ in memory, and to only add or modify the stored programs at each iteration.

One may notice τ_{CG} instead of 0 in the condition to add column. This hyper-parameter will be explained later in the chapter. For now consider $\tau_{CG} = 0$.

5.3 Examples

First example illustrates how iterates of column generation converge to the optimal solution. Second example, as previously, is on a bigger data set to have an idea of the performances of column generation, in comparison to row generation and the extended formulation.

Small Example : One Generator on Four Hours

Example 5.1 (One Generator Example : Column Generation)**.** *Consider the market defined by generators and consumer respectively detailed on tables 3.1 and 3.2, i.e. one generator and four hours. Table 5.1 shows the iterates of column generation.*

At initialization there is one feasible schedule, the one from UC.

$$\begin{aligned}
\hat{p} &= \left\{\begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix}\right\} \\
\hat{\hat{p}} &= \left\{\begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix}\right\} \\
\hat{u} &= \left\{\begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}\right\} \\
\hat{v} &= \left\{\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}\right\} \\
\hat{w} &= \left\{\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}\right\}
\end{aligned}$$

At iteration 1, master program $\bar{M}_{LP}$ *provides prices* $\lambda = \begin{pmatrix} 3000 & -3123.36 & 3000 & 3000 \end{pmatrix}$. *The price of hour 2 is completely out of scope of graph 5.1, showing the dual Lagrangian function, and of graph 5.2, showing uplifts function. Hence iterate 1 of prices from column generation is not represented on both graphs.*
For this iteration as expected the only available schedule is completely chosen, i.e. $z^{n_1} = 1$. *The slave program computes the reduced cost, and it is strictly negative* $-34,934$. *Hence it adds the corresponding feasible schedules.*

$$\begin{aligned}
\hat{p} &= \left\{\begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix}, \begin{pmatrix} 6 & 0 & 6 & 0 \end{pmatrix}\right\} \\
\hat{\hat{p}} &= \left\{\begin{pmatrix} 6 & 11 & 6 & 0 \end{pmatrix}, \begin{pmatrix} 6 & 0 & 6 & 0 \end{pmatrix}\right\} \\
\hat{u} &= \left\{\begin{pmatrix} 1 & 1 & 1 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}\right\} \\
\hat{v} &= \left\{\begin{pmatrix} 1 & 0 & 0 & 0 \end{pmatrix}, \begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}\right\} \\
\hat{w} &= \left\{\begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}, \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}\right\}
\end{aligned}$$

At iteration 2, there is one more feasible schedule. Master program $\bar{M}_{LP}$ *provides prices* $\lambda = \begin{pmatrix} -2822.33 & 52.4545 & 3000 & 3000 \end{pmatrix}$. *Price of hour 1 is completely out of scope of graph 5.1, showing dual Lagrangian function, and of graph 5.2, showing uplifts function. Hence iterate 2 is also not represented on both graphs.*
First schedule is again the one to be fully chosen, i.e. $z^{n_1} = 1$. *There are still negative reduced cost* $-17,467$ *strictly negative. The slave adds the corresponding*

feasible schedule to the master program $\bar{M}_{LP}$.

$$\begin{aligned}
\hat{p} &= \left\{\begin{pmatrix}6 & 11 & 6 & 0\end{pmatrix}, \begin{pmatrix}6 & 0 & 6 & 0\end{pmatrix}, \begin{pmatrix}0 & 0 & 6 & 0\end{pmatrix}\right\} \\
\hat{\hat{p}} &= \left\{\begin{pmatrix}6 & 11 & 6 & 0\end{pmatrix}, \begin{pmatrix}6 & 0 & 6 & \end{pmatrix}, \begin{pmatrix}0 & 0 & 6 & 0\end{pmatrix}\right\} \\
\hat{u} &= \left\{\begin{pmatrix}1 & 1 & 1 & 0\end{pmatrix}, \begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}, \begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}\right\} \\
\hat{v} &= \left\{\begin{pmatrix}1 & 0 & 0 & 0\end{pmatrix}, \begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}, \begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}\right\} \\
\hat{w} &= \left\{\begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}, \begin{pmatrix}0 & 1 & 0 & 1\end{pmatrix}, \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}\right\}
\end{aligned}$$

At iteration 3, there are three different schedules given to $\bar{M}_{LP}$. *First schedule is fully chosen* $z^{n_1} = 1$. *It computes prices* $\lambda = \begin{pmatrix}88.8333 & 52.4545 & 3000 & 3000\end{pmatrix}$. *Those prices can be represented on both graphs 5.1 and 5.2. With such prices there are no negative reduced costs. Column generation has converged. First graph 5.1 shows that last iterate of* $\bar{M}_{LP}$ *indeed reaches maximal value of dual Lagrangian function. Equivalently graph 5.2 shows that last iterate of* $\bar{M}_{LP}$ *reaches minimum uplifts.*

Iteration	$\bar{M}_{LP}$	Prices λ
1	-67,357	$\begin{pmatrix}3000 & -3123.36 & 3000 & 3000\end{pmatrix}$
2	-67,357	$\begin{pmatrix}-2822.33 & 52.4545 & 3000 & 3000\end{pmatrix}$
3	-67,357	$\begin{pmatrix}88.8333 & 52.4545 & 3000 & 3000\end{pmatrix}$

Table 5.1: Iterates of row generation method for example 5.1.

Table 5.2 reports the prices from LP, EF, RG *and* CG. *Prices from* EF, RG *and* CG *are CHP and lead to a 0[€] uplifts as one can see on table 5.3. The performances for each method is provided on table 5.3. However for such a small example, it is not really representative of a real-world electricity market. This is why the method is the method is applied to the second example.*

Real-World Example

Example 5.2 (Winter-WE : Column Generation)**.** *Consider Belgian generators on tables 9.5, 9.6, i.e. 68 generators, and demand* $Winter-WE$ *on a time horizon over 24 hours with a timestep of 1 hour, i.e.* $[T]$ *has of a length 24.*

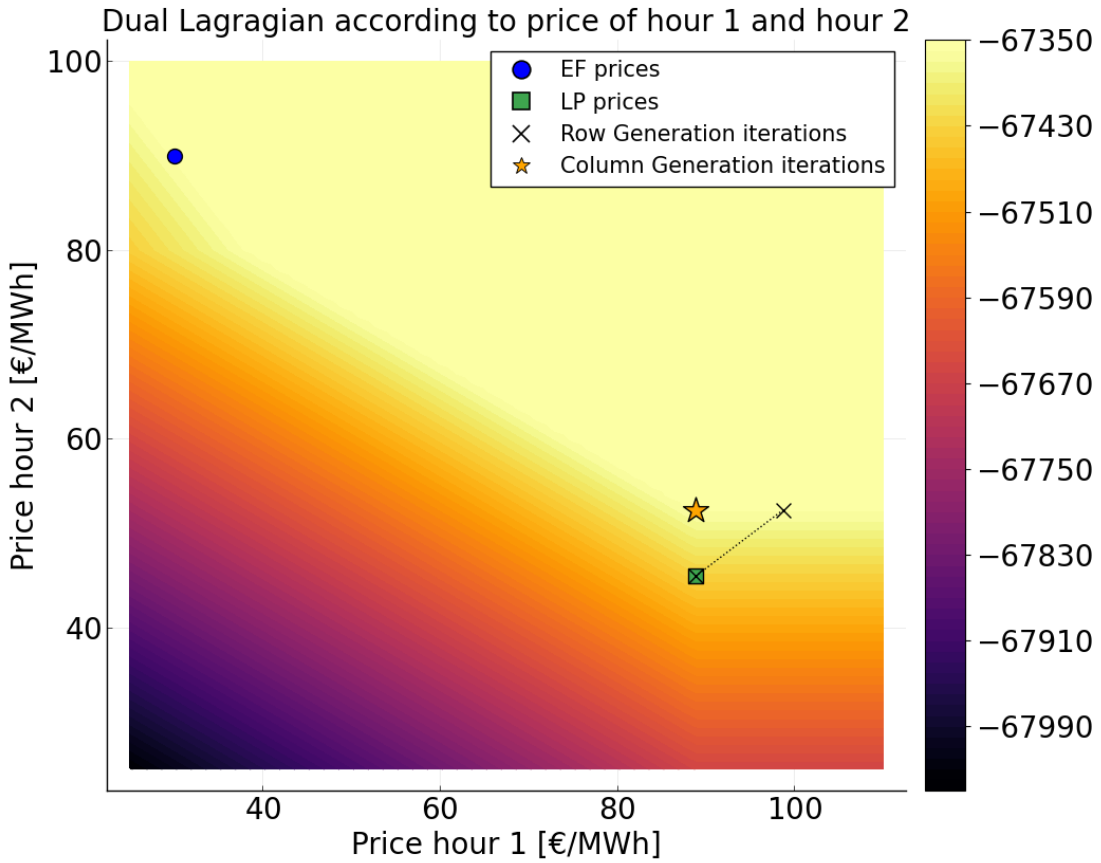


Figure 5.1: Dual Lagrangian function according to price hour 1 and price 2 for example 5.1.

Method	CHP ?	Prices [€/MWh]
LP	No	$\begin{pmatrix}88.8333 & 45.4375 & 3000 & 3000\end{pmatrix}$
EF	Yes	$\begin{pmatrix}30 & 90 & 3000 & 3000\end{pmatrix}$
RG	Yes	$\begin{pmatrix}98.7121 & 52.4545 & 3000 & 3000\end{pmatrix}$
CG	Yes	$\begin{pmatrix}88.833 & 52.4545 & 3000 & 3000\end{pmatrix}$

Table 5.2: Prices from different methods for example 5.1.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-67,357	/	0.013	0.001	/
LP	-67,434.187	77.187	0.013	0.001	/
EF	-67,357	0	0.016	0.0038	/
RG	-67,357	0	4.052	0.015	2
CG	-67,357	0	1.836	0.005	3

Table 5.3: Description of uplifts and their performances using LP and EF methods to compute prices for example 5.1.
RT : total Running Time of the program.
ST : Solving Time of the program.

$\bar{M}_{LP}$ and the slave programs are built only once and then kept in memory. CG converges in 29 iterations as visible on figure 5.3. The objective value of $\bar{M}_{LP}$ stagnates between iterations 1 and 10, and then between iterations 16 and 29. The impact of those iterations is less visible. Finally it provides CHP at iteration 29. Consumer has a 0[€] uplifts and uplift for each generator are provided on figure 5.4. On this figure, uplifts from the four tested methods EF, LP, RG, CG are represented.

CG algorithm struggles to converge due to floating point errors when $\tau_{CG} = 0$. Indeed, for this example CG still finds non-positive reduced cost of the order -10^{-10} after performing the 29 iterations. Therefore it continues to add columns and reach the maximal number of iterations that is 500. To avoid adding columns on CG when it should have converged, the hyper-parameter is set to $\tau_{CG} = 10^{-5}$. For such a value it prevents from looping until maximal number of iterations, and it preserves the precision for CHP. Solution $CG(\tau_{CG} = 10^{-5})$ is considered as optimal solution for the rest of this example.

Row generation and column generation perform almost the same number of iterations however CG converges in 10.463[s], that is 10 times faster than RG.

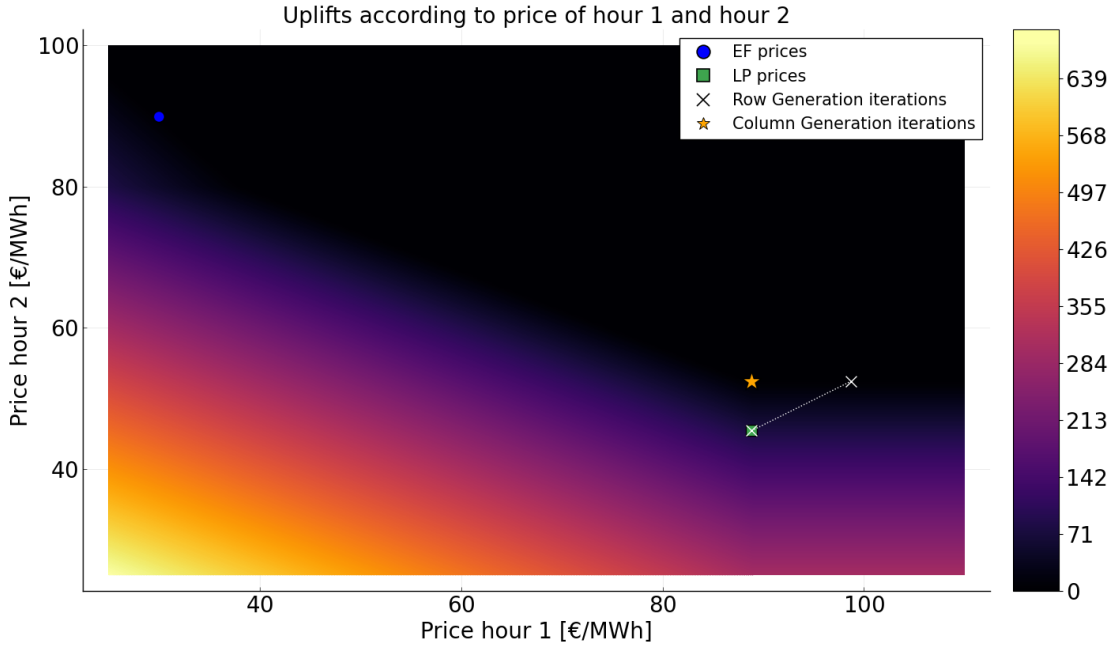


Figure 5.2: Dual Lagrangian function according to price hour 1 and price 2 for example 5.1.

Based on row generation, there is a improvement of 91% of time for RT. Based on the extended formulation, there is a improvement of 95% of time for RT.

Up to now there are three different methods providing CHP leading to minimum uplifts. For the same data sample, that is the Belgian generators and demand $Winter - WE$*, the extended formulation needs 242.347 [s], row generation needs 123.585 [s] and column generation needs 11.97 [s]. Up to now, column generation is outperforming the two other methods on the given data sample.*

5.4 Hyper-Parameter of CG

One may note there is a close similarity between the condition to add rows in RG and the condition to add columns in CG. RG adds a constraint if $z^* > 0$ and CG adds a column if $rc_g < 0$. However z^* and rc_g have complete different orders of magnitude. The first represents the distance of master solution to feasibility and the second represents reduced cost of a column for given dual multipliers λ_t and ν_g.

Due to numerically floating point errors, or because one is not interested in exact CHP but in an approximation, condition to add columns in RG might

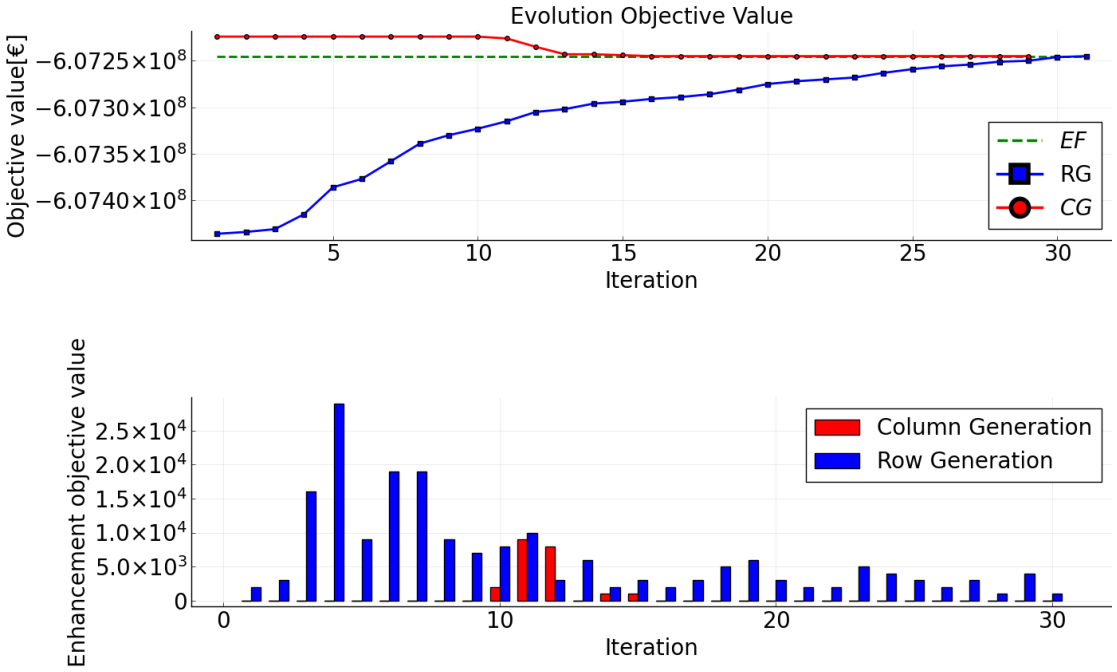


Figure 5.3: Evolution of row generation and column generation methods and the impact of their iterates on the objective value.

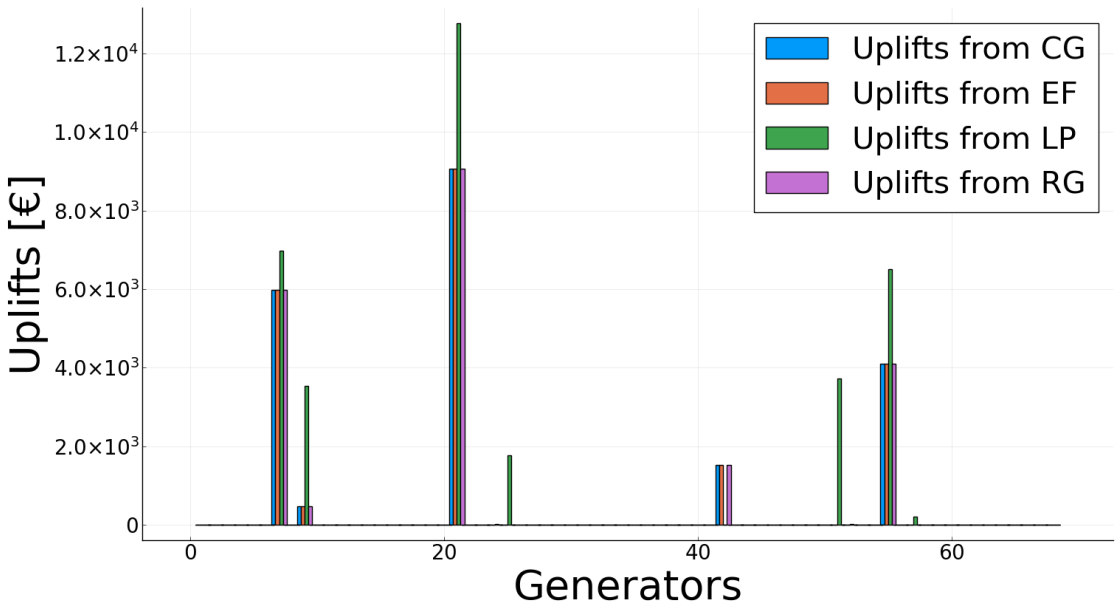


Figure 5.4: Uplift for each generator with CHP from EF, RG and CG and prices from LP.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-607,224,305	/	1.128	0.699	/
LP	-607,435,576	35,550	1.940	0.205	/
EF	-607,245,477	21,171	242.347	197.293	/
RG	-607,245,477	21,171	123.585	77.746	31
RG re-build	-607,245,477	21,171	1092.884	325.807	31
CG($\tau_{CG} = 0$)	-607,245,477	21,171	117.136	94.340	500 (iter max)
CG($\tau_{CG} = 10^{-5}$)	-607,245,477	21,171	11.976	5.346	29

Table 5.4: Uplifts and performances using LP, EF and RG methods to compute prices for example 4.2.
RT : total Running Time of the program.
ST : Solving Time of the program.

be changed such that $rc_g < -\tau_{CG}$, for $\tau_{CG} \geq 0$. Hyper-parameter τ_{CG} highly conditioned behaviour of the column generation algorithm and its results. For $\tau_{CRG} = 0$ example 5.2 reaches the maximal number of iterations, that is 500, because some reduced costs had very small negative values, i.e. $rc_g = -10^{-10}$. The proposed solution for this example was to use $\tau_{CG} = 10^{-5}$ and for such a τ_{CG} column generation converges after 29 iterations. What about other values for τ_{CG} ?

For column generation, the smaller is the objective value, the better is the solution. Indeed, column generation is an inner algorithm, adding columns enables CG to improve its objective function and thus to get closer to CHP.

On the one hand, for small values of τ_{CG} the condition $rc_g < -\tau_{CG}$ is often satisfied, and thus more columns are added to master program M_{LP}. The solution after convergence of column generation is closer to the maximal dual lagrangian and the prices lead to smaller uplifts, as one can see on figure 5.5. However it needs more iterations for the column algorithm to converge, as one can see on figure 5.7, and RT and ST are higher, as one can see on figure 5.6.

On the other hand, for bigger values of τ_{CG} the condition $rc_g < -\tau_{CG}$ is less often satisfied. Therefore less columns are added. Because column generation is an inner method, then the solution after convergence of CG is less good with less columns, i.e. prices lead to bigger uplifts as one can see on figure 5.5. In such cases, column generation provides an approximation, but it needs less iterations to converge, see on figure 5.7, and RT and ST are smaller 5.6.

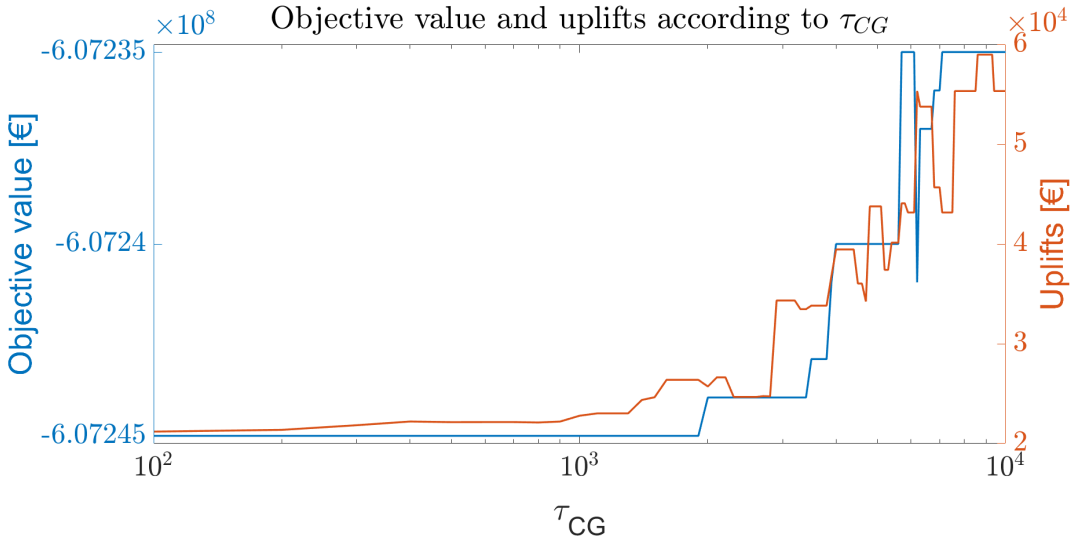


Figure 5.5: Objective value and uplifts for real-world example *Winter-WE* considering different values of τ_{CG}.

