TREBALL DE FI DE GRAU

Grau en Estadística

Títol: Generació d'arbres d'escenaris per a problemes d'oferta òptima en mercats d'electricitat

Autor: Roger Serra Castilla

Directors: F.- Javier Heredia, Marlyn D. Cuadrado

Departament: Estadística i Investigació Operativa, UPC

Convocatòria: Gener de 2019

UNIVERSITAT DE BARCELONA

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

AGRAÏMENTS

Al Javier Heredia, director d’aquest treball de fi de grau, per la seva implicació, paciència, orientació i, sobretot, recolzament durant tot el desenvolupament d’aquest projecte. Gràcies també per la confiança a l’hora de presentar-me aquesta proposta.

A la Marlyn Cuadrado, codirectora del treball, per l’ajuda, disposició i per haver sigut una molt bona companya d’equip.

A la meva família i amics, per la confiança. Especialment als meus pares, per tot el recolzament i suport que m’han donat durant el desenvolupament d’aquest projecte.

RESUM

El sistema de Mercats d’Electricitat (EM) es creà amb la finalitat de generar, transportar i distribuir energia elèctrica per satisfer la demanda a nivell nacional en cada moment. Aquest organisme inclou diferents mercats que interaccionen amb les plantes eòliques, les quals venen l’energia a aquests mercats. En aquets context, l’ús d’arbres d’escenaris ha de permetre a les plantes obtenir informació útil del comportament dels mercats per aconseguir el màxim de guanys en la venda d’energia. L’objectiu principal d’aquest projecte és l’obtenció d’arbres d’escenaris mitjançant algorismes de generació dinàmica per tal de satisfer la necessitat de resoldre, en un temps raonable, problemes complexos d’optimització d’oferta òptima de plantes eòliques en mercats elèctrics.

Paraules clau: arbres d’escenaris, planta eòlica, mercats elèctrics, distància niada, algorisme de generació d’arbres d’escenaris, model d’optimització WBVPP.
Classificació AMS: 05-C05 Trees, 68-W25 Approximation algorithms, 90C15 Stochastic programming

ABSTRACT

Scenario tree generation for optimal supply problems in electricity markets

The electricity markets (EM) is a regulated system which allows producers and consumers to sell and buy energy at a given price that is fixed through an auction mechanism. Thanks to the tasks done by this system, the safe generation, transportation and distribution of electricity needed to satisfy the demand of the national population is assured. The electricity markets is made up of several markets, that can be considered either spot markets (day-ahead and intraday markets, whose trading commodity is energy) or ancillary services markets (the commodity negotiated is energy used to assure the stability of the energy delivering).

On the other hand, interacting with the EM there is a wind power plant (WPP) which is in charge of the wind power energy production. A battery energy storage system (BESS) is usually associated to the WPP. The union of the WPP and the BESS is called virtual power plant (VPP).

In this context, an optimization model can be presented: the WBVPP model (WPP+BESS Virtual Power Plant). The aim of this optimization model is the maximization of the expected value of the total profit of the VPP. To calculate this value, it is necessary to quantify the amount of wind power energy that fluctuates between the VPP and the EM, as well as the clearing prices of the auctions.

The main purpose of this project is to obtain scenario trees using a dynamic generation algorithm in order to satisfy the need to solve, in a reasonable period of time, complex optimization problems of optimal supply of wind power plants in electricity markets. The scenarios trees that are going to be obtained include information regarding the production of wind power energy and the clearing prices of the auctions of the different studied electricity markets.

In this study a scenario tree generation algorithm is presented, together with all the theoretical background needed to understand and assure a correct implementation of it, as well as the definition of the different agents that perform in the context of electricity markets. In addition to that, some data will be applied to this algorithm in order to obtain a representation of the scenario trees and to understand the interpretation of them.

Keywords: scenario trees, wind power plant, nested distance, scenario tree generation algorithm, WBVPP optimization model.

ÍNDEX

ÍNDEX DE FIGURES

ÍNDEX DE TAULES

INTRODUCCIÓ

Els arbres d'escenaris són estructures bàsiques de dades per problemes d'optimització estocàstica multi-etapa. Són discretitzacions de processos estocàstics i suposen una aproximació al fenomen real. Un procés estocàstic caracteritza una successió de variables aleatòries que evolucionen en funció d'una altra variable.

En el context de les plantes eòliques en els mercats d'electricitat, esdevé la necessitat d'obtenir arbres d'escenaris que continguin informació de la generació d'energia eòlica i dels preus de venda en els mercats elèctrics. Els arbres d'escenaris han de permetre a les empreses aconseguir informació útil de com actuen els mercats d'electricitat i els han d'ajudar a obtenir el màxim de guanys.

Així doncs, en aquest context, la principal motivació d'aquest projecte és l'obtenció d'arbres d'escenaris mitjançant algorismes de generació dinàmica per tal de satisfer la necessitat de resoldre, en un temps raonable, problemes complexos d'optimització d'oferta òptima de plantes eòliques en mercats elèctrics. A partir d'aquesta motivació genèrica, s'han establert els objectius concrets i específics del treball:

- Estudi dels mercats elèctrics i de la relació que puguin establir amb una planta eòlica. Per poder entendre bé el context en el que s'està treballant, és imprescindible definir el paper de cada un dels elements en aquesta situació i les accions que poden dur a terme.
- Estudi de la distància de Wasserstein i de la distància niada. La implementació dels algorismes que crearan els arbres d'escenaris necessiten tenir una distància definida, que permeti calcular la distància entre clústers.
- Estudi d'un algorisme de *clustering* jeràrquic. Un dels principals passos per poder obtenir els arbres d'escenaris és agrupar les observacions en clústers. Un clúster es defineix com l'agrupació de diferents elements que tenen unes característiques semblants. En el cas d'estudi, aquestes agrupacions es faran a partir de les observacions que tinguin menor distància entre elles.
- Estudi d'un algorisme de generació dinàmica d'arbres d'escenaris. L'ús d'aquest algorisme, juntament amb l'algorisme mencionat en l'anterior objectiu, són els passos imprescindibles per aconseguir els arbres d'escenaris.
- Implementació dels algorismes estudiats en AMPL i obtenció d'arbres d'escenaris. Aplicant dades diàries basades en plantes eòliques en un mercat elèctric, cal aconseguir arbres d'escenaris.
- Obtenció del gràfic representatiu de l'arbre d'escenaris.
- Estudi del model de maximització de guanys d'una planta eòlica. Per poder observar una aplicació dels arbres d'escenaris obtinguts, cal presentar un model d'optimització que permeti a les plantes eòliques saber en quines condicions maximitzar els guanys.

- Comparació dels resultats dels guanys de la planta eòlica respecte el cas d'usar un model determinista.

Per poder realitzar aquest estudi, s'han seguit dues vies de treball diferenciades. Per una banda, ha sigut necessària la documentació i investigació, mitjançant articles i altra documentació bibliogràfica, per poder comprendre la base teòrica que sustenta els algorismes, així com el funcionament dels mercats elèctrics i de les plantes eòliques. Els articles que han aportat la major part de la informació han sigut Heredia et al. [3] i Plufg i Pichler [5].

Per altra banda, ha sigut necessari l'ús d'AMPL (*A Mathematical Programming Language*) per poder implementar els algorismes i, més endavant, aplicar-hi les dades amb les quals cal obtenir els arbres d'escenari: la generació d'energia eòlica i els preus d'aquesta en el mercat. A més, també s'ha utilitzat el *software* estadístic R per a l'obtenció de les representacions gràfiques dels arbres d'escenaris.

La memòria que es presentarà a continuació es divideix en cinc capítols principals. En el primer es presenta el mercat elèctric, la planta eòlica i el model que es pot obtenir en la interacció d'aquests dos elements. En el segon capítol s'introdueixen la distància de Wasserstein i la distància niada, imprescindibles per la posterior implementació dels algorismes. Seguidament, en el tercer capítol s'introdueixen els algorismes que s'usaran per la creació dels arbres d'escenaris i, en el quart, s'explica la implementació d'aquests a AMPL. En el darrer capítol es presenten els resultats obtinguts, així com una breu interpretació dels arbres d'escenaris. En últim lloc, la memòria conclou amb una síntesi de tot el projecte i amb la presentació de les conclusions obtingudes.

Fent referència als motius personals que van condicionar l'elecció d'aquest treball, el projecte m'ha permès aplicar molts dels conceptes apresos al llarg del Grau d'Estadística, especialment a l'hora de la programació dels algorismes i de la interpretació dels resultats. A més, i fent referència a la part més teòrica del treball, també s'han pogut introduir conceptes del Grau de Matemàtiques, que també estic cursant. La possibilitat de poder introduir nocions d'ambdós graus va ser una de les principals motivacions a l'hora d'escollir aquest projecte. A més, aquest estudi m'ha permès endinsar-me en un camp d'investigació que al llarg del Grau no s'havia tractat, i poder-ne veure una aplicació en un cas real. El fet d'utilitzar un llenguatge desconegut, AMPL, també va suposar una motivació addicional.

Finalment, cal puntualitzar que el treball realitzat s'emmarca en la línia de recerca d'optimització en mercats d'electricitat del grup de recerca de la UPC *Group on Numerical Optimization and Modeling.* Concretament, aquest projecte és un suport a la tesi de la codirectora del mateix, Marlyn Cuadrado, sobre generació d'arbres d'escenaris per a problemes d'oferta òptima al mercat elèctric de plantes de generació eòlica.

CAPÍTOL I: Context

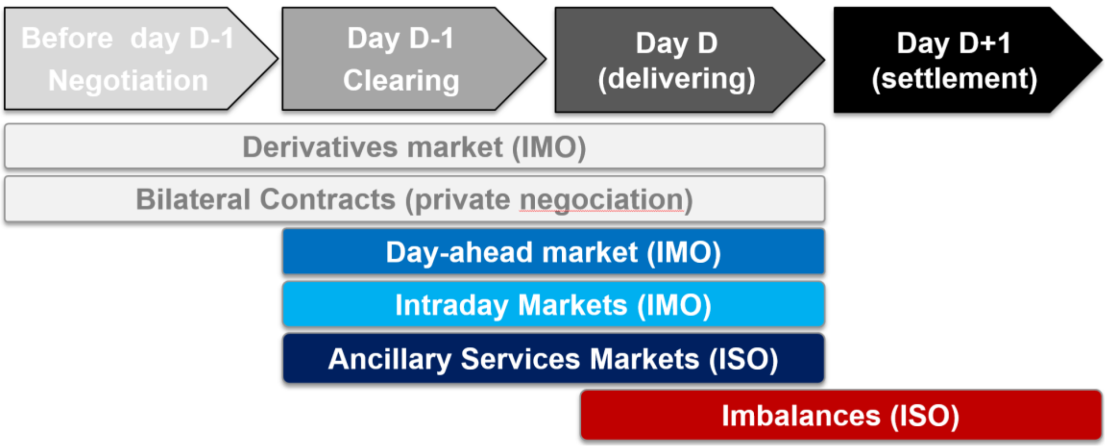
En aquest capítol es presentaran tots aquells conceptes necessaris per comprendre quin és l'entorn en el qual s'està treballant, amb la finalitat de poder entendre les dades i els resultats que s'obtinguin al final de l'estudi. La major part de la informació d'aquest capítol s'ha extret d'Heredia et al. [3].

1.1. Mercats elèctrics

L'objectiu del sistema nacional de producció d'energia és la generació, transport i distribució de l'energia elèctrica necessària per satisfer la demanda elèctrica a nivell nacional en cada moment. Per poder assolir aquest objectiu, es va crear un conjunt de Mercats d'Electricitat (EM, *Electricity Markets*), un sistema que permet als productors i consumidors vendre i comprar energia a un determinat preu fixat utilitzant mecanismes de subhasta. En els Mercats d'Electricitat hi ha dos elements principals:

- Agents. Els agents del mercat són empreses autoritzades a participar en el mercat de producció elèctrica com a compradors i venedors d'electricitat. Poden ser productors d'energia elèctrica, revenedors i consumidors directes a l'engròs.
- Operadors. Els operadors són empreses públiques encarregades d'organitzar i gestionar el mercat. Normalment es poden trobar dos tipus: els Operadors Independents de Mercat (IMO, *Independent Market Operators*), que són els encarregats de la gestió econòmica del sistema; i els Operadors Independents de Sistema (ISO, *Independent System Operators*), que són els encarregats de la gestió tècnica del sistema.

Figura I.1. Estructura dels Mercats d'Electricitat (EM)



El sistema de Mercats d'Electricitat inclou diferents mercats que, o bé es poden considerar mercats *spot* (els mercats diari i intradiari, aquells mercats on el bé comercialitzat és l'energia), o bé mercats de serveis auxiliars (aquells mercats on els béns comercialitzats són reserves d'energia o energia que s'utilitzarà per assegurar l'estabilitat i seguretat dels lliuraments d'energia). També hi ha un mercat de futurs i derivats per cobrir el risc de fluctuacions en els preus dels primers mercats, però queden fora de l'abast d'aquest projecte.
En aquest estudi es consideraran les operacions entre una planta elèctrica i els següents mercats:

- Mercat diari (DM, *Day-ahead Market*): és l'encarregat de gestionar les transaccions elèctriques per al dia següent a través d'ofertes de compra i venda d'electricitat a 24 hores fetes pels participants del mercat. Els resultats del mercat diari poden ser modificats posteriorment per la ISO, per garantir la seguretat i fiabilitat del subministrament.
- Mercat intradiari (IM, *Intraday Market*): els seus objectius són respondre als ajustos que la ISO fa dels resultats del mercat diari i respondre a les seves pròpies desviacions de la disponibilitat de generació prevista. Per aconseguir-ho, utilitza la presentació d'ofertes de compra i venda d'energia elèctrica per agents de mercat (a través de subhastes a 24 hores).
- Mercat de reserva (RM, *Secondary reserve market*): és l'encarregat de mantenir l'equilibri entre la generació i la demanda, corregint els desequilibris per omplir el buit entre el consum d'energia previst i el real.

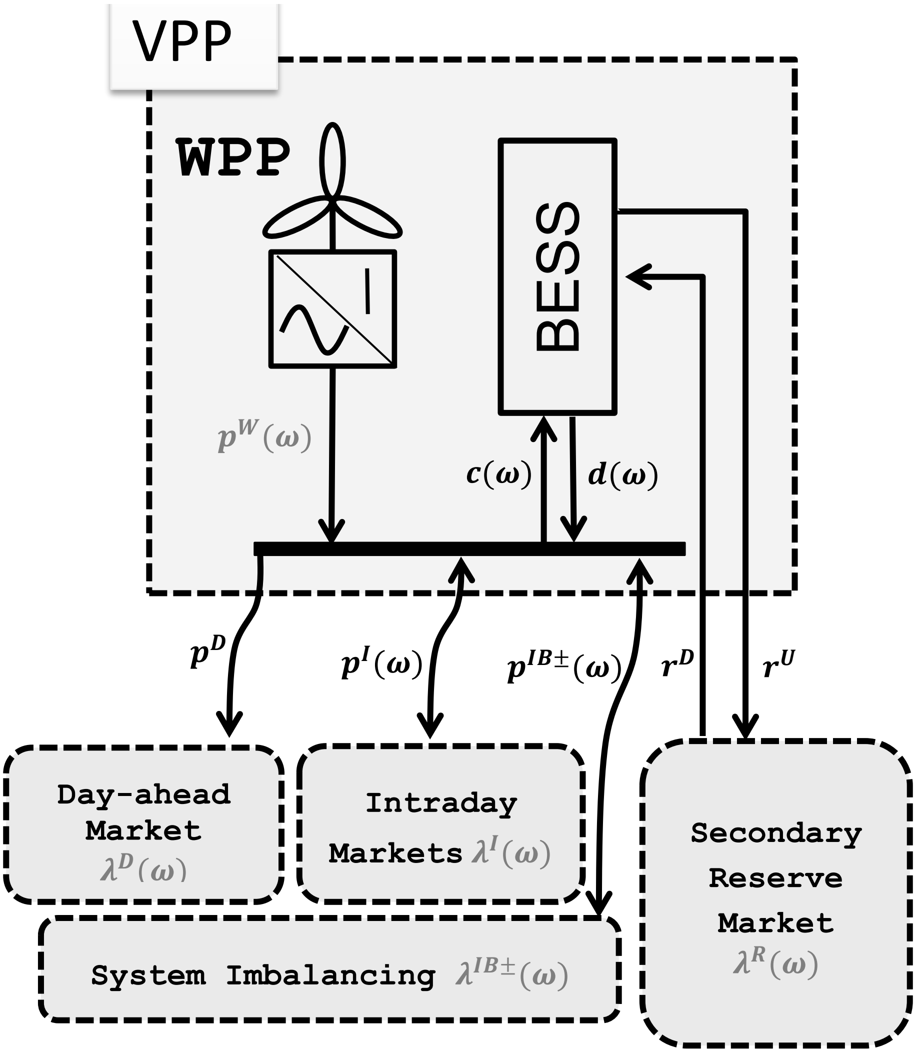
Juntament amb aquests mercats, també es tracta el sistema de desajustos (IB, *Imbalance System*). Aquest s'encarrega de calcular les desviacions reals entre la generació de la planta eòlica i l'energia casada[1] dels mercats diari i intradiari. Si la generació és superior a l'energia casada, es pagaran alguns drets de cobrament al propietari de la planta eòlica. Per altra banda, si la generació és inferior a l'energia casada, el propietari de la planta eòlica haurà de fer front a unes obligacions de pagament.
A nivell temporal, l'EM funciona de la següent manera: el dia D-1 representa el dia en el qual es fan les ofertes (període de negociació), el dia D és el dia en què es lliura l'energia acordada i el dia D+1 es determinen els desequilibris.

[1] Ofertes de venda i compra que es converteixen en compromisos ferms d'entrega d'energia.

1.2. La planta eòlica

L'element principal és una planta eòlica que genera energia elèctrica. Aquesta planta, juntament amb els diferents mercats d'energia elèctrica presentats i una bateria per emmagatzemar certa quantitat d'aquesta energia, són tots elements de la situació en la qual s'està treballant. Per visualitzar-ho millor, a continuació es presenta un gràfic on es pot veure com interactuen els elements entre sí:

Figura I.2. Paràmetres aleatoris (gris) i variables de decisió d'una VPP.



Tal i com es pot observar, es té una central eòlica (WPP, *Wind Power Plant*), que és la que s'encarrega de la generació de l'energia eòlica. A més a més, també es té una bateria, que és el sistema d'emmagatzematge d'energia (BESS, *Battery Energy Storage System)*. Aquesta pot realitzar dues accions: carregar-se o descarregar-se. La unió d'aquests dos elements formen la planta d'energia virtual (VPP, *Virtual Power Plant*), que és el conjunt que realment s'estudiarà.

Aquest sistema està en constant interacció amb els mercats d'energia elèctrica. Els mercats que es tractaran són:

- El mercat diari (DM, *Day-ahead market*): en aquest mercat, l'empresa només pot vendre-hi energia elèctrica.
- El mercat intradiari (IM, *Intraday market*): en aquest mercat, l'empresa pot vendre o comprar energia elèctrica.
- El mercat de reserva (RM, *Secondary reserve market*): en aquest mercat es corregeixen els desequilibris causats per l'empresa.

Per altra banda, hi ha un sistema de desajustos (IB, *Imbalance System*). La seva tasca és quantificar les manques o excedències d'energia en les transaccions entre la planta i els mercats. Un desajust positiu indica que la producció d'energia ha estat superior a l'energia casada, mentre que un desajust negatiu indica que la producció ha estat inferior a l'energia casada. Els dos casos tenen repercussions en els guanys de l'empresa.
La interacció de la VPP amb els mercats diari i intradiari es denotarà com VDI (*Virtual power plant with Day-ahead and Intra-day market*).

1.3. Definició dels paràmetres aleatoris i variables de decisió

El present estudi considera operacions diàries durant períodes de 24 hores, $\mathcal{T} \coloneqq \{1, \dots, 24\}$, d'una planta d'energia virtual (VPP) que opera ens els mercats diari (DM) i intradiari (IM). La Figura I.2. presenta aquesta VPP juntament amb les variables de decisió i els paràmetres aleatoris que actuen en les operacions d'aquesta planta.
L'objectiu és trobar el funcionament òptim d'aquesta VPP respecte qualsevol possible resultat de la variable aleatòria $\xi(\omega) \in \Xi$ associada a l'esdeveniment ω (són els valors escrits en gris de la Figura I.2.):

$$\xi(\omega)^T = [\lambda^D(\omega)^T, \lambda^R(\omega)^T, \lambda^I(\omega)^T, p_1^W(\omega)^T, \dots, p_{24}^W(\omega)^T, \lambda^{IB+}(\omega)^T, \lambda^{IB-}(\omega)^T].$$

El significat dels diferents paràmetres aleatoris es troba explicat en la següent taula:

Taula I.1. Paràmetres aleatoris.

Paràmetre	Significat
$\lambda^D(\omega)$	Preus de liquidació de les 24 subhastes del mercat diari [€/MW].
$\lambda^{I1\dots7}(\omega)$	Preus de liquidació de les 24 subhastes dels mercats intradiaris [€/MWh].
$p_t^W(\omega)$	Producció d'energia eòlica [MWh] al període $t \in \mathcal{T}$.
$\lambda^{IB+}(\omega), \lambda^{IB-}(\omega)$	Preus pels desajustos positius i negatius a cada hora [€/MWh].
$\lambda^R(\omega)$	Preus de liquidació de les 24 subhastes del mercat de reserva [€/MW].

Les variables de decisió associades a les operacions de la VPP corresponen a les quantitats ofertes dels tres mercats (diari, intradiari i de reserva) i a les operacions que realitza cada hora la bateria (càrrega i descàrrega), juntament amb els desequilibris de cada hora. Aquestes variables s'inclouen a la següent taula:

Taula I.2. Variables de decisió.

Variable	Significat
p^D	Quantitat d'energia de l'oferta acceptada per les 24 subhastes del mercat diari $[MWh]$.
$p^I(\omega)$	Quantitat d'energia de l'oferta acceptada per les 24 subhastes del mercat intradiari $[MWh]$.
$c_t(\omega), d_t(\omega)$	Càrregues i descàrregues de la BESS al període $t \in \mathcal{T}$ $[MW]$.
$p_t^{IB+}(\omega), p_t^{IB-}(\omega)$	Desajustos (positius i negatius) de la VPP al període $t \in \mathcal{T}$ $[MWh]$.
$r^U(\omega), r^D(\omega)$	Reserva de l'oferta acceptada per les 24 subhastes del mercat de reserva $[MW]$.

Per simplificar la notació, s'usarà $\lambda^{IB} = \begin{bmatrix} \lambda^{IB+} \\ \lambda^{IB-} \end{bmatrix}$ i $p^{IB} = \begin{bmatrix} p^{IB+} \\ p^{IB-} \end{bmatrix}$ per fer referència al conjunt dels desajustos positius i negatius en els preus i l'energia, respectivament.

1.4. Arbres d'escenaris

L'objectiu principal del projecte és construir arbres d'escenaris per representar les variables aleatòries (Taula I.2) en el model VPP.
Per aconseguir-ho, es disposarà d'unes matrius que contindran simulacions d'aquestes variables. En total es tenen 34 etapes, és a dir, variables aleatòries diferents. L'estructura d'aquests objectes és:

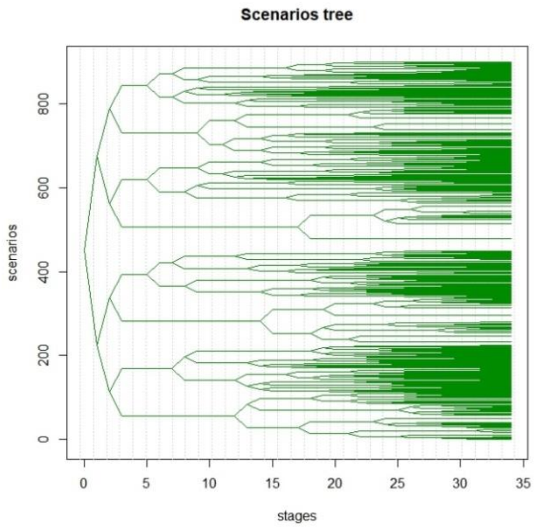
Taula I.3. Variables dels arbres d'escenaris.

