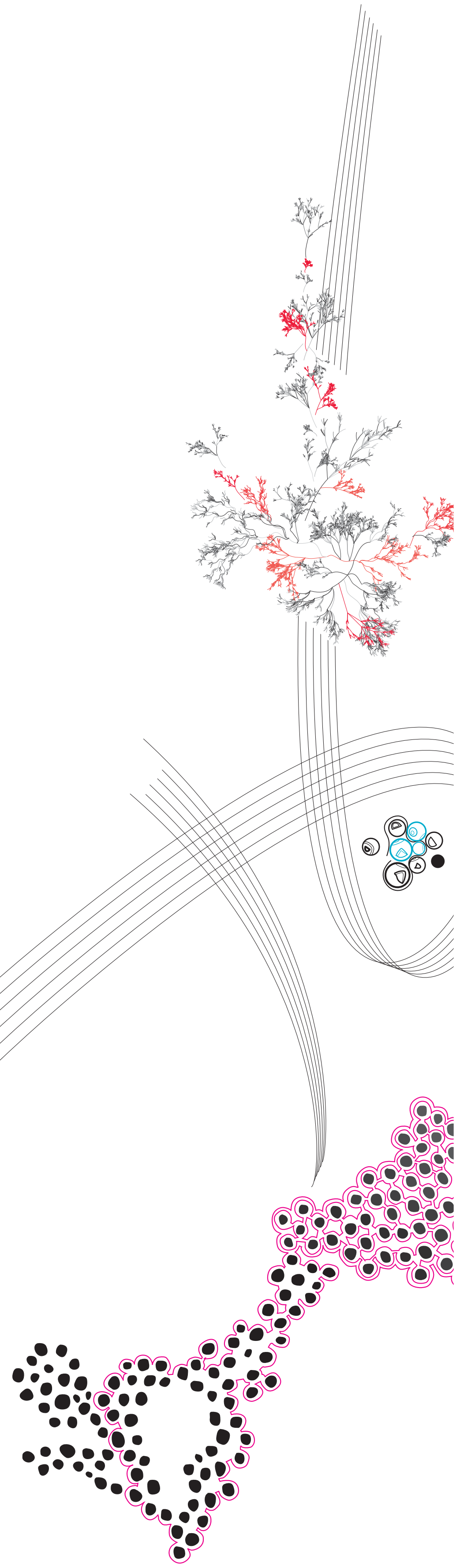
BSc Thesis Applied Mathematics



# Optimal Strategy to Charge a Car using Stochastic Dynamic Programming

Femke Wienk

Supervisor: N. van Dijk

July, 2022

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

## Preface

This report is written as part of my bachelor assignment. I would like to thank my supervisor prof. dr. N. van Dijk for his help during my research and the meetings with always useful feedback.

# Optimal Strategy to Charge a Car using Stochastic Dynamic Programming

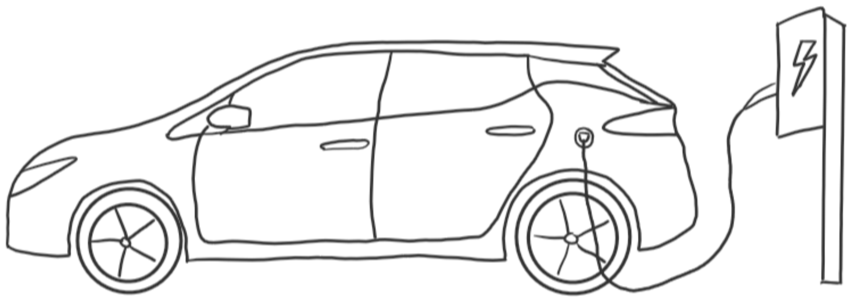
Femke Wienk

July, 2022

**Abstract**

In this paper we want to compare different strategies to charge an electric vehicle and see whether price dependency between periods can be used to an advantage. An electric car needs to be charged regularly. Usually, a car is connected to the network for a longer period than is needed to charge. This gives flexibility to charge the car during periods when the prices are low. First, we approach the problem as a knapsack problem with no uncertainty. Then we consider the prices as independently distributed. Next, we want to include the dependency of the prices between periods. We test different forecasting methods to predict the price of the next period. However, this makes the problem computationally intractable. Instead, we use a Markov chain to include the price dependency between periods. Another approach is also attempted, where we forecast a whole timeseries and proceed by considering this as a knapsack model. A conceptual model to include weather dependence is presented. Simulations with real-world data shows that the strategy with price-dependency performed on average significantly better than to assume the prices are independent between periods.