The increasing of the objective value and the uplifts on figure 5.5 is not monotonic. Depending on the hyper-parameter τ_{CG}, the columns added to the restricted master $\bar{M}_{LP}$ are not always the same. Hence from one value of τ_{CG} to another, the feasible schedules in the restricted master are not the same. The general trend is indeed to have an increase of the uplifts and the objective value. However sometimes with a looser constraint $rc_g < -\tau_{CG}$, there might have different schedules that combined together provide a better solution after convergence, than for a constraint less loose.

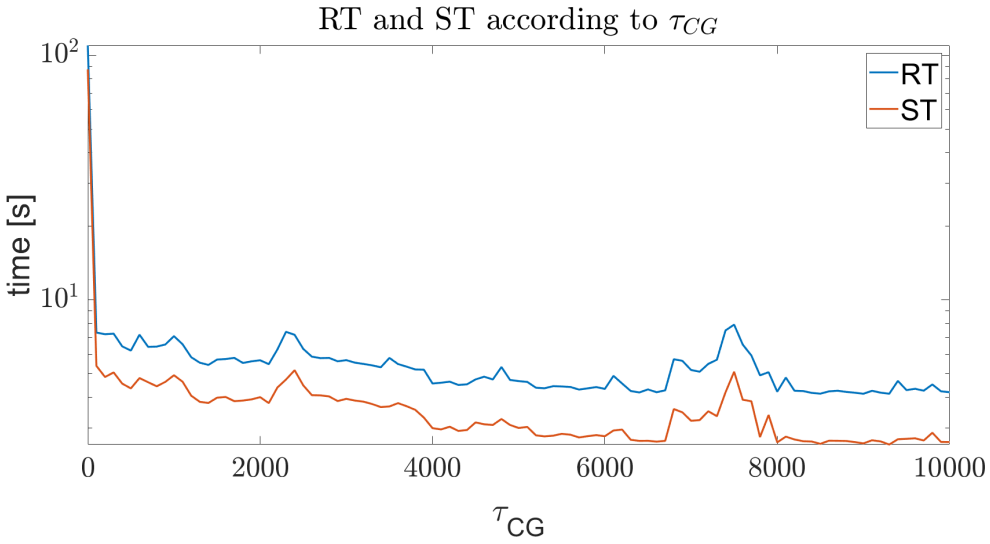


Figure 5.6: RT and ST for real-world example *Winter-WE* considering different values of τ_{CG}.
RT : total Running Time of the program.
ST : Solving Time of the program.

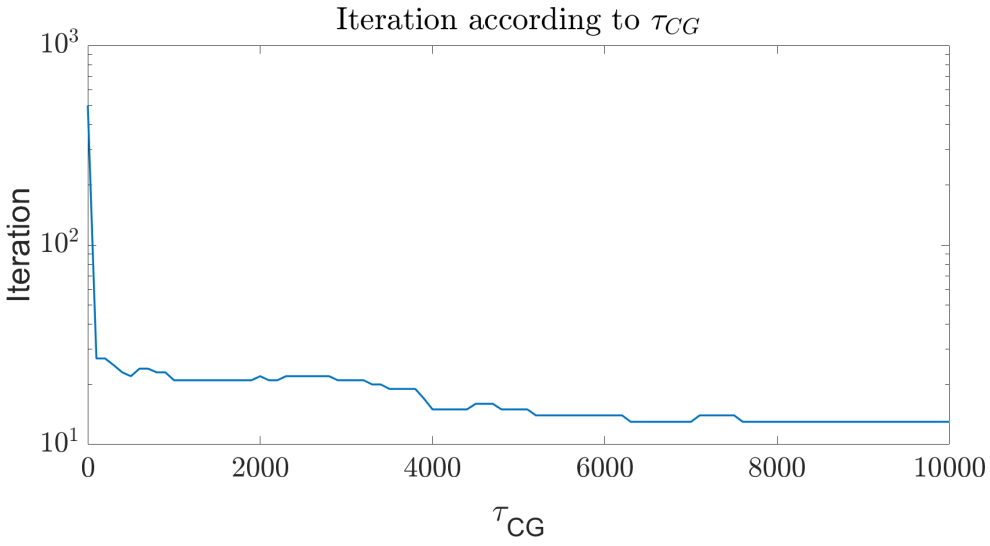


Figure 5.7: Number of iterations column generation needs to converge for real-world example *Winter-WE* considering different values of τ_{CG}.

Chapter 6

Column-and-Row Generation based on Extended Formulation

The third improved method to compute CHP is the **Column-and-Row Generation** (CRG) method, [24]. The extended formulation presented previously has the advantage to provide a tight and compact extended formulation for Π^g_{3-bin} as linear program. However it introduces a lot of constraints and variables. Row generation tackles the size of the linear program with an outer approach, by adding successive cuts to a master program. Column generation tackles the size of the linear program with an inner approach, by adding successive feasible schedules to its restricted master program.

Column-and-row generation is actually a column generation method for the extended formulation. Promising intervals of production along with their constraints are added to restricted master program in order to improve current solution. The slave program, that is the pricing problem, selects intervals as columns of the extended formulation with negative reduced cost.

6.1 Column-and-Row Generation : Basic Principles and General Formulations

Recall that the general MILP is $F = \mathbf{min}\ \{cx : Dx \geq d, x \in X\}$, where $X = P \cap (\mathbb{Z}^n \times \mathbb{R}^p)$ with polyhedron $P = \left\{x \in \mathbb{R}^{n+p}_+ : Bx \geq b\right\}$. Constraint $Dx \geq d$ represents a "complicate" constraint and set X represents tractable set.

Recall also that in previous sections, the general linear extended formulation R_{LP} is given by expression 6.1, the restricted linear master program of column generation $\bar{M}_{LP}$ is given by expression 6.2 and its dual D is given by expression 6.3.

They have been specified in order to apply different algorithms computing CHP for the Unit Commitment problem through the chapters **Extended Formulation** 3 and **Column Generation** 5. Those general formulations are also used in this section to provide the basic principles of column-and-row generation.

$$R_{LP} \equiv \textbf{min}\ cTz \tag{6.1a}$$
$$\text{s.t. } DTz \geq d \tag{6.1b}$$
$$Hz \geq h \tag{6.1c}$$
$$z \in \mathbb{R}^e \tag{6.1d}$$

$$\bar{M}_{LP} \equiv \textbf{min} \sum_{g \in G^t} (cx^g)\, z_g \tag{6.2a}$$
$$\text{s.t. } (\lambda) \sum_{g \in G^t} (Dx^g)\, z_g \geq d \tag{6.2b}$$
$$(\nu) \sum_{g \in G^t} z_g = 1 \tag{6.2c}$$
$$z \in [0,1]^{|G^t|} \tag{6.2d}$$

$$D \equiv \textbf{min}\ \lambda d + \nu \tag{6.3a}$$
$$\text{s.t. } \lambda D x^g + \nu \leq cx^g \quad \forall g \in G^t \tag{6.3b}$$
$$\lambda \in \mathbb{R}_+^{|G^t|}, \nu \in \mathbb{R} \tag{6.3c}$$

Assumption 6.1. *Assumptions made in source [24].*
There exists a polyhedron $Q = \left\{z \in \mathbb{R}^e_+ : Hz \geq h\right\}$*, defined by a rational matrix* $H \in \mathbb{Q}^{f \times e}$ *and a vector* $h \in \mathbb{Q}^f$*, and a linear transformation* F *defining projection* $: z \in \mathbb{R}^e_+ \longrightarrow x = (Tz) \in \mathbb{R}^{n+p}_+$ *such that*

(i) Q *defines an extended formulation for* $conv(X)$

$$conv(X) = proj_x Q = \left\{x \in \mathbb{R}^{n+p}_+ : x = Tz; Hz \geq h; z \in \mathbb{R}^e_+\right\}$$

(ii) $Z = Q \cap \mathbb{N}^e$ *defines an extended IP-formulation for* X*, i.e.*

$$X = proj_x Z = \left\{x \in \mathbb{R}^{n+p}_+ : x = Tz; Hz \geq h; z \in \mathbb{N}^e\right\}$$

Assumption 6.1-(i) is verified by proposition 3.2, where Q is actually the polytope D 3.9. Assumption 6.1-(ii) is also verified since the polytope D defines a tight extended formulation. The linear transformation is defined by expressions 3.11.

Restricted Extended Formulations

Let $\{z^s\}_{s \in S}$ be the enumerated set of solutions z^s of $Z \subseteq \mathbb{Z}^e_+$. Then $\bar{S} \subset S$ defines an enumerated subset of solutions : $\{z^s\}_{s \in \bar{S}}$.

Definition 6.1 (From source [24]). *Given a solution z^s of Z, let $J(z^s) = \left\{j : z^s_j > 0\right\} \subseteq \{1, ..., e\}$ be the support of solution vector z^s and let $I(z^s) = \{i : H_{ij} \neq 0$ for some $j \in J(z^s)\} \subseteq \{1, ..., f\}$ be the set of constraints of Q that involve some non zero components of z^s.*

From previous definition 6.1, refines the following terms :

- $\bar{z}$ is the restriction of z to the components of $\bar{J} = \cup_{s\in\bar{S}} J(z^s)$.
- $\bar{h}$ is the restriction of h to the components of $\bar{J} = \cup_{s\in\bar{S}} J(z^s)$.
- $\bar{H}$ is the restriction of H to the rows of $\bar{I} = \cup_{s\in\bar{S}} I(z^s)$ and the columns of $\bar{J} = \cup_{s\in\bar{S}} J(z^s)$
- $\bar{T}$ is the restriction of T to columns of $\bar{J} = \cup_{s\in\bar{S}} J(z^s)$

Define the general expression of Restricted Extended Formulation $\bar{R}$ that is given by expression 6.4, and its linear relaxation is $\bar{R}_{LP}$ that is given by expression 6.5. For those restricted formulations only a subset of variables and coefficients are available in the optimization programs. Both formulations $\bar{R}$ and $\bar{R}_{LP}$ are equivalent since packing polytopes D provide a tight extended formulation from proposition 3.2.

$$\bar{R} \equiv \textbf{min } c\bar{T}\bar{z} \quad (6.4a)$$
$$\text{s.t. } D\bar{T}\bar{z} \geq d \quad (6.4b)$$
$$\bar{H}\bar{z} \geq \bar{h} \quad (6.4c)$$
$$\bar{z} \in \mathbb{N}^{|\bar{J}|} \quad (6.4d)$$

$$\bar{R}_{LP} \equiv \textbf{min } c\bar{T}\bar{z} \quad (6.5a)$$
$$\text{s.t. } D\bar{T}\bar{z} \geq d \quad (6.5b)$$
$$\bar{H}\bar{z} \geq \bar{h} \quad (6.5c)$$
$$\bar{z} \in \mathbb{R}^{|\bar{J}|} \quad (6.5d)$$

Definition 6.2 (From source [24]). *Assume subset $\bar{S} \subset S$ properly chosen to guarantee the feasibility of $\bar{R}_{LP}$. The associated set is*

$$\bar{G} = G\left(\bar{S}\right) = \left\{g \in G : x^g = Tz^s \text{ for some } s \in \bar{S}\right\}$$

Proposition 6.1 (From source [24]). *Let $\bar{R}$ 6.4 and $\bar{M}$ be the restricted versions of formulations R and M, both associated with the same subset $\bar{S} \subset S$ of subproblem solutions and associated $\bar{G}$ as defined in 6.2. Under assumption 6.1, their linear relaxation values are such that*

$$v^* \leq v^{\bar{R}}_{LP} \leq v^{\bar{M}}_{LP} \quad (6.6)$$

Proof. See Appendix 9.3.13. □

Proposition 6.1 states that for a given subset $\bar{S}$, the restricted extended formulation $\bar{R}$ gives a better solution than the restricted Dantzig-Wolfe decomposition $\bar{M}$. Hence the idea is to do column generation on the restricted extended formulation $\bar{R}_{LP}$ 6.5, with the hope to converge in less iterations than the regular column generation, i.e. with Dantzig-Wolfe decomposition $\bar{M}_{LP}$ 6.2.

6.1.1 Column-and-Row Algorithm : General Procedure

The procedure proposed by [24] is a dynamic column-and-row generation for linear program $\bar{R}_{LP}$. It generates new columns, i.e. new production intervals, for $\bar{\mathcal{T}}$ while getting the prices as the dual multipliers of the balance constraint from $\bar{R}_{LP}$. It is not a simple column generation since adding a production interval $[a, b]$ in the basis $\bar{\mathcal{T}}$ requires to add constraints defining the production on this interval $[a, b]$, i.e. the dispatch polytope $D^{[a,b]}$.

Column-and-row method is based on several propositions summarised in 6.2.

Proposition 6.2. *From source [24].*
Let $v_{LP}^{\bar{R}}$ denote the optimal LP value of $\bar{R}_{LP}$, while (λ, σ) denotes an optimal dual solution respectively associated to the balance constraint and to the constraints defining extended formulation of convex hull. Let z^* be the subproblem solution given by the pricing problem in Step 2 of the procedure 6.1, and $\zeta = (c - \lambda D)Tz^*$ be its objective value. Then

(i) *The dual Lagrangian bound : $L(\lambda) = \lambda d + (c - \lambda D)Tz^* = \lambda d + \zeta$ defines a valid dual bound for M_{LP}, while (λ, ζ) defines a feasible solution of D, the dual of M_{LP}.*

(ii) *Let β denote the best of the above Lagrangian bound encountered in the procedure 6.1. If $v_{LP}^{\bar{R}} \leq \beta$, i.e. when the stopping condition is satisfied in Step 3, then $v^* = \beta$ and (λ, ζ) defines an optimal solution to D.*

(iii) *If $v_{LP}^{\bar{R}} > \beta$, then $[(c - \lambda D)\, T - \sigma H]\, z^* < 0$. Hence, some of the component of z^* were not present in $\bar{R}_{LP}$ and have a negative reduced cost for the current dual solution (λ, σ).*

(iv) *Inversely, when $[(c - \lambda D)\, T - \sigma H]\, z^* \geq 0$, i.e. if the generated column has non negative aggregate reduced cost for the dual solution of $\bar{R}_{LP}$, then $v_{LP}^{\bar{R}} \leq \beta$ (the stopping condition of Step 3 must be satisfied) and (λ, ν) defines a feasible solution for formulation D defined in 6.3 for $\nu = \sigma h$*

Proof. See Appendix. □

What utility for these propositions ?

- Proposition 6.2-(i) states that $L(\lambda)$ defines a valid dual bound for M_{LP}. It is used to define the dual bound update in Step 3.
- Proposition 6.2-(ii) defines the stopping criteria of the column-and-row generation algorithm 6.1. This stopping criteria is used in Step 3 and it involves the dual bound.
- Proposition 6.2-(iii) states that if the stopping criteria is not satisfied, then the columns with negative reduced cost have to be added to $\bar{R}_{LP}$. Those columns are the ones added in Step 4, if stopping criteria is not satisfied.
- Proposition 6.2-(iv) states that if stopping criteria is satisfied, then there are no more columns with negative reduced cost. Hence algorithm 6.1 has converged and it stops.

From the proposition 6.2 dynamic column-and-row generation algorithm can be built. It is provided for general formulations in table 6.1.

Step 0 :	Initialize the dual bound, $\beta := -\infty$, and the subproblem solution set $\bar{S}$ so that the linear relaxation of $\bar{R}$ is feasible.
Step 1 :	Solve the LP relaxation of $\bar{R}$ and record its value $v_{LP}^{\bar{R}}$ and the dual solution λ associated to constraint $D\bar{T}\bar{z} \geq d$.
Step 2 :	Solve the pricing problem ,i.e. slave program, $$z^* := \text{argmin}\{(c - \lambda D)Tz : z \in Z\}$$ and record its value $\zeta := (c - \lambda D)Tz^*$.
Step 3 :	Compute the Lagrangian dual bound: $L(\lambda) := \lambda d + \zeta$, and update the dual bound $\beta := \mathbf{max}\{\beta, L(\lambda)\}$. If $v_{LP}^{\bar{R}} \leq \beta$, STOP.
Step 4 :	Update the current bundle, $\bar{S}$, by adding solution $z^s := z^*$ and update the resulting restricted reformulation $\bar{R}$ according to definition 6.1. Then go to Step 1.

Table 6.1: Dynamic column-and-row generation for $\bar{R}_{LP}$,[24]

6.2 Restricted Extended Formulation, Pricing Problem and Column-and-Row Algorithm

Derive now the general formulation from previous section for the current problem, that is to solve $\mathbf{max}_\lambda \{L(\lambda)\}$ 2.2.

6.2.1 Restricted Extended Formulation

The extended formulation derived in previous chapter for the current problem is given by expressions 6.7. It is important to remember that the extended formulation uses intervals of production $[a,b] \in \mathcal{T}$, where $\mathcal{T}$ defines all production intervals respecting minimum up time UT.

$$\textbf{min} \left\{ \sum_{[a,b]\in\mathcal{T}} \left[\sum_{g\in\mathcal{G}} c^g \left(p_g^{[a,b]}, \gamma_{[a,b]}^g \right) \right] - VOLL \sum_t l_t \right\} \tag{6.7a}$$

$$\text{s.t } (\lambda_t) \sum_g \sum_{[a,b]} p_g^{[a,b],t} - l_t = 0 \qquad \forall t \in [T] \tag{6.7b}$$

$$A^{[a,b]} p^{[a,b]} + \bar{A}^{[a,b]} \bar{p}^{[a,b]} \leq \gamma_{[a,b]} \bar{b}^{[a,b]} \qquad \forall [a,b] \in \mathcal{T} \tag{6.7c}$$

$$\left(p^{[a,b]}, \bar{p}^{[a,b]} \right) \in \mathbb{R}_+^{2T} \qquad \forall [a,b] \in \mathcal{T} \tag{6.7d}$$

$$\sum_{\{[a,b]\in\mathcal{T} | t\in[a,b+DT]\}} \gamma_{[a,b]} \leq 1 \qquad \forall t \in [T] \tag{6.7e}$$

$$\gamma_{[a,b]} \geq 0 \qquad \forall [a,b] \in \mathcal{T} \tag{6.7f}$$

$$\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]} \right) \in \mathbb{R}_+^{2T} \qquad \forall g \in \mathcal{G} \tag{6.7g}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{6.7h}$$

For the problem of interest it means to restrict only to some intervals $[a,b] \in \bar{\mathcal{T}}$, and their related variables and constraints. Restricted Extended Formulation for the current problem is given by expressions 6.8.

Restricted Extended Formulation - Column-and-Row Generation

$$\bar{R}_{LP} \equiv \textbf{min} \left\{ \sum_{[a,b]\in\bar{\mathcal{T}}} \left[\sum_{g\in\mathcal{G}} c^g \left(p_g^{[a,b]}, \gamma_{[a,b]}^g \right) \right] - VOLL \sum_t l_t \right\} \tag{6.8a}$$

$$\text{s.t } (\lambda_t) \sum_g \sum_{[a,b]\in\bar{\mathcal{T}}} p_g^{[a,b],t} - l_t = 0 \qquad \forall t \in [T] \tag{6.8b}$$

$$A^{[a,b]} p^{[a,b]} + \bar{A}^{[a,b]} \bar{p}^{[a,b]} \leq \gamma_{[a,b]} \bar{b}^{[a,b]} \qquad \forall [a,b] \in \bar{\mathcal{T}} \tag{6.8c}$$

$$\left(p^{[a,b]}, \bar{p}^{[a,b]} \right) \in \mathbb{R}_+^{2T} \qquad \forall [a,b] \in \bar{\mathcal{T}} \tag{6.8d}$$

$$\sum_{\left\{[a,b]\in\bar{\mathcal{T}} | t\in[a,b+DT]\right\}} \gamma_{[a,b]} \leq 1 \qquad \forall t \in [T] \tag{6.8e}$$

$$\gamma_{[a,b]} \geq 0 \qquad \forall [a,b] \in \bar{\mathcal{T}} \tag{6.8f}$$

$$\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]} \right) \in \mathbb{R}_+^{2T} \qquad \forall g \in \mathcal{G} \tag{6.8g}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{6.8h}$$

6.2.2 Pricing Problem

The pricing problem is the slave program. It is the optimization program providing the dual bound $L(\lambda)$. Hence it dualizes the constraint $DTz \geq d$ from R_{LP} 6.5. Dual Lagrangian bound is described by following expression.

$$\begin{aligned} L(\pi) &= \textbf{min} \left\{ cTz - \lambda \left(DTz - d \right) : z \in Z \right\} \\ &= \textbf{min} \left\{ cTz - \lambda DTz : z \in Z \right\} + \lambda d = \left(c - \lambda D \right) Tz^* + \lambda d = \zeta + \lambda d \end{aligned}$$

For the current problem, the pricing problem is given by expression 6.9 in the dispatch polytopes basis, i.e. variables $\left(p_g^{[a,b]}, \bar{p}_g^{[a,b]}, \gamma_{[a,b]} \right) \in D^{[a,b]}$, that is the dual extended formulation.

$$\textbf{min} \left\{ \sum_{[a,b]\in\mathcal{T}} \left[\sum_{g\in\mathcal{G}} c^g \left(p_g^{[a,b]}, \gamma_{[a,b]}^g \right) \right] - VOLL \sum_t l_t \right. \tag{6.9a}$$

$$\left. - \sum_t \lambda_t \left(\sum_g \sum_{[a,b]\in\mathcal{T}} p_t^{[a,b]} - l_t \right) \right\} \tag{6.9b}$$

$$\text{s.t} \sum_{\{[a,b]\in\mathcal{T} | t\in[a,b+DT]\}} \gamma_{[a,b]}^g \leq 1 \qquad \forall g \tag{6.9c}$$

$$A_g^{[a,b]} p_g^{[a,b]} + \bar{A}_g^{[a,b]} \bar{p}_g^{[a,b]} - \gamma_g^{[a,b]} \bar{b}_g^{[a,b]} \leq 0 \qquad \forall g \tag{6.9d}$$

$$(p_g^{[a,b]}, \bar{p}_g^{[a,b]}) \in \mathbb{R}_+^{2T} \qquad \forall g \tag{6.9e}$$

$$0 \leq \gamma_{[a,b]}^g \qquad \forall g, \forall [a,b] \tag{6.9f}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{6.9g}$$

However solving this pricing problem requires to formulate all variables and constraints causing problems in EF 3.12. From proposition 3.2, the constraints 6.9c, 6.9d and 6.9e provide compact and tight extended formulation. The balance constraint is not binding all the generators any more, i.e. the "complicate" constraint $Dx \geq d$ is not in the feasible region any more. Hence, from proposition 2.3, solving the dual extended formulation problem is equivalent to directly solve the dual Lagrangian $L(\lambda)$.

$$\zeta = \textbf{min} \{ (c - \lambda D) Tz : z \in Z \} = \textbf{min} \{ (c - \lambda D) x : Bx \geq b \} \tag{6.10}$$

The pricing problem can be formulated as the dual Lagrangian problem 6.11, where $c^g := c^g (p_t^g, u_t^g, v_t^g, w_t^g)$.