$\xi \rightarrow$	$\boldsymbol{\lambda^D}$	$\boldsymbol{\lambda^R}$	$\boldsymbol{\lambda^{I1}}$	$\boldsymbol{\lambda^{I2}}$	$\boldsymbol{p^W_{1,2}}$	$\boldsymbol{\lambda^{I3}}$	$\boldsymbol{p^W_{3,4,5}}$	$\boldsymbol{\lambda^{I4}}$
etapa	1	2	3	4	5,6	7	8,9,10	11
$\boldsymbol{\mathcal{T}^\xi}$	1 ... 24	1 ... 24	1 ... 24	1 ... 24	1,2	5 ... 24	3, 4, 5	8 ... 24
$\boldsymbol{\lvert\mathcal{T}^\xi\rvert}$	24	24	24	24	2	20	3	17
$\boldsymbol{col(\xi_s)}$	1 ... 24	25 ... 48	49 ... 72	73 ... 96	97, 98	99 ... 118	119 ... 121	122 ... 138

$\boldsymbol{p^W_{6\ldots9}}$	$\boldsymbol{\lambda^{I5}}$	$\boldsymbol{p^W_{10\ldots13}}$	$\boldsymbol{\lambda^{I6}}$	$\boldsymbol{p^W_{14\ldots,19}}$	$\boldsymbol{\lambda^{I7}}$	$\boldsymbol{p^W_{20,\ldots,24}}$	$\boldsymbol{\lambda^{IB}}$
12 ... 15	16	17 ... 20	21	22 ... 27	28	29 ... 33	34
6 ... 9	12 ... 24	10 ... 13	16 ... 24	14 ... 19	22 ... 24	20 ... 24	1 ... 24
4	13	4	9	6	3	5	24
139 ... 142	143 ... 155	156 ... 159	160 ... 168	169 ... 174	175 ... 177	178 ... 182	183 ... 206

A partir de la implementació dels algorismes que es presentaran en el Capítol III, s'hi aplicaran aquestes dades per obtenir els arbres d'escenaris. Per poder entendre com es visualitzaran les variables en els arbres, a continuació s'adjunta un dels gràfics (Figura I.3.) dels arbres obtinguts:

Figura I.3. Arbre d'escenaris pel dia 6.



En l'eix de les abscisses s'hi representen les 34 etapes especificades en la Taula I.3. Per tant, si s'analitza el gràfic verticalment, els nodes que es troben sobre la mateixa etapa (línia vertical) són valors diferents per a la variable representada. Així, la primera etapa (λ^D) té un únic valor diferent, que és el primer node del qual en surten dos arcs. La segona etapa (λ^R) ja està formada per dos nodes diferents (dos valors diferents). Cal especificar, però, que en aquests gràfics no es representen els valors de les variables, sinó que serveixen per poder entendre l'estructura de cada arbre.

En el Capítol V, s'interpretaran aquests resultats en termes de la generació d'energia eòlica i de les transaccions diàries que ha de fer una VPP per aconseguir els màxims guanys.

CAPÍTOL II: Una distància

Per poder estudiar i implementar els algorismes, que es presentaran més endavant, és necessari definir abans una distància. En aquest apartat es presentarà la distància de Wasserstein, així com un cas particular d'ella: la distància niada.

Abans, però, és necessari definir els conceptes de semi-distància i distància. Sigui $\mathcal{P}$ un conjunt de mesures de probabilitat de $\mathbb{R}^m$.

Definició 2.1. Una semi-distància d a $\mathcal{P} \times \mathcal{P}$ satisfà les tres condicions següents:

I. No negativitat: $\forall P_1, P_2 \in \mathcal{P}$,
$$\mathrm{d}(P_1, P_2) \geq 0.$$
II. Simetria: $\forall P_1, P_2 \in \mathcal{P}$,
$$\mathrm{d}(P_1, P_2) = \mathrm{d}(P_2, P_1).$$
III. Desigualtat triangular: $\forall P_1, P_2, P_3 \in \mathcal{P}$
$$\mathrm{d}(P_1, P_2) \leq \mathrm{d}(P_1, P_3) + \mathrm{d}(P_3, P_2).$$

Una semi-distància $\mathrm{d}(\cdot,\cdot)$ s'anomena distància si satisfà la següent propietat:

IV. Si $\mathrm{d}(P_1, P_2) = 0$, aleshores $P_1 = P_2$.

Definició 2.2. S'anomena espai mètric a la parella (M, d), on M és un conjunt i d una distància.

2.1. La distància de Wasserstein

En el context de l'optimització estocàstica, una distància útil que s'hi adapti hauria de complir les següent propietats:

- Mesurar distàncies entre distribucions i ser independent dels espais de probabilitat.
- Permetre implementacions computacionals raonables.
- Estendre's a processos estocàstics generals.

La distància de Wasserstein, que es veurà que és la solució d'un problema d'optimització, cobreix totes les propietats desitjades de forma natural.

Per poder definir la distància de Wasserstein cal considerar una funció real i mesurable:
$$c: \Omega \times \widetilde{\Omega} \to \mathbb{R}$$
enllaçant dos espais de mostreig Ω i $\widetilde{\Omega}$.

Definició 2.1.1. La funció c sovint s'associa a la interpretació que moure un element $\omega \in \Omega$ a $\widetilde{\omega} \in \widetilde{\Omega}$ té cost $c(\omega, \widetilde{\omega})$. Per aquest motiu, a aquesta funció se l'anomena funció de cost.

La definició més comuna de la distància de Wasserstein considera la funció de cost:
$$c(\cdot,\cdot) \coloneqq \mathrm{d}(\cdot,\cdot)^r: \Omega \times \Omega \to \mathbb{R}$$

on d és una distància en Ω i $r \geq 1$. Es pot observar com en aquesta situació la funció de cost només actua en un sol espai.
Per poder definir la distància de Wasserstein és necessari conèixer el cost de transport òptim i la distància heretada. Aquesta última també fa servir el concepte de norma, que es definirà prèviament.
Definició 2.1.2. Sigui E un espai vectorial sobre $\mathbb{R}$. Una norma en E és una aplicació:

$$\| \cdot \| : E \to \mathbb{R}$$

tal que $\forall x, y \in E$ i $\forall \lambda \in \mathbb{R}$:

- $\|x\| \geq 0$
- $\|x\| = 0 \leftrightarrow x = 0$
- $\|\lambda x\| = |\lambda| \cdot \|x\|$
- $\|x + y\| \leq \|x\| + \|y\|$

Definició 2.1.3. S'anomena espai normat a la parella $(E, \| \cdot \|)$.
Definició 2.1.4. Sigui ξ una variable aleatòria de $\mathbb{R}^m$ en Ω i $\tilde{\xi}$ una altra variable aleatòria de $\mathbb{R}^m$ en $\widetilde{\Omega}$. Aleshores, la distància heretada entre Ω i $\widetilde{\Omega}$ es pot definir com el cost $c(\omega, \widetilde{\omega})$, és a dir:

$$\mathrm{d}(\omega, \widetilde{\omega}) \coloneqq c(\omega, \widetilde{\omega}) = \mathrm{d}\big(\xi(\omega), \tilde{\xi}(\widetilde{\omega})\big)$$

per a una distància d en $\mathbb{R}^m$. Normalment, es té $\mathrm{d}(u, v) = \|u - v\|$ per alguna norma $\| \cdot \|$ en $\mathbb{R}^m$.
Definició 2.1.5. Donats dos espais de probabilitat $(\Omega, \mathcal{F}, P)$ i $(\widetilde{\Omega}, \tilde{\mathcal{F}}, \tilde{P})$ i la funció de cost c, el cost òptim de transport és:

$$\inf_\pi \iint_{\Omega \times \widetilde{\Omega}} c(\omega, \widetilde{\omega}) \pi(d\omega, d\widetilde{\omega}),$$

on l'ínfim es pren en totes les mesures bivariants de probabilitat π en $\Omega \times \widetilde{\Omega}$. Amb les marginals P i $\tilde{P}$ es té:

$$\pi\big(A \times \widetilde{\Omega}\big) = P(A), \;\; \pi(\Omega \times B) = \tilde{P}(B)$$

per a tots els conjunts mesurables $A \in \mathcal{F}$ i $B \in \tilde{\mathcal{F}}$. La mesura òptima π s'anomena pla òptim de transport.
Així doncs, ara ja es pot definir la distància de Wasserstein.
Definició 2.1.6. La distància de Wasserstein d'ordre r $(r \geq 1)$ es defineix com:

$$\mathrm{d}_r\big(P, \tilde{P}\big) \coloneqq (\inf_\pi \iint_{\Omega \times \widetilde{\Omega}} \mathrm{d}(\omega, \widetilde{\omega})^r \pi(d\omega, d\widetilde{\omega}))^{1/r},$$

on l'ínfim està entre totes les mesures de probabilitat conjuntes π en $\Omega \times \widetilde{\Omega}$.

Donades dues mesures discretes $P = \sum_{i=1}^{n} P_i \delta_{\xi_i}$ i $\tilde{P} = \sum_{j=1}^{\tilde{n}} \tilde{P}_j \delta_{\tilde{\xi}_j}$, el càlcul de la distància de Wasserstein correspon a resoldre el problema lineal:

$$\begin{aligned} \underset{(en\,\pi)}{minimitza} &: \sum_{i,j} \pi_{i,j} \cdot d_{i,j}^r \\ subjecte\ a &\quad \sum_{j=1}^{\tilde{n}} \pi_{i,j} = P_i \qquad (i = 1, \dots, n) \\ &\quad \sum_{i=1}^{n} \pi_{i,j} = \tilde{P}_j \qquad (j = 1, \dots, \tilde{n}) \\ &\quad \pi_{i,j} \geq 0 \end{aligned}$$

on $d_{i,j} = \mathrm{d}(\xi_i, \tilde{\xi}_j)$ és una matriu $n \times \tilde{n}$ que conté les distàncies. La matriu $n \times \tilde{n}$ que conté $\pi_{i,j}$ correspon a les mesures de probabilitat bivariants:

$$\pi = \sum_{i,j} \pi_{i,j} \cdot \delta_{(\xi_i, \tilde{\xi}_j)}.$$

A més es té:

$$\sum_{i,j} \pi_{i,j} = \sum_i \sum_j \pi_{i,j} = \sum_i P_i = 1.$$

2.2. La distància niada

La distància de Wasserstein permet una generalització per a processos estocàstics: la distància niada.

Definició 2.2.1. La distància niada d'ordre r $(r \geq 1)$ de dos espais de probabilitat filtrats $\mathbb{P} = (\Omega, (\mathcal{F}_t), P)$ i $\widetilde{\mathbb{P}} = \left(\widetilde{\Omega}, (\tilde{\mathcal{F}}_t), \tilde{P}\right)$, pels quals es defineix una distància $\mathrm{d}: \Omega \times \widetilde{\Omega} \to \mathbb{R}$, és el valor òptim del problema d'optimització:

$$\begin{aligned} \underset{(en\,\pi)}{minimitza} &: \left(\int \mathrm{d}(\omega, \widetilde{\omega})^r \pi(d\omega, d\widetilde{\omega}) \right)^{1/r} \\ subjecte\ a &\quad \pi\left(A \times \widetilde{\Omega} \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t\right) = P(A \mid \mathcal{F}_t)\ (A \in \mathcal{F}_t, t \in \mathbf{T}) \\ &\quad \pi\left(\Omega \times B \mid \mathcal{F}_t \otimes \tilde{\mathcal{F}}_t\right) = \tilde{P}\left(B \mid \tilde{\mathcal{F}}_t\right) \left(B \in \tilde{\mathcal{F}}_t, t \in \mathbf{T}\right) \end{aligned}$$

on l'ínfim està entre totes les mesures de probabilitat conjuntes $\pi \in \mathcal{P}(\Omega \times \widetilde{\Omega})$ i es té $\mathbf{T} = \{0, 1, \dots, T\}$. El seu valor òptim, la distància niada, es denota per:

$$\mathbb{d}_r\left(\mathbb{P}, \widetilde{\mathbb{P}}\right).$$

Observació 2.2.2. La distància niada a vegades també s'anomena distància multi-etapa o distància de processos. Una mesura factible π s'anomena pla de transport niat.

Tal i com s'ha vist anteriorment, la distància de Wasserstein entre mesures discretes de probabilitat es pot calcular resolent un problema d'optimització lineal. Per establir el problema corresponent per la distància niada es fan servir arbres que modelitzin tot l'espai. Per poder-lo definir, es farà servir la notació $m \prec i$ per indicar que el node m és un predecessor del node i (no necessàriament el predecessor immediat).

Definició 2.2.3. La distància niada es pot obtenir resolent un problema d'optimització lineal. Siguin P i $\tilde{P}$ donats, el valor de la distància és el valor òptim de:

$$\begin{array}{cl} \underset{(en\,\pi)}{minimitza}: & \displaystyle\sum_{i,j} \pi_{i,j} \cdot d_{i,j}^r \\ subjecte\ a & P(i) \cdot \displaystyle\sum_{k \prec i', l \prec j'} \pi_{i',j'} = P(k) \cdot \sum_{l \prec j'} \pi_{i,j'}\ (k \prec i) \\ & \tilde{P}(j) \cdot \displaystyle\sum_{k \prec i', l \prec j'} \pi_{i',j'} = \tilde{P}(l) \cdot \sum_{k \prec i'} \pi_{i',j}\ (l \prec j) \\ & \pi_{i,j} \geq 0 \\ & \displaystyle\sum_{i,j} \pi_{i,j} = 1 \end{array}$$

Observació 2.2.4. Cal considerar el cas en el qual un dels dos arbres únicament tingui un node. Aleshores, la distància niada entre els arbres $\aleph$ i $\tilde{\aleph}$, on el segon és el que té només un node és:

$$\mathbb{d}_r(\aleph, \tilde{\aleph}) = \sum_i \pi_{i,1} \cdot d_{i,1}^r$$

i on es compleix que:

$$\sum_i \pi_{i,1} = 1.$$

CAPÍTOL III: Els algorismes

A continuació, es presentaran els tres algorismes necessaris per a l'obtenció d'arbres d'escenaris. L'estructura teòrica d'aquests, però, cal puntualitzar que no és com finalment s'han aplicat. Aquestes modificacions es presentaran en el proper capítol.
Per altra banda, el nom que s'usarà per fer referència a aquests algorismes és el mateix que es fa servir en l'article de Plufg i Pichler [5].

3.1. Algorisme 1: un algorisme de *clustering* jeràrquic

Inicialment, és necessari presentar un conegut algorisme per partir un gran conjunt de punts de $\mathbb{R}^m$ a s clústers. L'Algorisme 1 agrupa un conjunt de punts en subconjunts que tinguin una distància menor. Aquest algorisme es pot fer servir per generar escenaris representatius, però en el projecte s'usarà per aconseguir els clústers a partir dels quals es generaran les observacions de l'arbre d'escenaris.
Algorisme 3.1.1. Aquest algorisme es divideix en tres passos:

I. **Mostreig.** Se suposa que es tenen n punts $\{z^{(1)}, \dots, z^{(n)}\}$ en $\mathbb{R}^m$ dotat d'una mètrica d donada. El conjunt de tots aquests, $Z = \{z^{(i)} : i = 1, \dots, n\}$, se separa iterativament en clústers disjunts, de manera que el nombre vagi disminuint a cada pas. Al començament, cada punt és un clúster, és a dir, es tenen un total de n clústers.

II. **Iteració.** Se suposa que la partició actual del conjunt és $\mathbb{R}^m = \bigcup_j^+ C_j$. L'objectiu és trobar la parella $\left(C_j, C_k\right)$ per la qual:
$$\sup\{\mathrm{d}(z, z') : z \in C_j, z' \in C_k\}$$
és mínim. Aleshores es crea un nou clúster agrupant C_j amb C_k.

III. **Criteri d'aturada**. Si el nombre de clústers després de l'agrupació ha disminuït fins al nombre desitjat s, aleshores es finalitza l'execució de l'algorisme. Altrament, cal tornar al pas (II).

Així doncs, a l'hora d'implementar aquest algorisme és necessari definir una mètrica, per poder calcular la distància entre clústers.
Cal mencionar que hi ha més algorismes que s'encarreguen de fer agrupacions, però tensen un cost computacional més elevat (tarden més en aconseguir el mateix resultat) i són més difícils d'implementar.

3.2. Algorisme 3: aproximació estocàstica

A continuació, es presenta un algorisme d'aproximació estocàstica per aconseguir la discretització de la mesura de probabilitat P, a partir d'una mesura de probabilitat sobre s punts. Aquest és una modificació d'un algorisme determinista, que numèricament és molt complex.

L'algorisme requereix que es pugui obtenir una seqüència independent i idènticament distribuïda (i.i.d.)

$$\xi(1), \dots, \xi(n)$$

de vectors de llargada arbitrària n, cada un distribuït segons P.

Algorisme 3.2.1. Aquest algorisme es divideix en els següents passos:

I. **Mostreig.** Es té una mostra aleatòria de mida n de la distribució P. Usar un algorisme de *clustering* (per exemple, l'Algorisme 1 explicat anteriorment) per aconseguir s clústers. Es fixa $k = 0$, que indicarà la iteració que s'està realitzant, i sigui el conjunt $Z(0) = \{z^{(1)}(0), \dots, z^{(n)}(0)\}$ el clúster de les medianes. A més, cal escollir una sèrie a_k no negativa[2] i decreixent tal que:

$$\sum_{k=1} a_k^2 < \infty \ i \ \sum_{k=1} a_k = \infty.$$

II. **Iteració.** S'utilitza una mostra independent $\xi(k)$. Cal trobar l'índex i tal que:

$$\mathrm{d}\left(\xi(k), z^{(i)}(k)\right) = \min_l \mathrm{d}\left(\xi(k), z^{(l)}(k)\right),$$

i aleshores es defineix:

$$z^{(i)}(k+1) := z^{(i)}(k) - a_k \cdot r\mathrm{d}(\xi(k), z^{(i)}(k))^{r-1} \cdot \nabla_{\xi}\mathrm{d}\left(\xi(k), z^{(i)}(k)\right)$$

deixant els altres punts sense modificar. D'aquesta manera es crea un nou conjunt de punts $Z(k+1)$.

III. **Criteri d'aturada**. S'atura si, o bé s'ha executat el nombre predeterminat d'iteracions, o bé si el canvi relatiu del conjunt de punts Z està sota un límit ϵ. Sinó, es fixa $k = k + 1$ i es torna a (II).

IV. **Determinació de les probabilitats.** Després d'haver determinat el conjunt de punts final Z, es genera una nova mostra $(\xi(1), \dots, \xi(n))$ i es troben les probabilitats següents:

$$p_i = \frac{1}{n} \# \left\{l \colon \mathrm{d}\big(\xi(l), z^{(i)}\big) = \min_k \mathrm{d}\big(\xi(l), z^{(k)}\big)\right\}.$$

Al final, la distribució aproximada és $\tilde{P} = \sum_{i=1}^{s} p_i \cdot \delta_{Z^{(i)}}$, i la distància és:

$$\mathrm{d}_r(P, \tilde{P})^r \cong \frac{1}{n} \sum_{l=1}^{n} \min_k \mathrm{d}\big(\xi(l), z^{(k)}\big)^r.$$

---

[2] Una sèrie no negativa és el mateix que una sèrie de termes positius ($a_n \geq 0$).

Observació 3.2.2. Una sèrie que es podria escollir és:

$$a_k = \frac{1}{(k+30)^{\frac{3}{4}}}$$

Demostració. Cal veure que aquesta sèrie compleix les propietats demanades en l'algorisme. Per poder fer aquesta demostració, s'utilitzaran els conceptes inclosos en l'Annex 1 (Sèries numèriques).

És evident que $\forall k > 0, a_k > 0$, és a dir, és una sèrie no negativa. També es pot observar com:

$$\frac{\partial a_k}{\partial k} = \frac{-3}{4 \cdot (k+30)^{\frac{7}{4}}}$$

I, per tant, $\forall k > 0 \ \ \frac{\partial a_k}{\partial k} < 0$, és a dir, la seqüència és decreixent.

A continuació, s'observa que:

$$\sum_{k=1} a_k^2 = \sum_{k=1} \frac{1}{(k+30)^{\frac{6}{4}}} \leq \sum_{k=1} \frac{1}{k^{\frac{6}{4}}}$$

Que és la sèrie de Riemann amb $p > 1$ i, per tant, convergent. Utilitzant el criteri de comparació directa s'obté que la sèrie que s'està estudiant és convergent, és a dir:

$$\sum_{k=1} a_k^2 < \infty.$$

Utilitzant el mateix criteri, s'observa com:

$$\sum_{k=1} a_k = \infty.$$

Per tant, aquesta sèrie compleix totes les condicions per poder ser usada en l'algorisme d'aproximació estocàstica. ■

Observació 3.2.3. Una variant de l'Algorisme 3 evita determinar les probabilitats en un pas separat, ja que ho fa en cada iteració.

3.3. Algorisme 7: generació dinàmica d'arbres d'escenari amb espessor flexible

El següent algorisme és el que s'encarregarà d'obtenir l'arbre d'escenaris. Per poder-lo implementar és necessari tenir mostres disponibles de la distribució de probabilitat condicionada:

$$F_{t+1}(\cdot \,|\xi_t, \xi_{t-1}, \dots, \xi_0).$$

Aquestes observacions s'obtenen considerant les mostres de totes les etapes anteriors, és a dir, a partir dels valors de les etapes $0, \dots, t$ s'obté el de l'etapa següent (etapa $t+1$).

Algorisme 3.3.1. Aquest algorisme es divideix en els següents passos:

I. **Paràmetres.** Sigui T l'alçada desitjada de l'arbre, siguin $(b_1, \dots, b_T)$ els valors mínims de l'espessor per a cada etapa i $(d_1, \dots, d_T)$ les màximes distàncies de transport per etapes. Aquests dos vectors es fixen a l'inici.

II. **Determinar l'arrel.** El valor del procés a l'arrel és ξ_0 a l'etapa 0. Determinar l'arrel com l'actual node obert.

III. **MENTRE** hi hagi nodes oberts **FER:**

   A. Sigui k el següent node obert i sigui $t < T$ la seva etapa. Siguin $\xi_0, \dots, \xi_{t-1,}\xi_t$ els valors ja fixats en el node k i els seus predecessors. Es fixa el nombre inicial de successors de k a $s = b_{t+1}$.

   B. Cridar l'Algorisme 3 per generar s punts $z_1^*, \dots, z_s^*$ de la distribució:

$$F_{t+1}(\cdot\,|\xi_t, \xi_{t-1}, \dots, \xi_0).$$

   C. Calcular la distància $d = \mathrm{d}(F_{t+1}(\cdot\,|\xi_t, \xi_{t-1}, \dots, \xi_0), \sum_{i=1}^{s} p_i \cdot \delta_{z^{(i)}})$.

   D. Si la distància d és més gran que d_{t+1}, aleshores augmentar b_{t+1} en una unitat i tornar a (B).

   E. Guardar els b_{t+1} nodes successors del node k usant els valors de les variables z_i^* així com també les seves probabilitats condicionals p_i^* i marcar-los com a oberts.

IV. **Criteri d'aturada.** Si tots els nodes a l'etapa $T - 1$ s'han considerat com a nodes enllaçats, la generació de l'arbre s'ha acabat.

CAPÍTOL IV: Procediments

A continuació, s'explicaran els diferents passos que es van seguir per desenvolupar i implementar l'algorisme de generació dinàmica d'arbres d'escenari.
Inicialment es va aplicar manualment l'Algorisme 1; després, es va realitzar un altre exemple a mà de l'Algorisme 7, en el qual es van realitzar alguns canvis respecte el model original i, finalment, ja es va programar a AMPL i s'hi van aplicar les dades, per tal d'obtenir els arbres resultants.