*Keywords*: Electric vehicles, Markov chain, forecasting, electricity prices



## Contents

# 1 Introduction

Worldwide the sales of electric vehicles are increasing. In 2021 the sales increased with 108% in comparison to the year before [9]. Now charging an electric car the battery takes longer than filling a tank of a fossil fuel car. A typical electric car has a battery capacity of 60 kWh, which takes around 8 hours to completely charge (empty to full) with a 7 kW charging station.However, most drivers charge their car whenever the battery is not completely full. They charge whenever they park, this can be at home, at work, or at the supermarket. This is also know as "top up charging". There exist also fast chargers, these have a charging speed up to 50 kW, but these are expensive and difficult to install [14].

So the time to charge a battery can range from 30 minutes to 12 hours, depending on the battery capacity, state of the battery (empty or full) and the charging speed. Therefore, it is important to plan when and how much to charge. Usually, the car is connected to the network at a fixed position (at home) for a longer period than is needed to charge. This gives the owner flexibility to charge the car, and shift to periods when the load, and so the prices are low [14]. (generally overnight)

In the Netherlands there are currently around 81000 public charging stations and more than 3000 fast chargers, positioned at, for example, road restaurants, parking lots. This is the highest density of charger per 100 km road in Europe. In Europe the charging infrastructure is growing, especially the north of Europe has invested a lot in new charging stations [1].

But charging at your private charging stations, means you have to pay the costs. Electricity prices fluctuate during the day. This is because electricity prices are dependent on multiple factors. One part of the price consists of taxes and regulated components, the other part depends on market developments. Here, the largest component of the price depends on the cost of generating the electricity, which can change minute by minute [4]. Factors that can account for price changes are for example, the amount of load, market developments in other energy markets, also malfunctions at powerplants (this will increase the prices), or the generation of sustainable energy on sunny or windy days (this will decrease the energy prices) [3].

However, it depends on the contract with the energy supplier whether you will experience these price fluctuations during the day. Most energy suppliers charge the same rate for electricity during the day. Some suppliers do offer contracts with lower prices at off-peak hours [15].

The prices are determined by an auction, where every supplier delivers a bidding curve, that states how much energy it wants for all the different possible prices. Then all these bidding curves are combined and a price is determined that balances the supply and demand curves. Demand side management (DSM) methodologies aim to optimize the "consumption pattern of consumers and to exploit the potential of distributed generation and electricity storage systems" [13]. The Powermatcher is one of these management methodologies that make the electricity grid smart. The PowerMatcher effectively matches supply and demand, which results into better use of available sources of energy and allows more renewable energy to be integrated into the electricity system. It also avoids overload situations by shifting loads away from high peak demand moments. It is a methodology that uses the available capacity in the most optimal way possible. The Powermatcher is available on an open-source platform, so that is is widely accessible [11, 13].

In this paper we want to find a strategy that minimizes the charging costs of an electric car. A similar study has been done in [11]. Here, the prices were assumed to be independent between periods. Therefore, we want to compare different charging strategies, and see whether price dependency between periods (e.g. as by wind) can be used to an advantage.
We let the car charge overnight between 8 pm and 8 am. The main research question is: "Is there a significant cost reduction when price dependency is taken into account, as opposed to assume independent prices in determining a charging strategy? "

## 2 Knapsack Problem

First, we evaluate the problem as if we already know the time series of the electricity prices beforehand. Since there is no uncertainty about the prices, we can choose the cheapest periods to charge our car. This is known as a knapsack problem (see the example at the end of this section). In a knapsack problem we have a given set of items, each with a weight and value. We need to determine which item to include in a collection, such that the weight is below or equal to a given limit and the total value is maximized. Here, we want to minimize the total charging costs, while the amount of energy charged equals a certain amount [17]. Let