Pricing Problem - Column-and-Row Generation

$$z^* = \text{argmin} \left\{ \left(\sum_g c^g - VOLL \sum_t l_t \right) - \sum_t \lambda_t \left(\sum_g p_t^g - l_t \right) \right\} \tag{6.11a}$$

$$\text{s.t } (p_g, \bar{p}_g, u_g, v_g, w_g) \in \Pi_{3-bin}^g \qquad \forall g \in \mathcal{G} \tag{6.11b}$$

$$0 \leq l_l \leq L_t \qquad \forall t \tag{6.11c}$$

6.2.3 Column-and-Row Generation Algorithm

Column-and-row generation is based on the generic procedure 6.1. The implemented initialization is to define the initial production intervals using dispatch solution.

Then for each iteration, the algorithm first solves the restricted extended formulation linear program $\bar{R}_{LP}$, using the production intervals such that $[a,b] \in \bar{\mathcal{T}}$. $\bar{R}_{LP}$ defines a primal bound on the optimal solution v^* as it is an inner method. Then it solves the pricing problem with price λ from $\bar{R}_{LP}$. It provides a dual bound on the optimal solution v^*. β is updated to always represent the maximal dual bound reached. If both primal and dual bound are equal, then optimal solution has been found and CHP are λ. If optimality criteria is not satisfied, then the production intervals obtained from the pricing problem are added to set $\bar{\mathcal{T}}$.

This method adds columns, production intervals $[a,b]$, but also rows since those production intervals $[a,b]$ are defined by a dispatch polytope $D^{[a,b]}$. When adding the interval $[a,b]$ to $\bar{\mathcal{T}}$, the algorithm must also add the packing polytope $D^{[a,b]}$ to feasible region of restricted extended formulation $\bar{R}_{LP}$.

$\bar{\mathcal{T}}^g \longleftarrow$ initialization production intervals $\forall g \in \mathcal{G}$;
for *iter in 1..iterMax* **do**
  Solve $\bar{R}_{LP}$: $\left(\lambda, v_{LP}^{\bar{R}}\right) \longleftarrow \bar{R}_{LP}\left(\bar{\mathcal{T}}\right)$;
  $x^* \longleftarrow$ Pricing Problem (λ);
  Compute $L(\lambda) := \lambda d + \zeta = \lambda d + (c - \lambda D)x^*$;
  Update dual bound $\beta := \max\{\beta, L(\lambda)\}$;
  if $v_{LP}^{\bar{R}} \leq \beta + \tau_{CRG}$ **then**
    Optimal solution found : Stop;
  end
  $\bar{\mathcal{T}} := \bar{\mathcal{T}} \cup \{[a,b] \text{ s.t. } [a,b] \in x^*\}$;
end

Algorithm 3: Column-and-Row Generation (CRG)

$\bar{R}_{LP}$ and the pricing problem are only built once. At each iteration the objective value of the pricing problem is updated, and intervals of production $[a,b]$ with their packing polytopes $D^{[a,b]}$ are added to $\bar{R}_{LP}$.

Hyper-parameter τ_{CRG} ensures not to miss stopping criteria due to floating point errors. It could also be used if one is interested in approximation of CHP and make less iterations. For now consider $\tau_{CRG} = 10^{-5}$.

6.3 Examples

As previously two different example are analysed. First example is small example to illustrate how iterates of column-and-row generation converge towards maximum dual Lagrangian, equivalently minimum sum of uplifts. Second example is a

bigger one, to have an idea of the performances of column-and-row generation in comparison to the three other methods.

Small Example : One Generator on Four Hours

Example 6.1 (One Generator Example : Column-and-Row Generation)**.** *Consider a market where the generator and the consumer are respectively detailed in tables 3.1 and 3.2, i.e. one generator on four hours. Table 6.2 reports the iterates of column-and-row generation algorithm.*

At initialization, there is one interval, the one provided when looking a solution from UC, that is $\bar{\mathcal{T}} := \{[1,3]\}$.

At iteration 1, $\bar{R}_{LP}$ *provides prices* $\lambda = \begin{pmatrix} 30 & 30 & 3000 & 3000 \end{pmatrix}$. *It computes the pricing problem and updates the dual Lagrangian bound* $\beta = -67,957$. *The stopping criteria is not satisfied. Solution* x^* *from the pricing problem indicates what interval* $[a,b]$ *to add to* $\bar{R}_{LP}$. *The production from pricing problem is* $p_{x^*} = \begin{pmatrix} 0 & 0 & 6 & 0 \end{pmatrix}$ *hence production interval is* $[3,3]$. *Update of the set containing the intervals* $\bar{\mathcal{T}} = \{[1,3],[3,3]\}$.

At iteration 2, $\bar{R}_{LP}$ *provides prices* $\lambda = \begin{pmatrix} 30 & 84.5455 & 3000 & 3000 \end{pmatrix}$. *The dual bound is now* $\beta = -67,384.2727$. *The stopping criteria is not satisfied yet, since* $v_{LP}^{\bar{R}} = -67,357 \not\leq \beta = -67,384.272$. *The solution* x^* *from the pricing problem indicates what new interval to add to* $\bar{R}_{LP}$. *Production from pricing problem is* $p_{x^*} = \begin{pmatrix} 0 & 6 & 6 & 0 \end{pmatrix}$ *hence the production interval is* $[2,3]$. *Update of the set containing the intervals* $\bar{\mathcal{T}} = \{[1,3],[3,3],[2,3]\}$.

At iteration 3, $\bar{R}_{LP}$ *provides prices* $\lambda = \begin{pmatrix} 38.3333 & 80 & 3000 & 3000 \end{pmatrix}$. *The dual Lagrangian bound is now* $\beta = -67,357$ *and the stopping criteria is satisfied, i.e.* $\bar{R}_{LP} = 67,357 \leq \beta = -67,357$. *The optimal solution has been reached by column-and-row algorithm. Prices are CHP, as reported in table 6.3. Prices from EF, RG, CG and CRG are CHP and lead to a total amount of uplift that is 0[€] , as one can see in table 6.4. It also reports performances for each method. However for such a small example is not really representative of real-world example. This is why comes second example 6.2.*

Figure 6.1 shows the convergence of column-and-row iterates towards the maximal dual Lagrangian function, according to price hour 1 and hour 2. One can see that each iterate improves current dual Lagrangian bound, to finally obtain the optimal solution.

Iteration	$\bar{R}_{LP}$ [€]	$L(\lambda)$ [€]	Intervals	Prices λ
0	/	/	$\{[1,3]\}$	/
1	-67,357	-67,957	$\{[1,3];[3,3]\}$	$\begin{pmatrix}30 & 30 & 3000 & 3000\end{pmatrix}$
2	-67,357	-67,384.272	$\{[1,3];[3,3];[2,3]\}$	$\begin{pmatrix}30 & 84.5455 & 3000 & 3000\end{pmatrix}$
3	-67,357	-67,357	$\{[1,3];[3,3];[2,3]\}$	$\begin{pmatrix}38.3333 & 80 & 3000 & 3000\end{pmatrix}$

Table 6.2: Iterates of column-and-row generation for example 6.1.

Figure 6.2 shows the convergence of column-and-row iterates towards minimal uplifts. Each iteration improves the current uplifts to finally reach minimal uplifts. One can note CHP from extended formulation, row generation, column generation and column-and-row generation are all different but still provide the same maximal dual Lagrangian and minimum sum of uplifts.

Method	CHP ?	Prices [€/MWh]
LP	No	$\begin{pmatrix}88.8333 & 45.4375 & 3000 & 3000\end{pmatrix}$
EF	Yes	$\begin{pmatrix}30 & 90 & 3000 & 3000\end{pmatrix}$
RG	Yes	$\begin{pmatrix}98.7121 & 52.4545 & 3000 & 3000\end{pmatrix}$
CG	Yes	$\begin{pmatrix}88.833 & 52.4545 & 3000 & 3000\end{pmatrix}$
CRG	Yes	$\begin{pmatrix}38.333 & 80 & 3000 & 3000\end{pmatrix}$

Table 6.3: Prices from different methods for example 6.1.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-67,357	/	0.013	0.001	/
LP	-67,434.187	77.187	0.013	0.001	/
EF	-67,357	0	0.016	0.0038	/
RG	-67,357	0	4.052	0.015	2
CG	-67,357	0	1.836	0.005	3
CRG	-67,357	0	5.906	0.015	3

Table 6.4: Description of uplifts and performances for LP, EF, RG, CG and CRG methods to compute prices for example 6.1.
RT : total Running Time of the program.
ST : Solving Time of the program.

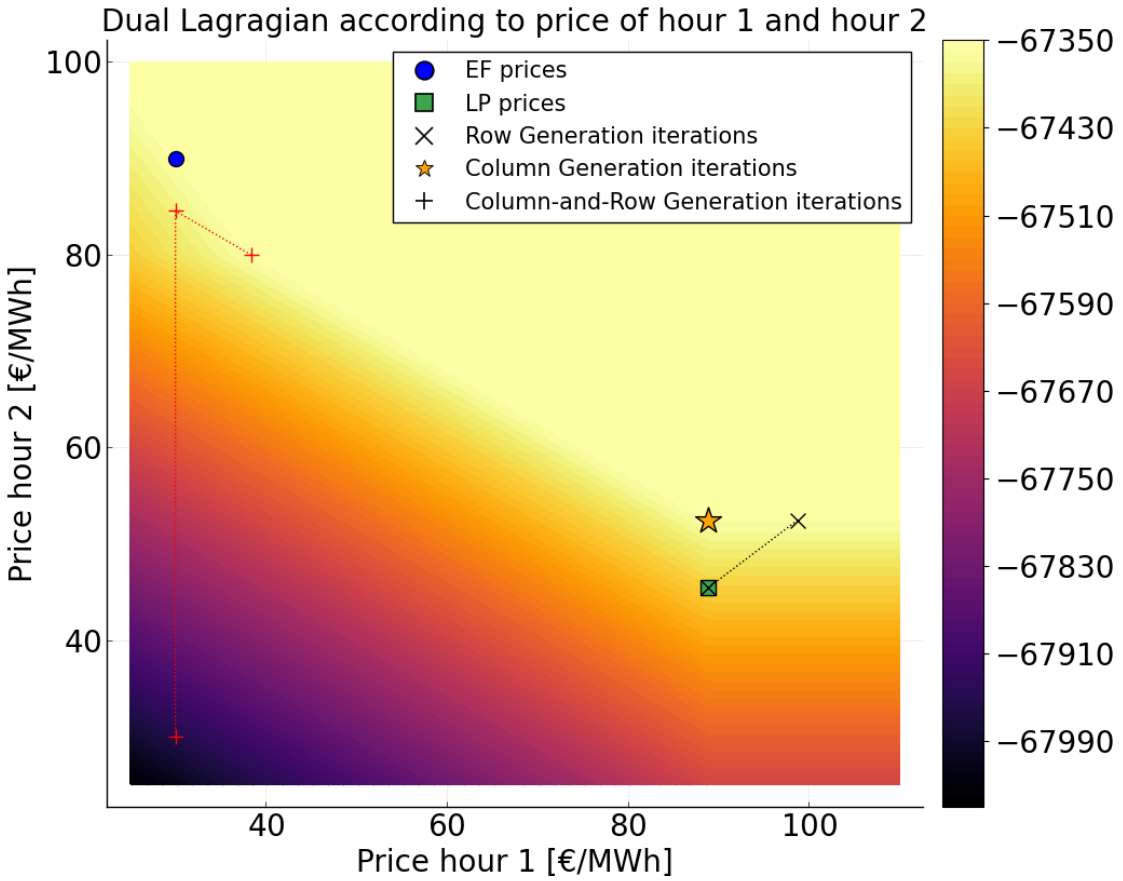


Figure 6.1: Dual Lagrangian function according to price hour 1 and hour 2 for example 6.1.

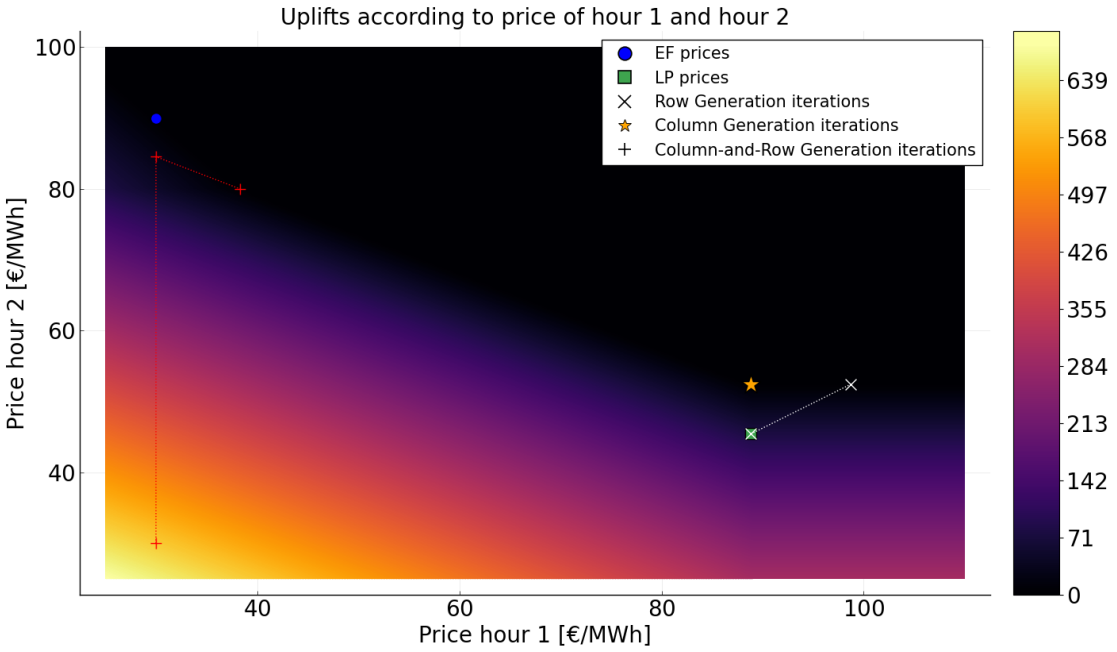


Figure 6.2: Uplifts according to price of hour 1 and hour 2 for example 6.1.

Real-World Example

Example 6.2 (Winter-We : Column-and-row Generation)**.** *This example considers the set of Belgian generators, given in tables 9.5, 9.6, and the Belgian demand Winter-WE on a time horizon over 24 hours with a timestep of 1 hour, i.e.* $[T]$ *is of length 24.*

CRG converges in 3 iterations, as one can see in table 6.5 and on figure 6.3. Second iteration has a notable impact to improve $\bar{R}_{LP}$*. Between iteration 2 and 3 objective value of* $\bar{R}_{LP}$ *does not improve but there were still columns with reduced cost. The stopping criteria was not satisfied. At iteration 3 it provides CHP since the optimal value is found* $v^* = v^{\bar{R}}_{LP}$*. Those CHP provide a 0[€] uplift for consumer and the uplifts for the generators are described on figure 6.4.*

It needs 11.4 seconds to perform the 3 iterations, however most of the time is spent building optimization programs. It only solves those optimization programs during 1.297 seconds. CG has a total running time a bit faster than CRG, but the CRG spends much less time solving optimization programs. One can also see that CRG needs 3 iterations to converge where CG needs 29.

Based on EF, column-and-row generation CRG provides an improvement of 95% for RT and an improvement of 99% for ST.

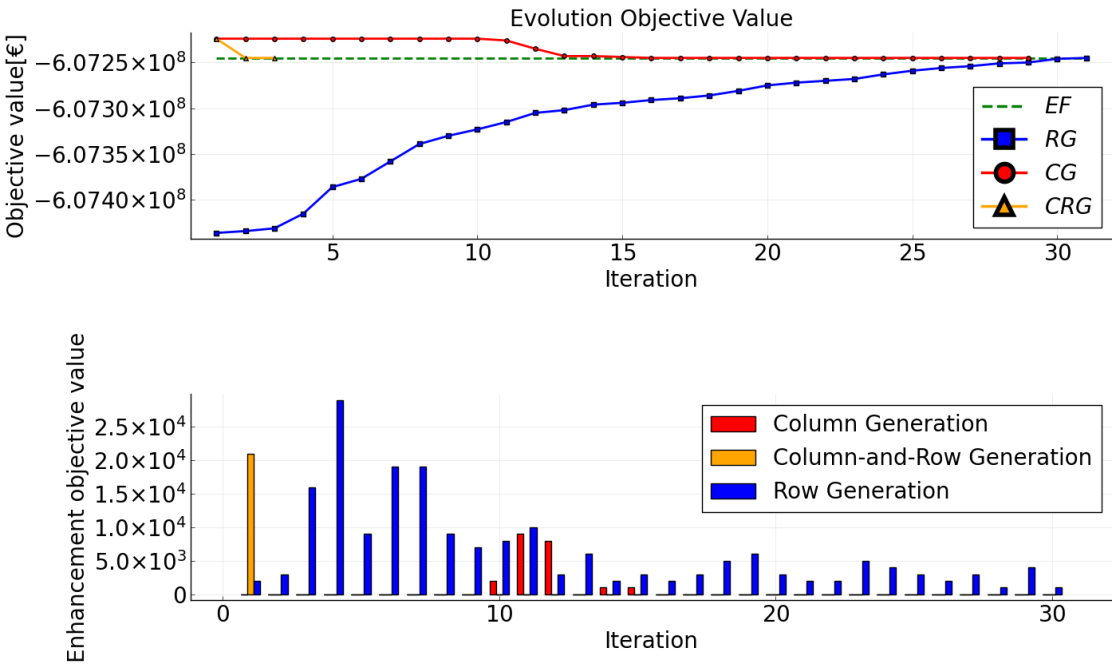


Figure 6.3: Evolution of row generation, column generation and column-and-row generation methods, and the impact of their iterates on the objective value.

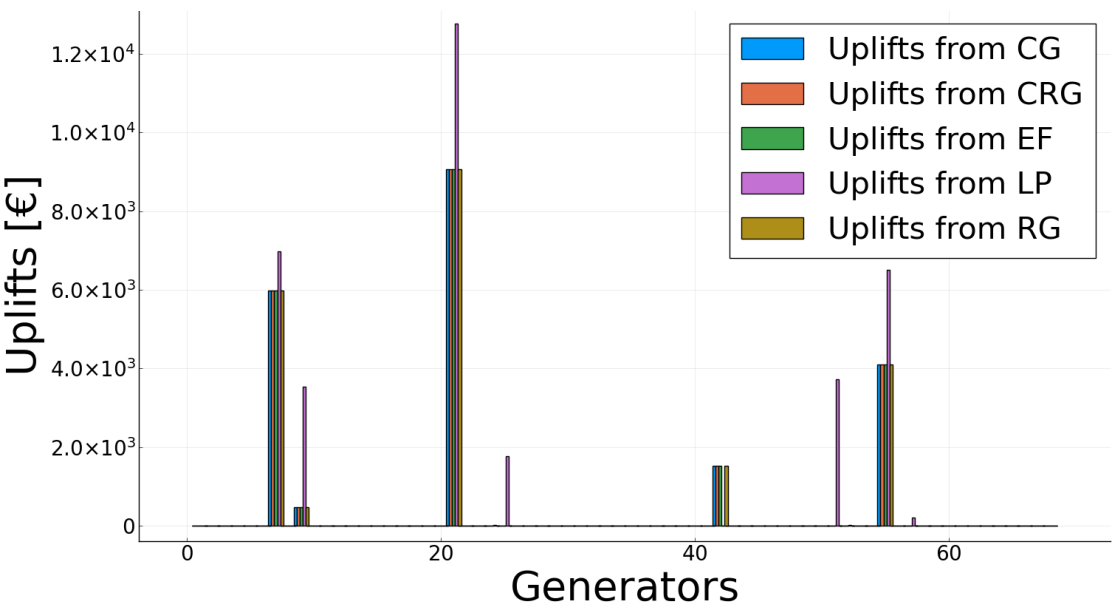


Figure 6.4: Uplift for each generator with CHP from EF, RG, CG and CRG, and prices from LP.

Method	Objective Value [€]	Uplifts [€]	RT [s]	ST [s]	Iterations
UC	-607,224,305	/	1.128	0.699	/
LP	-607,435,576	35,550	1.940	0.205	/
EF	-607,245,477	21,171	242.347	197.293	/
RG	-607,245,477	21,171	123.585	77.746	31
RG re-build	-607,245,477	21,171	1092.884	325.807	31
CG($\tau_{CG} = 0$)	-607,245,477	21,171	117.136	94.340	500 (iter max)
CG($\tau_{CG} = 10^{-5}$)	-607,245,477	21,171	11.976	5.346	29
CRG($\tau_{CRG} = 0$)	-607,245,477	21,171	204.391	188.237	500 (iter max)
CRG($\tau_{CRG} = 10^{-5}$)	-607,245,477	21,171	11.420	1.297	3

Table 6.5: Uplifts and performances using LP, EF, RG, CG and CRG methods to compute prices for example 4.2.
RT : total Running Time of the program.
ST : Solving Time of the program.

6.4 Hyper-Parameter of CRG

As for RG and CG, the stopping criteria may need to be changed for some data samples due to some floating point errors. Instead of considering the condition $v_{LP}^{\bar{R}} \geq \beta$ one may consider the condition $v_{LP}^{\bar{R}} \geq \beta + \tau_{CRG}$, with $\tau_{CRG} \geq 0$, as stopping criteria for the column-and-row generation algorithm.

One can see on figure 6.5 the uplifts and objective values $v_{LP}^{\bar{R}}$, and on figure 6.7 number of iterations according to different values of τ_{CRG} for real-world example 6.2. Note that both graphs consider values for τ_{CRG} from 0 to 10^{-5} with a step of 10^{-6}.

For $\tau_{CRG} = 0$, CRG iterates until 500 iterations, i.e. the maximum number of iterations. One can see that the total amount of uplifts is the same for $\tau_{CRG} = 0$ or $\tau_{CRG} = 10^{-6}$, but the difference is in the number of iterations. For $\tau_{CRG} = 0$, the algorithm reaches 500 iterations and for $\tau_{CRG} = 10^{-6}$ it converges in 3 iterations. Actually the algorithm has already converged for $\tau_{CRG} = 0$ after 3 iterations but the stopping criteria is not satisfied due to floating point error. There is a difference of the order of 10^{-10} between $v_{LP}^{\bar{R}}$ and the dual Lagrangian bound β.