4.1. Exercici pràctic: Algorisme 1

El primer pas que es va realitzar va ser implementar l'Algorisme 1, d'agrupació de les dades en clústers. En aquest exercici pràctic es consideren un total d'11 mostres aleatòries d'una distribució de probabilitat P i mesures de probabilitat discretes que venen donades a la següent taula:

Taula IV.1. Dades exercici pràctic Algorisme 1.

	NIVELL			
ESCENARIS	**0**	**1**	**2**	**Probabilitat**
1	10	13	15	0.02
2	10	13	14	0.04
3	10	13	13	0.08
4	10	13	11	0.06
5	10	11	12	0.21
6	10	11	9	0.09
7	10	8	10	0.15
8	10	8	8	0.06
9	10	8	6	0.09
10	10	6	7	0.08
11	10	6	5	0.12

L'objectiu d'aquest exercici és agrupar aquestes dades en un nombre determinat de clústers, que es va determinar que fos 7.
Tal i com s'ha comentat anteriorment, l'Algorisme 1 es basa en els següents passos:

ALGORISME 1:

I. **Mostreig.** Se suposa que es tenen n punts $\{z^{(1)}, \dots, z^{(n)}\}$ en $\mathbb{R}^m$ dotat d'una mètrica d donada. Al començament, cada punt és un clúster.

II. **Iteració.** Se suposa que la partició actual del conjunt és $\mathbb{R}^m = \bigcup_j^+ C_j$. L'objectiu és trobar la parella (C_j, C_k) per la qual

$$\sup\{\mathrm{d}(z, z'): z \in C_j, z' \in C_k\}$$

és mínim. Aleshores es crea un nou clúster agrupant C_j amb C_k.

III. **Criteri d'aturada**. Si el nombre de clústers ha disminuït fins al nombre desitjat s, aleshores s'atura. Altrament, cal tornar al pas (II).

Per aplicar aquest algorisme, cal definir una distància d:

$$\mathrm{d}_{i,j} = \sum_{t=0}^{2} \left| x_{i,t} - x_{j,t} \right|$$

que és la distància de Wasserstein calculada entre arbres que únicament tenen un sol escenari.

També cal mencionar què cal fer si el valor mínim de la distància es repeteix, cas que no surt explicat en la definició de l'algorisme. En aquesta situació, s'ha decidit escollir la primera parella de clústers que la tinguin.

A continuació, es prosseguirà a l'aplicació de l'algorisme en el cas a estudiar.

- **Mostreig**

En l'exemple que s'està treballant, es tenen un total de $n = 11$ punts, i s'ha establert obtenir $s = 7$ clústers. Com que inicialment cada punt és un clúster ell mateix, la situació de partida és:

$$C_1 = \begin{pmatrix} 0.02 \\ 10 \\ 13 \\ 15 \end{pmatrix} \quad C_2 = \begin{pmatrix} 0.04 \\ 10 \\ 13 \\ 14 \end{pmatrix} \quad C_3 = \begin{pmatrix} 0.08 \\ 10 \\ 13 \\ 13 \end{pmatrix} \quad C_4 = \begin{pmatrix} 0.06 \\ 10 \\ 13 \\ 11 \end{pmatrix}$$

$$C_5 = \begin{pmatrix} 0.21 \\ 10 \\ 11 \\ 12 \end{pmatrix} \quad C_6 = \begin{pmatrix} 0.09 \\ 10 \\ 11 \\ 9 \end{pmatrix} \quad C_7 = \begin{pmatrix} 0.15 \\ 10 \\ 8 \\ 10 \end{pmatrix} \quad C_8 = \begin{pmatrix} 0.06 \\ 10 \\ 8 \\ 8 \end{pmatrix}$$

$$C_9 = \begin{pmatrix} 0.09 \\ 10 \\ 8 \\ 6 \end{pmatrix} \quad C_{10} = \begin{pmatrix} 0.08 \\ 10 \\ 6 \\ 7 \end{pmatrix} \quad C_{11} = \begin{pmatrix} 0.12 \\ 10 \\ 6 \\ 5 \end{pmatrix}$$

Així doncs, es tenen 11 clústers.

- **Iteració 1**

Cal calcular les distàncies entre els diferents clústers. Per fer-ho, s'utilitzarà:

$$\mathrm{d}_{i,j} = \sum_{t=0}^{2} \left| x_{i,t} - x_{j,t} \right|$$

En aquesta primera iteració no cal aplicar el suprem, ja que cada clúster està format únicament per un punt. Les distàncies aconseguides són:

$\mathrm{d}_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}	C_{11}
C_1		1	2	4	5	8	10	12	14	15	17
C_2	1		1	3	4	7	9	11	13	14	16
C_3	2	1		2	3	6	8	10	12	13	15
C_4	4	3	2		3	4	6	8	10	11	13
C_5	5	4	3	3		3	5	7	9	10	12
C_6	8	7	6	4	3		4	4	6	7	9
C_7	10	9	8	6	5	4		2	4	5	7
C_8	12	11	10	8	7	4	2		2	3	5
C_9	14	13	12	10	9	6	4	2		3	3
C_{10}	15	14	13	11	10	7	5	3	3		2
C_{11}	17	16	15	13	12	9	7	5	3	2	

Es pot observar que és una matriu simètrica, ja que una de les propietats de la distància és la simetria:

$$\mathrm{d}_{i,j} = \mathrm{d}_{j,i}$$

Per poder entendre millor com es calculen aquestes distàncies, a continuació s'inclouen alguns exemples:

$$\mathrm{d}_{1,2} = \sum_{t=0}^{2} \left| x_{1,t} - x_{2,t} \right| = \left| x_{1,0} - x_{2,0} \right| + \left| x_{1,1} - x_{2,1} \right| + \left| x_{1,2} - x_{2,2} \right| =$$

$$= |10 - 10| + |13 - 13| + |15 - 14| = 1$$

$$\mathrm{d}_{1,11} = \sum_{t=0}^{2} \left| x_{1,t} - x_{11,t} \right| = \left| x_{1,0} - x_{11,0} \right| + \left| x_{1,1} - x_{11,1} \right| + \left| x_{1,2} - x_{11,2} \right| =$$

$$= |10 - 10| + |13 - 6| + |15 - 5| = 17$$

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\left\{ \mathrm{d}(z, z') \colon z \in C_j, z' \in C_k, j \neq k \right\} \right) = 1$$

que és la distància que hi ha entre els clústers 1 i 2 i entre els clústers 2 i 3 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, per tant, es crearà un nou clúster format per C_1 i C_2.

Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2}) = \begin{pmatrix} 0.02 & 0.04 \\ 10 & 10 \\ 13 & 13 \\ 15 & 14 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0.08 \\ 10 \\ 13 \\ 13 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 0.06 \\ 10 \\ 13 \\ 11 \end{pmatrix}$$

$$C_4 = \begin{pmatrix} 0.21 \\ 10 \\ 11 \\ 12 \end{pmatrix} \qquad C_5 = \begin{pmatrix} 0.09 \\ 10 \\ 11 \\ 9 \end{pmatrix} \qquad C_6 = \begin{pmatrix} 0.15 \\ 10 \\ 8 \\ 10 \end{pmatrix} \qquad C_7 = \begin{pmatrix} 0.06 \\ 10 \\ 8 \\ 8 \end{pmatrix}$$

$$C_8 = \begin{pmatrix} 0.09 \\ 10 \\ 8 \\ 6 \end{pmatrix} \qquad C_9 = \begin{pmatrix} 0.08 \\ 10 \\ 6 \\ 7 \end{pmatrix} \qquad C_{10} = \begin{pmatrix} 0.12 \\ 10 \\ 6 \\ 5 \end{pmatrix}$$

- **Criteri d'aturada**

En finalitzar aquesta iteració es tenen 10 clústers. Com que $s = 7$, s'ha de realitzar una altra iteració.

- **Iteració 2**

Cal calcular les distàncies entre els 10 clústers. En aquesta segona iteració ja s'ha d'aplicar el suprem, degut a que el primer clúster està format per més d'un punt. Les distàncies aconseguides són:

$\mathrm{d}_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
C_1		2	4	5	8	10	12	14	15	17
C_2	2		2	3	6	8	10	12	13	15
C_3	4	2		3	4	6	8	10	11	13
C_4	5	3	3		3	5	7	9	10	12
C_5	8	6	4	3		4	4	6	7	9
C_6	10	8	6	5	4		2	4	5	7
C_7	12	10	8	7	4	2		2	3	5
C_8	14	12	10	9	6	4	2		3	3
C_9	15	13	11	10	7	5	3	3		2
C_{10}	17	15	13	12	9	7	5	3	2	

Per tal de poder entendre com es calcula la distància utilitzant el suprem, a continuació se n'inclou un exemple:

$$\mathrm{d}_{1,2} = \sup\{\mathrm{d}(z, z') : z \in C_1, z' \in C_2\} = \sup\{\mathrm{d}(C_{1,1}, C_2), \mathrm{d}(C_{1,2}, C_2)\} = \sup\{2, 1\} = 2$$

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z, z') : z \in C_j, z' \in C_k, j \neq k\}\right) = 2$$

que és la distància que hi ha entre els clústers 1 i 2, entre els clústers 2 i 3, entre els clústers 6 i 7, entre els clústers 7 i 8 i entre els clústers 9 i 10 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, per tant, es crearà un nou clúster format per C_1 i C_2. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2} \quad C_{1,3}) = \begin{pmatrix} 0.02 & 0.04 & 0.08 \\ 10 & 10 & 10 \\ 13 & 13 & 13 \\ 15 & 14 & 13 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0.06 \\ 10 \\ 13 \\ 11 \end{pmatrix}$$

$$C_3 = \begin{pmatrix} 0.21 \\ 10 \\ 11 \\ 12 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 0.09 \\ 10 \\ 11 \\ 9 \end{pmatrix} \qquad C_5 = \begin{pmatrix} 0.15 \\ 10 \\ 8 \\ 10 \end{pmatrix} \qquad C_6 = \begin{pmatrix} 0.06 \\ 10 \\ 8 \\ 8 \end{pmatrix}$$

$$C_7 = \begin{pmatrix} 0.09 \\ 10 \\ 8 \\ 6 \end{pmatrix} \qquad C_8 = \begin{pmatrix} 0.08 \\ 10 \\ 6 \\ 7 \end{pmatrix} \qquad C_9 = \begin{pmatrix} 0.12 \\ 10 \\ 6 \\ 5 \end{pmatrix}$$

- **Criteri d'aturada**

En finalitzar aquesta iteració es tenen 9 clústers. Com que $s = 7$, s'ha de realitzar una altra iteració.

- **Iteració 3**

Les distàncies aconseguides entre els 9 clústers són:

$d_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
C_1		4	5	8	10	12	14	15	17
C_2	4		3	4	6	8	10	11	13
C_3	5	3		3	5	7	9	10	12
C_4	8	4	3		4	4	6	7	9
C_5	10	6	5	4		2	4	5	7
C_6	12	8	7	4	2		2	3	5
C_7	14	10	9	6	4	2		3	3
C_8	15	11	10	7	5	3	3		2
C_9	17	13	12	9	7	5	3	2	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{d(z, z') : z \in C_j, z' \in C_k, j \neq k\}\right) = 2$$

que és la distància que hi ha entre els clústers 5 i 6, entre els clústers 6 i 7 i entre els clústers 8 i 9 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, es crearà un nou clúster format per C_5 i C_6. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2} \quad C_{1,3}) = \begin{pmatrix} 0.02 & 0.04 & 0.08 \\ 10 & 10 & 10 \\ 13 & 13 & 13 \\ 15 & 14 & 13 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0.06 \\ 10 \\ 13 \\ 11 \end{pmatrix}$$

$$C_3 = \begin{pmatrix} 0.21 \\ 10 \\ 11 \\ 12 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 0.09 \\ 10 \\ 11 \\ 9 \end{pmatrix} \qquad C_5 = (C_{5,1} \quad C_{5,2}) = \begin{pmatrix} 0.15 & 0.06 \\ 10 & 10 \\ 8 & 8 \\ 10 & 8 \end{pmatrix}$$

$$C_6 = \begin{pmatrix} 0.09 \\ 10 \\ 8 \\ 6 \end{pmatrix} \qquad C_7 = \begin{pmatrix} 0.08 \\ 10 \\ 6 \\ 7 \end{pmatrix} \qquad C_8 = \begin{pmatrix} 0.12 \\ 10 \\ 6 \\ 5 \end{pmatrix}$$

- **Criteri d'aturada**

En finalitzar aquesta iteració es tenen 8 clústers. Com que $s = 7$, s'ha de realitzar una altra iteració.

- **Iteració 4**

Les distàncies aconseguides entre els 8 clústers són:

$d_{i,j}$	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8
C_1		4	5	8	12	14	15	17
C_2	4		3	4	8	10	11	13
C_3	5	3		3	7	9	10	12
C_4	8	4	3		4	6	7	9
C_5	12	8	7	4		4	5	7
C_6	14	10	9	6	4		3	3
C_7	15	11	10	7	5	3		2
C_8	17	13	12	9	7	3	2	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z, z') : z \in C_j, z' \in C_k, j \neq k\}\right) = 2$$

que és la distància que hi ha entre els clústers 7 i 8 (casella marcada en color blau). Pel criteri escollit a l'hora dels empats, es crearà un nou clúster format per C_7 i C_8. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2} \quad C_{1,3}) = \begin{pmatrix} 0.02 & 0.04 & 0.08 \\ 10 & 10 & 10 \\ 13 & 13 & 13 \\ 15 & 14 & 13 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 0.06 \\ 10 \\ 13 \\ 11 \end{pmatrix}$$

$$C_3 = \begin{pmatrix} 0.21 \\ 10 \\ 11 \\ 12 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 0.09 \\ 10 \\ 11 \\ 9 \end{pmatrix} \qquad C_5 = (C_{5,1} \quad C_{5,2}) = \begin{pmatrix} 0.15 & 0.06 \\ 10 & 10 \\ 8 & 8 \\ 10 & 8 \end{pmatrix}$$

$$C_6 = \begin{pmatrix} 0.09 \\ 10 \\ 8 \\ 6 \end{pmatrix} \qquad C_7 = (C_{7,1} \quad C_{7,2}) = \begin{pmatrix} 0.08 & 0.12 \\ 10 & 10 \\ 6 & 6 \\ 7 & 5 \end{pmatrix}$$

- **Criteri d'aturada**

En acabar aquesta iteració es tenen 7 clústers. Com que $s = 7$, l'algorisme finalitza.

4.2. Exercici pràctic: Algorisme 7

L'objectiu d'aquest exercici és generar dinàmicament un arbre d'escenaris amb espessor flexible. Cal mencionar que en la implementació d'aquest algorisme es van aplicar canvis, però s'explicaran més endavant. L'estructura a seguir és:

ALGORISME 7:

I. **Paràmetres.** Sigui T l'alçada desitjada de l'arbre, siguin $(b_1, \dots, b_T)$ els valors mínims de l'espessor per a cada etapa i $(d_1, \dots, d_T)$ les màximes distàncies de transport per etapes. Aquests dos vectors es fixen a l'inici.

II. **Determinar l'arrel.** El valor del procés a l'arrel és ξ_0 a l'etapa 0. Determinar l'arrel com l'actual node obert.

III. **MENTRE** hi hagi nodes oberts **FER:**

   A. Sigui k el següent node obert i sigui $t < T$ la seva etapa. Siguin $\xi_0, \dots, \xi_{t-1,}\xi_t$ els valors ja fixats en el node k i els seus predecessors. Es fixa el nombre inicial de successors de k a $s = b_{t+1}$.

   B. Cridar l'Algorisme 3 per generar s punts $z_1^*, \dots, z_s^*$ de la distribució
   $$F_{t+1}(\cdot\,|\xi_t, \xi_{t-1}, \dots, \xi_0).$$

   C. Calcular la distància $d = \mathrm{d}(F_{t+1}(\cdot\,|\xi_t, \xi_{t-1}, \dots, \xi_0), \sum_{i=1}^{s} p_i \cdot \delta_{z^{(i)}})$.

   D. Si $d > d_{t+1}$, aleshores augmentar b_{t+1} en una unitat i tornar a (B).

   E. Guardar els b_{t+1} nodes successors del node k usant els valors de z_i^* així com també les seves probabilitats condicionals p_i^* i marcar-los com a oberts.

IV. **Criteri d'aturada.** Si tots els nodes a l'etapa $T-1$ s'han considerat com a nodes enllaçats, la generació de l'arbre s'ha acabat.

ALGORISME 3:

Es té una mostra aleatòria de mida n de la distribució P. Usar un algorisme de clustering per aconseguir s clústers. Es fixa k i sigui $Z(0) = \{z^{(1)}(0), \dots, z^{(n)}(0)\}$ el clúster de les medianes, que es passarà a l'Algorisme 7.

ALGORISME 1:
• **Mostreig.** Se suposa que es tenen n punts $\{z^{(1)}, \dots, z^{(n)}\}$ en $\mathbb{R}^m$ dotat d'una mètrica d donada. Al començament, cada punt és un clúster. • **Iteració.** Se suposa que la partició actual del conjunt és $\mathbb{R}^m = \bigcup_j^{+} C_j$. L'objectiu és trobar la parella (C_j, C_k) per la qual $\sup\{\mathrm{d}(z,z'): z \in C_j, z' \in C_k\}$ és mínim. Aleshores es crea un nou clúster agrupant C_j amb C_k. • **Criteri d'aturada**. Si el nombre de clústers ha disminuït fins al nombre desitjat s, aleshores s'atura. Altrament, cal tornar al pas (II).

L'exemple que s'aplicarà és el de la creació d'un arbre de dues etapes, obtenint els valors de cada una d'elles mitjançant un generador de dades.

- **Algorisme 7: Paràmetres**

Es vol obtenir un arbre d'alçada $T = 2$. Es defineixen els vectors:

$$b = (2, 3), \qquad d = (6, 7)$$

El primer determina el nombre mínim d'arcs que han de sortir de cada node segons l'etapa i el segon la distància màxima que hi pot haver entre tots els arcs que surten d'un node.

- **Algorisme 7: Determinar l'arrel**

El valor del procés a l'arrel és $\xi_0 = 10$ a l'etapa 0. Aquest serà l'actual node obert. Es generen 4 valors aleatoris per a la primera etapa. L'arbre inicial és el següent:

Taula IV.2. Dades inicials exercici pràctic Algorisme 7.

	ETAPES	
ESCENARIS	**0**	**1**
1	10	13
2	10	11
3	10	8
4	10	6

I es fixa $s = b_1 = 2$. A continuació, s'aplicarà l'Algorisme 1 per obtenir el nombre de clústers determinats.

- **Algorisme 1**

En l'exemple que s'està treballant, es tenen un total de $n = 4$ punts, i s'ha establert obtenir $s = 2$ clústers. Com que inicialment cada punt n'és un ell mateix, la situació de partida és:

$$C_1 = \begin{pmatrix}10\\13\end{pmatrix} \qquad C_2 = \begin{pmatrix}10\\11\end{pmatrix} \qquad C_3 = \begin{pmatrix}10\\8\end{pmatrix} \qquad C_4 = \begin{pmatrix}10\\6\end{pmatrix}$$

Així doncs, es tenen 4 clústers.

ITERACIÓ 1

Cal calcular les distàncies entre els diferents clústers.

$\mathrm{d}_{i,j}$	C_1	C_2	C_3	C_4
C_1		2	5	7
C_2	2		3	5
C_3	5	3		2
C_4	7	5	2	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z,z'): z \in C_j, z' \in C_k, j \neq k\}\right) = 2$$

que és la distància que hi ha entre els clústers 1 i 2 i entre els clústers 3 i 4 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, per tant, es crearà un nou clúster format per C_1 i C_2. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2}) = \begin{pmatrix} 10 & 10 \\ 13 & 11 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 10 \\ 8 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 10 \\ 6 \end{pmatrix}$$

En finalitzar aquesta iteració es tenen 3 clústers. Com que $s = 2$, s'ha de realitzar una altra iteració.

ITERACIÓ 2

Cal calcular les distàncies entre els diferents clústers.

$\mathrm{d}_{i,j}$	C_1	C_2	C_3
C_1		5	7
C_2	5		2
C_3	7	2	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z,z'): z \in C_j, z' \in C_k, j \neq k\}\right) = 2$$

que és la distància que hi ha entre els clústers 2 i 3 (casella marcada en color blau). Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2}) = \begin{pmatrix} 10 & 10 \\ 13 & 11 \end{pmatrix} \qquad C_2 = (C_{2,1} \quad C_{2,2}) = \begin{pmatrix} 10 & 10 \\ 8 & 6 \end{pmatrix}$$

En finalitzar aquesta iteració es tenen 2 clústers. Com que $s = 2$, s'acaba l'Algorisme 1.

- **Algorisme 3**

Cal determinar el clúster de les medianes. Aquest és:

$$Z(0) = \left\{\begin{pmatrix} 10 \\ 12 \end{pmatrix}, \begin{pmatrix} 10 \\ 7 \end{pmatrix}\right\}$$

- **Algorisme 7: Iteració 1**

Ara cal calcular la distància.

$$d = (|10 - 10| + |12 - 7|) = 5$$

Com que $d = 5 < 6 = d_1$, aleshores cal guardar els valors z_i^* i crear nous valors per a l'etapa 2.

Taula IV.3. Dades fins al segon nivell exercici pràctic Algorisme 7.

	NIVELL		
ESCENARIS	**0**	**1**	**2**
1	10	12	15
2	10	12	14
3	10	12	13
4	10	12	11
5	10	7	10
6	10	7	9
7	10	7	8
8	10	7	7

En aquest cas, s'han escollit els valors de manera que l'etapa 2 depèn de les medianes obtingudes.

Ara s'escull com a node obert (10,12) i es fixa $s = b_2 = 3$. A continuació, s'aplicarà l'Algorisme 1 per obtenir el nombre de clústers determinats.

- **Algorisme 1**

En l'exemple que s'està treballant, es tenen un total de $n = 4$ punts, i s'ha establert obtenir $s = 3$ clústers. Com que inicialment cada punt n'és un ell mateix, la situació de partida és:

$$C_1 = \begin{pmatrix} 10 \\ 12 \\ 15 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 10 \\ 12 \\ 14 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 10 \\ 12 \\ 13 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 10 \\ 12 \\ 11 \end{pmatrix}$$

Així doncs, es tenen 4 clústers.

ITERACIÓ 1

Cal calcular les distàncies entre els diferents clústers.

$\mathbf{d}_{i,j}$	C_1	C_2	C_3	C_4
C_1		1	2	4
C_2	1		1	3
C_3	2	1		2
C_4	4	3	2	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z,z'): z \in C_j, z' \in C_k, j \neq k\}\right) = 1$$

que és la distància que hi ha entre els clústers 1 i 2 i entre els clústers 2 i 3 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, per tant, es crearà un nou clúster format per C_1 i C_2. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2}) = \begin{pmatrix} 10 & 10 \\ 12 & 12 \\ 15 & 14 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 10 \\ 12 \\ 13 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 10 \\ 12 \\ 11 \end{pmatrix}$$

En finalitzar aquesta iteració es tenen 3 clústers. Com que $s = 3$, s'acaba l'Algorisme 1.

- **Algorisme 3**

Cal determinar el clúster de les medianes. Aquest és:

$$Z(0) = \left\{\begin{pmatrix} 10 \\ 12 \\ 14.5 \end{pmatrix}, \begin{pmatrix} 10 \\ 12 \\ 13 \end{pmatrix}, \begin{pmatrix} 10 \\ 12 \\ 11 \end{pmatrix}\right\}$$

- **Algorisme 7: Iteració 2**

Ara cal calcular la distància.

$$d = (|14.5 - 13| + |14.5 - 11| + |13 - 11|\ ) = 7$$

Com que $d = 5\ < 7 = d_2$ i encara queden nodes oberts en l'etapa, s'escull el següent que és (10,7) i es fixa $s =\ b_2 = 3$. A continuació, s'aplicarà l'Algorisme 1 per obtenir el nombre de clústers determinats.