- $L$ be the total amount of energy to be charged,

- $T$ be the total amount of periods,
- $p_t$ be the price value per unit energy at time $t$ (during period $t$)
- $u_{max}$ be the maximum energy that can be charged in one period,
- $u_t \in [0, u_{max}]$ be the amount of energy we charge in period $t$,

The decision variable is $u_t$, which can range from 0 to $u_{max}$. But $u_t$ equals either 0 or $u_{max}$ when $L$ is a multiple of $u_{max}$. The problem can be described by a linear program with $t = 1, \dots, T$:

$$\min c = u_1 p_1 + u_2 p_2 + \cdots + u_T p_T \tag{1}$$
$$\text{s.t.} \quad u_1 + u_2 + \cdots + u_T = L \tag{2}$$

There are generic methods to solve such linear programs (such as the Branch-and-Bound method). The knapsack problem can also be solved efficiently with a dynamic program. We use the value function $V_t(x)$, where

- $x_t$ is the state, which denotes the amount of energy left to charge at time $t$,
- $V_t(x)$ is the minimal expected costs for periods $t \dots T$.

If the car is not fully charged at the end of period $T$, then we have a penalty cost. So for $t = T + 1$, we obtain;

$$V_{T+1}(x) = \begin{cases} 0 & \text{if} \quad x = 0 \\ \infty & \text{otherwise,} \end{cases} \tag{3}$$

and for $t = T, \dots, 1$

$$V_t(x) = \min_u \left[ u p_t + V_{t+1}(x - u) \right]. \tag{4}$$

**Example**

We describe a simple knapsack problem, where the total weight must be below or equal to a given limit and the total value is maximized

Suppose we have a backpack (knapsack), which can hold a weight of maximum 6 kg. We have a number of items with weights in kg and values in euros (€), as shown in Figure 1.

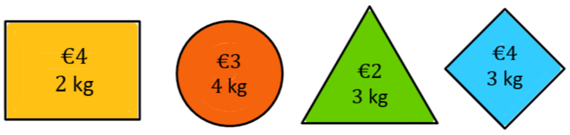


FIGURE 1: Available items to put in the backpack

To solve this problem, we first compute the value/weight ratio for each item. We then put the items with the best ratio (here, the highest ratio) in our backpack. Each item can only be chosen once. The ratios for the items shown in Figure 1 are, respectively:

$$\frac{4}{2} = 2, \quad \frac{3}{4} = 0.75, \quad , \frac{2}{3} = 0.66 \dots, \frac{4}{3} = 1.33 \dots.$$

We see that the yellow triangle has the highest ratio and then the blue diamond. The backpack weights 5 kg with these two items and hence, there's no room for another item. The most optimal solution is to put the yellow rectangle and the blue diamond in the backpack. Then the total value equals €8. No other combination will result in a higher value.

## 3 Stochastic dynamic program with independent prices

Now, we analyze the stochastic dynamic program (SDP) as described in [11, 2]. The prices for the different periods are not known beforehand (else it would be a Knapsack problem as in Section 2). The prices are assumed to be independent between periods. The objective is again to minimize the total cost to charge an electric car within $T$ intervals. We use the following parameters and variables:

- $L$ : The total amount to be charged,
- $u_t \in [0, u_{max}]$ : The amount of energy to charge in period $t$,
- $p_t$ : Price value per unit energy at time $t$ (during period $t$),
- $P_t$ : Stochastic price variable for period $t$,
- $x_t$ : The amount of energy left to charge at time $t$,
- $V_t(x, p)$: The expected minimized costs for periods $t, \ldots, T$, when state at time $t$ is $(x,\, p)$