τ_{CRG} has big impact on CRG in the sens that the number of iterations drops from 500 to 3, when changing τ_{CRG} from 0 to 10^{-6}. However τ_{CRG} has little impact on the behaviour when varying from 10^{-5} to 10^{-6}. As soon as $\tau_{CRG} > 10^{-10}$, then CRG converges in 3 iterations. For the rest of the manuscript, assume $\tau_{CRG} = 10^{-5}$.

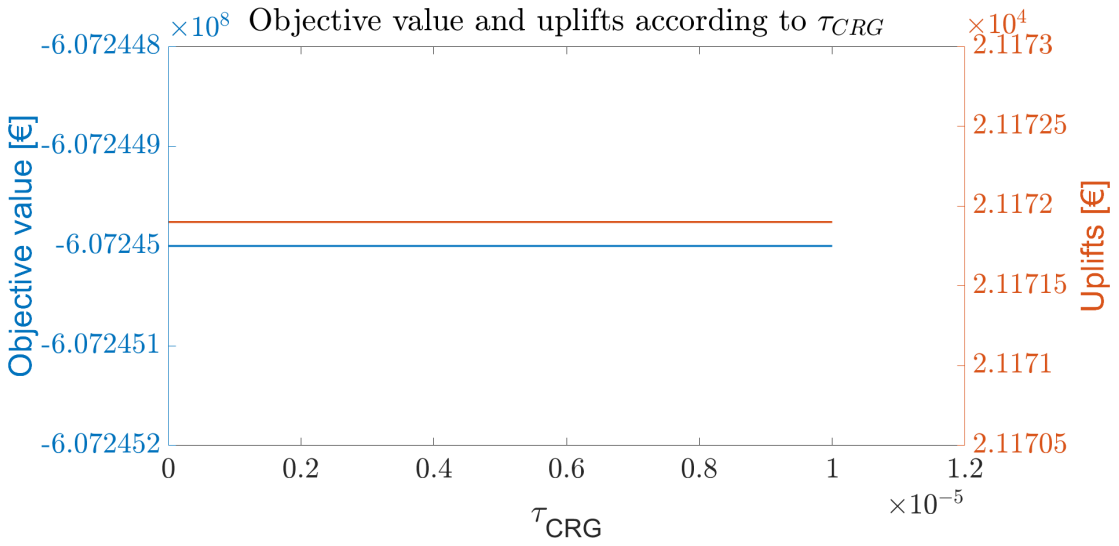


Figure 6.5: Objective value and uplifts for real-world example *Winter-WE* considering different values of τ_{CRG}.

6.5 Column-and-Row Generation versus Column Generation

Column generation and column-and-row generation are both algorithm adding columns with negative reduced cost to a restricted master program. But the restricted master program is not the same. Primary interest to implement $\bar{R}_{LP}$, instead of $\bar{M}_{PL}$, is to exploit second inequality of proposition 6.1, i.e. $v^R_{LP} \leq v^M_{LP}$.

Consider the column generation restricted master program $\bar{M}_{LP}$ 5.6. It is a Dantzig-Wolfe decomposition considering the balance constraint and convex combinations as its feasible constraints. Columns are feasible schedules, i.e. fixed set of vector $(\hat{p}, \hat{\bar{p}}, \hat{u}, \hat{v}, \hat{w})$. Schedules are considered as parameters by restricted master program CG of column generation.

Consider the column-and-row generation restricted master program $\bar{R}_{LP}$ 6.8. It is a restricted extended formulation considering packing polytope D 3.12 of its production intervals, and the balance constraint as its feasible region. The columns are production intervals $[a, b]$. Adding columns to $\bar{R}_{LP}$ means to add all variables $(p^{[a,b]}, \bar{p}^{[a,b]}, \gamma_{[a,b]})$ and constraints $D^{[a,b]}$ 3.12 to $\bar{R}_{LP}$, where for the regular column generation CG, it is only means to add feasible schedules and variables z^{n_g}.

In practice $\bar{R}_{LP}$ considers more variables and constraints than $\bar{R}_{LP}$. Table 6.6 reports number of variables and constraints needed to build the master programs

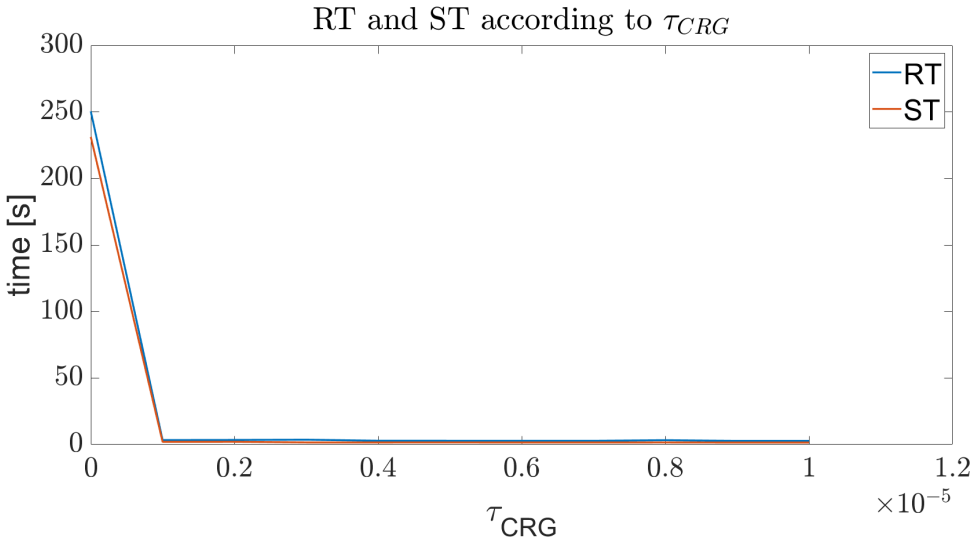


Figure 6.6: RT and ST for real-worl example *Wintr-WE* considering different values of τ_{CRG}.
RT : total Running Time of the program.
ST : Solving Time of the program.

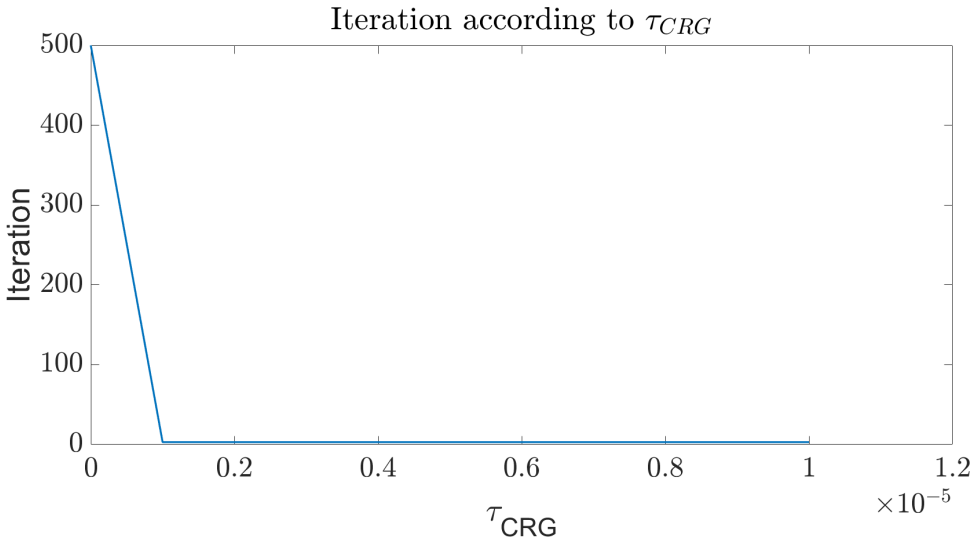


Figure 6.7: Number of iterations column-and-row generation needs to converge for real-world example *Winter-WE* considering different values of τ_{CRG}.

Iteration	Row Generation		Column-and-Row Generation	
	Number Variables	Number Constraints	Number Variables	Number Constraints
1	12,786	35,368	92	276
2	13,581	37,213	153	398
3	13,634	37,392	214	520
⋮			⋮	⋮
29			861	1,814

Table 6.6: Number of variables and constraints building restricted master program according to the iterate of CG and CRG considering real-world example 6.2.

of column generation and column-and-row generation. Since the first iteration, $\bar{R}_{LP}$ considers more variables and constraints than $\bar{M}_{LP}$. However this difference is also visible in the time required to perform restricted master programs. One can see detailed solving time for column generation and column-and-row generation on figure 6.8. Column-and-row iterations converges in less iteration, but each iteration for its restricted master program $\bar{R}_{LP}$ 6.8 and its slave program *Pricing Problem* 6.11 needs more time to be solved, than the ones of column generation. Restricted master program $\bar{M}_{LP}$ 5.6 of column generation is simply a linear program with variables z^{n_g}, since feasible schedules are parameters It is thus solved in a small amount of time. The solving time required for all the slave programs 5.8, i.e. one per generator, of column generation are summed up and considered as one big slave program for each iteration. But still, they need less time than *Pricing Problem* to be solved.

Key Properties of the interest of Column-and-Row Generation

Source [24] advances **Recombination** property 6.1. One may not need to generate further columns to achieve some solution $Q \backslash \text{conv}\left(Z\left(\bar{S}\right)\right)$, i.e. that is in the extended formulation of conv (X), but not the part already representing conv $\left(Z\left(\bar{S}\right)\right)$. Intuitively, production intervals are fixed, but the way generators produce in those intervals is free. Hence it can represent different feasible schedules $\left(\hat{p}, \hat{\bar{p}}, \hat{u}, \hat{v}, \hat{w}\right)$ from CG. With one iteration CRG can thus represent several iterations of CG. For all the coming data samples, CRG always needs less iterations than CG to converge.

Property 6.1 (Recombination)**.** *From sources [24].*
Given $\bar{S} \subset S$, $\exists \tilde{z} \in R_{LP}\left(\bar{S}\right)$, *such that* $\tilde{z} \notin conv\left(Z\left(\bar{S}\right)\right)$

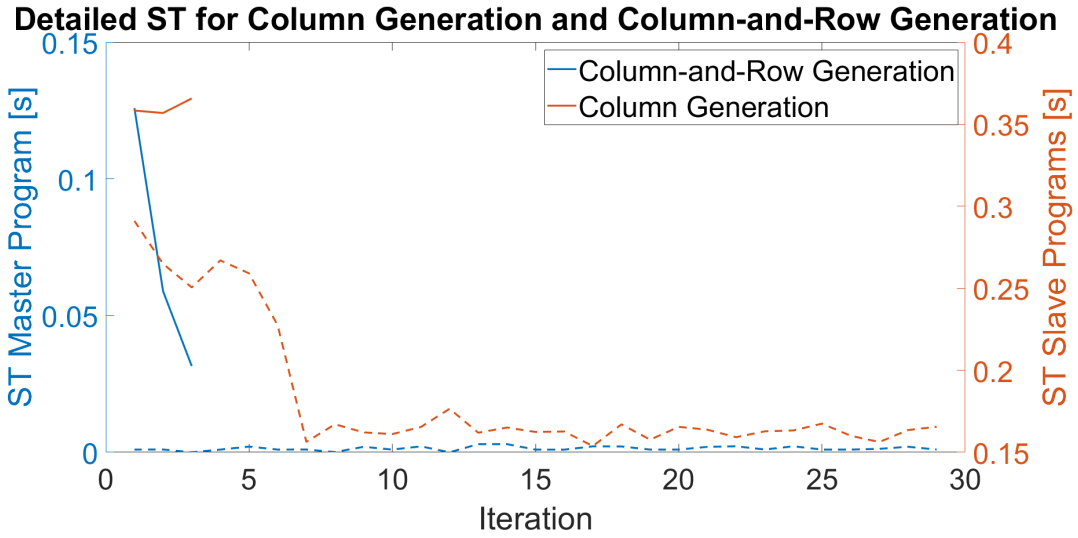


Figure 6.8: Solving Time (ST) for master and slave programs considering both column generation and column-and-row generation.

Part III

Comparison of the Methods

Chapter 7

Computational Experiments and Conclusion

All different methods, the extended formulation, row generation, column generation, and column-and-row generation, compute CHP for real-world day-ahead electricity markets. They have been detailed in previous chapters and tested on one real-world example, that is *Winter-WE*. Table 6.5 summarises performances of the different methods for this specific example. This chapter compares the four methods, when possible, on four data sets considering different amounts of generators and different time horizons.

RT is the *total Running Time* needed to provide CHP, for a given method, and ST is the *total Solving Time* `Gurobi` needs to provide CHP, for a given method. Those two quantities are used to describe the behaviour of the methods when computing CHP. The main performance indicator is RT. ST is an additional performance indicator allowing to see what percentage of RT is dedicated to solving optimization problems.

All computational experiments were performed on an Asus Harkort STR 24 laptop with a 7 Intel 6500U processor, and 8 Go of RAM, running Windows 10. `Gurobi` 9.1.1 was used to solve all the presented optimization problems using Julia as the programming language. All parameters of `Gurobi` were preserved at default, in particular the feasibility tolerance, the optimality tolerance and the barrier convergence tolerance, for which the default value is $1e-8$.

For all the data sets, prior and posterior productions are set to 0, as explained in section **Prior and Posterior Production** 3.7.

7.1 `small_belgian` Data Set

The first data set is called `small_belgian`. It considers the 68 Belgian generators given in tables 9.5 and 9.6. Time horizon $[T]$ is over 24 hours with a timestep of 1 hour, i.e. $[T]$ has a length 24. Table 9.1 is in the appendix, and it reports performances of the four methods providing CHP for all the data samples of the data set considered. In table 9.1, the smallest RT value between all the four methods is highlighted in green. Figure 7.1 reports the average performances of the methods considering all data samples.

The extended Formulation As expected the extended formulation is the method with the highest average RT and ST values, as one can see on figure 7.1. This means it is the slowest method to provide CHP. Average RT and ST values are respectively 206.265 seconds and 168.763 seconds. Such values are a consequence of the number of variables and constraints required to build the complete optimization program. Indeed on average among all the data samples there are 607,446 variables and 2,080,438 constraints required to model the extended formulation. On average solving the optimization program takes 81% of the total running time RT. The algorithm does not spend much time doing other things than solving the optimization program. The aim for the other three methods is to improve both RT and ST values of this method.

Row Generation It provides CHP faster than the extended formulation. Average RT is 143.595 seconds and average ST is 59.96 seconds. As one can see on figure 7.1 those values for RT and ST are already lower, for row generation, than for the extended formulation. For all the data samples it was possible to build the slave programs only once, and then to store them in memory. Solving optimization programs takes on average only 41% of RT. The rest of the time is spent building slave programs, among other things. Even if this percentage is not very high, keeping slave programs in memory increases the percentage of solving time ST among the total running time RT. This percentage would be higher if all slave programs would need to be re-built at each step.

The expensive step of the algorithm is checking if the master solution is feasible for all the slave programs. Solving master programs takes on average 0.829 seconds, i.e. 1.3% of ST, and solving slave programs takes on average 59.131 seconds, i.e. 98.7% of ST. It makes sense since the master program is a simple linear program and slave programs are extended formulations for the generators. Based on the extended formulation, row generation makes improvements of 30% for RT and 64% for ST.

Column generation It is faster than row generation, and as a consequence it also improves the extended formulation. Average RT and ST values for column generation are represented on figure 7.1. Both values are lower than the ones of row generation. They are respectively 8.953 seconds and 4.739 seconds. The method spends 52% of its running time solving optimization programs. This percentage is less than the one of the extended formulation because column generation needs to build all the slave programs.

Solving restricted master programs $\bar{M}_{LP}$ of column generation takes on average 0.06 seconds, i.e. 1% of ST, and solving slave programs takes on average 4.679 seconds, i.e. 99% of ST. As for row generation, the expensive step is running slave programs. However it needs less time to run slave programs for column generation than for row generation. This has an impact on both RT and ST values, that are on average smaller for column generation than for row generation. Based on the extended formulation, column generation makes improvements of 95% for RT and 97% for ST. Those improvements are already better than the ones of row generation.

Column-and-row Generation For column-and-row generation, average RT and ST values are respectively 5.349 seconds and 1.475 seconds. They are the smallest ones, as one can see on figure 7.1. Solving optimization programs takes on average 27% of RT. This percentage is smaller than the one of the other methods because the steps in the algorithm, that are not solving optimization programs, require more time. Indeed the method needs time to generate production intervals, and to add the related constraints and variables to the master program.

Solving restricted master programs $\bar{R}_{LP}$ takes on average 0.108 seconds, i.e. 7% of ST, and solving slave programs takes on average 1.367 seconds, i.e. 93% of ST. Again, the expensive step is running the salve programs. It is on average faster to solve slave programs of column-and-row generation than to solve the ones of row generation or column generation. Column-and-row generation also needs less iterations than column generation to converge. Hence column-and-row generation provides CHP faster than column generation. It is actually the fastest method to provide CHP for this data set. Based on the extended formulation, column-and-row generation improves even more RT and ST values than column generation. The improvements for RT and ST respectively reach 97% and 99%.

Column-and-row generation provides the highest improvements based on the extended formulation. It is on average the fastest method to provide CHP for the data set `small_belgian`.

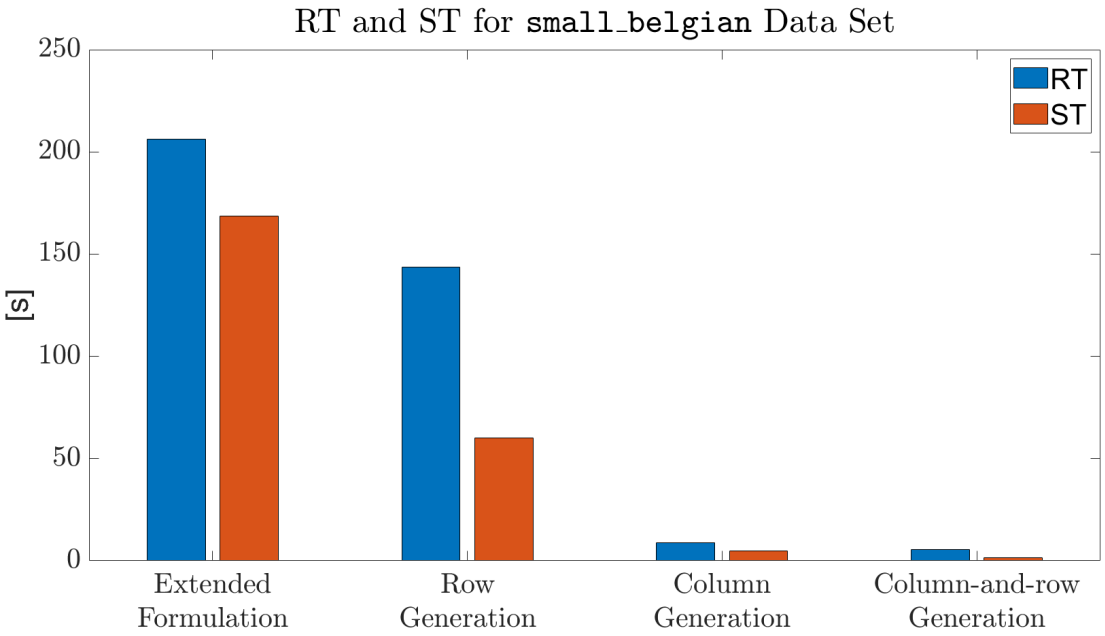


Figure 7.1: Average RT and ST for the data set `small_belgian`.

7.2 `rts_gmlc` Data Set

The second data set is `rts_gmlc`, [3]. It considers 73 generators from US. The time horizon $[T]$ is over 48 hours with a timestep of 1 hour, i.e. $[T]$ has a length of 48. This data set has almost the same number of generators as the previous data set `small_belgian` but it considers a bigger time horizon. For this data set, as prior and posterior productions are set to 0, the demand has been cut in two to have prices closer to the reality, i.e. different than $VOLL$ for each hour. Table 9.2, in the appendix, reports performances of the four different methods. For each data sample the smallest RT value between all the four methods is highlighted in green. Figure 7.2 reports the average performances of the methods considering all data samples of the current data set `rts_gmlc`.

The Extended Formulation For the `rts_gmlc` data set it is not possible to build the extended formulation. `Gurobi` needs too much memory for the optimization program to be completely modelled. Hence it is not possible to provide RT and ST values. Neither RT, nor ST are described on figure 7.2.

Row Generation Since the extended formulation can not be built, it is straightforward that all the slave programs of row generation can not be built once and then kept in memory. The number of variables and constraints of each slave program

depends on the technical constraints of generator g. On average, among all the 78 generator from the data sample `2020-01-27`, a slave program has 100,120 variables and 327,633 constraints. One may think it is not a big issue to re-build slave programs at each iteration. However for the data sample `2020-01-27`, the first iteration takes 451.921 seconds, i.e. 7 minuntes and 31 seconds. If row generation converges after 30 iterations it would roughly take 3 hours and 45. Row generation does not converge in less than 30 minutes for any of the data samples from `rts_gmlc`. In comparison to column generation and column-and-row generation, row generation is completely outdated. Row generation is considered as unable to provide CHP, therefore neither RT, nor ST are described on figure 7.2.

Column Generation This method is able to provide CHP in a reasonable amount of time, and it does even better than reasonable time. Average RT value is 66.694 seconds and average ST value is 54.812 seconds. As for the previous data set, RT and ST values are described on figure 7.2. The method spends 82% of its total running time solving optimization programs. This percentage is higher than for the previous data set because there are much more iterations performed for this data set than for the previous one.

Solving restricted master programs $\bar{M}_{LP}$ takes on average 2.747 seconds, i.e. 5% of ST, and solving slave programs takes on average 52.065 seconds, i.e. 95% of ST. As for the previous data set, the expensive step of the algorithm is solving the slave programs. Since neither the extended formulation, nor row generation provide CHP, column generation serves as a basis to be improved by column-and-row generation.

Column-and-row Generation It provides CHP faster than column generation, as one can see on figure 7.2. The method has average RT value of 10.871 seconds and average ST value of 4.797 seconds. It spends 44% of the total running time RT solving optimization programs. This percentage is lower than the one of column generation because the steps in the algorithm, that are not solving optimization programs, require more time. However this percentage is higher than the one of the previous data set because there are more performed iterations for this data set. The total amount of time needed to build the slave programs is the same, but the percentage is different because the method runs longer.

Restricted master program $\bar{R}_{LP}$ of column-and-row generation takes on average 1.184 seconds to be solved, i.e. 24.7% of ST, and slave program takes on average 3.614 second to be solved, i.e. 75.3% of ST. As for all the previous methods, the expensive step is solving slave programs. Based on column generation, there is an improvement of 83% for RT and 91% for ST.