- **Algorisme 1**

En l'exemple que s'està treballant, es tenen un total de $n = 4$ punts, i s'ha establert obtenir $s = 3$ clústers. Com que inicialment cada punt n'és un ell mateix, la situació de partida és:

$$C_1 = \begin{pmatrix} 10 \\ 7 \\ 10 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 10 \\ 7 \\ 9 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 10 \\ 7 \\ 8 \end{pmatrix} \qquad C_4 = \begin{pmatrix} 10 \\ 7 \\ 7 \end{pmatrix}$$

Així doncs, es tenen 4 clústers.

ITERACIÓ 1

Cal calcular les distàncies entre els diferents clústers.

$\mathbf{d}_{i,j}$	C_1	C_2	C_3	C_4
C_1		1	2	3
C_2	1		1	2
C_3	2	1		1
C_4	3	2	1	

Un cop calculades les distàncies, cal buscar el mínim d'elles, és a dir:

$$\min\left(sup\{\mathrm{d}(z,z'): z \in C_j, z' \in C_k, j \neq k\}\right) = 1$$

que és la distància que hi ha entre els clústers 1 i 2, entre els clústers 2 i 3 i entre els clústers 3 i 4 (caselles marcades en color blau). Pel criteri escollit a l'hora dels empats, per tant, es crearà un nou clúster format per C_1 i C_2. Així doncs, els clústers que es tenen ara són:

$$C_1 = (C_{1,1} \quad C_{1,2}) = \begin{pmatrix} 10 & 10 \\ 7 & 7 \\ 10 & 9 \end{pmatrix} \qquad C_2 = \begin{pmatrix} 10 \\ 7 \\ 8 \end{pmatrix} \qquad C_3 = \begin{pmatrix} 10 \\ 7 \\ 7 \end{pmatrix}$$

En finalitzar aquesta iteració es tenen 3 clústers. Com que $s = 3$, s'acaba l'Algorisme 1.

- **Algorisme 3**

Cal determinar el clúster de les medianes. Aquest és:

$$Z(0) = \left\{\begin{pmatrix} 10 \\ 7 \\ 9.5 \end{pmatrix}, \begin{pmatrix} 10 \\ 7 \\ 8 \end{pmatrix}, \begin{pmatrix} 10 \\ 7 \\ 7 \end{pmatrix}\right\}$$

- **Algorisme 7: Iteració 3**

Ara cal calcular la distància.

$$d = (|9.5 - 8| + |9.5 - 7| + |8 - 7|\ ) = 5$$

Com que $d = 5\ < 7 = d_2$ i no queden nodes oberts en l'etapa 1. Com que s'ha acabat l'etapa $T - 1$, l'Algorisme 7 finalitza.

- **Criteri d'aturada**

L'arbre aconseguit al final de l'Algorisme 7 és:

Taula IV.4. Arbre final exercici pràctic Algorisme 7.

	NIVELL		
ESCENARIS	**0**	**1**	**2**
1	10	12	14.5
2	10	12	13
3	10	12	11
4	10	7	9.5
5	10	7	8
6	10	7	7

4.3. Implementació

A continuació, es presenta la implementació en AMPL de l'algorisme de generació dinàmica d'arbres d'escenari.

4.3.1. Definició de paràmetres i conjunts

Inicialment, cal definir el paràmetre que fa referència al nombre d'etapes que tenen els escenaris (*T*). A partir d'aquí, es defineixen els conjunts de les etapes (*ETAPES*), el vector que indica la mínima espessor (*b*) i el que fa referència a la distància màxima que hi pot haver entre els arcs sortint dels nodes (*d*).

```
param T default 34;
set ETAPES  := 1 .. T;
param b { ETAPES };
param d { 0 .. T };
```

També s'introdueixen un vector que indica quines etapes fan referència a la generació d'energia (*gener*), un vector que indica a quina posició comença cada etapa (*col_var*) i un vector que fa referència al nombre de columnes que té cada etapa (*nRVSG*). A més, es defineix un paràmetre que fa referència a la dimensió (*coordenades*, *columnes*) que té l'etapa que s'està estudiant (*coordnivellactual*) i el conjunt associat (*COORD*).

```
param nRVSG { 0 .. T }; #elements de cada etapa (important per vectors)
param gener { ETAPES };
param col_var { ETAPES };
param coordnivellactual, integer;
set COORD := 1 .. coordnivellactual;
```

Per altra banda, s'hi inclouen els paràmetres que fa referència al nombre d'escenaris generats (*n*), al nombre de clústers que es vol obtenir a l'aplicar l'Algorisme 1 (*s*) i els seus conjunts associats (*ESCENARIS* i *CLUSTERS*). També es defineix un paràmetre que fa referència al nombre d'observacions que tenen les dades que s'utilitzen (*aproxs*) i la matriu que les contindrà (*E*).

```
param n, integer;
set ESCENARIS  := 1 ..n;
param s, integer;
set CLUSTERS := 1 .. s;
param aproxs;
param E { 1 .. aproxs, 1 .. 206 };
```

A més, cal definir un seguit de matrius que serviran per anar guardant els resultats. Per poder-ho fer, es té un paràmetre que indica quantes files té la matriu de resultats (*fila*) i a quina columna s'està escrivint (*columna*). També es creen els conjunts associats (*FILES* i *COLUMNES*). Amb aquests elements, es crea una matriu que tindrà la probabilitat de cada node (*mat_prob*), una altra que indica segons l'etapa a quin clúster es troba (*mat_clust*), una que inclou les posicions de cada node i que s'utilitzarà a l'hora de realitzar el gràfic (*mat_grafic*) i la matriu que inclou els resultats, és a dir, el valor dels nodes de cada etapa per cada escenari (*matriu*).

```
param columna default 0;
param fila default 1;
set FILES := 1 .. fila;
set COLUMNES := 1 .. columna;
param mat_prob{ FILES, ETAPES};
param mat_clust { FILES, ETAPES };
param mat_grafic { FILES, 1 .. T+1 };
param matriu{ FILES, COLUMNES};
```

A més, s'utilitzen molts més elements, però que són auxiliars i que s'usen únicament per a l'execució de diferents processos.

4.3.2. Inicialització.

A l'inici del programa, primer s'hi introdueixen totes les definicions, els elements que s'utilitzaran i es llegeixen les dades.

```
data ("20181219_10000_" & jj &".txt");
data Algoritme7.dat;
```

En el primer *data* s'hi troben les observacions de cada etapa. Aquests documents estaven generats anteriorment, a partir d'un *script* d'R, i contenien un total de 10000 casos. L'element *jj* fa referència al dia que s'està simulant ja que, com s'ha explicat anteriorment, es volen aconseguir un conjunt d'arbres d'escenaris per un nombre concret de dies.
En el document *.dat*[3] hi ha els valors que pren el vector *b*, *gener*, *col_var* i *nRVSG*.
Aleshores, per cada node de l'última etapa generada es defineixen els valors *s* i *n*, és a dir, el nombre de clústers a obtenir i els escenaris que cal simular.

```
let s:= b[e];
let n:= s + 4;
```

Es va decidir que el nombre d'escenaris que calia obtenir fos l'espessor mínima de l'etapa, és a dir, el mínim nombre d'arcs que han de sortir d'un node més 4.
Un cop decidit quants escenaris cal aconseguir, és necessari diferenciar si l'etapa que s'està estudiant fa referència a la generació d'energia eòlica o bé a preus de mercat, ja que la forma d'obtenir les observacions és diferent.

[3] Observar Annex 2.

Si l'etapa actual és de preus, s'escullen aleatòriament *n* valors entre 1 i 10000 segons una distribució Uniforme i, aleshores, se seleccionen les observacions que es troben en aquestes posicions.

```
for{i in ESCENARIS}{
 let vect_aleat[i] := round(Uniform(1, aproxs));
}
let {i in ESCENARIS, j in COORD} valors[i,j] := E[vect_aleat[i], antcol + j];
```

L'element *vect_aleat* emmagatzema les posicions que s'escullen i la matriu *valors* les observacions seleccionades.

Per altra banda, si l'etapa fa referència a generació eòlica cal considerar les etapes anteriors que també tenen la mateixa naturalesa. Per aquest motiu, hi ha una matriu definida, *mat_gen*, que conté aquesta informació. Aquest procés cal fer-lo ja que abans de començar aquest projecte s'havia comprovat que les diferents etapes de generació estaven relacionades entre elles. Així doncs, i per poder-ho aconseguir, només se seleccionen aquelles que es troben dins d'un interval definit en funció de les etapes anteriors.

```
let naprox := 0;
let aleat := round(Uniform(1, aproxs));
repeat while naprox < n {
	let trobat := 1;
	let etapa_gener := 5;
	repeat while trobat = 1  and etapa_gener < e {
		if gener[etapa_gener] > 0 then {
			if E[aleat,col_var[etapa_gener]] > mat_gen[pos_escriure,
			2*gener[etapa_gener]] or E[aleat,col_var[etapa_gener]] <
			mat_gen[pos_escriure, 2*gener[etapa_gener]-1] then {
				let trobat := 0;
			}
		}
		let etapa_gener := etapa_gener + 1;
	}
	if trobat = 1 then {
		let naprox := naprox + 1;
		let valors[naprox,1] := E[aleat,col_var[e]];
	}
	if aleat = aproxs then {
		let aleat := 1;
	} else {
		let aleat := aleat + 1;
	}
}
```

Un cop obtinguts els *n* escenaris, cal definir la distància màxima que hi ha d'haver entre els clústers que s'obtinguin. En la definició de l'algorisme, aquest és un paràmetre que es defineix prèviament però, en aquest cas, es va decidir aplicar una modificació, perquè no es trobava una manera adient de definir aquest valor.

Per fer-ho, es va calcular el màxim de les distàncies niades entre els *n* escenaris obtinguts i cada un d'ells.

```
for{i in 1 .. n-1}{
	for{j in i+1 .. n}{
		let suma := 0;
		for{k in COORD}{
			let suma := suma + (valors[i,k] - valors[j,k])^2;
		}
		let distancies[i,j] := suma^(1/2);
		let distancies[j,i] := distancies[i,j];
	}
	let distancies[i,i] := 0;
}
let distancies[n,n] := 0;
if gener[e] > 0 then {
	let max_dist[e] := erel_gen*(max{ i in ESCENARIS } (sum{j in
	ESCENARIS} distancies[i,j]))/n;
} else {
	let max_dist[e] := erel_preu*(max{ i in ESCENARIS } (sum{j in
	ESCENARIS} distancies[i,j]))/n;
}
let dist_a7 := max_dist[e] + 1;
```

En el primer bucle, es calcula la taula de les distàncies euclidianes, dels *n* escenaris entre ells. Un cop obtinguda, cal diferenciar entre els casos de generació d'energia eòlica (el valor del vector gener és major a 0) o del cas dels preus (el valor de gener és igual a 0). Això és degut a que el paràmetre *erel* té un valor diferent en cada cas. Aquest paràmetre serveix per modificar el nombre final d'escenaris.

Finalment, l'Algorisme 7 es realitzarà mentre la distància entre els clústers obtinguts sigui major al valor *dist_a7*.

4.3.3. Implementació Algorisme 1

Inicialment, cal crear la taula en la qual s'hi indiquin les distàncies entre les diferents observacions de cada clúster, tal i com s'han realitzat en l'exemple anterior d'aquest algorisme i aplicant-hi la mateixa distància.

```
for{i in 1 .. n-1}{
	for{j in i+1 .. n}{
		let suma := 0;
		for{k in COORD}{
			let suma := suma + (valors[i,k] - valors[j,k])^2;
		}
		let distancies[i,j] := suma^(1/2);
		let distancies[j,i] := distancies[i,j];
	}
	let distancies[i,i] := 0;
}
let distancies[n,n] := 0;
```

A continuació, cal buscar la distància mínima de totes les calculades, a més de determinar entre quins clústers hi ha la distància mínima. Per aconseguir-la, primer es dona un valor elevat a la distància mínima i, seguidament, es va comparant cada distància amb aquest valor. Si la distància que s'està escollint en la iteració és menor al valor mínim, es modifica el valor de la distància mínima i es guarden els clústers (fila i columna de la matriu) entre els quals hi ha la distància mínima.

```
let minim_dist := 10000000;
for{i in 1 .. n-1}{
	for{j in i+1 .. n}{
		if distancies[i,j] < minim_dist and distancies[i,j] != 0 then {
			let minim_dist := distancies[i,j];
			let ci := i;
			let cj := j;
		}
	}
}
```

El següent pas és l'actualització dels clústers, ajuntant els dos entre els quals hi ha la mínima distància.

```
let clust_anter := clust[cj];
for{i in ESCENARIS}{
	if clust[i] = clust_anter then {
		let clust[i] := clust[ci];
	}
}
```

També cal modificar la matriu de les distàncies ja que, si cal fer noves iteracions de l'Algorisme 1 per aconseguir el nombre desitjat de clústers, no sigui necessari calcular un altre cop aquest element. Per fer-ho, es dona el valor de 0 a les distàncies entre punts del mateix clúster i, en els altres casos, es calcula el mínim de les distàncies entre cada clúster.

Tots els passos que s'han fet fins aquest punt de l'algorisme s'han de repetir tantes vegades com siguin necessaris per aconseguir el nombre de clústers desitjat (que és el de valor *s*).

Finalment, l'última acció que cal realitzar és crear un vector de totes les observacions on s'indiqui a quin clúster pertany.

```
for{i in CLUSTERS}{
	let minim_clust := n + 1;
	for{j in ESCENARIS}{
		if clust[j] >= i and clust[j] < minim_clust then let
		minim_clust := clust[j];
	}
	for{j in ESCENARIS}{
		if clust[j] = minim_clust then let clust[j] := i;
	}
}
```

4.3.4. Implementació Algorisme 3

Tal i com s'ha dit anteriorment, s'ha modificat l'ús d'aquest algorisme i només es fa servir per calcular el vector amb les medianes de cada clúster, que és el que s'utilitzarà com a valors per a l'arbre d'escenari que s'està generant.

Per poder-ho aconseguir, cal considerar que AMPL no té una funció que calculi la mediana. Així doncs, primer cal ordenar les dades i després calcular aquest percentil. Per obtenir les medianes de cada clúster, inicialment cal saber quants casos constitueixen cada grup. A partir del vector creat anteriorment, que conté a quin conjunt pertany cada observació, es crea un vector nou que contingui quantes observacions té cada un dels clústers.

```
for{i in CLUSTERS}{
      for{j in ESCENARIS}{
            if clust[j] = i then {
                  let n_punts[i] := n_punts[i] + 1;
            }
      }
}
```

Seguidament, cal ordenar els valors dels clústers, pas imprescindible per aconseguir calcular la mediana. Dins de cada grup es va seleccionant el valor menor que encara no s'ha escollit i, d'aquesta manera, es crea un vector ordenat creixentment. Per saber si un valor s'ha seleccionat, a l'última columna de l'objecte *ultim_nivell* s'hi posa un 0, si l'element no s'ha seleccionat, o un 1, en el cas que sí.

```
let posicio := 1;
for{i in CLUSTERS}{
      for{j in 1 .. n_punts[i]}{
            let minim := max{ f in ESCENARIS } ultim_nivell[f,co];
            for{l in ESCENARIS}{
                  if ultim_nivell[l,co] <= minim and clust[l] = i and
                  ultim_nivell[l,coordnivellactual+1] = 0 then {
                        let minim := ultim_nivell[l,co];
                        let cluster := l;
                  }
            }
            let vect_ordenat[posicio,co] := minim;
            let ultim_nivell[posicio,coordnivellactual+1] := 1;
            let posicio := posicio + 1;
      }
}
let {esc in ESCENARIS} ultim_nivell[esc,coordnivellactual+1] := 0;
```

Finalment, es calculen les medianes de tots els clústers, que seran les observacions que s'utilitzaran per crear l'arbre d'escenaris final. Per calcular aquest quantil, cal considerar si el nombre de dades és senar o parell. En el primer cas, la mediana és el nombre que es troba en la posició central. En l'altre, la mediana és la mitjana dels dos nombres centrals.

```
let posicio := 0;
for{i in CLUSTERS}{
      if n_punts[i] mod 2 = 0 then {
```

```
                let median[i,j] := (ultim_nivell[posicio + n_punts[i]/2, j] +
                ultim_nivell[posicio + n_punts[i]/2 + 1, j])/2;
        } else {
                let median[i,j] := ultim_nivell[posicio + n_punts[i] div 2 + 1,
                j];
        }
        let posicio := n_punts[i] + posicio;
}
```

4.3.5. Implementació Algorisme 7

Un cop es té el vector amb les medianes dels s clústers, cal calcular la distància de transport entre les *s* dades obtingudes i les *n* observacions inicials. Per aconseguir-ho, es calcula la distància euclidiana entre tots aquests valors, de manera que s'obté una matriu amb *n* files i *s* columnes.

```
for{i in ESCENARIS}{
        for{j in CLUSTERS}{
                let suma := 0;
                for{k in COORD}{
                        let suma := suma + (valors[i,k] - median[j,k])^2;
                }
        let dist_trans[i,j] := suma^(1/2);
        }
}
```

El següent pas és calcular la distància niada. Per obtenir-la, es diferencien dos casos:

I. Si $s = 1$, per calcular aquesta distància no és necessari aplicar el model de programació lineal i es pot resoldre manualment, utilitzant l'Observació 2.2.4.

```
let dist_a7 := (sum{i in ESCENARIS} dist_trans[i,1])/n;
```

II. Si $s \geq 2$, aleshores s'utilitza el fitxer *.mod* que conté la implementació del problema de programació lineal de la distància niada.

```
option solver cplex;
option solver_msg 0;
solve;
let dist_a7 := z;
```

Un cop es té calculada aquesta distància, cal observar si és inferior o superior a la que s'havia establert inicialment com a màxima distància que hi pot haver entre els arcs sortint d'un node. Si la distància és superior s'augmenta en 1 el nombre d'arcs que poden sortir del node obert que s'està estudiant.

```
let s := s + 1;
```

Si la distància és inferior o *s* és el nombre d'observacions escollides a l'inici (*n*), es passa al següent node obert.

4.3.6. Emmagatzematge del resultats en matrius

A continuació, cal guardar les dades obtingudes. En aquest procés es modifiquen els següents objectes:

- *matriu*: s'hi inclouen els valors que s'utilitzen per a la creació de l'arbre d'escenaris.
- *mat_prob*: s'hi inclou la probabilitat que té cada dada. Aquesta probabilitat és 1/*s*.
- *mat_clust*: s'hi inclou el clúster al qual pertany cada dada.

A part d'aquestes tres matrius, també es modifica la *mat_gen*. Aquest element és molt important en les etapes que tracten etapes de generació d'energia eòlica. Quan les dades que cal guardar són d'aquest tipus, es creen dues columnes noves a la matriu, que formaran els valors inferior i superior d'un interval. Aquest és el que s'observa en etapes posteriors per saber quines observacions cal escollir per tal que hi hagi relació entre les diferents etapes de generació.

```
let {i in CLUSTERS} mat_gen[i + pos_escriure - 1, 2*gener[e] - 1] :=
median[i,1] - 5*max_dist[e];
let {i in CLUSTERS} mat_gen[i + pos_escriure - 1, 2*gener[e]] := median[i,1]
+ 5*max_dist[e];
```

Un cop s'han guardat els resultats, es prossegueix a estudiar el següent node obert i, així, iterativament fins arribar a l'última etapa.

4.3.7. Creació de documents

Un cop s'ha obtingut l'arbre d'escenaris cal guardar alguns dels objectes en diferents documents, necessaris per a la representació dels arbres o per a la implementació de futurs processos.
Primer, es calcula un vector que conté la probabilitat de cada escenari. A partir de la matriu *mat_prob*, es multipliquen els valors de totes les columnes per cada fila i, d'aquest manera, s'obté la probabilitat esperada.

```
for{i in FILES}{
	let prob_final[i] := 1;
	for{j in ETAPES}{
		let prob_final[i] := prob_final[i]*mat_prob[i,j];
	}
}
```

A continuació, es crea un document *.dat* que contindrà l'arbre d'escenaris i altres dades rellevants. El nom d'aquest serà:

data _ número observacions inicials _ número d'arbre.dat

Aquest arxiu contindrà:

- *erel_gen*: *erel* per a les etapes de generació.

- *erel_preu*: *erel* per a les etapes de preus.
- *nS*: número d'escenaris.
- *nRVSG*: espessor mínima de cada etapa.
- *Scen0*: els valors de l'arbre d'escenaris.
- *Prob0*: el vector amb les probabilitats de cada escenari.
- Uns conjunts que indiquen quins escenaris surten de cada node.

Seguidament, es crea el document *.txt* següent:

stock_results_ *número d'arbre.*txt

L'arxiu contindrà els valors que s'utilitzaran per a la representació de l'arbre, continguts en la matriu *mat_grafic*. Aquest element s'explicarà en el següent apartat.
Finalment, es crea el document *.txt* següent:

tree_parameters_ *número d'arbre.*txt

L'arxiu ha de tenir el nombre d'escenaris i d'etapes.

4.3.8. Representació

Per poder realitzar el gràfic, cal crear la matriu *mat_grafic*, que conté les posicions de cada node.
El primer pas és indicar la posició del node arrel. Aquest se situarà a una alçada que tindrà el valor de la meitat del nombre de nodes.

```
for{i in FILES}{
      let mat_grafic[i,1] := nnodes[T+1]/2;
}
```

A continuació, es van calculant les següents etapes. Si el nombre d'arcs (es notarà *n_clusts*) que surt d'un node és senar, aleshores la representació tindrà un arc que serà horitzontal, (*n_clusts*-1)/2 que aniran amunt i el mateix nombre que aniran avall. En canvi, si el nombre d'arcs que surt del node és parell, aleshores la representació tindrà *n_clusts*/2 que aniran amunt i el mateix nombre que aniran avall. En el cas que només surti un node es manté el valor, de manera que gràficament es veurà una línia horitzontal.
La distància que se sumarà o restarà als nous nodes serà una proporció de la distància que hi ha entre el node de l'etapa, que s'està estudiant, i el primer predecessor que fa que aquesta sigui diferent de 0.
Un cop obtinguda la matriu amb les posicions dels diferents nodes, es guarda aquest element en el document *stock_results_ número d'arbre.*txt.
A partir d'aquest arxiu i del *tree_parameters_ número d'arbre.*txt es realitza el gràfic de l'arbre, aplicant l'*script* arbol-Rnuevo180911.R, el qual primer dibuixa els diferents nodes i després els arcs que els uneixen.

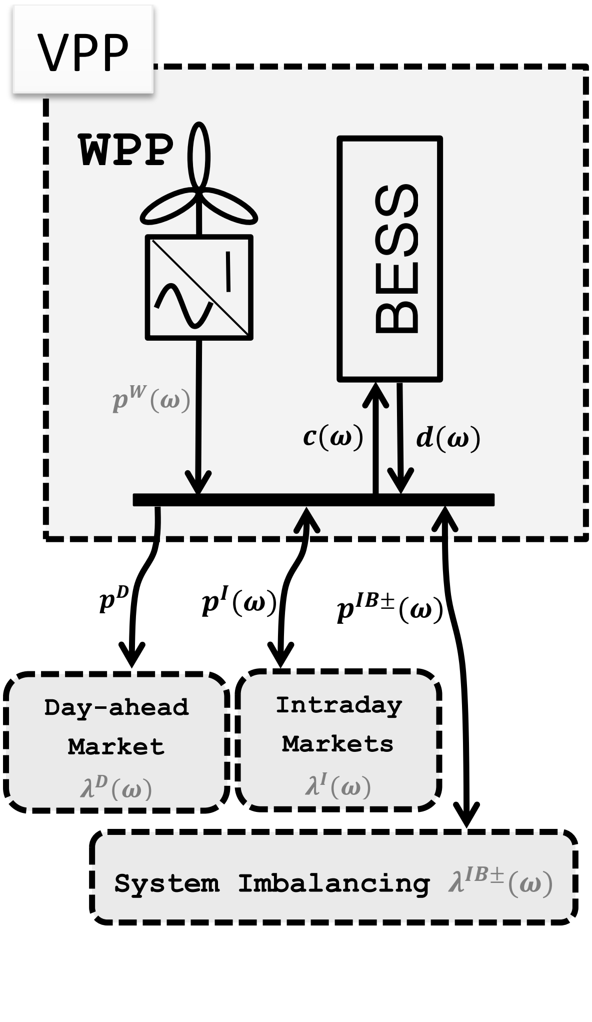
CAPÍTOL V: Resultats i interpretació

L'objectiu d'aquest capítol és presentar els resultats obtinguts en la implementació dels algorismes anteriors després d'haver-hi aplicat les dades del projecte, així com de la interpretació d'aquests en el mercat d'energia eòlica.