Recall that the decision variable $u_t$ can range from 0 to $u_{max}$, but equals either 0 or $u_{max}$ when $L$ is a multiple of $u_{max}$. It is assumed the prices $P_t$ are independent and identically distributed. Since the prices are not known beforehand, the price $p_t$ at time $t$ is included in the state. The state is denoted by $(x_t, p_t)$. Further, we want the car to be fully charged after $T$ periods, else we have a penalty cost. Then the minimal expected costs $V_t(x, p)$ given that $x_t = x$ and $P_t = p$, is described by

$$V_{T+1}(x) = \begin{cases} 0 & \text{if} \quad x = 0 \\ \infty & \text{otherwise,} \end{cases} \tag{5}$$

and for $t = T, \ldots, 1$

$$V_t(x, p) = \min_u \left[ up + \sum_{p'} \mathbb{P}(P_{t+1} = p') V_{t+1}(x - u, p') \right] [11]. \tag{6}$$

In [11] the SDP is further approached by a heuristic with an optimal control law for $u_t$ (See Appendix 12.2). However, we are only interested in Equations (5) and (6).

## 4 Forecasting Methods

Next, we want to consider the problem without the assumption that the prices are independent between periods. Therefore, we consider the past price value(s). We can use the past values of the price to make a forecast of the price for the next period. Using this forecast, we can make a better prediction of the state for the next period.

The price distribution at time $t$ is conditional and its expectation would be equal to the forecast price for time $t + 1$. So, suppose at period $t$ we have past values $h_t$ and current value $p_t$. Then we can use $h_t$ and $p_t$ to forecast the price value $p$ for next period . Denote the forecast value by $\tilde{p}$. Then the probability distribution for next period $\mathbb{P}(P_{t+1} = p \,|\, P_t = p_t, h_t)$ can be described by a normal distribution with expectation $\tilde{p}$.

There exist multiple forecasting methods. Forecasting methods can be divided into extrapolation and causal forecasting methods. Extrapolation methods use past values of a time series to forecast future values of that time series. It assumes that past patterns and trends will continue in future time periods. Causal forecasting methods attempt to forecast future values by taking into account what caused past data. So causal forecasting methods have a dependent variable and one or multiple independent variables. Then past data is used to estimate the relation between the dependent and independent variable [17].

We describe the Simple Exponential Smoothing method with Holt-Winter's seasonal component and Simple Linear regression. The Moving-Average method and Auto Regression Integrated Moving Average (ARIMA) method are described in Appendix 12.3.

## Simple Exponential Smoothing

Simple exponential smoothing is an extrapolation forecast method. It is specifically, suitable for time series that fluctuate around a base level. The forecast $A_t$ is the smoothed average after observing $x_t$. $A_t$ is described by

$$A_t = \alpha x_t + (1-\alpha)A_{t-1}, \tag{7}$$

here $\alpha$ is a smoothing constant with $0 < \alpha < 1$. Further, the forecast value of $x_{t+k}$ is denoted by $A_t = f_{t,k}$. We only forecast one period ahead so we have $k = 1$. Then the error $e_t$ is denoted by

$$e_t = x_t - f_{t-1,1} = x_t - A_{t-1}[17]. \tag{8}$$

The mean absolute deviation (MAD) is the measure of forecast accuracy and given by

$$MAD = \frac{\sum_t |e_t|}{\#forecasts}. \tag{9}$$

If the optimal $\alpha$ that minimizes the MAD exceeds 0.5, then trend, seasonality, or a cyclical variation is possibly present. Another form of the exponential smoothing may be better suitable. For example, the Holt-Winter's smoothing method [17].

## Holt-Winter's smoothing method

Holt-Winter's smoothing method captures the seasonality and trend. If we want to forecast $k$ periods in the future and we are now at time $t$, then the forecast $F_{t+k}$ is given by

$$F_{t+k} = L_t + kT_t + S_{t+k-M}, \tag{10}$$

with $L_t$ the level estimate for time $t$, $T_t$ denotes the trend estimate at time $t$, and $S_t$ is the seasonal estimate at time $t$ [8].

## Simple Linear Regression

Simple linear regression is a causal forecast method. Here, we have a dependent variable $Y$ that we want to forecast using an independent variable $X$. As stated before, one of the factors influencing the electricity prices is the wind and solar production [12]. Now, since solar production peaks around noon and has no production during night, we focus on the wind production. Therefore, we take the electricity price as the dependent variable and the measured wind power production as the independent variable.