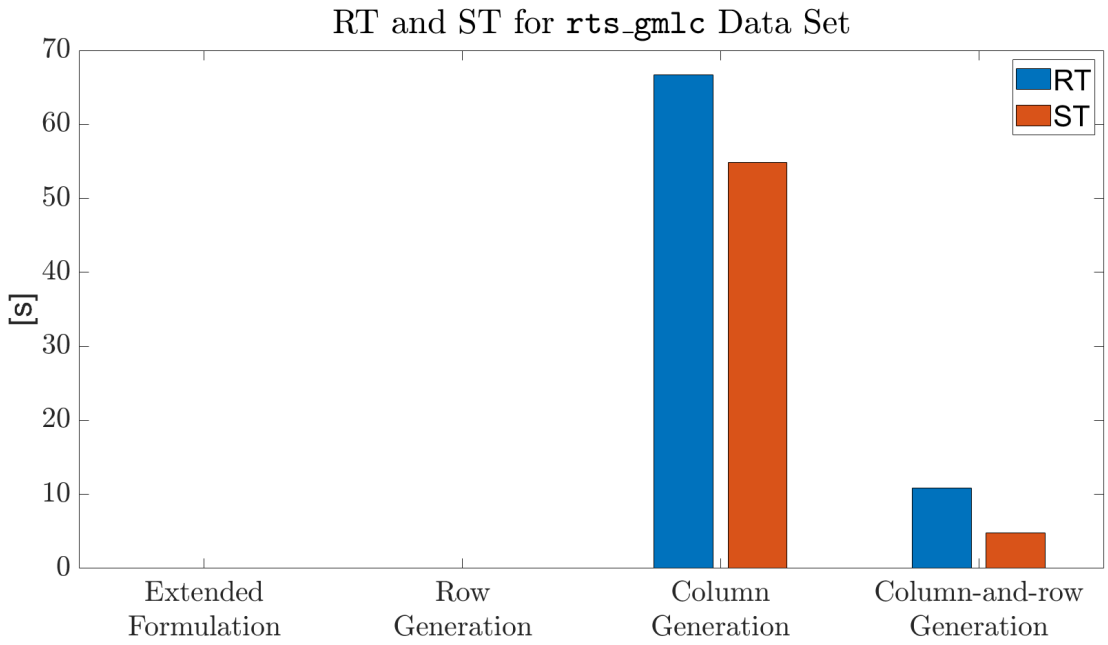


Figure 7.2: Average RT and ST for the data set `rts_gmlc`.

Column-and-row generation is clearly the fastest method to compute CHP for the `rts_gmlc` data set, just like for the `small_belgian` data set.

7.3 `ferc` Data Set

The third data set is `ferc` [17], [19]. It considers between 934 and 978 generators, depending on the data sample. The two first data samples `2015-03-01_hw` and `2015-03-01_lw` consider 934 generators. The other data samples consider 978 generators. Table 9.3 is in the appendix, and it reports the performances of the four different methods. The time horizon $[T]$ is over 48 hours with a timestep of 1 hour, i.e. $[T]$ has a length of 48. For this data set, as prior and posterior productions are set to 0, the demand has been cut in twelve to have prices closer to the reality, i.e. different than $VOLL$ for each hour. For each data sample the smallest RT value between all the four methods is highlighted in green. Figure 7.3 reports average RT and ST values for all the four methods considering the current data set `ferc`.

The Extended Formulation As expected, it is not possible to build the extended formulation. The `ferc` data set considers more generators than the previous `rts_gmlc` data set, for which it was already impossible to build the extended formulation. There are too many variables and constraints for `Gurobi` to store the

optimization problem in memory. It is not possible for the extended formulation to provide RT and ST values. They are both not described on figure 7.3.

Row Generation Row generation can not store the slave programs in memory. It was expected since there are more generators for this data set than for the `rts_gmlc` data set, and the time horizon [T] is the same as the one in `rts_gmlc`. On average among all the 934 generators from the data sample `2015-03-01_hw`, a slave program has 88,408 variables and 293,636 constraints. At each iteration of row generation all the 934 slave programs are re-built from scratch. It is costly and one iteration roughly takes 4336.371 seconds, i.e. 1 hour and 12 minutes. Assuming row generation has to perform 30 iterations, the average number of iterations from the `small_belgian` data set, it would roughly take 36 hours and 7 minutes for row generation to provide CHP. Again, for this data set row generation never provides CHP in less than 30 minutes. In comparison to column generation and column-and-row generation, row generation is again completely outdated. Row generation is considered as unable to provide CHP, therefore neither RT, nor ST are described on figure 7.3.

Column Generation This method is able to provide CHP and does even better than simply providing CHP. Average RT and ST values, among all the data samples, are respectively 240.602 seconds and 165.127 seconds. This means that column generation spends on average 68% of its total running time RT solving optimization programs. The time needed to build the slave programs is bigger because there are more generators. Hence this building time is bigger in the total running time. It decreases the percentage of solving time ST. This percentage is also lower than the one of column generation applied to the `rts_gmlc` data set, because there are less iterations performed by column generation.

Solving master programs takes on average 3.395 seconds, i.e. 2% of ST, and solving slave programs takes on average 161.732 seconds, i.e. 98% of ST. As previously, solving slave programs is the expensive step of the algorithm. Column generation is the fastest method for two lasts data samples in table 9.3. However on average it is not the fastest one, as one can see on figure 7.3.

Column-and-row Generation Column-and-row generation is able to provide CHP faster than column generation. Average RT value is 188.655 seconds and average ST value is 138.091 seconds. This means that the method spends 73% of its total running solving optimization problems. This percentage is higher because the problems to be solved are bigger and take more time. Hence the method spends more time solving problems and the percentage increases.

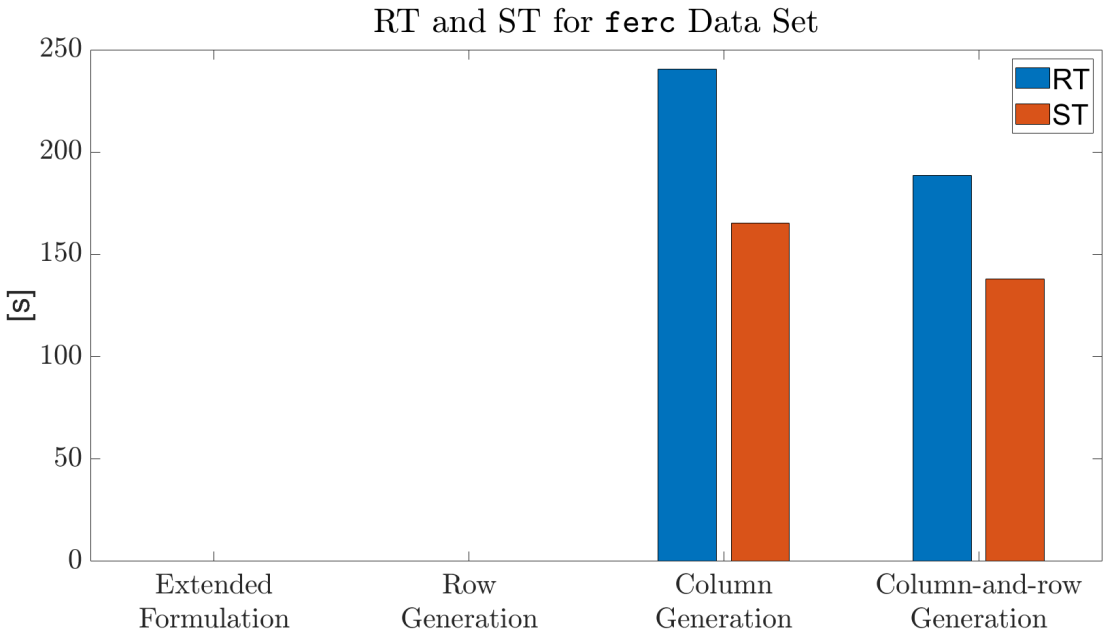


Figure 7.3: Average RT and ST for the data set `ferc`.

Column-and-row generation spends on average 38.184 seconds solving restricted master programs $\bar{R}_{LP}$, i.e. 27.6% of ST, and it spends on average 99.907 seconds solving slave programs, i.e. 72.4% of ST. The expensive step is again solving the slave programs. Based on column generation, column-and-row generation makes improvements of 21% for RT and 16% for ST.

As for the `small_belgian` and `rts_gmlc` data sets, column-and-row generation provides CHP faster than the other methods.

7.4 `big_belgian` Data Set

The fourth data set is `big_belgian`. It considers the same 68 Belgian generators as for the previous `small_belgian` data set, given in tables 9.5 and 9.6. The demand is still over 24 hours but it considers a timestep of 15 minutes. Hence the length of the time horizon vector $[T]$ is 96. Table 9.4, in the appendix, reports performances of the four different methods. For each data sample of the `big_belgian` data set, the smallest RT value between all the four methods is highlighted in green.

The Extended Formulation It is not possible to build the extended formulation for any of the data samples. There are too many variables and constraints for `Gurobi` to store the optimization problem in memory. It is not possible to have

RT and ST values for the extended formulation. They are both not described on figure 7.4

Row Generation As expected, row generation can not store the slave programs in memory. Indeed, since the extended formulation can not be built due to a lack of memory, it is straightforward that building all the salve programs representing the extended formulation of the corresponding generators can not be stored in memory. On average, considering all the 68 generators from data sample `AutumnWE`, a slave program has 395,082 variables and 1,306,185 constraints. At each iteration of row generation, all the 68 slave programs must be re-built from scratch. Therefore one iteration roughly takes 12,728.92 seconds, i.e. 3 hours and 31 minutes. Assuming row generation has to perform 30 iterations, it would take 106 hours, i.e. 4 days. In comparison to column generation and column-and-row generation, row generation is completely outdated, just like for the `rts_gmlc` and `ferc` data sets. Row generation is considered as unable to provide CHP, therefore neither RT, nor ST are described on figure 7.4.

Column Generation Column generation is able to provide CHP in a reasonable amount of time. On average among all the data samples, RT and ST values are respectively 320.485 seconds and 294.532 seconds. It means that column generation spends on average 91% of the time total running RT solving optimization problem. In comparison to previous data sets, this percentage for column generation is higher. It increases due to the number of iterations performed. Column generation simply solves more optimization programs on average for this data sets than for the other data sets.

Solving restricted master programs $\bar{M}_{LP}$ takes on average 4.302 seconds, i.e. 1.4% of ST, and solving slave programs takes on average 290.229 seconds, i.e. 98.6% of ST. As previously, solving slave programs is the expensive step in the column generation algorithm, and by far. Column generation is not the fastest method to provide CHP, as one can see on figure 7.4.

Column-and-row Generation Column-and-row generation is on average the fastest method. Average RT value is 45.691 seconds and average ST value is 33.664 seconds. Solving optimization programs takes on average 73% of the total running time RT. This percentage is higher than the previous data sets, because the method as to perform more iterations to provide CHP. Hence it solves more optimization programs.

Solving restricted master programs $\bar{R}_{LP}$ takes on average 23.604 seconds, i.e. 12.5% of ST, and solving slave programs takes on average 165.050 seconds, i.e.

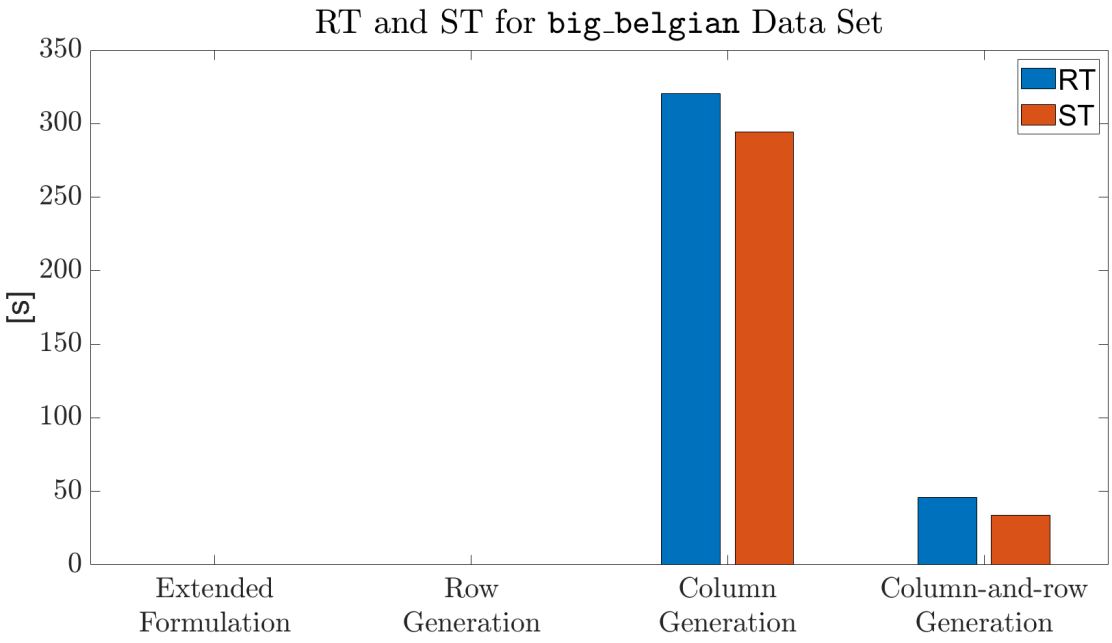


Figure 7.4: Average RT and ST for the data set `big_belgian`.

87.5% of ST. Average RT and ST values from column-and-row generation are smaller than the ones from column generation. As all previous data sets, the expensive step is solving slave programs. Based on column generation, column-and-row generation makes improvements of 21% for RT and 16% for ST.

Column-and-row generation is the fastest method to provide CHP for the `big_belgian` data set, as one can see on figure 7.4.

Chapter 8

Conclusion

The extended formulation provides a compact and tight extended formulation for the convex hull conv $\left(\Pi^g_{3-bin}\right)$. The linear program provided is too big to be efficiently solved, or even sometimes to be built, for real-world markets. Hence row generation, column generation, and column-and-row generation are iterative methods that improve the total running time and the solving time of the extended formulation.

8.1 Main Results and Discussion

The four methods providing CHP are analyzed according to the three following questions.

1. Is the method able to provide CHP at some point ?
   This question defines whether the method is going to provide CHP, even after a long time of computation.

2. Can the method be considered as a solution to provide CHP ?
   If the method can provide CHP, then it looks if they can be provided in a reasonable amount of time. The reasonable limit time is set to 30 minutes. For a total running time RT bigger than 30 minutes the method is considered as outdated, as one can see on previous figures 7.1, 7.2, 7.3, 7.4.

3. Is the method considered as THE solution to provide CHP ?
   If several methods can provide CHP in a reasonable amount of time, then the method considered as the solution to provide CHP is the fastest method on average among the four data sets `small_belgian`, `rts_gmlc`, `ferc` and `big_belgian`.

The Extended Formulation This method is the basic one. It simply considers the large linear optimization program EF 3.12. The extended formulation quickly reaches its limits. It can not be built for the three last data sets `rts_gmlc`, `ferc` and `big_belgian`. It considers more variables and constraints than `Gurobi` can handle. Hence a method being able to provide CHP for any of the data sets is already an improved method, in comparison to the extended formulation.

Row Generation This method can be used to solve all the data sets, therefore it already improves the extended formulation. For the first data set `small_belgian`, row generation provides CHP faster than the extended formulation, but clearly not as fast as the two other improved methods. Even if row generation can be used to have CHP, it does not mean that row generation is able to always provide CHP. Indeed for the three last data sets, namely `rts_gmlc`, `ferc` and `big_belgian`, row generation is not able to build the slave programs only once and to keep them in memory. They must be re-built at each iteration. It impacts the total running time. Considering that row generation has to perform 30 iterations, it needs more than 30 minutes to provide CHP. The estimated convergence times for the three last data sets are :

- 3 hours and 45 minutes for `rts_gmlc`.
- 36 hours and 7 minutes for `ferc`.
- 106 hours for `big_belgian`.

Even if row generation improves the extended formulation, in the sense that it is able to provide CHP, it is not considered as a solution to obtain CHP.

Column Generation Column generation is able to provide CHP for any of the data sets in a reasonable time, i.e. always lower than 30 minutes. Hence column generation is an improved method in comparison to the extended formulation. It is also an improved method in comparison to row generation because it provides CHP in a reasonable time and therefore can be considered as a solution to have CHP.

Column-and-Row Generation As for column generation, column-and-row generation is able to provide CHP for any of the data sets in a reasonable time, i.e. lower than 30 minutes. Hence this method is already an improvement, in comparison to the extended formulation and row generation. It is actually the fastest method to provide CHP, on average. To illustrate this, table 8.1 reports improvements of RT and ST that column-and-row generation brings to column generation on average. Column-and-row generation is on average always improving column generation for any of the data sets.

Data Set	CG RT [s]	CG ST [s]	CRG RT [s]	CRG ST [s]	Improvements of CRG based on CG RT	Improvements of CRG based on CG ST
`small_belgian`	8.953	4.739	5.349	1.475	40%	68%
`rts_gmlc`	66.694	54.812	10.871	4.797	83%	91%
`ferc`	240.602	165.127	188.655	138.091	21%	16%
`big_belgian`	320.485	294.532	45.691	33.664	86%	89%

Table 8.1: Average performances of the four methods computing CHP according the different data sets.
RT : Total Running Time of the method.
ST : Solving Time of the method.

The Solution to Provide CHP is Column-and-row Generation This method provides CHP in an efficient way. Regarding the use of this method for an industrial purpose, it provides CHP in less than 4 minutes on a simple laptop, for any of the tested data sets.

Table 8.2 provides a summary of the methods and their answers to the three questions. Symbol ✗ means that the answer for the method to the related question is negative, and symbol ✓ means that the answer is positive.

	Able to Provide CHP ?	Considered as a Solution ?	Considered as THE Solution ?
Extended Formulation	✗	✗	✗
Row Generation	✓	✗	✗
Column Generation	✓	✓	✗
Column-and-row Generation	✓	✓	✓

Table 8.2: Analysis of the methods and their answers to the three questions.

8.2 Improving Paths

Modelling the Network
The day-ahead electricity market considered in this manuscript does not model the electricity network. It only considers one node where all producers input their production and where the one big consumer outputs its consumption. A good improvement to the current unit commitment problem would be to model the

transmission and distribution of the Belgian electricity network. Hence the model would be closer to the reality.

Source [9] is a good start to tackle non-convexities in the modelling of the network. It gives a general version providing CHP solving multiobjective minimum uplift problem and potential congestion revenue shortfall.

Consider Prior and Posterior Production
Section **Prior and Posterior Production** 3.7 shows why considering prior production intervals $[a = 0, b]$, or posterior production intervals $[a, b = T + 1]$, is not always sufficient. Hence in this manuscript all prior and posterior productions have been set to zero production. To have a complete modelling of the demand it would be interesting to be able to have non-zero production for prior and posterior intervals.

A possible solution could be to take the smallest value a^* such that all committed production, i.e. $p_{t=0} > 0$, are feasible in the time horizon $[a^*, b^*]$, and to take the biggest value b^* such that all posterior production, i.e. $p_{t=T+1} > 0$, are feasible in the time horizon $[a^*, b^*]$. The considered time horizon would be $[T^*] = [a^*, b^*]$, where $a^* \leq 0$ and $T + 1 \leq b^*$. Column-and-row generation dynamically adds intervals. Hence it is not necessary to consider the complete time horizon $[a^*, b^*]$ and the complete set of intervals $\mathcal{T}^* := \{[a, b] \,|a^* \leq a \leq a + (UT - 1) \leq b \leq b^*\}$.

$$a^* = \mathbf{min}_{p_{t=0}>0} \left\{0 - \tfrac{p_{t=0}-SU}{RU}\right\} \qquad b^* = \mathbf{max}_{p_{t=T+1}>0} \left\{(T+1) + \tfrac{p_{t=T+1}-SD}{RD}\right\}$$

a^* and b^* are found using constraint 3.4f and constraint 3.4g.

Spinning Reserves
Reserves have not been implemented in the market. It could be interesting to add this feature to the market and compute the related prices. Therefore the method would be able to both provide prices for the electricity and the reserves.

Column-and-row generation method is already providing promising time performances. Implementing the improving paths would allow to have a closer look at the performances of this method for an industrial purpose.

Part IV

Appendix

Chapter 9

Appendix

9.1 Computational Experiments : Performances

Data Sample	Extended Formulation RT [s]	Extended Formulation ST [s]	Row Generation RT [s]	Row Generation ST [s]	Row Generation # iter	Column Generation RT [s]	Column Generation ST [s]	Column Generation # iter	Column-and-row Generation RT [s]	Column-and-row Generation ST [s]	Column-and-row Generation # iter
Autumn-WD	134.518	90.148	70.898	37.904	23	9.822	2.396	9	12.720	1.085	2
Spring-WD	166.572	132.223	65.421	37.231	24	5.321	2.220	11	2.896	1.202	3
Summer-WD	170.152	134.801	69.917	41.927	26	6.063	3.487	18	2.233	1.234	3
Winter-WD	110.623	75.182	62.181	36.956	22	3.539	1.259	7	1.740	0.779	2
Autumn-WE	242.648	208.194	98.402	69.761	31	9.085	5.703	30	4.023	2.099	5
Winter-WE	242.347	197.293	123.585	77.746	31	11.976	5.346	29	11.42	1.297	3
Spring-WE	291.981	256.638	120.471	90.647	41	13.667	9.417	38	4.056	2.166	5
Summer-WE	291.283	255.625	117.309	87.510	43	12.153	8.089	39	3.710	1.943	5
On average	206.265	168.763	143.595	59.960	30.125	8.953	4.739	22.625	5.349	1.475	3.5
Improvement of EF	0%	0%	30%	64%	-	95%	97%	-	97%	99%	-

Table 9.1: Performances of extended formulation, row generation, column generation and column-and-row generation on the data set `small_belgian`. The data set considers 68 generators and time horizon is on 24 hours with timestep of 1 hour.
RT : Total Running Time of the method.
ST : Solving Time of the method.
iter : number of iterations performed by the method.