Cal mencionar que, per tal de no complicar excessivament la descripció només es presentarà i analitzarà el cas sense el mercat de reserva, malgrat tenir els seus preus inclosos dins de l'arbre. Tanmateix, els arbres obtinguts en aquest projecte permeten resoldre satisfactòriament el problema considerant també el mercat de reserva.

Així doncs, la situació que s'estudia és la d'una planta virtual d'energia (VPP) amb els mercats diari i intradiari (VDI, *Virtual power plant with Day-ahead and Intra-day market*).

Figura V.1. Paràmetres aleatoris (gris) i variables de decisió d'una VPP sense considerar mercat de reserva.



5.1. Les dades

Un cop es va tenir tot l'algorisme implementat, es va prosseguir a aplicar els fitxers amb les dades.

Les dades a usar per aconseguir els arbres d'escenaris es troben en uns documents *.txt* que contenen un total de 10000 observacions per 34 etapes. Algunes d'aquestes etapes fan referència a un sol valor (una columna) i d'altres són un vector (més d'una columna, segons la llargada de l'element). Així doncs, es té una matriu de 10000 files i un total de 206 columnes.

Cal matisar que a l'inici del projecte es tenien 1000 observacions en cada arxiu, però en veure que en alguns casos s'obtenien arbres amb una amplitud major, es va decidir augmentar la dimensió de la matriu de dades inicial.

Cada una d'aquestes etapes fa referència a la generació d'energia eòlica o a preus dels diferents mercats i cada arxiu representa un dia (per aquest motiu es tenen 24 etapes de generació eòlica, una per cada hora). Aquelles etapes que facin referència a la generació, les dades s'expressaran en [*MWh*] i, en els casos que s'estiguin tractant preus, les unitats seran de [*€/MWh*].

L'explicació de cada una de les etapes ve definida en la taula següent:

Taula V.1. Explicació de les dades a partir de les quals s'obtindran els arbres d'escenaris.

Etapa	**Informació**	**Nº columnes**	**Significat**
1	λ^{D}	24	Preus del DM[4]
2	λ^{R}	24	Preus del RM[5]
3	λ^{I1}	24	Preus del IM1[6] (primer intradiari)
4	λ^{I2}	24	Preus del IM2 (segon intradiari)
5	p_1^W	1	Generació eòlica a l'hora 1
6	p_2^W	1	Generació eòlica a l'hora 2
7	λ^{I3}	20	Preus del IM3 (tercer intradiari)
8	p_3^W	1	Generació eòlica a l'hora 3
9	p_4^W	1	Generació eòlica a l'hora 4
10	p_5^W	1	Generació eòlica a l'hora 5
11	λ^{I4}	17	Preus del IM4 (quart intradiari)
12	p_6^W	1	Generació eòlica a l'hora 6
13	p_7^W	1	Generació eòlica a l'hora 7
14	p_8^W	1	Generació eòlica a l'hora 8

[4] Mercat diari.
[5] Mercat de reserva.
[6] Mercat intradiari.

15	p_9^W	1	Generació eòlica a l'hora 9
16	λ^{I5}	13	Preus del IM5 (cinquè intradiari)
17	p_{10}^W	1	Generació eòlica a l'hora 10
18	p_{11}^W	1	Generació eòlica a l'hora 11
19	p_{12}^W	1	Generació eòlica a l'hora 12
20	p_{13}^W	1	Generació eòlica a l'hora 13
21	λ^{I6}	9	Preus del IM6 (sisè intradiari)
22	p_{14}^W	1	Generació eòlica a l'hora 14
23	p_{15}^W	1	Generació eòlica a l'hora 15
24	p_{16}^W	1	Generació eòlica a l'hora 16
25	p_{17}^W	1	Generació eòlica a l'hora 17
26	p_{18}^W	1	Generació eòlica a l'hora 18
27	p_{19}^W	1	Generació eòlica a l'hora 19
28	λ^{I7}	3	Preus del IM7 (setè intradiari)
29	p_{20}^W	1	Generació eòlica a l'hora 20
30	p_{21}^W	1	Generació eòlica a l'hora 21
31	p_{22}^W	1	Generació eòlica a l'hora 22
32	p_{23}^W	1	Generació eòlica a l'hora 23
33	p_{24}^W	1	Generació eòlica a l'hora 24
34	λ^{IB}	24	Preus del IB[7]
		206	

5.2. Resultats

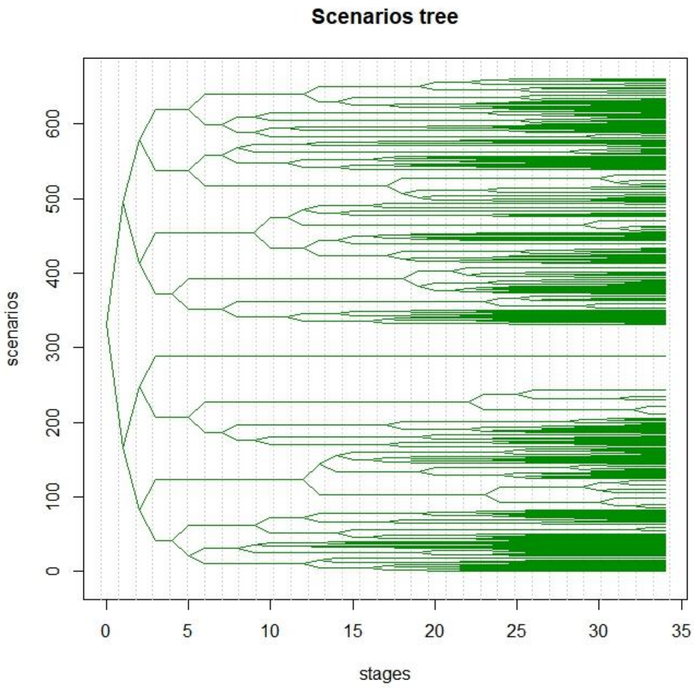
L'algorisme es va executar dues vegades, utilitzant un nombre diferent de dies (arxius). En la primera es van executar 14 fitxers, els arbres dels quals es troben inclosos a l'Annex 3 d'aquest projecte. En la segona es van executar 90 dies, amb els que es van obtenir les dades necessàries per aconseguir els gràfics del següent apartat i que, en etapes futures del projecte, serviran per poder realitzar la comparació d'aquests algorismes aplicats amb altres que també s'han usat.

Per tant, i fent referència al primer cas, es va executar l'*script* per als 14 arxius, de manera que es van obtenir 14 fitxers de *stock_results*, *tree_parameters_* i l'arxiu *.dat* amb l'arbre d'escenaris. A continuació, cal aplicar els dos primers documents en el programa *.R* per obtenir el gràfic de l'arbre d'escenaris.

Seguidament s'inclou el gràfic del primer dia, on es pot observar com les dades obtingudes acaben formant una estructura d'arbre; la resta es troben a l'Annex 3.

[7] Preus deguts als desviaments.

Figura V.2. Arbre d'escenaris pel dia 1.



Així doncs, amb els arxius obtinguts a partir de l'execució de l'algorisme i els gràfics realitzats posteriorment, s'ha aconseguit l'objectiu principal d'aquest projecte, que era la generació dinàmica d'arbres d'escenari.

5.3. Maximització de guanys

Cal aclarir que la intenció no és fer una descripció detallada d'un model elèctric, perquè queda fora de l'abast d'aquest projecte, sinó que es donaran unes petites indicacions per així poder interpretar els resultats obtinguts.

Abans de realitzar la interpretació dels arbres obtinguts, cal plantejar el context plantejat com un problema d'optimització. Fent referència a la Figura V.1., es té un node principal (que és la barra horitzontal que es troba al mig del gràfic) i les fletxes són els arcs (el flux). Les variables de decisió d'aquest model són les que estan escrites en color negre.

Així doncs, el node principal pot rebre energia (les fletxes que hi arriben) o en pot treure (les fletxes que en surten). Per poder representar més clarament com es mou l'energia, s'utilitzarà la següent taula:

Taula V.2. Explicació de les fluctuacions d'energia en el model.

ENERGIA ENTRANT	ENERGIA SORTINT
- $p^W(\omega)$: energia produïda per la central eòlica. - $p^I(\omega)$: energia que es compra al mercat intradiari. - $d(\omega)$: energia que es descarrega de la bateria.	- p^D: energia que es ven al mercat diari. - $p^I(\omega)$: energia que es ven al mercat intradiari. - $c(\omega)$: energia que es carrega a la bateria.

Cal especificar que totes les variables es mesuren en $[MWh]$.
A partir d'aquestes variables, es té la següent equació d'equilibri:

$$p_{t,s}^{IB} = p_{t,s}^{IB+} - p_{t,s}^{IB-} = \left(p_{t,s}^{W} + \Delta t \cdot d_{t,s}\right) - \left(p_{t,s}^{D} + p_{t,s}^{I} + \Delta t \cdot c_{t,s}\right)$$

on a la primera part de l'equació hi ha la quantitat d'energia que arriba al node i, a la segona, la que en surt. Es pot veure, però, que només surt una vegada el valor de l'energia relacionada amb el mercat intradiari. Si aquest és positiu, implica que la sortida és major que l'entrada i, en el cas negatiu, que l'entrada és major que la sortida.
La situació teòrica seria que el resultat fos 0, però s'admet que hi hagi un desequilibri (el valor $p^{IB}(\omega)$). A partir d'aquí, es poden diferenciar dos casos:

- $p^{IB}(\omega) > 0$, és a dir, desviament positiu. Aleshores $p^{IB+}(\omega) > 0$ i $p^{IB-}(\omega) = 0$. En aquest cas, la producció d'energia ha estat superior a la quantitat d'energia casada.
- $p^{IB}(\omega) < 0$, és a dir, desviament negatiu. Aleshores $p^{IB+}(\omega) = 0$ i $p^{IB-}(\omega) > 0$. En aquest cas, la producció d'energia ha estat inferior a la quantitat d'energia casada.

Un cop definides les diferents quantitats d'energia, ja es pot plantejar el model WPP+BESS d'una planta d'energia virtual (WBVPP, *WPP+BESS Virtual Power Plant model*). L'objectiu d'aquest problema d'optimització és maximitzar l'esperança matemàtica dels guanys de la planta eòlica en la venda de l'energia, tal i com es defineix en la següent equació:

$$EP^{VPP}(p^D, p^I, p^{IB+}, p^{IB-}) = DM(p^D) + IM(p^I) + IB^+(p^{IB+}) - IB^-(p^{IB-}).$$

Cada valor és:

- $DM(p^D)$: guanys obtinguts amb la venda d'energia al mercat diari. El preu en aquest cas és λ^D.
- $IM(p^I)$: guanys obtinguts en la venda d'energia als mercats intradiaris. El preu en aquest cas és λ^I.
- $IB^+(p^{IB+})$: guanys obtinguts amb els desviaments positius. El preu en aquest cas és λ^{IB+}.
- $IB^-(p^{IB-})$: pèrdues obtingudes amb els desviaments negatius. El preu en aquest cas és λ^{IB-}. Com que en aquest cas s'està parlant d'uns diners que ha de pagar l'empresa per

no haver produït l'energia casada, aquesta quantitat s'ha de restar dels guanys obtinguts.

Així doncs, totes les λ que apareixien en l'anterior gràfic fan referència als preus de l'energia eòlica en els diferents mercats. S'expressen en [*€/MWh*].
El resultat que s'obtingui en el model d'optimització serà una quantitat expressada en €.

5.4. Interpretació

A continuació, es presentaran un seguit de gràfics que ajudaran a la interpretació dels arbres d'escenaris aconseguits anteriorment. Cal especificar que les següents representacions no s'han realitzat com a part d'aquest projecte, sinó que només s'inclou la interpretació per poder visualitzar l'aplicació que tenen els resultats obtinguts en la implementació dels diferents algorismes.

5.4.1. Informació diària

La interpretació dels gràfics que es presentaran a continuació està relacionada amb les plantes d'energia eòlica (WPP) amb sistemes d'emmagatzematge d'energia usant una bateria[8]. Les variables que es representen són les variables de decisió del model estocàstic (VPP), i no són les mateixes que s'obtenen en els algorismes implementats, és a dir, aquestes dades no es poden observar en cap de les etapes dels arbres d'escenaris.
Com es podrà observar, per cada dia es tindran quatre gràfics, que representaran les dades en funció del temps (en hores, eix de les X). Les interpretacions de cada un d'ells són les següents:

- Gràfic a: permet veure com evolucionen les ofertes al mercat diari i intradiari segons la generació eòlica.
  La línia de color verd fa referència a l'esperança matemàtica de generació eòlica, la línia blava contínua a l'oferta del mercat diari i la blava discontínua a la suma de l'oferta del mercat diari més la mitjana del mercat intradiari (que s'anomenarà energia total casada). Aquesta última línia és l'esperança matemàtica del que es vendrà realment.
- Gràfic b: permet veure com evolucionen els desviaments.
  Un desviament positiu implica que s'ha produït més energia de l'energia total casada (mercat diari més intradiari), mentre que un desviament negatiu vol dir que s'ha produït menys quantitat de l'energia casada. Econòmicament, un desviament negatiu

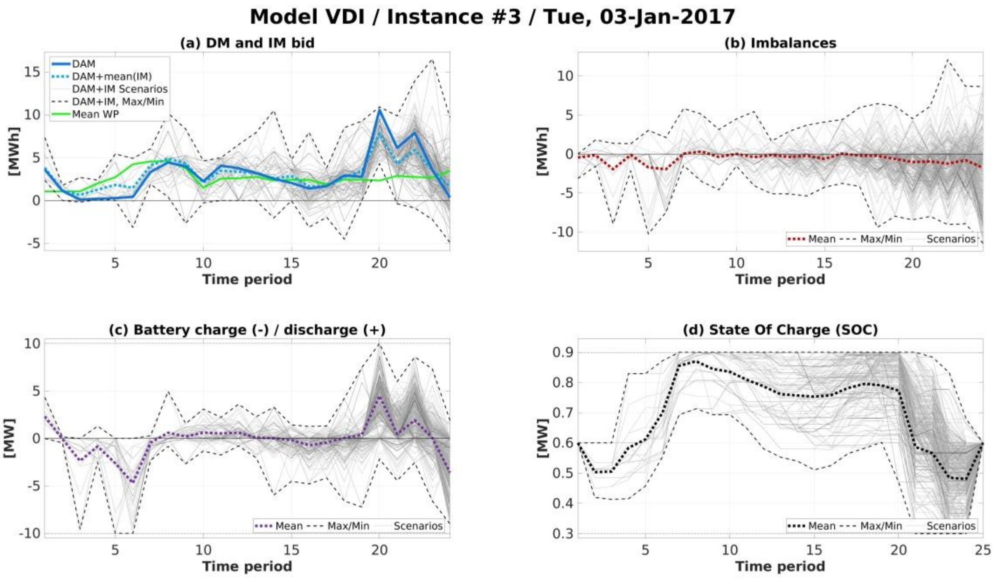
[8] Consultar Capítol I.

comporta a l’empresa una penalització i un de positiu comporta que l’empresa possiblement vengui l’energia que s’excedeix a un preu menor.

- Gràfic c: permet veure les càrregues i descàrregues de la bateria.
  La línia discontínua de color lila indica quina quantitat d’energia s’està carregant o descarregant de la bateria. Si aquesta línia es troba per sota de 0, vol dir que s’està carregant, mentre que si és superior, implica que la bateria s’està descarregant.
- Gràfic d: permet veure quin és el percentatge d’ús de la bateria.
  Aquest gràfic és redundant amb l’anterior i permet observar quin és l’estat de càrrega (*State Of Charge*), que és el percentatge d’energia emmagatzemada a la bateria respecte de la seva capacitat màxima (línia discontínua de color negre). Així, si en l’anterior gràfic s’observa que la bateria s’està carregant, en aquest es veurà com l’estat de càrrega augmenta, i viceversa. Per convenció, la bateria ha de començar i acabar el dia al 60% i mai es pot omplir al 100% ni deixar-la completament buida (0%) per evitar escurçar la seva vida útil.

Cal matisar, com es veurà, que en tots aquests gràfics es veuen representades un conjunt de línies de color gris, que són els valors de les variables de decisió. El nombre total d’aquestes línies és el mateix que les branques que tingui l’arbre d’escenaris del dia estudiat. Les línies negres discontínues fan referència al màxim i al mínim de les observacions, i serveixen per veure el rang en el qual es mouen les variables de decisió.
A continuació, s’inclou el gràfic dels resultats aconseguits pel dimarts 3 de gener de 2017. A l’Annex 4 es poden observar les representacions dels 7 primers dies d’aquell any.

Figura V.3. Resultats dimarts 3 de gener de 2017.



Gràcies a aquests gràfics es pot observar com a les primeres hores del dia es comprava energia, justament quan el preu era més baix (de 2:00 a 6:00). A més, la venda d’energia eòlica va ser menor a la generació, es va carregar la bateria i, com a conseqüència, l'estat de càrrega de la bateria va augmentar. Tot i així, cal matisar que just començar el dia la situació era al revés, motiu pel qual l'estat de càrrega de la bateria havia disminuït. Seguint en aquest període, es pot observar com hi ha desequilibris negatius. Aquest fet podria ser degut a que l'empresa aprofita per carregar la bateria malgrat no arribar a subministrar tota l'energia casada.

A partir de les sis del matí i fins a les nou del vespre, la situació s'estabilitza i s'aprofita per anar carregant la bateria (cal recordar que aquesta ha de tancar el dia a un 60% de la seva capacitat). Es veu també com, en aquest període, els desviaments són pràcticament nuls.

Finalment, es pot observar com a les nou del vespre (hora punta) l'energia casada augmenta considerablement, motiu pel qual la bateria es descarrega. A més, és l'hora en la qual l'energia va més cara.

Cal especificar que els gràfics poden canviar molt entre dies diferents, ja que el mercat varia segons el dia, depenent de moltes altres variables.

5.4.2. Generació boxplots[9]

A part de les representacions diàries individuals, també s'han realitzat uns boxplots per poder visualitzar altres aspectes dels resultats obtinguts. Tal i com s'ha especificat anteriorment, la generació d'aquests gràfics no forma part d'aquest projecte, però sí que s'explicarà què es visualitza i com s'ha calculat.

En el primer gràfic s'hi representen els valors òptims diaris obtinguts en el problema d'optimització estudiat (RP, *Expected value of the profit, recourse problem*), és a dir, el valor del model que té per funció objectiu:

$$EP^{VPP}(p^D, p^I, p^{IB+}, p^{IB-}) = DM(p^D) + IM(p^I) + IB^+(p^{IB+}) - IB^-(p^{IB-}).$$

Així doncs, s'hi representa l'esperança matemàtica del que es guanyaria cada dia per la venda d'energia eòlica.

Per altra banda, es vol observar el percentatge de millora en el valor òptim entre utilitzar un arbre d'escenaris, el cas que s'està tractant, i fer servir el valor de les variables aleatòries proporcionat pel mateix model de previsió usat en la generació dels arbres d'escenaris. Per aconseguir-ho, cal seguir els passos que s'explicaran a continuació.

Primer és necessari calcular el valor dels guanys (valor òptim del problema) sota la hipòtesi que les prediccions són exactes, és a dir, que els preus de mercat fossin els mateixos als de la predicció. Aquests valors són el resultat del problema de previsió (FV, *Forecasted Value*) utilitzant ξ_d^F, que són les previsions pel dia d, és a dir, el valor de les variables aleatòries a partir de les prediccions del dia d.

[9] Diagrama de caixa.

Però les previsions no són exactes, ja que el preu de mercat no és sempre el mateix, va variant. Així doncs, i utilitzant els valors obtinguts amb la previsió, aquests s'apliquen als resultats aconseguits utilitzant els escenaris de cada dia, és a dir, es pren l'oferta òptima obtinguda amb les previsions conjuntament amb els valors dels preus dels mercats obtinguts dels arbres, per calcular els guanys en la venda de l'energia. Com que cada escenari té una probabilitat, aleshores se'n fa l'esperança del benefici en el dia (EFV, *Expectation of the Forecasted Value*).

A partir d'aquí s'obté el que es buscava, que és el percentatge de millora que s'aconsegueix utilitzant escenaris en lloc del valor esperat (FVSS, *Forecasted Value of the Stochastic Solution*) pel dia d:

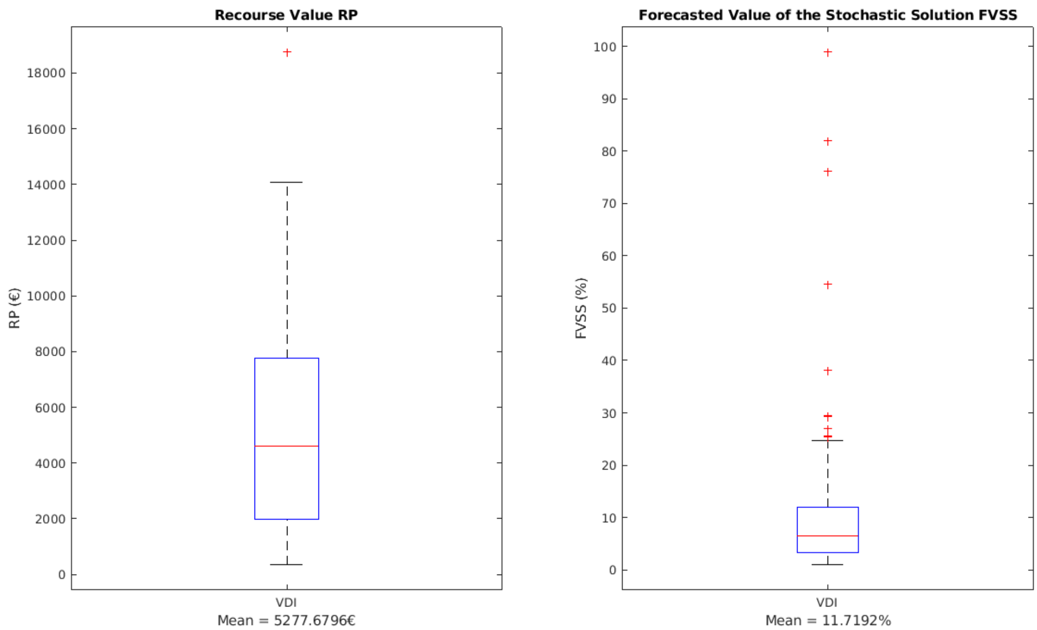
$$FVSS_d = \frac{RP_d - EFV_d}{EFV_d} \times 100.$$

L'objectiu és veure que s'obtenen millors resultats usant arbres d'escenaris que no pas prediccions, és a dir, la millora dels beneficis utilitzant un model estocàstic en lloc d'un model determinista.

5.4.3. Informació 90 dies

A partir dels resultats aconseguits en els 90 dies, els boxplots obtinguts són els següents:

Figura V.4. Boxplots obtinguts amb les dades dels 90 dies.



En el primer, es pot observar com en un 50% dels dies els guanys es troba entre els 2000 € i 8000 € (els límits de la capsa). De fet, també es podria dir que un 50% dels dies es guanya menys de 5000 €, i que la resta de dies els guanys són superiors. A més, la mitjana d'aquests

valors és de 5277.68 €. Visualitzant aquest gràfic, es podria concloure també que la distribució dels guanys té asimetria a la dreta (asimetria positiva).
Fent referència al percentatge de millora en els guanys utilitzant els valors dels arbres, es pot observar com en un 75% dels casos està entre el 2% i el 12%, és a dir, en 68 dies s'obté que els guanys en la venda d'energia eòlica és entre un 2% i 12% superior utilitzant arbres d'escenaris. En aquest cas, la mitjana no és una mesura representativa ja que hi ha un elevat nombre d'*outliers*[10].

[10] Dada de valor extrem.

CONCLUSIÓ

A partir de les dades de generació i preus d’energia eòlica i els algorismes presentats a Plufg i Pichler [5], s’han pogut aconseguir els objectius que es tenien a l’inici del projecte. Per una banda, s’han obtingut els arbres d’escenaris, així com la seva representació gràfica i, a més, s’han interpretat els resultats en els termes relacionats amb el mercat elèctric i el model WBVPP. A més, s’han pogut veure les diferències entre usar un model estocàstic (com és el cas estudiat) i un model determinista.

Per crear els arbres d’escenaris, inicialment s’ha hagut de fer un estudi teòric dels algorismes a aplicar, extrets de l’article de Plufg i Pichler [5]. Els tres algorismes utilitzats han sigut:

- Un algorisme de *clustering* jeràrquic, per ajuntar les dades en un nombre especificat de clústers (Algorisme 1).
- Un algorisme d’aproximació estocàstica, per aconseguir els valors que s’usaran per crear l’arbre d’escenaris (Algorisme 3).
- Un arbre de generació d’arbres d’escenaris (Algorisme 7).

La implementació d’aquests tres algorismes, a més, implicava la definició prèvia d’una distància. Els dos casos estudiats han sigut la distància de Wasserstein i la distància niada (aquesta és un cas particular de la primera), on ambdues s’obtenen com a conseqüència del càlcul del valor objectiu d’un problema d’optimització lineal. Finalment, s’ha aplicat la distància niada i, per calcular-la, també han sigut necessaris els conceptes de distància i norma euclidianes.

Per tal d’aplicar correctament els algorismes, inicialment es van realitzar dos casos pràctics, per així entendre la interpretació dels diferents passos d’aquests. El primer cas es va basar en l’aplicació de l’Algorisme 1 (*clustering* jeràrquic). Donades unes dades, i un nombre predeterminat de clústers a obtenir, l’objectiu era aconseguir els clústers demanats. En l’execució d’aquest exercici, es va poder observar com hi havia situacions que l’algorisme no plantejava, com és el criteri a seguir en cas d’empat en la distància mínima entre clústers. Un cop resolt l’exemple, es va realitzar la implementació de l’algorisme a AMPL.

Seguidament, es va realitzar l’aplicació de l’Algorisme 7 en un exercici pràctic. L’execució d’aquest implica també usar els altres dos algorismes escollits. Durant la realització de l’exercici pràctic, s’observà com en l’Algorisme 3 calia aplicar-hi modificacions i, per aquest motiu, únicament s’ha usat per obtenir el clúster de les medianes, que conté els valors que s’usaran per realitzar l’arbre d’escenaris. Un cop resolt l’exercici, es va prosseguir amb la implementació d’aquest algorisme, utilitzant també el codi de l’Algorisme 1.

Durant la programació dels algorismes, es van haver d’anar aplicant modificacions en la manera d’implementar els processos i de definir els paràmetres, de manera que s’adaptessin millor a les dades. És el cas, per exemple, dels passos a seguir a l’hora d’escollir les observacions de generació d’energia (que té en compte les etapes anteriors). En el cas de la definició dels paràmetres, no s'han utilitzat valors predeterminats a l'hora de determinar quina ha de ser la distància màxima entre els nodes que surten del node obert estudiat, sinó que a cada iteració es fixava segons la distància entre les observacions escollides.

Un cop finalitzada la implementació, es van aplicar les dades i es van obtenir les taules amb els arbres d'escenaris. El fet de no determinar l'amplitud que havia de tenir l'arbre ha sigut rellevant, ja que el nombre d'escenaris per cada arbre va ser molt variable. En alguns casos es tenien uns 200 escenaris i, en altres, es superaven els 1000.

A continuació, es va realitzar la interpretació dels resultats fent referència a una planta eòlica en els mercats d'energia. Els gràfics analitzats no es van obtenir com a part d'aquest projecte, però es van introduir per poder il·lustrar l'aplicació dels arbres d'escenaris en el context d'estudi.

La situació que s'ha estudiat és la relació que hi ha entre una planta eòlica i una bateria (VPP) amb els mercats diari i intradiari. La planta és l'encarregada de la generació de l'energia eòlica, i aquesta es pot vendre a un dels mercats mencionats o bé es pot carregar a la bateria, i així usar-la en un futur. A més, també es pot obtenir energia comprant-la al mercat intradiari. En el moment de la interacció entre la planta i els mercats es poden produir dos casos: o bé que es produeixi més energia (desequilibri positiu) o bé que no s'arribi a la quantitat casada (desequilibri negatiu). Aquests escenaris són molt importants a l'hora de plantejar el model que s'ha estudiat: maximitzar els guanys que té una planta eòlica. Aquest objectiu s'ha vist enunciat en la següent equació, que és la funció objectiu del model d'optimització:

$$EP^{VPP}(p^D, p^I, p^{IB+}, p^{IB-}) = DM(p^D) + IM(p^I) + IB^+(p^{IB+}) - IB^-(p^{IB-}).$$

Per aconseguir calcular el valor òptim d'aquesta funció ha sigut imprescindible l'obtenció dels arbres d'escenaris.

Tenint en compte aquest context, es van interpretar uns gràfics que mostraven la relació entre la generació de la planta eòlica, l'evolució de l'energia cassada segons l'hora del dia i els processos que es duien a terme a la bateria. Així doncs, s'observà com la planta eòlica intentava reservar energia, guardant-la a la bateria, per poder satisfer la quantitat d'energia cassada a les hores en què el preu era superior. A més, també es va veure com podien influir els desequilibris en les accions de la planta, ja que en alguns casos es tenien desequilibris negatius deguts a que es guardava energia per transaccions futures i no es destinava a satisfer l'energia cassada.

Finalment, l'últim pas dut a terme en aquest projecte ha estat la interpretació d'uns boxplots que presentaven informació referent a 90 dies. Per una banda, s'ha pogut observar com en un 50% del casos, els guanys de la planta es troben entre els 2000 € i 8000 €. La mitjana de tots els valors ha estat de 5277.68 €.

Per altra banda, s'ha obtingut un boxplot que ha permès la comparació entre el valor òptim de benefici amb els procediments utilitzats i el valor aconseguit utilitzant un model determinista. En un 75% del casos (uns 68 dies), la millora del model d'aquest estudi es troba entre el 2% i el 12%.

Ja per finalitzar, es pot concloure que tots els objectius establerts a l'inici de l'estudi s'han aconseguit. Gràcies a l'estudi i implementació de l'algorisme de generació d'arbres d'escenaris, i de dos algorismes necessaris per al correcte funcionament d'aquest, s'han obtingut els arbres d'escenaris basats en dades de generació i preus d'energia eòlica. A més, s'han pogut aplicar i interpretar els resultats en el context de les plantes eòliques en els mercats d'energia, i veure quines aplicacions tenien els arbres d'escenaris.

Com a línies de continuació d'aquest treball, cal ressenyar:

- L'extensió de l'estudi del Capítol V al conjunt complet de dies de 2017.
- La comparativa en termes de la distribució del paràmetre $FVSS_d$ entre els arbres generats en aquest projecte i els generats mitjançant algorismes *Forward Tree Construction* Römisch i Heitsch [7] usats a Heredia et al. [3].
- L'elaboració d'un article amb l'extensió del treball Heredia et al. [3] per al cas amb set mercats intradiaris i l'anàlisi derivada dels dos punts anteriors.

BIBLIOGRAFIA

[1] AMPL [Online]. 2016. Disponible: http://ampl.com/.

[2] Heredia, F.-J., Cuadrado, M. (2018). Methodology for scenario tree generation for multi-stage stochastic programming models.

[3] Heredia, F.-J., Cuadrado, M., Corchero, C. (2018). On optimal paricipation in the electricity markets of wind power plants with battery energy storage systems. *Computers and Operations Research*, 96 (2018): 316–329.

[4] Pflug, G. C., Pichler, A. (2014). Multistage Stochastic Optimization. *Springer Series in Operations Research and Financial Engineering*, DOI:10.1007/978-3-319-08843-3__2.

[5] Pflug, G. C., Pichler, A. (2015). Dynamic generation of scenario trees. *Computational Optimization and Application*s, 62(3):641–668.

[6] Pflug, G. C., Pichler, A. (2016). From empirical observations to tree models for stochastic optimization: converge properties. *Society for Industrial and Applied Mathematics*, pp. 1715–1740.

[7] Römisch, W., Heitsch, H. (2009). Scenario tree modelling for multistage stochastic programs. *Mathematical Programming*, vol. 118, no. 2, pp. 371-406.

ANNEX 1: Sèries numèriques

Definició A1.1. Una sèrie de nombres reals és un parell de successions de nombres reals $(a_n)_{n\geq 0}$, $(s_n)_{n\geq 0}$ relacionades per:

$$s_n = \sum_{k=0}^{n} a_k.$$

S'anomena terme n-èssim de la sèrie a l'element a_n, i suma parcial n-èssima de la sèrie a s_n.

Observació A1.2. Les sumes parcials defineixen els termes:

$$a_0 = s_0$$

$$a_n = s_n - s_{n-1} \ \ (n \geq 1)$$

Definició A1.3. S'anomena suma de la sèrie a:

$$s = \lim_{n\to\infty} s_n = \lim_{n\to\infty} \sum_{k=0}^{n} a_k,$$

suposant que existeixi.

Definició A1.4. Una sèrie $\sum a_n$ es diu convergent o divergent si ho és la successió de sumes parcials.

- Convergent: $\lim s_n \in \mathbb{R}$
- Divergent: $\lim s_n = \pm\infty$
- Oscil·lant: $\lim s_n$ no existeix

Proposició A1.5 (Condició necessària de convergència). Si $\sum a_n$ és convergent, aleshores $\lim a_n = 0$.

Demostració. Se sap que $a_n = s_n - s_{n-1}$, per tant, $\lim a_n = \lim\,(s_n - s_{n-1})$. Com que és convergent, el $\lim s_n$ existeix i, com a conseqüència, també ho fa $\lim s_{n-1}$. Així doncs, s'obté que:

$$\lim a_n = \lim\,(s_n - s_{n-1}) = \lim s_n - \lim s_{n-1} = 0. \ \blacksquare$$

Definició A1.6. Sigui $r \in \mathbb{R}$. Es defineix la sèrie geomètrica de raó r a la sèrie:

$$\sum_{n\geq 0} r^n.$$

Proposició A1.7. La sèrie geomètrica és convergent si i només si $|r| < 1$ i la seva suma és:

$$\sum_{n\geq 0} r^n = \frac{1}{1-r}.$$

Demostració. Primer cal calcular el terme n-èssim:

$$s_n = 1 + r + \cdots + r^n = \begin{cases} n+1 & \text{si } r = 1 \\ \frac{r^{n+1}-1}{r-1} & \text{si } r \neq 1 \end{cases}$$

Aleshores es tenen diferents casos. Si:

- $r = 1$, $\lim s_n = \lim_{n\to\infty} n + 1 = +\infty$.

- $|r| > 1,\ \lim s_n = \lim_{n\to\infty} \frac{r^{n+1}-1}{r-1} = +\infty.$
- $|r| < 1,\ \lim s_n = \lim_{n\to\infty} \frac{r^{n+1}-1}{r-1} = \frac{-1}{r-1}.$
- $r = -1,\ s_n = 0$ si n parell i $s_n = 1$ si n imparell. Per tant, la sèrie és oscil·lant. ■

Definició A1.8. La sèrie $\sum a_n$ és de termes positius si $a_n \geq 0, \forall n \geq 0$.

Proposició A1.9. Si una sèrie $\sum a_n$ és de termes positius, aleshores la successió $(s_n)_{n\geq 0}$ de sumes parcials és creixent i, per tant, sempre es té límit:

$$\sum a_n = \lim_{n\to\infty} s_n = \sup_{n\in\mathbb{N}} s_n.$$

Aquest límit pot ser finit (si la successió de sumes parcials és acotada) o infinit (en cas contrari).

Demostració. La prova d'aquesta proposició és trivial. ■

Proposició A1.10 (Criteri de comparació directa). Siguin $\sum a_n$ i $\sum b_n$ dues sèries de termes positius. Si $\exists n_0 \in \mathbb{N}$ tal que $a_n \leq b_n$ $(\forall n \geq n_0)$, aleshores:

$$\sum_{n=n_0}^{\infty} a_n \leq \sum_{n=n_0}^{\infty} b_n.$$

Per tant, la convergència de $\sum b_n$ implica la de $\sum a_n$, i la divergència de $\sum a_n$ implica la de $\sum b_n$.

Demostració. Per l'enunciat:

$$\sum_{i=n_0}^{n} a_i \leq \sum_{k=n_0}^{n} b_k \xRightarrow{es\ fa\ el\ límit} \sum_{i=n_0}^{\infty} a_i \leq \sum_{k=n_0}^{\infty} b_k.$$

Els termes $a_0, \cdots, a_{n_0}$ es poden afegir al sumatori i no alteren la convergència. ■

Definició A1.11. S'anomena sèrie harmònica a la sèrie:

$$\sum_{n\geq 1} \frac{1}{n}.$$

Definició A1.12. Sigui $p \in \mathbb{R}$. S'anomena sèrie harmònica generalitzada o sèrie de Riemman de paràmetre p a la sèrie:

$$\sum_{n\geq 1} \frac{1}{n^p}.$$

Proposició A1.13. La sèrie de Riemann és convergent si i només si $p > 1$.

Demostració. Es distingiran entre diferents casos:

- Si $p = 1$. Se suposa que la sèrie és convergent amb suma s:

  $$s = \left(1 + \frac{1}{2}\right) + \left(\frac{1}{3} + \frac{1}{4}\right) + \cdots > \left(\frac{1}{2} + \frac{1}{2}\right) + \left(\frac{1}{4} + \frac{1}{4}\right) + \cdots = 1 + \frac{1}{2} + \frac{1}{3} + \cdots = s,$$

  que és una contradicció, ja que s'obté $s > s$. Per tant, en aquest cas és divergent.

- Si $p < 1$.

$$n^p \leq n \xRightarrow{es\ fa\ invers} \frac{1}{n^p} \geq \frac{1}{n}$$

i per comparació directa amb la sèrie harmònica, divergeix.

- Si $p > 1$.

$$\sum_{n\geq 1} \frac{1}{n^p} = 1 + \left(\frac{1}{2^p} + \frac{1}{3^p}\right) + \left(\frac{1}{4^p} + \frac{1}{5^p} + \frac{1}{6^p} + \frac{1}{7^p}\right) + \cdots \leq$$

$$\leq 1 + \left(\frac{1}{2^p} + \frac{1}{2^p}\right) + \left(\frac{1}{4^p} + \frac{1}{4^p} + \frac{1}{4^p} + \frac{1}{4^p}\right) + \cdots == 1 + \frac{1}{2^{p-1}} + \frac{1}{2^{2(p-1)}} + \cdots$$

que és una sèrie geomètrica de raó $\frac{1}{2^{p-1}} < 1$ i, per tant, convergent.■

ANNEX 2: Fitxers utilitzats

2.1. Fitxer distance.mod

Aquest programa serveix per calcular la distància niada.

```
var prob { cl_i in ESCENARIS, cl_j in CLUSTERS } >= 0;
minimize z : sum{ cl_i in ESCENARIS, cl_j in CLUSTERS }
dist_trans[cl_i,cl_j]*prob[cl_i,cl_j];
subject to restriccion1 { cl_i in ESCENARIS }:
      sum{ cl_j in CLUSTERS } prob[cl_i, cl_j] = 1/n;
subject to restriccion2 { cl_j in CLUSTERS }:
      sum{ cl_i in ESCENARIS } prob[cl_i, cl_j] = 1/s;
```

També es va realitzar una altra versió, utilitzant la notació node i arc.

```
minimize z;
node restriccion1 { cl_i in ESCENARIS }: net_in = 1/n;
node restriccion2 { cl_j in CLUSTERS }: net_in = 1/s;
arc prob { (cl_i, cl_j) in LINKS } >= 0,
     from restriccion1[cl_i], to restriccion2[cl_j], obj z dist_trans[cl_i,cl_j];
```

2.2. Fitxer Algorisme7.dat

Aquest fitxer conté dades necessàries per a l'execució de l'algorisme, com són el vector amb l'espessor mínima, un vector que indica quines etapes fan referència a la generació d'energia, un vector que indica a quina posició comença cada etapa i un vector que fa referència al nombre de columnes que té cada etapa.

```
param b :    1     2   3   4   5      6     7     8     9     10     11   12
13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29
30   31   32   33   34 :=
             1     2     2   2   1   1   1   1   1   1    1    1    1    1    1
1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1    1
1    1    1;

param gener :      1     2   3   4   5      6     7     8     9     10     11
12   13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28
29   30   31   32   33   34 :=
                   1     0     0   0   0   1   2   0   3   4    5    0    6    7
8    9    0   10   11   12   13    0   14   15   16   17   18   19    0   20   21
22   23   24    0;

param col_var : 1   2   3   4   5      6     7     8     9     10     11   12
13   14   15   16   17   18   19   20   21   22   23   24   25   26   27   28   29
30   31   32   33   34 :=
                   1     1  25  49  73  97  98  99 119 120  121  122  139  140
141  142  143  156  157  158  159  160  169  170  171  172  173  174  175  178
179  180  181  182  183;

param nRVSG [*] :=
 0  0    4 24    8  1   12  1   16 13   20  1   24  1   28  3   32  1
 1 24    5  1    9  1   13  1   17  1   21  9   25  1   29  1   33  1
 2 24    6  1   10  1   14  1   18  1   22  1   26  1   30  1   34 24
 3 24    7 20   11 17   15  1   19  1   23  1   27  1   31  1;
```