Data Sample	Extended Formulation RT [s]	Extended Formulation ST [s]	Row Generation RT [s]	Row Generation ST [s]	Row Generation # iter	Column Generation RT [s]	Column Generation ST [s]	Column Generation # iter	Column-and-row Generation RT [s]	Column-and-row Generation ST [s]	Column-and-row Generation # iter
2020-01-27	-	-	>30min	>30min	-	101.776	82.361	156	20.373	3.014	4
2020-02-09	-	-	>30min	>30min	-	69.414	56.480	126	11.599	5.951	5
2020-03-05	-	-	>30min	>30min	-	85.815	71.744	147	7.440	3.754	5
2020-04-03	-	-	>30min	>30min	-	82.991	69.437	144	8.292	3.576	4
2020-05-05	-	-	>30min	>30min	-	81.678	69.257	143	11.304	5.286	5
2020-06-09	-	-	>30min	>30min	-	62.065	50.228	111	11.608	4.831	5
2020-07-06	-	-	>30min	>30min	-	53.158	44.586	106	7.483	3.968	5
2020-08-12	-	-	>30min	>30min	-	25.624	18.939	51	11.018	5.564	5
2020-09-20	-	-	>30min	>30min	-	53.331	43.324	105	10.403	4.835	6
2020-10-27	-	-	>30min	>30min	-	55.328	45.424	92	10.190	5.573	6
2020-11-25	-	-	>30min	>30min	-	67.794	54.588	120	12.083	6.646	6
2020-12-23	-	-	>30min	>30min	-	61.358	51.384	107	8.662	4.577	5
Average	-	-	-	-	-	66.694	54.812	106.5	10.871	4.797	5.08
Improvement of *CG*	-	-	-	-	-	0%	0%	-	83%	91%	-

Table 9.2: Performances of column generation and column-and-row generation on the data set `rts_gmlc`. The data set considers 73 generators and time horizon is on 48 hours with a timestep of 1 hour.
RT : Total Running Time of the method.
ST : Solving Time of the method.
iter : number of iterations performed by the method.

Data Sample	Extended Formulation RT [s]	Extended Formulation ST [s]	Row Generation RT [s]	Row Generation ST [s]	Row Generation # iter	Column Generation RT [s]	Column Generation ST [s]	Column Generation # iter	Column-and-row Generation RT [s]	Column-and-row Generation ST [s]	Column-and-row Generation # iter
2015-03-01_hw	-	-	>30min	>30min	-	181.940	124.016	45	120.125	75.359	4
2015-03-01_lw	-	-	>30min	>30min	-	166.124	112.732	42	111.618	79.234	4
2015-04-01_hw	-	-	>30min	>30min	-	369.113	276.190	78	158.328	105.721	4
2015-04-01_lw	-	-	>30min	>30min	-	318.927	229.850	67	168.865	113.626	4
2015-05-01_hw	-	-	>30min	>30min	-	102.703	46.458	11	54.914	18.571	1
2015-05-01_lw	-	-	>30min	>30min	-	90.395	36.186	8	53.507	16.848	1
2015-06-01_hw	-	-	>30min	>30min	-	353.580	249.001	70	315.378	240.260	7
2015-06-01_lw	-	-	>30min	>30min	-	400.565	296.711	83	291.098	227.436	7
2015-07-01_hw	-	-	>30min	>30min	-	314.361	229.230	65	298.616	235.297	5
2015-07-01_lw	-	-	>30min	>30min	-	262.500	182.937	56	251.439	199.698	4
2015-08-01_hw	-	-	>30min	>30min	-	174.187	100.728	28	220.687	164.655	5
2015-08-01_lw	-	-	>30min	>30min	-	152.830	97.489	27	219.295	180.390	5
Average	-	-	-	-	-	240.602	165.127	48.33	188.655	138.091	4.25
Improvement of CG	-	-	-	-	-	0%	0%	-	21%	16%	-

Table 9.3: Performances of column generation and column-and-row generation on the data set `ferc`. The two first data samples consider 934 geenrators. The other data samples consider 978 generators . Time horizon is on 48 hours with a timestep of 1 hour.
RT : Total Running Time of the method.
ST : Solving Time of the method.
iter : number of iterations performed by the method.

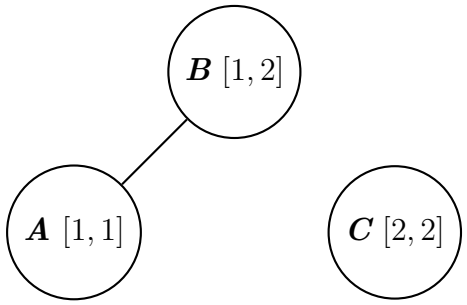
Data Sample	Extended Formulation RT [s]	Extended Formulation ST [s]	Row Generation RT [s]	Row Generation ST [s]	Row Generation # iter	Column Generation RT [s]	Column Generation ST [s]	Column Generation # iter	Column-and-row Generation RT [s]	Column-and-row Generation ST [s]	Column-and-row Generation # iter
Autumn-WE	-	-	>30min	>30min	-	438.571	401.993	273	69.984	52.114	11
Spring-WE	-	-	>30min	>30min	-	705.126	659.137	354	43.792	34.469	10
Summer-WE	-	-	>30min	>30min	-	725.615	677.638	363	56.753	43.646	11
Winter-WE	-	-	>30min	>30min	-	290.674	267.276	195	78.149	68.593	11
Autumn-WD	-	-	>30min	>30min	-	91.032	73.979	65	40.196	15.239	3
Spring-WD	-	-	>30min	>30min	-	105.460	93.476	73	21.444	14.996	3
Summer-WD	-	-	>30min	>30min	-	159.694	144.293	114	23.800	16.568	3
Winter-WD	-	-	>30min	>30min	-	47.710	38.468	32	31.410	23.689	5
Average	-	-	-	-	-	320.485	294.532	183.625	45.691	33.664	7.125
Improvement of CG	-	-	-	-	-	0%	0%	-	86%	89%	-

Table 9.4: Performances of column generation and column-and-row generation on the data set `big_belgian`. There are 68 generators. Time horizon is on 24 hours with a timestep of 15 minutes, $[T]$ has a legnth of 96.
RT : Total Running Time of the method.
ST : Solving Time of the method.
iter : number of iterations performed by the method.

9.2 Illustrative Representations of Restriction Γ

Restriction Γ enforces a generator g to only chose non-overlapping intervals using clique inequalities. Example 9.1 gives an intuition on the graphs that are built at each time instance t and the cliques on those graphs.

Example 9.1. *Consider a generator such that its minimum up time and down time are $UT = 1 = DT$. Consider a time horizon of two time instances, i.e. $T = 2$ and $[T] = \{1, 2\}$. $\mathcal{T} = \{[1, 1]; [1, 2]; [2, 2]\}$ represent all intervals respecting the minimum up time. From undirected graph $G = (V, E)$, the set of vertices is $V = \{[1, 1]; [1, 2]; [2, 2]\}$ and an edge connects two vertices when there are overlapping intervals. Edge x_{ij} connects node i to node j.*

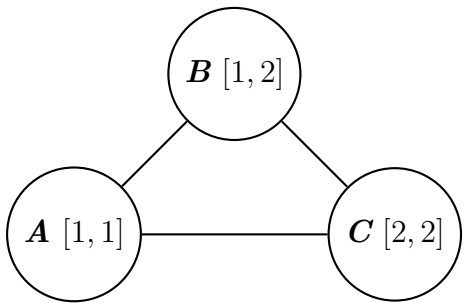


For $t = 1$ the basic clique inequality is given by the following expressions : $x_{AB} \leq 1$

*Considering the graph problem with clique inequalities at $t = 1$ both nodes **A** $[1, 1]$ and **B** $[1, 2]$ are overlapping. Therefore those two nodes are connected by the edge x_{AB}.*

Figure 9.1 also expresses the fact that intervals $[1, 1]$ and $[1, 2]$ are overlapping intervals at time $t = 1$. One can see that intervals $[a, b + DT]$ intersecting vertical black line at $t = 1$ have their indicator variable in the inequality. The basic clique inequality $x_{AB} \leq 1$ can be rewritten with indicator variables $\gamma_{[1,1]}$ and $\gamma_{[1,2]}$.

$$x_{AB} \leq 1 \Longrightarrow \gamma_{[1,1]} + \gamma_{[1,2]} \leq 1$$



For $t = 2$ clique inequality is given by $x_{AB} + x_{AC} + x_{BC} \leq 1$.

*Three nodes **A** $[1, 1]$, **B** $[1, 2]$ and **C** $[2, 2]$ are overlapping intervals at time $t = 2$.*

One can see on figure 9.1 that interval $[1, 1]$ is indeed overlapping with two other intervals. Intervals $[a, b + DT]$ for $[1, 1]$, $[1, 2]$ and $[2, 2]$ are intersecting black vertical line at $t = 2$. The basic clique inequality can be rewritten with indicator variables $\gamma_{[1,1]}$, $\gamma_{[1,2]}$ and $\gamma_{[2,2]}$.

$$x_{AB} + x_{AC} + x_{BC} \leq 1 \Longrightarrow \gamma_{[1,1]} + \gamma_{[1,2]} + \gamma_{[2,2]} \leq 1$$

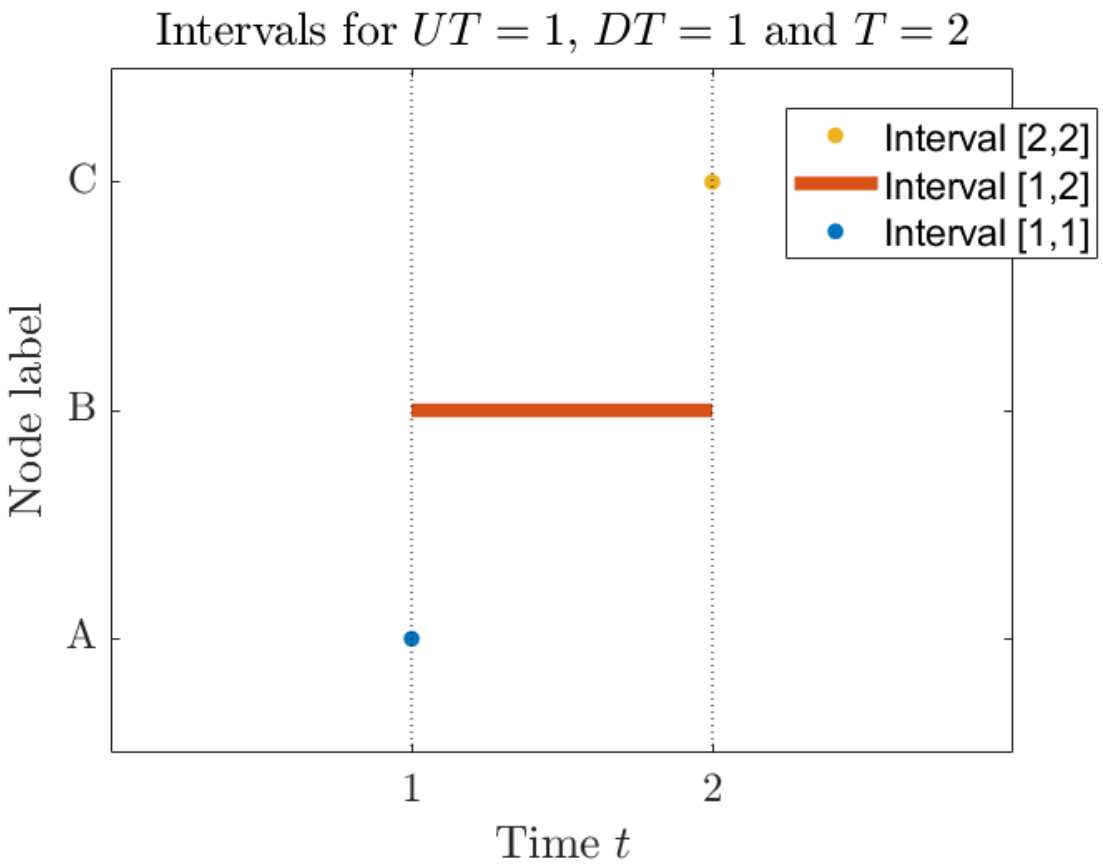
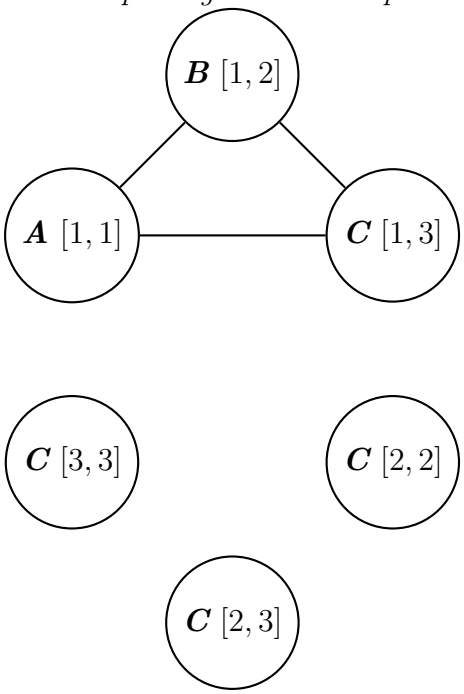


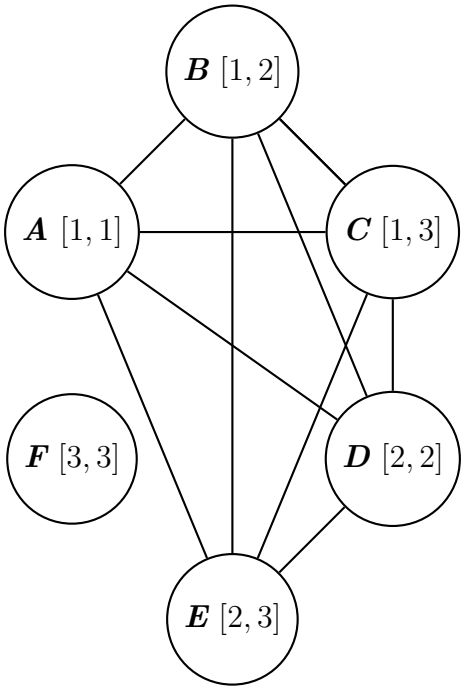
Figure 9.1: Illustrative representation of intervals for example 9.1.

Example 9.2. *Consider a more complex example than example 9.1. the time horizon is bigger,* $T = 3$*, but it keeps the minimum up time and down time* $UT = 1 = DT$*. Therefore* $\mathcal{T} = \{[1,1]; [1,2]; [1,3]; [2,2]; [2,3]; [3,3]\}$ *represents all intervals respecting minimum up time* UT*.*



One can see on figure 9.2 for $t = 1$ *that the three intervals* $[1,1]$*,* $[1,2]$ *and* $[1,3]$ *are overlapping. The basic clique inequality is the following expression. Then it can be rewrite using indicator variables* $\gamma_{[a,b]}$*.*

$$\begin{aligned} & x_{AB} + x_{AC} + x_{BC} \leq 1 \\ \Longrightarrow\ & \gamma_{[1,1]} + \gamma_{[1,2]} + \gamma_{[1,3]} \leq 1 \end{aligned}$$



One can see on figure 9.2 for $t = 2$ *that the five intervals* $[1,1]$*,* $[1,2]$*,* $[1,3]$*,* $[2,2]$ *and* $[2,3]$ *are overlapping. The basic clique inequality is the following expression. Then it can be rewrite using indicator variables* $\gamma_{[a,b]}$*.*

$$\begin{aligned} & x_{AB} + x_{AC} + x_{AD} + x_{AE} + \\ & x_{BX} + x_{BD} + x_{BE} + \\ & x_{CD} + x_{CE} + \\ & x_{DE} \end{aligned} \qquad \leq 1$$

$$\Longrightarrow$$

$$\gamma_{[1,1]} + \gamma_{[1,2]} + \gamma_{[1,3]} + \gamma_{[2,2]} + \gamma_{[2,3]} \leq 1$$

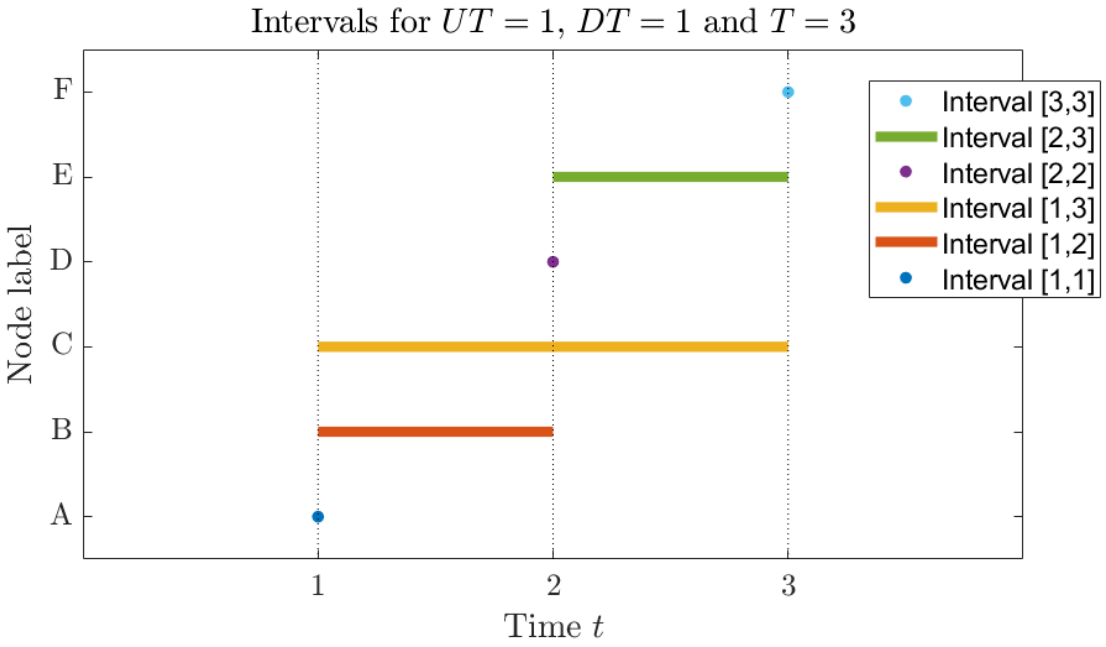
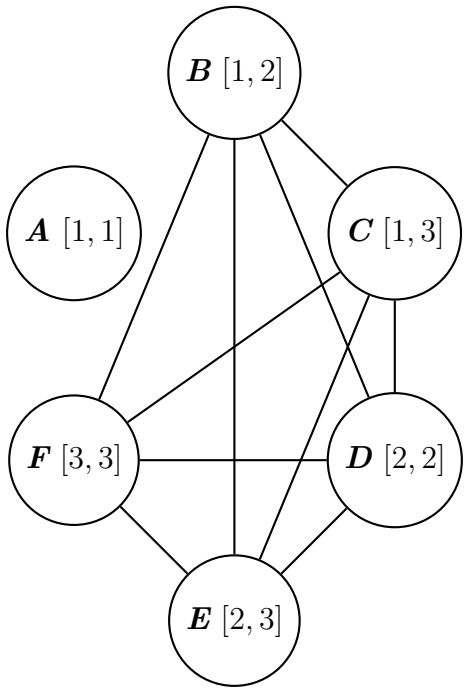


Figure 9.2: Illustrative representation of intervals for example 9.2



One can see on figure 9.2 for $t = 3$ *five intervals* $[1,2]$, $[1,3]$, $[2,2]$, $[2,3]$ *and* $[3,3]$ *are overlapping. The basic clique inequality is the following expression. Then it can be rewrite using indicator variables* $\gamma_{[a,b]}$.

$$\begin{array}{l} x_{BC} + x_{BD} + x_{BE} + x_{BF} + \\ x_{CD} + x_{CE} + x_{CF} + \\ x_{DE} + x_{DF} + \\ x_{EF} \end{array} \leq 1$$

$$\Longrightarrow$$

$$\gamma_{[1,2]} + \gamma_{[1,3]} + \gamma_{[2,2]} + \gamma_{[2,3]} + \gamma_{[3,3]} \leq 1$$

9.3 Proofs Propositions

9.3.1 Proof Proposition 1.1

Proof. Let (p^*, u^*, v^*, w^*) be the optimal solution of UC. Objective value of UC can be rewrite using optimal solution. Dualizing this constraint with price $\lambda \geq 0$ does not penalise the objective value since balance constraint is tight $\left(\sum_{g\in\mathcal{G}} p_g^{t*}\right) - l_t^* = 0$. Dual Lagrangian is smaller than the rewrite objective value because (p^*, u^*, v^*, w^*) is in the feasibility region of $L(\lambda)$.

$$\begin{aligned} UC &= \sum_t \left[\sum_{g\in\mathcal{G}} c^{g*}\right] - VOLL\left(\sum_t l_t^*\right) \\ &= \sum_t \left[\sum_{g\in\mathcal{G}} c^{g*}\right] - VOLL\left(\sum_t l_t^*\right) - \sum_t \lambda_t \left[\left(\sum_{g\in\mathcal{G}} p_g^{t*}\right) - l_t^*\right] \geq L(\lambda) \end{aligned}$$

□

9.3.2 Proof Proposition 2.1

Proof. The proof comes from source [26].
Consider general integer minimization progam 9.1 and general dual Lagrangian relaxation dualizing constraint $Dx \geq d$ with price λ 9.2. Constraint $Dx \geq d$ is considered as complicated and constraint $Bx \geq b$ is considered as easy.

$$z = \mathbf{min}_x cx \tag{9.1a}$$
$$\text{s.t. } Dx \geq d \tag{9.1b}$$
$$Bx \geq b \tag{9.1c}$$
$$x \in \mathbb{Z}_+^n \tag{9.1d}$$

$$z(\lambda) = \mathbf{min}_x cx + \lambda^T (d - Dx) \tag{9.2a}$$
$$\text{s.t. } Bx \geq b \tag{9.2b}$$
$$x \in \mathbb{Z}_+^n \tag{9.2c}$$

The considered problem is to find price λ such that lower bound $z(\lambda)$ is the closest possible to z. Define z_{LD} as the solution the maximization problem.

$$z_{LD} = \mathbf{max}_{\lambda\geq 0} z(\lambda) = \mathbf{max}_{\lambda\geq 0} \left\{ \begin{array}{c} \mathbf{min}_x cx + \lambda^T (d - Dx) \\ \text{s.t. } Bx \geq b \\ x \in \mathbb{Z}_+^n \end{array} \right\} \tag{9.3}$$

Assuming constraints set $Z := \left\{x \in \mathbb{Z}_+^n : Bx \geq b\right\}$ is non-empty and bounded, problem 9.3 is reformulated as a linear program. The dual Lagrangian sub-problem achieves its optimum at an extreme point x^t of conv(Z). $\{x^t\}_{t=1,...,T}$ is the set of extreme points of conv(Z) or alternatively $\{x^t\}_{t=1,...,T}$ is the set of all points of Z.

$$z_{LD} = \mathbf{max}_{\lambda \geq 0}\mathbf{min}_{t=1,...,T}\left\{cx^t + \lambda^T\left(d - Dx^t\right)\right\} \tag{9.4}$$

$$= \mathbf{max}_{\lambda \geq 0}\mathbf{min}_{t=1,...,T}\left\{\lambda^T d + \left(c - \lambda^T D\right)x^t\right\} \tag{9.5}$$

To reformulate the problem introduce variable σ representing a lower bound on values $\left(c - \lambda^T D\right)x^t$.

$$z_{LD} = \mathbf{max}_{\lambda \geq 0}\left\{\begin{array}{cc} \mathbf{min}_{t=1,...,T} & \lambda^T d + \sigma \\ \text{s.t.} & \sigma \leq \left(c - \lambda^T D\right)x^t \forall t = 1, ..., T \\ & \sigma \in \mathbb{R} \end{array}\right\} \tag{9.6}$$

Since σ is a lower bound inequality $\sigma \leq \left(c - \lambda^T D\right)x^t$ must hold $\forall t$. The dual Lagrangian sub-problem does not depend on t anymore and the "max-min" problem of the dual Lagrangian can be rewrite by expressions 9.7.

$$z_{LD} = \mathbf{max}_{\lambda,\sigma}\lambda^T d + \sigma \tag{9.7a}$$

$$\text{s.t. } (\mu_t)\lambda^T Dx^t + \sigma \leq cx^t \forall t = 1, ..., T \tag{9.7b}$$

$$\lambda \geq 0 \tag{9.7c}$$

$$\sigma \in \mathbb{R} \tag{9.7d}$$

In its matrix form is given by expressions 9.8.

$$z_{LD} = \mathbf{max}_{\lambda,\sigma}\begin{pmatrix} d^T & 1 \end{pmatrix}\begin{pmatrix} \lambda^T \\ \sigma \end{pmatrix} \tag{9.8a}$$

$$\text{s.t. } \begin{pmatrix} D^T x^1 & 1 \\ D^T x^2 & 1 \\ D^T x^3 & 1 \\ \vdots & \vdots \\ D^T x^T & 1 \end{pmatrix}\begin{pmatrix} \lambda \\ \sigma \end{pmatrix} \leq \begin{pmatrix} cx^1 \\ cx^2 \\ cx^2 \\ \vdots \\ cx^T \end{pmatrix} \qquad \lambda \geq 0 \tag{9.8b}$$

$$\sigma \in \mathbb{R} \tag{9.8c}$$

The linear programming dual gives 9.9.

$$z_{LD} = \mathbf{min}_{\mu_t} \sum_{t=1}^{T} \left(cx^t\right) \mu_t \tag{9.9a}$$

$$\text{s.t. } D\left(\sum_{t=1}^{T} x^t \mu_t\right) \geq d \tag{9.9b}$$

$$\sum_{t}^{T} \mu_t = 1 \tag{9.9c}$$

$$\mu_t \geq 0 \ \forall t \tag{9.9d}$$

By definition of the set of point $\{x^t\}_{t=1}^{T}$ convex hull is
$\text{conv}(Z) = \left\{x = \sum_{t=1}^{T} x^t \mu_t : \sum_{t=1}^{T} \mu_t = 1, \mu_t \geq 0 \ \forall t\right\}$. Hence the value of the maximum dual Lagrangian is equal to the value of the linear minimization of cx over the intersection of the complicate constraint $Dx \geq d$ with the convex hull of set Z.

$$\begin{aligned} z_{LD} = \mathbf{min}_x cx \\ \text{s.t. } Dx \geq d \\ x \in \text{conv}(Z) \end{aligned}$$

In the considered case the complicate constraint $Dx \geq d$ is balance constraint $\left(\sum_{g\in\mathcal{G}} p_t^g\right) - l_t = 0$. The set Z is made of feasible constraints of all generators Π_{3-bin}^g and the set $[0, L_t]$ bounding demand l_t.