2.3. Fitxer Algorisme7bo2.run

Aquest fitxer és el que inclou la implementació de l'Algorisme 7 i tots els processos necessaris per després poder dibuixar els resultats.

```
param T default 34; #nombre etapes
set ETAPES  := 1 .. T;
param erel_gen := 0.99;
param erel_preu := 1;
param b { ETAPES };
param max_dist { ETAPES };
param dist_a7;
param nRVSG { 0 .. T }; #elements de cada etapa (important per vectors)
param aproxs;
param E { 1 .. aproxs, 1 .. 206 }; #aproximacions
param valor_anterior;
param gener { ETAPES };
param col_var { ETAPES };
param d { 0 .. T };
param ScenO {1 .. 206};
param ScenF {1 .. 206};

######Paràmetres necessaris per Algorisme 1######
param n, integer;
set ESCENARIS  := 1 ..n;
param s, integer;
set CLUSTERS := 1 .. s;
param max_coord;
param max_clust;
param distancies { 1 ..  max_clust, 1 ..  max_clust };
param clust { 1 ..  max_clust } default 0;
param minim_dist default 10000000;
param ci default 0;
param cj default 0;
param suprem default 0;
param minim_clust default n + 1;
param coordnivellactual, integer;
set COORD := 1 .. coordnivellactual;
param valors { 1 ..  max_clust, 1 .. max_coord}; ###valors aproximacions que es
faran servir a cada iterció
param clust_anter default 0;

####Per escollir aproximacions es faran servir nombres aleatoris
param vect_aleat { 1 ..  max_clust };

######Paràmetres necessaris per Algorisme 3######
param ultim_nivell { 1 ..  max_clust, 1 ..  max_coord + 1};
param n_punts { 1 ..  max_clust} default 0;
param vect_ordenat { 1 ..  max_clust, 1 .. max_coord };
param minim default 0;
param cluster default 1;
param posicio default 1;
param median { 1 .. max_clust, 1 .. max_coord };

#####Model de la nested distance#####
param dist_trans { 1 .. max_clust, 1 .. max_clust};
model distance.mod;
```

```
##CREACIÓ DE LA MATRIU FINAL
param ultima, integer;
param columna default 0;
param fila default 1;
set FILES := 1 .. fila;
set COLUMNES := 1 .. columna;
param matriu{ FILES, COLUMNES};
param suma default 0;
param nnodes { 1 .. 1 + T}; #nodes que hi ha a cada etapa en la matriu final
param pos_escriure, integer;
param antcol, integer;
param cont, integer;

##CREACIÓ DE LA MATRIU DE GENERACIÓ
param mat_gen { FILES, 1 .. 48};
param hora default 0;
param naprox default 0;
param aleat, integer;
param trobat, integer;
param etapa_gener, integer;

##CREACIÓ DE LA MATRIU DE PROBABILITATS
param mat_prob{ FILES, ETAPES};
param prob_final { FILES };

##CREACIÓ DE LA MATRIU DELS CLÚSTERS
param mat_clust { FILES, ETAPES };

###CREACIÓ MATRIU PER FER EL GRÀFIC
param mat_grafic { FILES, 1 .. T+1 };
param valor;
param mig;
param clust_actual;
param pos_clust;
param pos_col;
param nclusts;
param pos_1;

param bases := 90;

##INICI ALGORISME##
for{jj in 1 .. bases}{
   reset data;
   data ("20181219_10000_" & jj &".txt");
   data Algoritme7.dat;
   let max_clust := max{ f in ETAPES } b[f] + 4;
   let max_coord := max{ f in ETAPES } nRVSG[f];
   let fila := 1;
   let aproxs := 10000;
   let columna := 0;
   let hora := 0;
   let nnodes[1] := 1;
   for{e in ETAPES}{
      if gener[e] > 0 then {
         let hora := hora + 1;
   }
   let antcol := columna;
   let columna := columna + d[e];
   let coordnivellactual := d[e];
```

```
let nnodes[e+1] := 0;
let pos_escriure := 1;
for{nod in 1 .. nnodes[e]}{
   let s:= b[e];
   let n:= s + 4;
   let {i in ESCENARIS, j in COORD} valors[i,j] := 0;
   ##Escollir aproximacions que es faran servir
   if gener[e] > 1 then {
      let naprox := 0;
      let aleat := round(Uniform(1, aproxs));
      repeat while naprox < n {
         let trobat := 1;
         let etapa_gener := 5;
         repeat while trobat = 1  and etapa_gener < e {
         if gener[etapa_gener] > 0 then {
            if E[aleat,col_var[etapa_gener]] > mat_gen[pos_escriure,
            2*gener[etapa_gener]] or E[aleat,col_var[etapa_gener]] <
            mat_gen[pos_escriure, 2*gener[etapa_gener]-1] then {
               let trobat := 0;
            }
         }
         let etapa_gener := etapa_gener + 1;
      }
      if trobat = 1 then {
         let naprox := naprox + 1;
         let valors[naprox,1] := E[aleat,col_var[e]];
      }
      if aleat = aproxs then {
         let aleat := 1;
      } else {
         let aleat := aleat + 1;
      }
   }
} else {
   for{i in ESCENARIS}{
      let vect_aleat[i] := round(Uniform(1, aproxs));
   }
   let {i in ESCENARIS, j in COORD} valors[i,j] := E[vect_aleat[i], antcol +
   j];
}
for{i in 1 .. n-1}{
   for{j in i+1 .. n}{
      let suma := 0;
      for{k in COORD}{
         let suma := suma + (valors[i,k] - valors[j,k])^2;
      }
      let distancies[i,j] := suma^(1/2);
      let distancies[j,i] := distancies[i,j];
   }
   let distancies[i,i] := 0;
}
let distancies[n,n] := 0;
if gener[e] > 0 then {
   let max_dist[e] := erel_gen*(max{ i in ESCENARIS } (sum{j in ESCENARIS}
   distancies[i,j]))/n;
} else {
   let max_dist[e] := erel_preu*(max{ i in ESCENARIS } (sum{j in ESCENARIS}
   distancies[i,j]))/n;
}
```

```
   let dist_a7 := max_dist[e] + 1;
   repeat while dist_a7 > max_dist[e]{
##Inici Algorisme 1##
##Matriu de distàncies de l'Algorisme 1
      if s != b[e] then {
         for{i in 1 .. n-1}{
            for{j in i+1 .. n}{
               let suma := 0;
               for{k in COORD}{
                  let suma := suma + (valors[i,k] - valors[j,k])^2;
               }
               let distancies[i,j] := suma^(1/2);
               let distancies[j,i] := distancies[i,j];
            }
            let distancies[i,i] := 0;
         }
         let distancies[n,n] := 0;
      }
      for{i in ESCENARIS}{
         let clust[i] := i;
      }
      for{k in s+1 .. n}{
#Es busca la distància mínima i entre quins clústers és
         let minim_dist := 10000000;
         for{i in 1 .. n-1}{
            for{j in i+1 .. n}{
               if distancies[i,j] < minim_dist and distancies[i,j] != 0 then {
                  let minim_dist := distancies[i,j];
                  let ci := i;
                  let cj := j;
               }
            }
         }
#S'introdueix el nou punt al clúster nou
         let clust_anter := clust[cj];
         for{i in ESCENARIS}{
            if clust[i] = clust_anter then {
               let clust[i] := clust[ci];
            }
         }
#S'actualitza la matriu de distàncies calculant el suprem
         let suprem := 0;
         for{i in ESCENARIS}{
            if clust[i] != clust[ci] then {
               let suprem := 0;
               for{j in ESCENARIS}{
                  if clust[j] = clust[ci] then {
                     let suprem := max(suprem, distancies[i,j]);
                  }
               }
               for{j in ESCENARIS}{
                  if clust[j] = clust[ci] then {
                     let distancies[j,i] := suprem;
                     let distancies[i,j] := suprem;
                  }
               }
            } else {
               for{j in i+1 .. n}{
                  if clust[i] = clust[j] then {
```

```
                    let distancies[i,j] := 0;
                    let distancies[j,i] := 0;
                 }
              }
           }
        }
     }
     for{i in CLUSTERS}{
        let minim_clust := n + 1;
        for{j in ESCENARIS}{
           if clust[j] >= i and clust[j] < minim_clust then
              let minim_clust := clust[j];
        }
        for{j in ESCENARIS}{
           if clust[j] = minim_clust then
                let clust[j] := i;
        }
     }
##Fi Algorisme 1##
##Inici Algorisme 3##
     for{i in ESCENARIS}{
        for{j in COORD}{
           let ultim_nivell[i,j] := valors[i,j];
        }
        let ultim_nivell[i,coordnivellactual+1] := 0;
     }
     let {i in 1 .. s} n_punts[i] := 0;
#Es calcula el número de punts de cada clúster
     for{i in CLUSTERS}{
        for{j in ESCENARIS}{
           if clust[j] = i then {
              let n_punts[i] := n_punts[i] + 1;
           }
        }
     }
     for {co in COORD}{
        let posicio := 1;
        for{i in CLUSTERS}{
           for{j in 1 .. n_punts[i]}{
              let minim := max{ f in ESCENARIS } ultim_nivell[f,co];
              for{l in ESCENARIS}{
                 if ultim_nivell[l,co] <= minim and clust[l] = i and
                 ultim_nivell[l,coordnivellactual+1] = 0 then {
                    let minim := ultim_nivell[l,co];
                    let cluster := l;
                 }
              }
              let vect_ordenat[posicio,co] := minim;
              let ultim_nivell[posicio,coordnivellactual+1] := 1;
              let posicio := posicio + 1;
           }
        }
        let {esc in ESCENARIS} ultim_nivell[esc,coordnivellactual+1] := 0;
     }
#Es calcula el clúster de medianes
     for{j in COORD}{
        let posicio := 0;
        for{i in CLUSTERS}{
           if n_punts[i] mod 2 = 0 then {
```

```
                let median[i,j] := (ultim_nivell[posicio + n_punts[i]/2, j] +
                ultim_nivell[posicio + n_punts[i]/2 + 1, j])/2;
            } else {
                let median[i,j] := ultim_nivell[posicio + n_punts[i] div 2 + 1,
                j];
            }
            let posicio := n_punts[i] + posicio;
        }
    }
##Fi Algorisme 3##
##Inici Algorisme 7##
    for{i in ESCENARIS}{
        for{j in CLUSTERS}{
            let suma := 0;
            for{k in COORD}{
                let suma := suma + (valors[i,k] - median[j,k])^2;
            }
            let dist_trans[i,j] := suma^(1/2);
        }
    }
    if s = 1 then {
            let dist_a7 := (sum{i in ESCENARIS} dist_trans[i,1])/n;
    } else {
        option solver cplex;
        option solver_msg 0;
        solve;
        let dist_a7 := z;
    }
    if s = n then let dist_a7 := 0;
    if dist_a7 > max_dist[e] then let s := s + 1;
  }
  let fila := fila + s - 1;
  if nnodes[e] != nod then {
     let cont := fila - s + 1;
     let ultima := fila;
     repeat while cont > pos_escriure {
        for{j in 1 .. e-1}{
            let mat_prob[ultima,j] := mat_prob[(cont),j];
            let mat_clust[ultima,j] := mat_clust[(cont),j];
        }
        for{j in 1 .. columna - d[e]}{
            let matriu[ultima,j] := matriu[(cont),j];
        }
        let cont := cont - 1;
        let ultima := ultima - 1;
     }
  }
   let {i in pos_escriure + 1.. pos_escriure + s - 1, j in 1 .. columna - d[e]}
   matriu[i,j] := matriu[pos_escriure,j];
   let {i in CLUSTERS, j in COORD} matriu[i + pos_escriure - 1,antcol + j] :=
   median[i,j];
   let {i in pos_escriure + 1.. pos_escriure + s - 1, j in 1 .. e - 1}
   mat_prob[i,j] := mat_prob[pos_escriure,j];
   let {i in CLUSTERS} mat_prob[i + pos_escriure - 1, e] := 1/s;
   let {i in pos_escriure + 1.. pos_escriure + s - 1, j in 1 .. e - 1}
   mat_clust[i,j] := mat_clust[pos_escriure,j];
   let {i in CLUSTERS} mat_clust[i + pos_escriure - 1, e] := i + pos_escriure -
   1;
  if gener[e] != 0 then {
```

```
      if nnodes[e] != nod then {
         let cont := fila - s + 1;
         let ultima := fila;
         repeat while cont > pos_escriure {
            for{j in 1 .. 2*hora - 2}{
               let mat_gen[ultima,j] := mat_gen[(cont),j];
            }
            let cont := cont - 1;
            let ultima := ultima - 1;
         }
      }
      let {i in pos_escriure + 1.. pos_escriure + s - 1, j in 1 .. 2*hora - 2}
      mat_gen[i,j] := mat_gen[pos_escriure,j];
      let {i in CLUSTERS} mat_gen[i + pos_escriure - 1, 2*gener[e] - 1] :=
      median[i,1] - 5*max_dist[e];
      let {i in CLUSTERS} mat_gen[i + pos_escriure - 1, 2*gener[e]] := median[i,1]
      + 5*max_dist[e];
   } else {
      let cont := fila - s + 1;
      let ultima := fila;
      repeat while cont > pos_escriure {
         for{j in 1 .. 2*hora}{
            let mat_gen[ultima,j] := mat_gen[(cont),j];
         }
         let cont := cont - 1;
         let ultima := ultima - 1
      }
      let {i in pos_escriure + 1.. pos_escriure + s - 1, j in 1 .. 2*hora}
      mat_gen[i,j] := mat_gen[pos_escriure,j];
   }
   let pos_escriure := pos_escriure + s;
   let nnodes[e+1] := nnodes[e+1] + s;
##Fi Algorisme 7##
   }
   display e;
##Es fa un vector amb les probabilitats
   for{i in FILES}{
      let prob_final[i] := 1;
      for{j in ETAPES}{
         let prob_final[i] := prob_final[i]*mat_prob[i,j];
      }
   }
###Guardar arbre escenari
   if jj < 10 then {
      printf "#param erel_gen:= %5f;\n", erel_gen> ("20181219_"& nnodes[T+1]&"-
      00"& jj &".dat");
      printf "#param erel_preu:= %5f;\n", erel_preu> ("20181219_"& nnodes[T+1]&"-
      00"& jj &".dat");
      printf "param nS:= %5i;\n", nnodes[T+1]> ("20181219_"& nnodes[T+1]&"-00"& jj
      &".dat");
      printf "param nSG:= %5i;\n", T> ("20181219_"& nnodes[T+1]&"-00"& jj
      &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
      display nRVSG> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
      display ScenO> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
      display ScenF> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
```

```
#Scen0
        printf"param Scen0 (tr)\n " > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf":" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf{l in COLUMNES} "%5i\t",l > ("20181219_"& nnodes[T+1]&"-00"& jj
        &".dat");
        printf":=\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        for{i in FILES}{
           printf "%5i\t",i > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           for{j in COLUMNES}{
               printf "%.6f\t",matriu[i,j] > ("20181219_"& nnodes[T+1]&"-00"& jj
               &".dat");
           }
        printf "\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        }
        printf";" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf"\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
#Prob0
        printf"param Prob0 :=\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        for{i in FILES} {
           printf "%5i\t",i > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
          printf"%.10f\n",prob_final[i] > ("20181219_"& nnodes[T+1]&"-00"& jj
          &".dat");
        }
        printf";\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
#Conjunts clústers
        printf"set c["> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf "%.0f", 0> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf ","> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf "%.0f", 1> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        printf "] := "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        for{i in FILES}{
               printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        }
        printf "\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
        for{j in ETAPES}{
           printf"set c["> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           printf "%.0f",j> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           printf ","> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           printf "%.0f",1> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           printf "] := "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           printf "%.0f\t",1> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
           for{i in FILES}{
               if i != 1 then {
                  if mat_clust[i,j] = mat_clust[i-1,j] then {
                     printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-00"& jj
                     &".dat");
                  } else {
                     printf";\n"> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf"set c["> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf "%.0f",j> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf ","> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf "%.0f",i> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf "] := "> ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
                     printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-00"& jj
                     &".dat");
                  }
               }
           }
           printf";\n" > ("20181219_"& nnodes[T+1]&"-00"& jj &".dat");
```

```
      }
   } else {
      printf "#param erel_gen:= %5f;\n", erel_gen> ("20181219_"& nnodes[T+1]&"-0"&
      jj &".dat");
      printf "#param erel_preu:= %5f;\n", erel_preu> ("20181219_"& nnodes[T+1]&"-
      0"& jj &".dat");
      printf "param nS:= %5i;\n", nnodes[T+1]> ("20181219_"& nnodes[T+1]&"-0"& jj
      &".dat");
      printf "param nSG:= %5i;\n", T> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      display nRVSG> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      display ScenO> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "param "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      display ScenF> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
#Scen0
      printf"param Scen0 (tr)\n " > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf":" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf{l in COLUMNES} "%5i\t",l > ("20181219_"& nnodes[T+1]&"-0"& jj
      &".dat");
      printf":=\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      for{i in FILES}{
         printf "%5i\t",i > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         for{j in COLUMNES}{
            printf "%.6f\t",matriu[i,j] > ("20181219_"& nnodes[T+1]&"-0"& jj
            &".dat");
         }
         printf "\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      }
      printf";" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf"\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
#Prob0
      printf"param Prob0 :=\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      for{i in FILES} {
         printf "%5i\t",i > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf"%.10f\n",prob_final[i] > ("20181219_"& nnodes[T+1]&"-0"& jj
         &".dat");
      }
      printf";\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
#Conjunts clústers
      printf"set c["> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "%.0f", 0> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf ","> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "%.0f", 1> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      printf "] := "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      for{i in FILES}{
         printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      }
      printf ";\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
      for{j in ETAPES}{
         printf"set c["> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf "%.0f",j> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf ","> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf "%.0f",1> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf "] := "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         printf "%.0f\t",1> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
         for{i in FILES}{
            if i != 1 then {
               if mat_clust[i,j] = mat_clust[i-1,j] then {
```

```
                    printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-0"& jj
                    &".dat");
                } else {
                    printf";\n"> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf"set c["> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf "%.0f",j> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf ","> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf "%.0f",i> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf "] := "> ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
                    printf "%.0f\t", i> ("20181219_"& nnodes[T+1]&"-0"& jj
                    &".dat");
                }
            }
        }
        printf";\n" > ("20181219_"& nnodes[T+1]&"-0"& jj &".dat");
    }
}
###Per fer el gràfic
for{i in FILES}{
    let mat_grafic[i,1] := nnodes[T+1]/2;
}
if nnodes[2] mod 2 = 0 then {
    let valor := mat_grafic[1,1]/nnodes[2];
    let mig := nnodes[2]/2;
    for{i in FILES}{
        if i = 1 then {
            let mat_grafic[i,2] := mat_grafic[1,1] + mig*valor;
        } else {
            if mat_clust[i,1] = mat_clust[i-1,1] then {
            let mat_grafic[i,2] := mat_grafic[1,1] + mig*valor;
        } else {
            if mig = 1 then {
                let mig := - 1;
            } else {
                let mig := mig - 1;
            }
            let mat_grafic[i,2] := mat_grafic[1,1] + mig*valor;
        }
    }
} else {
    if nnodes[2] = 1 then {
        for{i in FILES}{
            let mat_grafic[i,2] := mat_grafic[1,1];
        }
    } else {
        let valor := mat_grafic[1,1]/(nnodes[2]-1);
        let mig := (nnodes[2] + 1)/2;
        for{i in FILES}{
           let mat_grafic[i,2] := mat_grafic[1,1]+(mig-mat_clust[i,1])*valor;
        }
    }
}
for{e in 2 .. T}{
    let pos_clust := 2;
    let pos_1 := 1;
    let nclusts := 1;
    repeat while pos_clust <= nnodes[T+1] {
        if mat_clust[pos_clust, e] != mat_clust[pos_clust-1, e] then {
            if mat_clust[pos_clust, e-1] = mat_clust[pos_clust-1, e-1] then {
```

```
    let nclusts := nclusts + 1;
} else {
    if nclusts mod 2 = 0 then {
        if mat_grafic[pos_clust-1,e] != mat_grafic[1,1] then {
            let pos_col := e-1;
            let valor := 0;
            repeat while valor = 0 {
                let valor := abs(mat_grafic[pos_clust-1,pos_col]-
                mat_grafic[pos_clust-1,e])/nclusts;
                let pos_col := pos_col - 1;
            }
        } else {
            let valor := mat_grafic[1,1]/nclusts;
        }
        let mig := nclusts/2;
        for{i in pos_1 .. pos_clust-1}{
            if i = pos_1 then {
                let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
                if i != 1 then {
                    if mat_grafic[i,e+1] >= mat_grafic[i-1,e+1] then {
                        let valor := valor/2;
                        let mat_grafic[i,e+1] := mat_grafic[i,e] +
                        mig*valor;
                    }
                }
            } else {
                if mat_clust[i,e] = mat_clust[i-1,e] then {
                    let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
                } else {
                    if mig = 1 then {
                        let mig := - 1
                    } else {
                        let mig := mig - 1;
                    }
                    let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
                }
            }
        }
    } else {
        if nclusts = 1 then {
            for{i in pos_1 .. pos_clust-1}{
                let mat_grafic[i,e+1] := mat_grafic[pos_clust-1,e];
            }
        } else {
            if mat_grafic[pos_clust-1,e] != mat_grafic[1,1] then {
                let pos_col := e-1;
                let valor := 0;
                repeat while valor = 0 {
                    let valor := abs(mat_grafic[pos_clust-1,pos_col]-
                    mat_grafic[pos_clust-1,e])/(nclusts-1);
                    let pos_col := pos_col - 1;
                }
            } else {
                let valor := mat_grafic[1,1]/nclusts;
            }
            let mig := (nclusts + 1)/2;
            let clust_actual := 1;
            for{i in pos_1 .. pos_clust-1}{
                if i = pos_1 then {
```

```
                        let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                        clust_actual)*valor;
                        if i != 1 then {
                            if mat_grafic[i,e+1] >= mat_grafic[i-1,e+1] then {
                                let valor := valor/2;
                                let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                                clust_actual)*valor;
                            }
                        }
                    } else {
                        if mat_clust[i,e] = mat_clust[i-1,e] then {
                            let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                            clust_actual)*valor;
                        } else {
                            let clust_actual := clust_actual + 1;
                            let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                            clust_actual)*valor;
                        }
                    }
                }
            }
        }
        let nclusts := 1;
        let pos_1 := pos_clust;
    }
  }
  let pos_clust := pos_clust + 1;
}
if nclusts mod 2 = 0 then {
  if mat_grafic[nnodes[T+1],e] != mat_grafic[1,1] then {
    let pos_col := e-1;
    let valor := 0;
    repeat while valor = 0 {
      let valor := abs(mat_grafic[nnodes[T+1],pos_col]-
      mat_grafic[nnodes[T+1],e])/nclusts;
      let pos_col := pos_col - 1;
    }
  } else {
    let valor := mat_grafic[1,1]/nclusts;
  }
  let mig := nclusts/2;
  for{i in pos_1 .. pos_clust-1}{
    if i = pos_1 then {
      let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
      if i != 1 then {
        if mat_grafic[i,e+1] >= mat_grafic[i-1,e+1] then {
          let valor := valor/2;
          let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
        }
      }
    } else {
      if mat_clust[i,e] = mat_clust[i-1,e] then {
        let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
      } else {
        if mig = 1 then {
          let mig := - 1
        } else {
          let mig := mig - 1;
        }
```

```
                    let mat_grafic[i,e+1] := mat_grafic[i,e] + mig*valor;
                }
            }
        }
    } else {
        if nclusts = 1 then {
            for{i in pos_1 .. pos_clust-1}{
                let mat_grafic[i,e+1] := mat_grafic[pos_clust-1,e];
            }
        } else {
            if mat_grafic[nnodes[T+1],e] != mat_grafic[1,1] then {
                let pos_col := e-1;
                let valor := 0;
                repeat while valor = 0 {
                    let valor := abs(mat_grafic[nnodes[T+1],pos_col]-
                    mat_grafic[nnodes[T+1],e])/(nclusts-1);
                    let pos_col := pos_col - 1;
                }
            } else {
                let valor := mat_grafic[1,1]/nclusts;
            }
            let mig := (nclusts + 1)/2;
            let clust_actual := 1;
            for{i in pos_1 .. pos_clust-1}{
                if i = pos_1 then {
                    let mat_grafic[i,e+1] := mat_grafic[pos_clust-1,e]+(mig-
                    clust_actual)*valor;
                    if i != 1 then {
                        if mat_grafic[i,e+1] >= mat_grafic[i-1,e+1] then {
                            let valor := valor/2;
                            let mat_grafic[i,e+1] := mat_grafic[pos_clust-1,e]+(mig-
                            clust_actual)*valor;
                        }
                    }
                } else {
                    if mat_clust[i,e] = mat_clust[i-1,e] then {
                        let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                        clust_actual)*valor;
                    } else {
                        let clust_actual := clust_actual + 1;
                        let mat_grafic[i,e+1] := mat_grafic[i,e]+(mig-
                        clust_actual)*valor;
                    }
                }
            }
        }
    }
}
for{j in ETAPES}{
    printf "%.0f\t",j-1> ("stock_results_" & jj & ".txt");
    printf "%.0f\t",j> ("stock_results_" & jj & ".txt");
    printf "%.10f\t",mat_grafic[1,j]> ("stock_results_" & jj & ".txt");
    printf "%.10f",mat_grafic[1,j+1]> ("stock_results_" & jj & ".txt");
    printf "\n"> ("stock_results_" & jj & ".txt");
    for{i in FILES}{
        if i != 1 then {
            if mat_clust[i,j] != mat_clust[i-1,j] then {
                printf "%.0f\t",j-1> ("stock_results_" & jj & ".txt");
                printf "%.0f\t",j> ("stock_results_" & jj & ".txt");
```

```
                printf "%.10f\t",mat_grafic[i,j]> ("stock_results_" & jj & ".txt");
                printf "%.10f",mat_grafic[i,j+1]> ("stock_results_" & jj & ".txt");
                printf"\n"> ("stock_results_" & jj & ".txt");
              }
           }
        }
    }
    printf "%.10f\t",nnodes[T+1]> ("tree_parameters_" & jj & ".txt");
    printf "%.10f\t",T> ("tree_parameters_" & jj & ".txt");
    printf "%.10f\t",nnodes[T+1]> ("tree_parameters_" & jj & ".txt");
}
```

2.4. Fitxer arbol-Rnuevo180911

El següent codi és el que s'ha utilitzat per aconseguir les representacions dels arbres d'escenaris.

```
#####################################################
#grafica los nodos preservados
x1<- read.delim("stock_results_1.txt", header = F, sep = "\t")$V1
x2<- read.delim("stock_results_1.txt", header = F, sep = "\t")$V2

y1 <- read.delim("stock_results_1.txt", header = F, sep = "\t")$V3
y2 <- read.delim("stock_results_1.txt", header = F, sep = "\t")$V4

cantesc <- read.delim("tree_parameters_1.txt", header = F, sep = "\t")$V1
cantetp <- read.delim("tree_parameters_1.txt", header = F, sep = "\t")$V2
#cantescpres <- read.delim("tree_parameters_1.txt", header = F, sep = "\t")$V3
#erel<-read.delim("paramerel_1.txt", header = F, sep = "\t")$V1
#eps<-read.delim("parameps_1.txt", header = F, sep = "\t")$V1

punto12<-data.frame(x=x1, y=y1)
####################################################
#Etapas
for(j in 1:(cantetp-1)){
 cod=cbind(y1[j],1:length(x1))
 row.names(cod)=cod[,1]
 b1=cod[as.character(y1[j]),2]
}

win.graph()
#leg.txt <- c("scenarios number", "scenarios reduced")
#y.leg <- c(cantesc, cantescpres)
```

```
plot(punto12[1:2],type="n",          xlim=c(0,cantetp),          ylim=c(-10,cantesc),xlab="stages",
ylab="scenarios",main="Scenarios tree")
grid((cantetp+1), NA)
#for (i in 1:2) {
#  text((cantetp-4),-5, paste("scenario number=",formatC(y.leg[1])),adj=1, cex=.7)
#  text((cantetp-8),-10, paste("scenario preserved=",formatC(y.leg[2])),adj=1, cex=.7)
#}

for(j in 1:length(y1)){
  segments(x1[j],y1[j],x2[j],y2[j], col="green4")
}
```

ANNEX 3: Arbres dels 14 dies

Figura A3.1. Arbre d’escenaris pel dia 1.

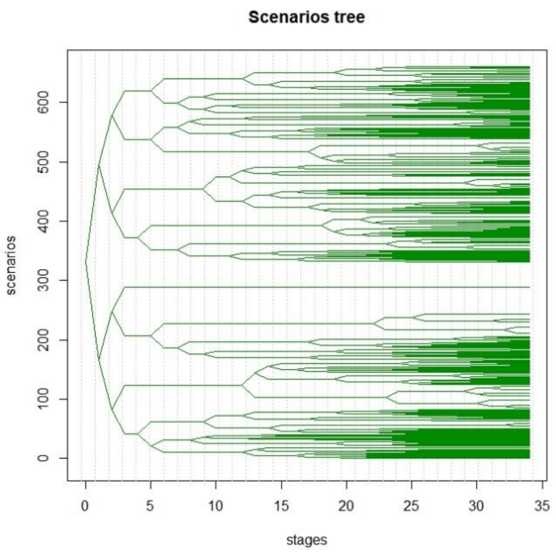


Figura A3.2. Arbre d’escenaris pel dia 2.

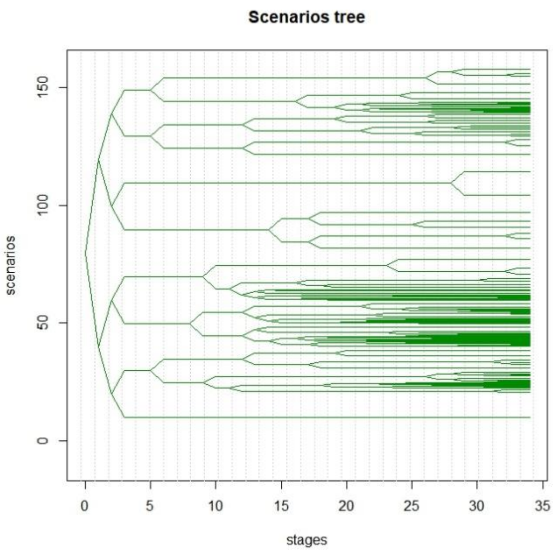


Figura A3.3. Arbre d’escenaris pel dia 3.