The linear objective function cx is in the objective coming from UC 1.2. It is given by $\sum_t \left[\sum_{g\in\mathcal{G}} c^g\left(p_g, u_g, v_g, w_g\right)\right] - VOLL \sum_t l_t$.

□

9.3.3 Proof Proposition 2.2

Proof. The set $[0, L_t]$ is simply a line. All the convex combinations using points $x^t \in [0, L_t]$ are also on the line. It is already a convex set.
Therefore $\text{conv}([0, L_t]) = [0, L_t]$. □

9.3.4 Proof Proposition 2.3

Proof. The proof comes from source [6].
Proof is made of two points.

1. Since $S \subseteq \text{conv}(S)$ and it is a minimization.
   Then $\min\{cx : x \in S\} \geq \min\{cx : x \in \text{conv}(S)\}$.

2. Let $z^* = \min\{cx : x \in S\}$. Assume $-\infty < z^*$ otherwise it is trivially satisfied. Let $H := \{x \in \mathbb{R}^n : z^* \leq cx\}$. Note that H is convex and by definition is contained in every convex set containing S. Therefore $\text{conv}(S) \subseteq H$ and thus $\min\{cx : x \in \text{conv}(S)\} \geq z^* = \min\{cx : x \in S\}$.

Combining point 1 and 2 gives $\min\{cx : x \in S\} = \min\{cx : x \in \text{conv}(S)\}$.

□

9.3.5 Proof Proposition 2.4

Proof. Proof of proposition 2.4 is given by source [21]. □

9.3.6 Proof Proposition 2.5

Proof. The proof comes from source [6].
From definition 2.1 of the convex hull $\text{conv}(S)$ is the smallest convex set containing S. It must contain all convex combinations of points in S. Hence convex hull can be written such that

$$\text{conv}(S) = \{x \in \mathbb{R}^n : x \text{is convex combination of points in } S\}$$

Assume S is finite and all points in S are $\{x^t\}_{t=1}^T$, then

$$\text{conv}(S) = \left\{x = \sum_{t=1}^{T} x^t \mu_t : \sum_{t=1}^{T} \mu_t = 1, \mu_t \geq 0 \ \forall t\right\}$$

For the "only if" direction, assume $\inf\{cx : x \in S\} = c\bar{x}$ for some $\bar{x} \in S$. Then clearly $\bar{x} \in \text{conv}(S)$ and it follows from proposition 2.3 that
$\inf\{cx : x \in \text{conv}(S)\} = c\bar{x}$.
For the "if" direction, assume there exists $\bar{x} \in \text{conv}(S)$ such that $\inf\{cx : x \in \text{conv}(S)\} = c\bar{x}$. By definition 2.1 of convex hull $\bar{x} = \sum_{t=1}^{T} \mu_t x^t$ for some finite number of points $x^1, ..., x^T \in S$ and $\mu_1, ..., \mu_T > 0$ such that $\sum_{t=1}^{T} = 1$. By definition of $\bar{x}$, $c\bar{x} \geq cx^t$ $\forall t$

$$c\bar{x} = \sum_{t=1}^{T} \mu_t \left(cx^t\right) \leq c\bar{x} \sum_{t=1}^{T} \mu_t = c\bar{x}$$

Since $\mu_t > 0$ it follows that $c\bar{x} = cx^t$ $\forall t$. Thus the infimum over S is achieved by $x^1, ..., x^T$.

□

9.3.7 Proof Proposition 3.1

Proof. Graph $G = (V, E)$, where the set of vertices $V = \mathcal{T}$ and where an edge connects two vertices $([a, b])$ and $([c, d])$ if their time intervals $[a, b]$ and $[c, d]$ overlap each other, is a graph intervals. From [12] intervals graph is a perfect graph.
From Theorem 3.1, [5], cliques inequalities in Γ 3.8 defines the convex hull of a stable set in a perfect graph. □

Definition 9.1 (Intervals Graph, [12])**.** *A graph is an interval graph if each vertex can be represented by an interval on the real line in such a way that two vertices are adjacent if and only if their corresponding intervals intersect. These graphs are perfect graphs.*

Definition 9.2 (Perfect Graph, [12])**.** *A graph is called perfect graph if the chromatic number and the clique number have the same value for each of its induced subgraphs.*

Definition 9.3 (Stabel Set, [5])**.** *A stable set in graph G is a set of vertices, no two of which are adjacent.*

9.3.8 Proof Proposition 3.2 : Constrained Minkowski Sums of Polyhedra

Proof. Proof comes from [16]. All definitions and theorem used are given just after the proof.

The goal of this proof is to arrive at a representation of constrained Minkowski sums of polyhedra using indicator variables. In this proof scalar multiples is defined by definition 9.4 and Minkowski sum is defined by definition 9.5.
The current proof generalizes theorems 9.1 and 9.2 to allow for different combinations of polyhedra. Assume polyhedron $\Gamma \subseteq \mathbb{R}^n$ defining a restriction on indicator variables γ to define the set $\cup \{\sum_{i=1}^{m} \gamma_i P^i \ \gamma \in \Gamma\}$. The idea is to derive similar results in the spirit of the two preceding theorems but considering indicator variables γ and restriction Γ.

Consider the set $S := \cup_{\gamma \in \Gamma} (\sum_{i=1}^{m} \gamma_i P^i)$ where P^i, $i \in [m]$, are non-empty polyhedra in $\mathbb{R}^n$ and $\Gamma \subseteq \mathbb{R}^m_+$ is a non-empty and non-negative polyhedron. The goal is to arrive at a polyhedral representation for S.
From proposition 9.3
$P = \text{proj}_x (Y) := \{x \in \mathbb{R}^n | \exists (x^1, ..., x^m, \gamma) \in \mathbb{R}^{nm+m} \text{ s.t. } (x, x^1, ..., x^m, \gamma) \in Y\}$ and P is a polyhedron, where $Y$is such that

$$Y := \left\{ A^i x^i \leq \gamma_i b^i \ i \in [m]; \sum_{i=1}^{m} x^i = x; (\gamma_1, ..., \gamma_m) = \gamma \in \Gamma \right\} \tag{9.11}$$

Hence the restriction Γ of indicator variables γ provides a polyhedron.

Remark 9.1. *For all $\Gamma \subseteq \mathbb{R}^m_+$, Y provides a polynomial-size (in dim(P^i) and dim(Γ)) polyhedral representation of P. Further, if for all $i \in [m]$, P^i is bounded (i.e. $R^i = \{0\}$), then $P = S$ and Y provides a compact formulation for S.*

Remark 9.2. *If $\Gamma \subseteq \mathbb{R}^m_{++}$ (then open, strictly positive orthant), then $\gamma_i P^i = \gamma Q^i + P^i \ \forall(\gamma_1, ..., \gamma_m) \in \Gamma$. Therefore $P = S$ and so Y provides a compact formulation for S.*

Theorem 9.4 demonstrates that $\mathrm{cl}\,(S) = P$ with a restriction on Γ. This restriction is required to have $\hat{\gamma} \in \Gamma$ such that $\hat{\gamma}_i > 0 \ \forall i \in [m]$. The requirement should not been seen as overly restrictive. If for some i there is $\gamma_i = 0 \ \forall \gamma \in \Gamma$ then this particular P^i should be discarded since it never contributes to the sum.

Remark 9.3. *If there exists $\hat{\gamma} \in \Gamma$ such that $\hat{\gamma} > 0$ and $P^1, ...P^m$ are all non-empty, then theorems 9.3 and 9.3 together imply that $cl(S) = proj_x(Y)$.*

From theorem 9.5, for $P^1, ..., P^m$ non-empty polyhedra and identical recession cones, and considering $0 \notin \Gamma$, then the set defined previously $S = \cup_{\gamma \in \Gamma} \left(\sum_{i=1}^m \gamma_i P^i\right)$ is a polyhedron and is such that $S = \mathrm{proj}_x\,(Y)$. It means Y 9.11 is an extended formulation in the sens of definition 3.3 for set S.

Remark 9.4. *To see the necessity of $0 \notin \Gamma$ in theorem 9.5, consider the sets S and P when $\gamma = 0$. If $P^1, ..., P^m$ have the same recession cone R, then $P|_{\gamma=0} = \sum_{i=1}^m 0Q^i + \sum_{i=1}^m R^i = R$, whereas $S|_{gamma=0} = \sum_{i=1}^m 0P^i = \{0\}$, and $R = \{0\}$ if and only if all the P^i's are bounded. Hence it can be done without the assumption $0 \notin \Gamma$ in theorem 9.5 if all P^i's are bounded.*

It may be that Γ is the continuous relaxation of some integers set which determines the polyhedra P^i simultaneously allowed in the sum. The next theorem shows that vertices and extreme rays of Y have γ components which are vertices and extreme rays of Γ, hence if Γ is a perfect formulation for some integer set, vertices of Y will have integer γ. Further even if Γ is not a perfect formulation, this shows that to find solution with integer γ one needs only to consider cuts on Γ and not on the entire polyhedron Y. For ease of notation, for $y \in Y$ define y_Γ to be components of y in Γ.

Theorem 9.6 demonstrates that if Γ is a perfect formulation of some integer set and the variables x^i are continuous, then Y (under the given assumptions) provides a perfect formulation for $S|_{\mathbb{Z}_+} = \cup_{\gamma \in \Gamma \cap \mathbb{Z}_+} \{\sum_{i=1}^m \gamma_i P^i\}$.
Note the polyhedron D defined by proposition 3.2 is exactly of this form. Vertices of polytope D must have integer γ.

□

Definition 9.4. *Assume a set $C \subseteq \mathbb{R}^n$ then scalar multiplication in previous proof is defined such that*

$$\mu C := \{\mu x | x \in C\}$$

Definition 9.5. *Assume two sets $C_1 \subseteq \mathbb{R}^n$ and $C_2 \subseteq \mathbb{R}^n$, Minkowski sum is defined such that*

$$C_1 + C_2 := \{x_1 + x_2 | x_1 \in C_1, x_2 \in C_2\}$$

Definition 9.6. *The conic hull of a non-empty set $S \subseteq \mathbb{R}^n$ is*

$$cone\,(S) := \left\{ x \in \mathbb{R}^n : x = \sum_{i=1}^{k} \mu_i x^i where k \geq 1, \mu \in \mathbb{R}^k_+ and x^1, ..., x^k \in S \right\}$$

If S is a finite set, $cone\,(S)$ is said to be finitely generated, [16].

Theorem 9.1. *Consider m polyhedra $P\hat{\imath} = \{x in \mathbb{R}^n | A^i x \leq b\hat{\imath}\}$ and their polyhedral recession cones $R^i = \{x \in \mathbb{R}^n | A^i \leq 0\}$ and let Q^i be a (bounded) polytope such that $P^i = Q^i + R^i$. The notation $[m]$ refers to the set $\{1, ..., m\}$.*
Define the set $S = conv\left(\cup_{i \in [m]} P^i\right)$ and polyhedron $P = conv\left(\cup_{i \in [m]} Q^i\right) + cone\left(\cup_{i \in [m]} R^i\right)$.
Then the polyhedron

$$Y := \left\{ \begin{array}{cc} A^i x^i \leq \gamma_i b^i & i \in [m] \\ \sum_{i \in [m]} x^i = x & \\ \sum_{i \in [m]} \gamma_i = 1 & \\ \gamma_i \geq 0 & i \in [m] \end{array} \right\} \tag{9.12}$$

provides an extended formulation of P. If each P^i $i \in [m]$ is nonempty then $cl\,(S) = P$. Additionally, the vertices of Y have binary γ_i.

Proof. Proof of theorem 9.1 is given by sources [16], [2]. □

Theorem 9.2. *If $P^1, ..., P^m$ are all nonempty and have identical recession cones then $S = P$ and so Y provides a polyhedral extended formulation for S.*

Proof. Proof of theorem 9.2 is given by source [13]. □

Theorem 9.3. *Consider m non-empty polyhedra $P^i = \{x \in \mathbb{R}^n | A^i x \leq b^i\}$, $i \in [m]$, and for each $i \in [m]$ let Q^i be a (bounded) polytope in $\mathbb{R}^n$ and R^i be a (closed convex) cone in $\mathbb{R}^n$ such that $P^i = Q^i + R^i$. Let $\Gamma \subseteq \mathbb{R}^m_+$ be a nonempty polyhedron. Consider the set $P := \cup_{\gamma \in \Gamma} \left(\sum_{i=1}^m \gamma_i Q^i + \sum_{i=1}^m R^i\right)$ and consider the polyhedron $Y \subseteq \mathbb{R}^{n+nm+m}$ defined by*

$$Y := \left\{ \begin{array}{cc} A^i x^i \leq \gamma_i b^i & i \in [m] \\ \sum_{i=1}^m x^i = x & \\ (\gamma_1, ..., \gamma_m) = \gamma \in \Gamma & \end{array} \right\} \tag{9.13}$$

Then $P = proj_x(Y) := \{x \in \mathbb{R}^n | \exists (x^1, ..., x^m, \gamma) \in \mathbb{R}^{nm+m} s.t. (x, x^1, ..., x^m, \gamma) \in Y\}$. In particular, P is a polyhedron.

Proof. Proof from source [16].
Let $x \in P$. P is non-empty as the union of the sum is non-empty sets. There exists points $q^i \in Q^i$, $r^i \in R^i$ and $\gamma \in \Gamma$ such that $x = \sum_{i=1}^m \gamma_i q^i + \sum_{i=1}^m r^i$. Define $x^i = \gamma_i + r^i$ for $i \in [m]$.
Now by construction $x = \sum_{i=1}^m x^i$ and $A^i x^i = A^i(\gamma_i q^i + r^i) = \gamma_i A^i q^i + A^i r^i \leq \gamma_i b^i + 0$ since $\gamma_i A^i q^i \leq \gamma_i b^i$ and $A^i r^i \leq 0$. Hence $(x, x^1, ..., x^m, \gamma) \in Y$, so $P \subseteq \text{proj}_x(Y)$.
Conversely, let $(x, x^1, ..., x^m, \gamma) \in Y$. Consider $I^+ := \{i | \gamma_i > 0\}$ and $I^0 := \{i | \gamma_i = 0\}$. For $i \in I^+$, $A^i x^i \leq \gamma_i b^i$ and so $x^i \in Q^i + R^i$. For $i \in I^0$, $A^i x^i \leq 0$ and so $x^i \in R^i = \gamma_i Q^i + R^i$. Since $x = \sum_{i=1}^m x^i \in \sum_{i=1}^m (\gamma_i Q^i + R^i)$ and $\gamma \in \Gamma$, this shows $x \in P$, and hence $\text{proj}_x(Y) \subseteq P$.
As the projection of a polyhedron, P is itself a polyhedron. □

Theorem 9.4. *Let $\Gamma \subseteq \mathbb{R}^m_+$ and $P^1, ..., P^m \subseteq \mathbb{R}^n$ be non-empty polyhedra. Suppose there exists $\hat{\gamma} \in \Gamma$ such that $\hat{\gamma}_i > 0$ $\forall i \in [m]$. Then for P and S defined as above, $cl(S) = P$.*

Proof. Proof from source [16].
First consider $\text{cl}(S) \subseteq P$. Since P as a polyhedron is closed, it suffices to show $S \subseteq P$. Hence let $x \in S$. Then $\exists \gamma \in \Gamma$, $p^i \in P^i$ for $i \in [m]$ such that $x = \sum_{i=1}^m \gamma_i p^i$. As above for each $i \in [m]$, consider $P^i = Q^i + R^i$, so for each $i \in [m]$ we have $p^i = q^i + r^i$ for $q^i \in Q^i$ and $r^i \in R^i$. Thus $x = \sum_{i=1}^m \gamma_i q^i + \sum_{i=1}^m r^i$, and since $\gamma_i q^i \in \gamma_i Q^i$ and $\gamma_i r^i \in R^i$ (as R^i is a closed convex cone, $\gamma_i \geq 0$), we have $x \in P$.
Conversely, let $x \in P$. Then there exists $\gamma \in \Gamma$, $q^i \in Q^i$ and $r^i \in R^i$ such that $x = \sum_{i=1}^m \gamma_i q^i + \sum_{i=1}^m r^i$. By assumption $\exists \hat{\gamma} \in \Gamma$ that is strictly positive. By convexity, $(1-\epsilon)\gamma + \epsilon\hat{\gamma} \in \Gamma$ $\forall \epsilon \in (0,1)$; further $(1-\epsilon)\gamma + \epsilon\hat{\gamma} > 0$ $\forall \epsilon \in (0,1)$. Define $x^\epsilon := \sum_{i=1}^m [(1-\epsilon)\gamma_i + \epsilon\hat{\gamma}_i] q^i + \sum_{i=1}^m r^i$. Clearly $\lim_{\epsilon \longrightarrow 0^+} x^\epsilon = x$ and we see that $x^\epsilon := \sum_{i=1}^m [(1-\epsilon)\gamma_i + \epsilon\hat{\gamma}_i] (q^i + r^i) \left(q^i + \frac{r^i}{[(1-\epsilon)\gamma_i + \epsilon\hat{\gamma}_i]}\right) \in P^i$ for $i \in [m]$, $\epsilon \in (0,1)$ and $1-\epsilon)\gamma + \epsilon\hat{\gamma} \in \Gamma$ $\forall \epsilon \in (0,1)$ we have that $x^\epsilon \in S$ $\forall \epsilon \in (0,1)$. Hence $x \in \text{cl}(S)$.
□

Theorem 9.5. *Suppose $P^1, ..., P^m$ are non-empty polyhedra with identical recession cones and $\Gamma \subset \mathbb{R}^m_+$ is a polyhedron such that $0 \notin \Gamma$. Then $S = \cup_{\gamma \in \Gamma} \left(\sum_{i=1}^m \gamma_i P^i\right)$ is a polyhedron and $S = proj_x(Y)$.*

Proof. Proof from source [16].
Let $x \in P$. Then there exists $q^i \in Q^i$, $r^i \in R^i$ and $\gamma \in \Gamma$ such that $x = \sum_{i=1}^m \gamma_i q^i + \sum_{i=1}^m r^i$. By assumption there exists $j \in [m]$ such that $\gamma_j > 0$. As the P^i's have identical recession cones, we have $\sum_{i=1}^m r^i \in \gamma_j P^j$. Define $p^j = q^j + \sum_{i=1}^m \frac{r^i}{\gamma_j}$ and $p^i = q^i$ for $i \neq j$ and it follows that $x = \sum_{i=1}^m \gamma_i p^i$. Hence $x \in S$. The results then follows from theorem 9.3. □

Theorem 9.6. *$Y = conv(V) + cone(R)$, for finite set V and R, where for each vertex $v \in V$, v_Γ is a vertex of Γ and for each extreme ray $r \in R$, r_Γ is an extreme ray of Γ. That is $proj_\gamma = \Gamma$.*

Proof. Proof from source [16].
Let $y \in Y$ such that $y = (x, x_1, ..., x_m, \gamma_1, ..., \gamma_m)$ and define $\gamma := y_\Gamma$. Since Γ is a polyhedron, by the Minkowski-Weyl theorem 9.7 (compact form is given by theorem 9.8) there exist vectors $v^1, ..., v^p, r^1, ..., r^q \in \mathbb{R}^m$ and $\lambda \in \mathbb{R}^p_+$, $\mu \in \mathbb{R}^q_+$ such that $\gamma = \sum_{k=1}^p \lambda_k v^k + \sum_{l=1}^q \mu_l r^l$ and $\sum_{k=1}^p \lambda_k = 1$. In particular

$$\gamma_i = \sum_{k=1}^p \lambda_k v_i^k + \sum_{l=1}^q \mu_l r_i^l \text{ , with } \sum_{k=1}^p \lambda_k = 1 \tag{9.14}$$

Let $I_+ = \{i | \gamma_i > 0\}$ and $I_0 = \{i | \gamma_i = 0\}$. Define

$$x_i^k := \begin{cases} x_i \frac{v_i^k}{\gamma_i} & \text{if } i \in I_+ \\ x_i & \text{if } i \in I_0 \text{ and } \lambda_k = 0 \\ \hat{x}_i^k \in v_i^k P^i & \text{if } i \in I_0 \text{ and } \lambda_k = 0 \end{cases} \tag{9.15}$$

$$x_i^l := \begin{cases} x_i \frac{r_i^l}{\gamma_i} & \text{if } i \in I_+ \\ 0 & \text{if } i \in I_0 \text{ and } \mu_l > 0 \\ \hat{x}_i^l \in r_i^l P^i & \text{if } i \in I_0 \text{ and } \mu_l = 0 \end{cases} \tag{9.16}$$

and $x^k = sum_{i=1}^m x_i^k$ for $k \in [p]$ and $x^l = \sum_{i=1}^m x_i^l$ for $l \in [q]$. For $k \in [p]$ define $y^k := \left(x^k, x_1^k, ..., x_m^k, v_1^k, ..., v_m^k\right)$ and for $l \in [q]$ define $y^l := (x^l, x_1^l, ..., x_m^l, r_1^l, ..., r_m^l)$.