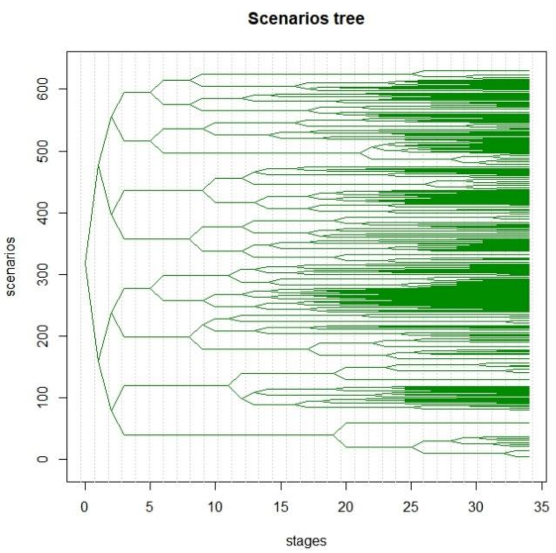


Figura A3.4. Arbre d’escenaris pel dia 4.

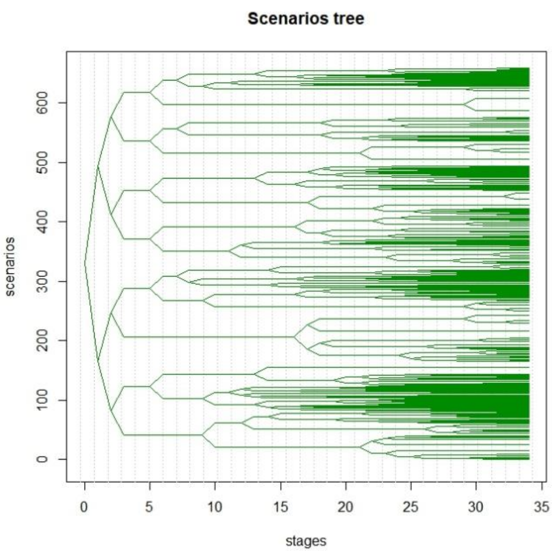


Figura A3.5. Arbre d'escenaris pel dia 5.

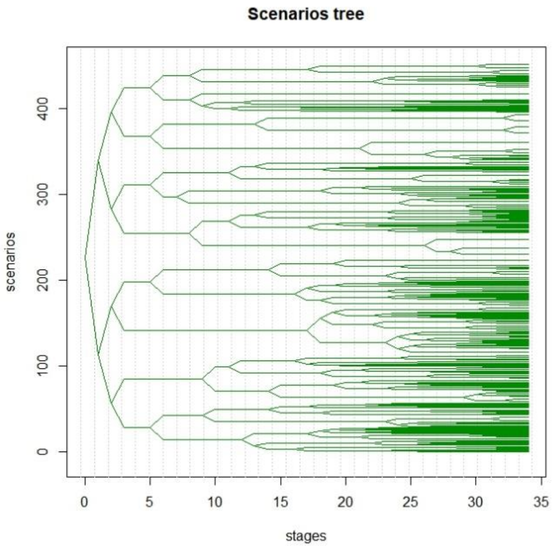


Figura A3.6. Arbre d'escenaris pel dia 6.

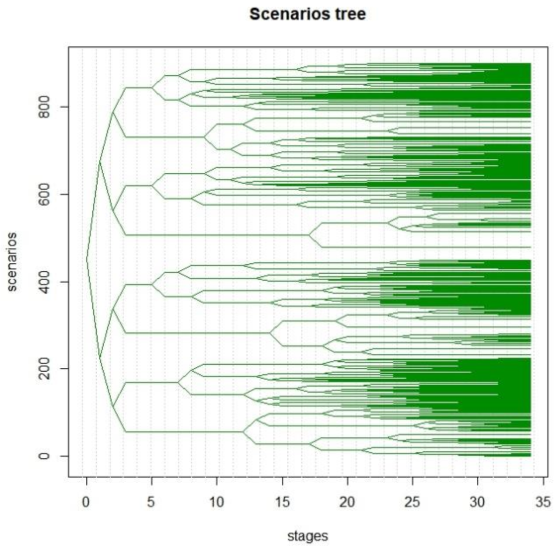


Figura A3.7. Arbre d'escenaris pel dia 7.

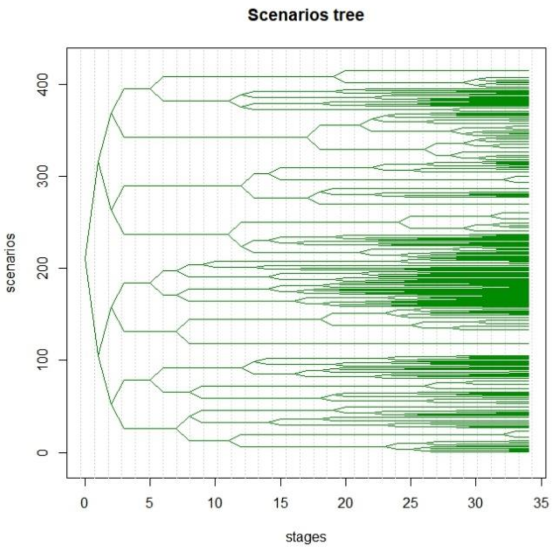


Figura A3.8. Arbre d'escenaris pel dia 8.

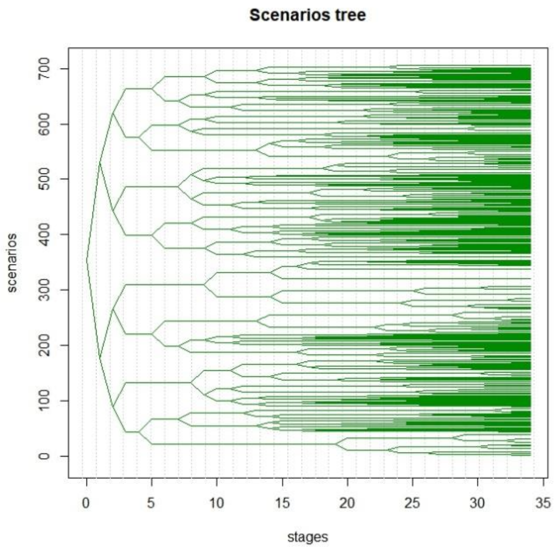


Figura A3.9. Arbre d'escenaris pel dia 9.

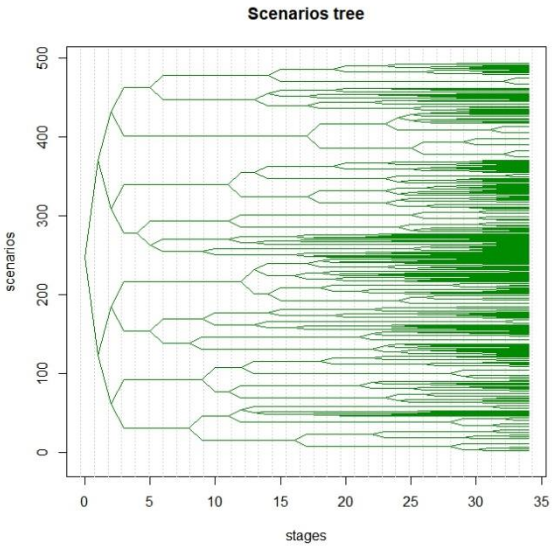


Figura A3.10. Arbre d'escenaris pel dia 10.

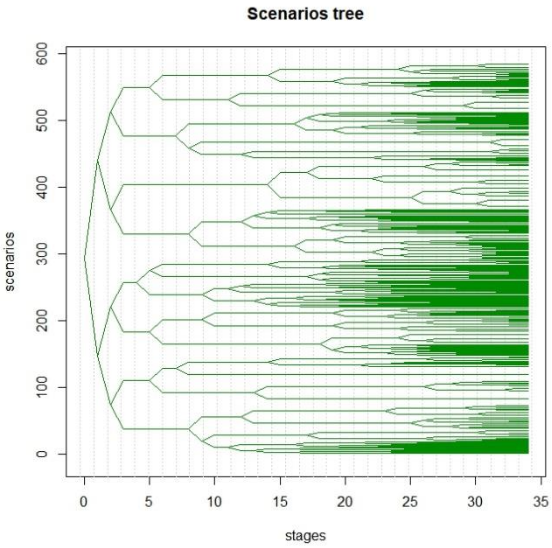


Figura A3.11. Arbre d'escenaris pel dia 11.

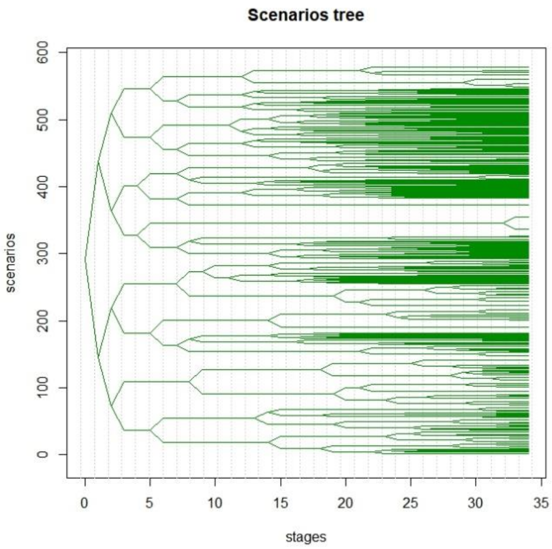


Figura A3.12. Arbre d'escenaris pel dia 12.

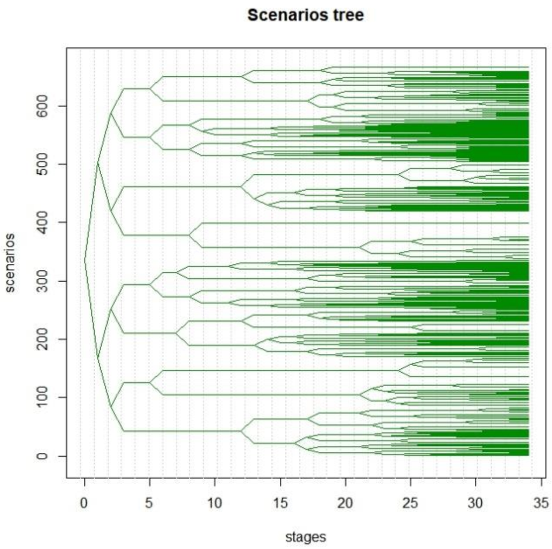


Figura A3.13. Arbre d'escenaris pel dia 13.

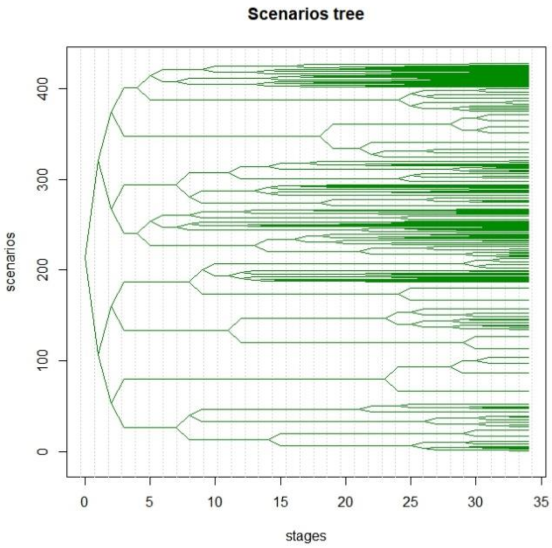
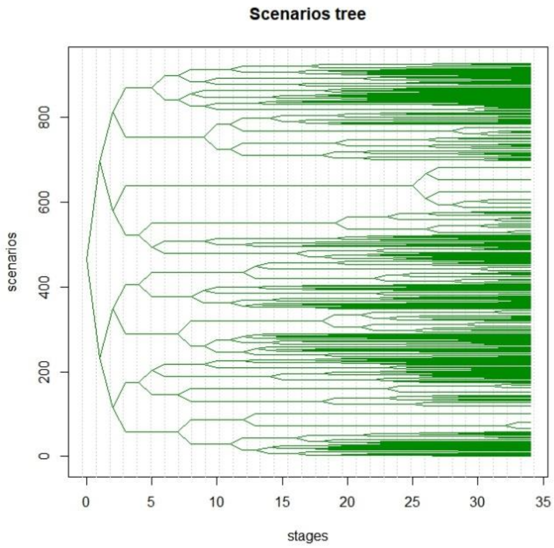


Figura A3.14. Arbre d'escenaris pel dia 14.



ANNEX 4: Resultats dels 7 dies

Figura A4.1. Resultats diumenge 1 de gener de 2017.

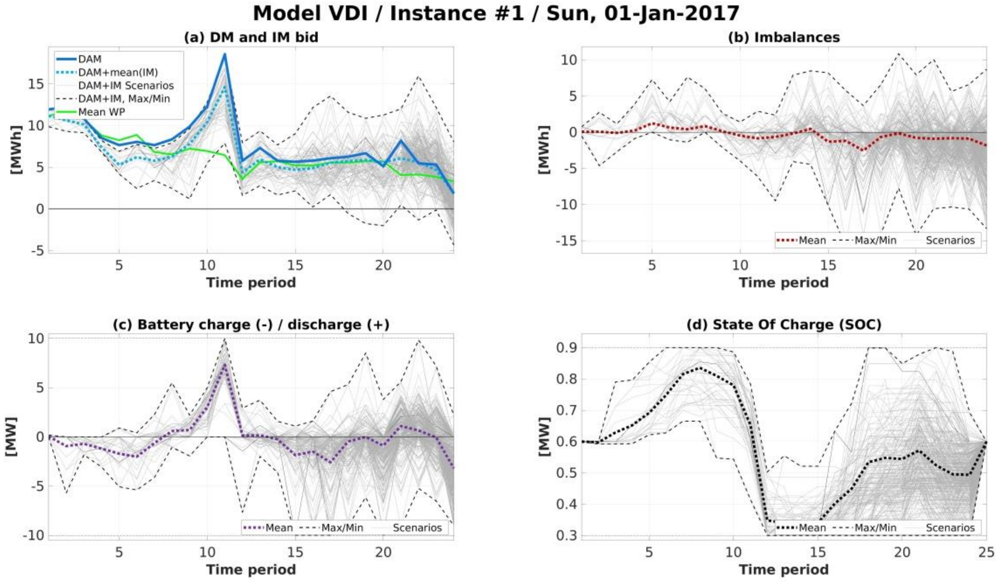


Figura A4.2. Resultats dilluns 2 de gener de 2017.

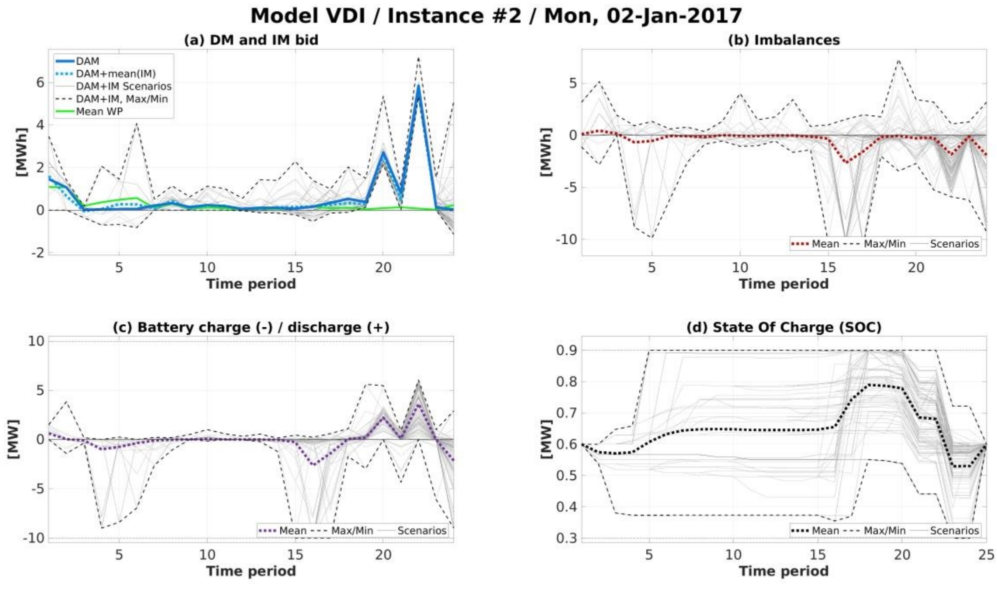


Figura A4.3. Resultats dimarts 3 de gener de 2017.

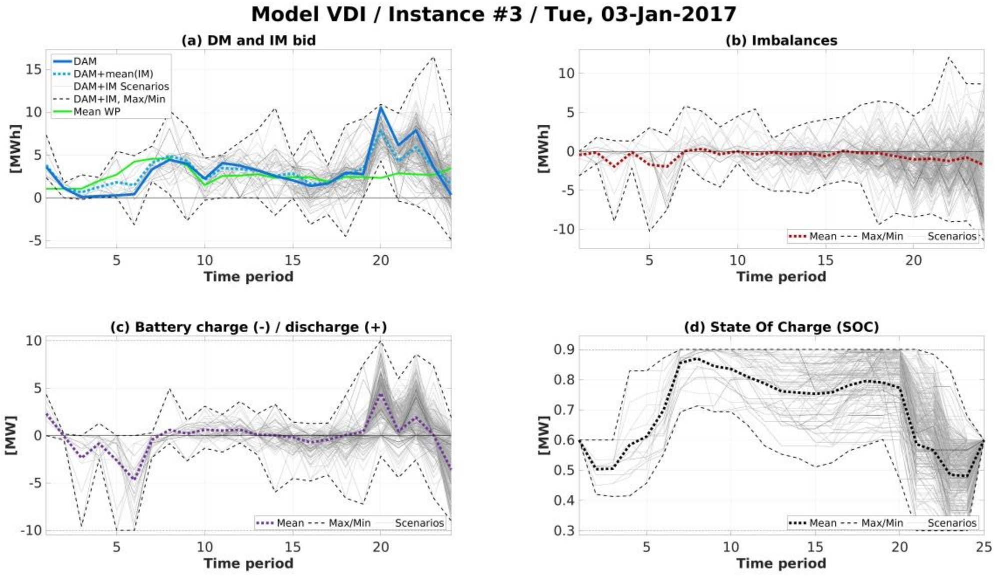


Figura A4.4. Resultats dimecres 4 de gener de 2017.

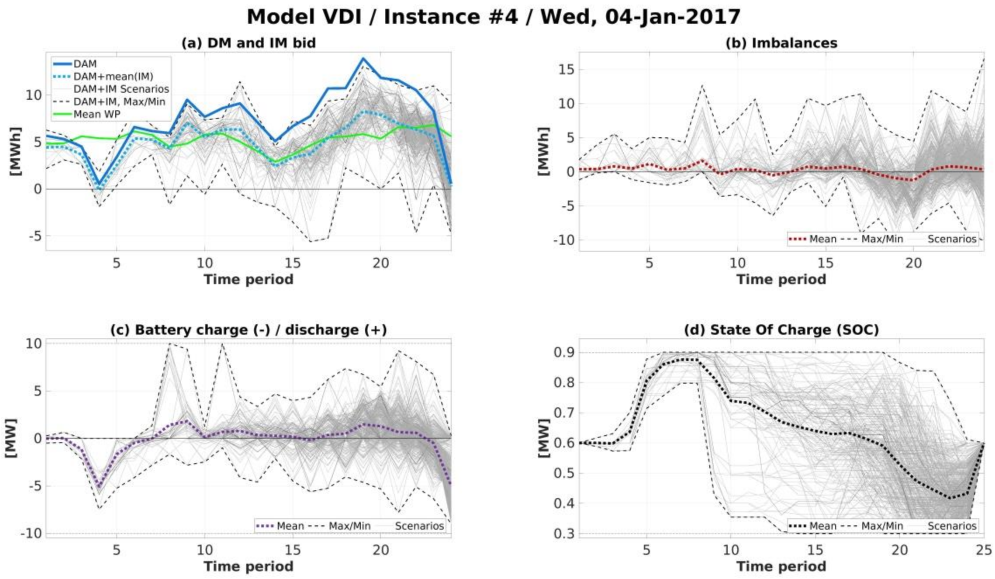


Figura A4.5. Resultats dijous 5 de gener de 2017.

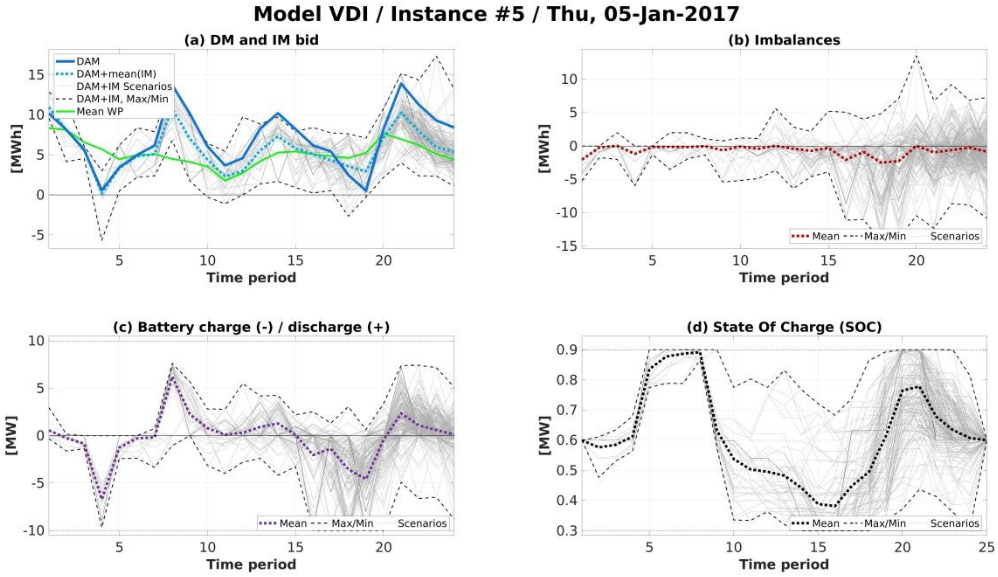


Figura A4.6. Resultats divendres 6 de gener de 2017.

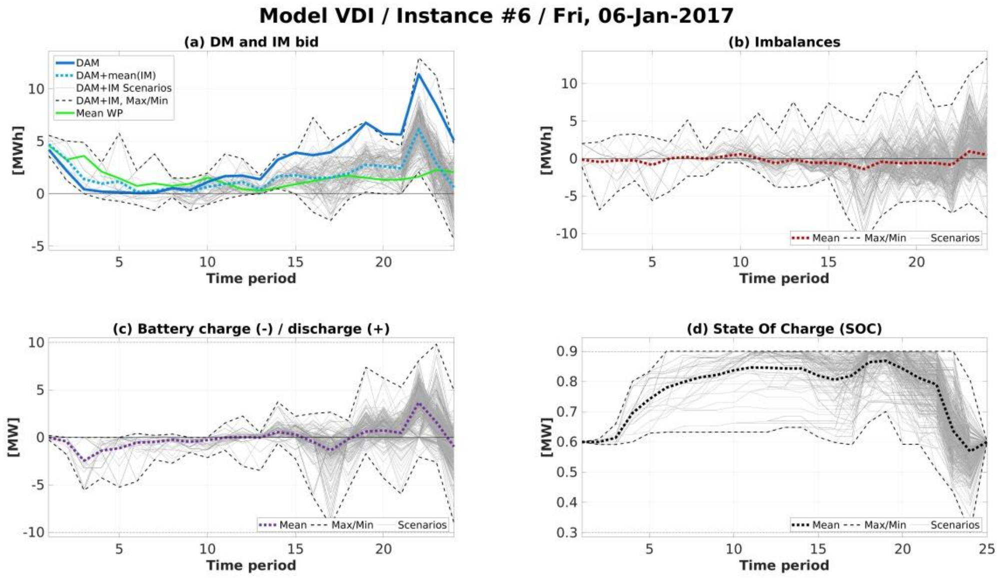


Figura A4.7. Resultats dissabte 7 de gener de 2017.