Firstly check the feasibility of the points constructed above. So for each $k \in [p]$, consider y^k. By construction $y_\Gamma^k \in \Gamma$ and $x^k = \sum_{i=1}^m x_i^k$, so for feasibility it needs to verify $A^i x_i^k \leq v_i^k b^i$.
Suppose $i \in I_+$, then $A^i x_i \leq \gamma_i b^i$, and multiplying both sides by v_i^k and dividing by γ_i shows x_i^k is feasible.
Now suppose $i \in I_0$ and so $A^i x_i \leq 0$. If $\lambda_k > 0$, then we must have $v_i^k = 0$, so x_i^k is feasible. If $\lambda_k = 0$, x_i^k is feasible by construction (since each P^i is non-empty it is always possible to find a point $\hat{x}_i^k$). The feasibility of y^l for each $l \in [q]$ is similar.

To complete the proof $y = \sum_{k=1}^{p} \lambda_k y^k + \sum_{l=1}^{q} \mu_l y^l$ must be shown.
First suppose $i \in I_+$, then

$$\sum_{k=1}^{p} \lambda_k x_i^k + \sum_{l=1}^{q} \mu_l x_i^l = \sum_{k=1}^{p} \lambda_k x_i^k \frac{v_i^k}{\gamma_i} + \sum_{l=1}^{q} \mu_l x_i^l \frac{r_i^l}{\gamma_i} = \frac{x_i}{\gamma_i} \left(\sum_{k=1}^{p} \lambda_k v_i^k + \sum_{l=1}^{q} \mu_l r_i^l \right) = x_i$$

Conversely, suppose $i \in I_0$ then

$$\begin{aligned} \sum_{k=1}^{p} \lambda_k x_i^k + \sum_{l=1}^{q} \mu_l x_i^l &= \sum_{k:\gamma_k>0} \lambda_k x_i + \sum_{k:\gamma_k=0} \mu_l 0 + \sum_{l:\mu_l>0} \mu_l 0 + \sum_{l:\mu_l=0} \mu_l \hat{x}_i^l \\ &= \sum_{k:\gamma_k>0} \lambda_k x_i + 0 + 0 + 0 = x_i \sum_{k:\gamma_k>0} \lambda_k = x_i \end{aligned}$$

It then follows

$$\sum_{k=1}^{p} \lambda_k x^k + \sum_{l=1}^{q} \mu_l x^l = \sum_{k=1}^{p} \lambda_k \sum_{i=1}^{m} x_i^k + \sum_{l=1}^{q} \mu_l \sum_{i=1}^{m} x_i^l = \sum_{i=1}^{m} \left(\sum_{k=1}^{p} \lambda_k x_i^k + \sum_{l=1}^{q} \mu_l x_i^l \right) = \sum_{i=1}^{m} x_i = x$$

Hence it has been shown that $y = \sum_{k=1}^{p} \lambda_k y^k + \sum_{l=1}^{q} \mu_l y^l$ with $\lambda, \mu \geq 0$ and $\sum_{k=1}^{p} \lambda_k = 1$, proving theorem 9.6.

□

Theorem 9.7 (Minkowski-Weyl Theorem - full version)**.** *Let $X \subseteq \mathbb{R}^d$. Then the following are equivalent.*

1. *($\mathcal{H}$-description) There exists $m \in \mathbb{N}$, a matrix $A \in \mathbb{R}^{m \times d}$ and a vector $b \in \mathbb{R}^m$ such that $X = \left\{ x \in \mathbb{R}^d : Ax \leq b \right\}$.*
2. *($\mathcal{V}$-description) There exist finite sets $V, R \subseteq \mathbb{R}^d$ such that $X = conv(V) + cone(R)$.*

Proof. Proof of theorem 9.7 is given by source [4]. □

Theorem 9.8 (Minkowski-Weyl Theorem - compact version)**.** *Let $X \subseteq \mathbb{R}^d$. Then X is a bounded polyhedron if and only if X is the convex hull of a finite set of points.*

Proof. Proof of theorem 9.8 is given by source [4]. □

9.3.9 Proof Proposition 3.3

Proof. From definition 1.4 EF is a relaxation of UC.

1. The feasible set of the EF is at least as big as feasible set of UC. Indeed feasible set of EF is the convex hull of feasible set of UC.
2. Objective function of EF is equal than the one of UC.

□

9.3.10 Proof Proposition 4.1

Proof. Continuous Linear Program 2.5 considers $5 \cdot |\mathcal{G}| \cdot T = \mathcal{O}(T)$ variables. It considers $2 \cdot T + \sum_{g \in \mathcal{G}} (4 \cdot T + (T - UT^g) + (T - DT^g)) = \mathcal{O}(T)$ constraints. □

9.3.11 Proof Proposition 5.1

Proof. Objective function from SPP and UC are equivalent. Constraints 5.5b defines the balance constraint as in UC. Three-bin feasible region Π^g_{3-bin} is not explicitly represented in the formulation of SPP but it is implicitly since feasible schedule $(\hat{p}, \hat{\bar{p}}, \hat{u}, \hat{v}, \hat{w}) \in \Pi_{3-bin}$ and since in SPP all feasible schedule are considered.

□

9.3.12 Proof Proposition 5.2

Proof. Apply theorem 9.9 from [7]
λ_t and π_t are respectively dual multipliers of the two following constraints.

$$\begin{array}{ll} (\lambda_t) & \sum_{g \in \mathcal{G}} \sum_{n_g \in \mathcal{N}_g} \hat{p}_g^{n_g} = l_t \quad \forall t \in [T] \\ (\mu_g) & \sum_{n_g \in \mathcal{N}_g} z_g^{n_g} = 1 \qquad \forall g \in \mathcal{G} \end{array}$$

Selection of the schedule to join the basis already in RMP is chosen for each generator g such that it improves the current optimal solution of RMP.

$$c_g(p_g, \bar{p}_g, u_g, v_g, w_g) - \sum_{t \in [T]} \left(\lambda_t p_g^t\right) - \mu_g = rc_g(p_g, \bar{p}_g, u_g, v_g, w_g) < 0$$

The expression corresponds well to reduced cost defined by 5.7. □

Theorem 9.9. *Only a finite number of iterations of the simple algorithm is required if each basic feasible solution is improved by introducing into the basis either an extreme point $P_{j^*} \in C_{j^*}$ chose so that*

$$\pi P_{j^*} = min_{P_j \in C_j} \pi P < 0, j = 1, 2, ..., n \tag{9.17}$$

where π are the simplex multipliers of the basis or any homogeneous solution $\bar{P}_{j^*}$ from the finite set such that $\pi P_{j^*} < 0$.

Proof. Proof of theorem 9.9 is given by source [7]. □

9.3.13 Proof Proposition 6.1

Proof. Proof from source [24].
The first inequality results from the fact that $\bar{R}_{LP}$ only includes a subset of variables of R whose LP value is $v^R_{LP} = v^*$, while the missing constraints are satisfied by the current restricted reformulation $\bar{R}_{LP}$, which we denote by $\tilde{z}$. More explicitely, all the missing constraints must be satisfied by the solution that consists in extending $\tilde{z}$ with zero components, i.e. $(,0)$ defines a solution R_{LP}; otherwise, such missing constraints is not satisfied by solution of $\bar{S}$ in constradiction to the fact solutions of $\bar{S}$ satisfy all constraints of Q.

The second inequality is derived from the fact that any solution $\tilde{\pi}$ to the LP relaxation of M has its counterpart $\tilde{z} = \sum_{g \in G} z^g \tilde{\pi}_g$ that defines a valid solution for $\bar{R}_{LP}$, where $z^g \in Z$ is obtained by lifting solution $x^g \in X$. The existence of the associated z^g is guaranteed by assumption 6.1-(ii). □

9.3.14 Proof Proposition 6.2

Proof. From source [24].

(i) For any λ, and in particular for the current dual solution associated with balance constraint, $L(\lambda)$ defines a valid Lagrangian dual bound on F 1.1. As $\zeta = \min\{(c - \lambda D)Tz : z \in Z\} = \min\{(c - \lambda D)x : x \in X\}$, (λ, ζ) defines a feasible solution of D and hence its value, $\lambda d + \zeta$, is a valid dual bound on M_{LP}.

(ii) From point (i) and proposition 6.1, we have $L(\lambda) \leq \beta \leq v^* \leq v^{\bar{R}}_{LP}$. When the stopping condition in Step 3 is satisfied, the inequalities turn into equalities.

(iii) When $v^{\bar{R}}_{LP} > \beta \geq L(\lambda)$, we note that $\sigma h > (c - \lambda D)Tz^*$ because $v^{\bar{R}}_{LP} = \lambda d + \sigma h$ and $L(\lambda) = \lambda d + \zeta = \lambda d + (c - \lambda D)Tz^*$. As $Hz^* \geq h$ and $\sigma \geq 0$, this implies that $[(c - \lambda D)T - \sigma H]z^* < 0$. Assume by contradiction that each component of z^* has non negative reduced cost for the current dual solution of $\bar{R}_{LP}$. Then, the aggregate sum, $[(c - \lambda D)T - \sigma H]z^*$ cannot be strictly negative for $z^* \geq 0$. As (λ, σ) is an optimal dual solution to $\bar{R}_{LP}$, all variables of $\bar{R}_{LP}$ have positive reduced cost. Thus, the negative reduced cost components of z^* must have been absent from $\bar{R}$.

(iv) Because $Hz^* \geq h$, $[(c - \lambda D)T]z^* \geq \sigma Hz^*$ implies $(c - \lambda D)Tz^* \geq \sigma H$, i.e. $\zeta \geq \sigma h$. In turn, $\zeta \geq \sigma h$ implies that (λ, ν) with $\nu = \sigma h$ is feasible for D (all constraints of D are satisfied by (λ, ν)). Note that $\zeta \geq \sigma h$ also implies $v^{\bar{R}}_{LP} \leq \beta$, as $v^{\bar{R}}_{LP} = \lambda d + \sigma h \leq \lambda d + \zeta = L(\lambda) \leq \beta$

□

9.4 Belgian Generators

Generator	Min Run Capacity [MW]	Max Run Capacity [MW]	Ramp Up [MW/min]	Ramp Down [MW/min]	UT [hours]	DT [hours]	No Load Consumption [MW]	Startup Cost [€]	Marginal Cost [€/MWh]
G_1	525.5	1051	35.03333	35.03333	168	168	0	22624.2	7.2276
G_2	10.3	43	2.15	2.15	2	2	41.0889	0	59.7183
G_3	10.3	43	2.15	2.15	2	2	57.3333	0	61.3861
G_4	240.4	437	13.10667	13.10667	8	4	422.1467	11903.56121	44.4739
G_5	5.1	64	0.64	0.64	12	8	54.3515	3871.126193	53.5481
G_6	3.7	46	0.46	0.46	12	8	0	2782.371951	46.117
G_7	16.8	70	3.5	3.5	2	2	0	0	87.585
G_8	15.4	64	3.2	3.2	2	2	70.7657	1103.313312	67.1551
G_9	15.4	64	3.2	3.2	2	2	70.7657	1103.313312	73.2508
G_10	192.5	350	10.5	10.5	8	4	255.566	0	48.0159
G_11	14.5	14.5	0	0	8	8	9.396	0	49.582
G_12	504	1008	33.6	33.6	168	168	0	21844	7.2445
G_13	253	460	13.8	13.8	8	4	0	0	48.0159
G_14	211.8	385	11.54667	11.54667	8	4	281.1226	0	45.6913
G_15	48.7	203	10.15	10.15	2	2	0	0	44.9865
G_16	48.7	203	10.15	10.15	2	2	0	0	50.0534
G_17	48.7	203	10.15	10.15	2	2	0	0	47.4682
G_18	48.7	203	10.15	10.15	2	2	0	0	47.6574
G_19	48.7	203	10.15	10.15	2	2	0	0	44.8836
G_20	48.7	203	10.15	10.15	2	2	0	0	44.2396
G_21	18.7	78	3.9	3.9	2	2	104.8125	0	86.5361
G_22	10.8	45	2.25	2.25	2	2	59.7427	0	18.9648
G_23	481	962	32.06667	32.06667	168	168	1574.1818	0	7.315
G_24	23.5	294	2.94	2.94	12	8	0	16267.29631	64.4349
G_25	7.6	95	0.95	0.95	12	8	142.5	8302.837604	71.0255
G_26	15.6	52	0.52	0.52	3	2	127.6364	0	224.4062
G_27	94	94	0	0	8	8	0.0025	0	44.1246
G_28	225.5	410	12.3	12.3	8	4	398.6316	0	42.2347
G_29	96	140	2.8	2.8	8	8	122.5069	0	48.8841
G_30	96	135	2.6	2.6	8	8	0.0014	0	42.2347
G_31	34	85	1.7	1.7	8	8	65.79	0	51.3525
G_32	42	58	1.06667	1.06667	8	8	31.9464	0	50.1702
G_33	48	48	0	0	8	8	14.8135	4694.354261	68.3508
G_34	48.7	203	10.15	10.15	2	2	0	0	43.4767
G_35	43	43	0	0	8	8	8.9301	4205.359026	46.117
G_36	43	43	0	0	8	8	25.3293	0	47.0009

Table 9.5

Generator	Min Run Capacity [MW]	Max Run Capacity [MW]	Ramp Up [MW/min]	Ramp Down [MW/min]	UT [hours]	DT [hours]	No Load Consumption [MW]	Startup Cost [€]	Marginal Cost [€/MWh]
G_37	43	43	0	0	8	8	7.9423	0	47.4614
G_38	22.8	22.8	0	0	8	8	5.13	0	49.4535
G_39	231	420	12.6	12.6	8	4	0	0	48.8841
G_40	270	375	15	15	8	4	0	0	47.4682
G_41	280	305	1.66667	1.66667	12	8	0	0	20.3643
G_42	42	140	1.4	1.4	3	2	0	0	222.6571
G_43	4.8	20	1	1	2	2	34.3511	0	36.1915
G_44	2.5	10.5	0.525	0.525	2	2	18.9	0	26.9735
G_45	48.7	203	10.15	10.15	2	2	0	0	43.8617
G_46	42	50	0.53333	0.53333	8	8	10.8	0	49.0903
G_47	40	40	0	0	8	8	0.0006	0	47.6022
G_48	266.8	485	14.54667	14.54667	8	4	0	0	45.6913
G_49	192.5	350	10.5	10.5	8	4	0	0	20.3643
G_50	23.5	294	2.94	2.94	12	8	411.1888	0	64.4349
G_51	23.2	290	2.9	2.9	12	8	0	0	69.4517
G_52	17.2	215	4.3	4.3	12	8	237.7286	0	68.5876
G_53	10.4	130	1.3	1.3	12	8	0	0	32.2246
G_54	9.8	122	1.22	1.22	12	8	0	0	34.6503
G_55	4.2	52	0.52	0.52	12	8	0	0	82.6683
G_56	154	154	0	0	8	8	0	0	44.4739
G_57	25	104	5.2	5.2	2	2	114.9943	0	69.0237
G_58	10.3	43	2.15	2.15	2	2	0	0	62.0372
G_59	10.8	36	0.36	0.36	3	2	88.3636	0	217.1073
G_60	264.2	480.3	14.40667	14.40667	8	4	489.3623	0	20.3643
G_61	7.2	30	1.5	1.5	2	2	0	0	7.5107
G_62	522	1044	34.8	34.8	168	168	1708.3636	0	7.658
G_63	503	1006	33.53333	33.53333	168	168	0	22775.9	7.7644
G_64	216.3	432.5	14.41333	14.41333	168	168	0	9372.6	7.9161
G_65	216.3	432.5	14.41333	14.41333	168	168	0	9372.6	7.415
G_66	43	43	0	0	8	8	27.864	0	42.2203
G_67	20.6	258	2.58	2.58	12	8	339.8049	0	21.7826
G_68	20.6	258	2.58	2.58	12	8	243.5268	0	62.5744

Table 9.6

Bibliography

[1] Panagiotis Andrianesis, Michael C Caramanis, and William W Hogan. "Computation of Convex Hull Prices in Electricity Markets with Non-Convexities using Dantzig-Wolfe Decomposition". In: *arXiv preprint arXiv:2012.13331* (2020).

[2] Egon Balas. "Disjunctive programming: Properties of the convex hull of feasible points". In: *Discrete Applied Mathematics* 89.1-3 (1998), pp. 3–44.

[3] Clayton Barrows et al. "The IEEE Reliability Test System: A Proposed 2019 Update". In: *IEEE Transactions on Power Systems* 35.1 (2020), pp. 119–127. DOI: 10.1109/TPWRS.2019.2925557.

[4] Amitabh Basu. *Introduction to Convexity*. https://www.ams.jhu.edu/~abasu9/AMS_550-465/notes-without-frills.pdf. 2019.

[5] Vašek Chvátal. "On certain polytopes associated with graphs". In: *Journal of Combinatorial Theory, Series B* 18.2 (1975), pp. 138–154.

[6] Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, et al. *Integer programming*. Vol. 271. Springer, 2014.

[7] George B Dantzig and Philip Wolfe. "Decomposition principle for linear programs". In: *Operations research* 8.1 (1960), pp. 101–111.

[8] *Evolution prix de l'énergie Belgique et pays voisins | CREG : Commission de Régulation de l'Électricité et du Gaz*. URL: https://www.creg.be/fr/professionnels/fonctionnement-et-monitoring-du-marche/evolution-prix-de-lenergie-belgique-et-pays (visited on 06/05/2021).

[9] Manuel Garcia, Harsha Nagarajan, and Ross Baldick. "Generalized Convex Hull Pricing for the AC Optimal Power Flow Problem". In: *IEEE Transactions on Control of Network Systems* 7.3 (2020), pp. 1500–1510. DOI: 10.1109/TCNS.2020.2982572.

[10] Paul Gribik and S Pope. *Market Clearing Electricity Prices and Energy Uplift*. https://www.researchgate.net/publication/263047745_Market_Clearing_Electricity_Prices_and_Energy_Uplift. Jan. 2007.

[11] Paul R Gribik, William W Hogan, Susan L Pope, et al. "Market-clearing electricity prices and energy uplift". In: *Cambridge, MA* (2007).

[12] Stefan Hougardy. "Classes of perfect graphs". en. In: *Discrete Mathematics* 306.19-20 (Oct. 2006), pp. 2529–2571. ISSN: 0012365X. DOI: 10.1016/j.disc.2006.05.021. URL: https://linkinghub.elsevier.com/retrieve/pii/S0012365X06004055 (visited on 05/29/2021).

[13] Robert G Jeroslow. "Representability in mixed integer programmiing, I: Characterization results". In: *Discrete Applied Mathematics* 17.3 (1987), pp. 223–243.

[14] Michael Jünger et al. *50 Years of integer programming 1958-2008: From the early years to the state-of-the-art*. Springer Science & Business Media, 2009. Chap. Chapter 9. Lagrangian Relaxation for Integer Programming.

[15] Gunnar W. Klau. *Solving the MWT*. http://www.mi.fu-berlin.de/wiki/pub/Main/GunnarKlauP1winter0708/discMath_klau_ILP_II.pdf.

[16] Ben Knueven, Jim Ostrowski, and Jianhui Wang. "The Ramping Polytope and Cut Generation for the Unit Commitment Problem". In: *INFORMS Journal on Computing* 30.4 (2018), pp. 739–749. DOI: 10.1287/ijoc.2017.0802. eprint: https://doi.org/10.1287/ijoc.2017.0802. URL: https://doi.org/10.1287/ijoc.2017.0802.

[17] Bernard Knueven, James Ostrowski, and Jean-Paul Watson. "On mixed-integer programming formulations for the unit commitment problem". In: *INFORMS Journal on Computing* 32.4 (2020), pp. 857–876.

[18] Bernard Knueven et al. "A Computationally Efficient Algorithm for Computing Convex Hull Prices". In: (2019). URL: http://www.optimization-online.org/DB_FILE/2019/09/7370.pdf.

[19] Eric Krall, Michael Higgins, and Richard P O'Neill. "RTO unit commitment test system". In: *Federal Energy Regulatory Commission* 98 (2012).

[20] Mehdi Madani et al. *Convex Hull, IP and European Electricity Pricing in a European Power Exchanges setting with efficient computation of Convex Hull Prices*. 2018. arXiv: 1804.00048 [math.OC].

[21] Robert R Meyer. "On the existence of optimal solutions to integer and mixed-integer programming problems". In: *Mathematical Programming* 7.1 (1974), pp. 223–235.

[22] Anthony Papavasiliou. *Optimization Models in Electricity Markets*.

[23] Anthony Papavasiliou and Yves Smeers. *Remuneration of Flexibility using Operation Reserve Demand Curves : A Case Study of Belgium*. https://ap-rg.eu/wp-content/uploads/2020/07/J11.pdf. 2016.

[24] Vanderbeck François Sadykov Ruslan. “Column generation for extended formulations”. In: *EURO Journal on Computational Optimization* (2013). DOI: 10.1007/s13675-013-0009-9. URL: https://doi.org/10.1007/s13675-013-0009-9.

[25] *Statistiques électricité*. fr. URL: https://www.febeg.be/fr/statistiques-electricite (visited on 06/04/2021).

[26] François Vanderbeck and Laurence A. Wolsey. “Reformulation and Decomposition of Integer Programs”. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Ed. by Michael Jünger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 431–502. ISBN: 978-3-540-68279-0. DOI: 10.1007/978-3-540-68279-0_13. URL: https://doi.org/10.1007/978-3-540-68279-0_13.

UNIVERSITÉ CATHOLIQUE DE LOUVAIN
École polytechnique de Louvain
Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve, Belgique | **www.uclouvain.be/epl**