If both are related with a linear relation, then we can use simple linear regression. The linear relation can be verified by finding the correlation coefficient $r$. This coefficient measures the strength and direction of correlation. It is described by the following ratio between the covariance and the product of the standard deviations:

$$r = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}. \tag{11}$$

The coefficient $r$ has a value between $-1$ and 1, where $r = 1$ means a perfect positive correlation, and $r = -1$ a perfect negative correlation. A value of $r = 0$ means that the variables have no linear dependency. If the variables have a linear relation, then this can be described by [17];

$$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i, \tag{12}$$

here $\varepsilon_i$ is the error. The parameters $\beta_0$ and $\beta_1$ have to be estimated, as the true values are unknown. We denote the estimation by $\hat{\beta}_0$ and $\hat{\beta}_1$ respectively. The values of $\hat{\beta}_0$ and

$\hat{\beta}_1$ are determined by using least squares. Further, we take $\varepsilon_i = 0$, since $\varepsilon_i$ is expected to average out to zero. We obtain the least squares regression line for the prediction for $y_i$, denoted by $\hat{y}_i$;

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i. \tag{13}$$

The values $\hat{\beta}_0$ and $\hat{\beta}_1$, called the least squares estimates of $\beta_0$ and $\beta_1$ minimize the errors $e_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i$:

$$F(\hat{\beta}_0, \hat{\beta}_1) = \sum_i e_i^2 = \sum_i (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i)^2 \tag{14}$$

Then $\hat{\beta}_0$ and $\hat{\beta}_1$ are found by setting

$$\frac{\partial F}{\partial \hat{\beta}_0} = \frac{\partial F}{\partial \hat{\beta}_1} = 0. \tag{15}$$

This results into

$$\hat{\beta}_1 = \frac{\sum_i (x_1 - \bar{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} \quad \text{and} \quad \hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}, \tag{16}$$

here $\bar{x}$ and $\bar{y}$ are the average values of, respectively all $x_i$ and $y_i$ [17].

## 5 Stochastic dynamic program with history of prices

### 5.1 Considering the history of prices

Now we want to include the history of prices and weather dependence in Equation (6), by using one of the forecast methods from Section 4 and 12.3.

We no longer assume that the prices are independent. Instead, we will model the prices as dependent random variables. More precisely, with the history of prices at time $1, \ldots, t-1$, then the probability is described by $P(p_t = p | p_1, \ldots, p_{t-1})$. Therefore, we also need to include the history

$$h_t = (p_1, \ldots, p_{t-1})$$

of the prices to our state. The state is then described by $(x_t, p_t, h_t)$. Given the state at the start of a period, a decision $u$ has to be made. We let

- $L$ be the total amount of energy to be charged,
- $T$ be the total amount of periods,
- $P_t$ be the stochastic price variable for period $t$,
- $p_t$ be the price value at time $t$ (during period $t$),
- $u_{max}$ be the maximum energy that can be charged in one period,
- $u_t \in [0, u_{max}]$ be the amount of energy we charge in period $t$ (the decision variable),
- $h_t$ be an array of past price values $p_1, \ldots, p_{t-1}$ ,
- $V_t(x_t, p_t, h_t)$ be the expected minimized costs for periods $t, \ldots, T$, when state at period $t$ is $(x_t, p_t, h_t)$.

We want the car to be fully charged after $T$ periods, else we have a penalty cost. With this history included, we obtain the value function $V_t(x_t, p_t, h_t)$ with decision variable $u_t$. For $t = T + 1$ we have

$$V_{T+1}(x) = \begin{cases} 0 & \text{if} \quad x = 0 \\ \infty & \text{otherwise,} \end{cases} \tag{17}$$

and for $t = T, T-1, \ldots, 1$;

$$V_t(x_t, p_t, h_t) = \min{}_u \left[ up_t + \sum_{p'} \mathbb{P}(P_{t+1} = p' | P_t = p_t, h_t) V_{t+1}(x - u, p', h_{t+1}) \right] [11]. \tag{18}$$

**Example**

To illustrate the stochastic process, we describe a fictitious situation. Suppose we want to charge the car overnight (8 pm - 8 am). We have a total of $T = 24$ periods of 30 minutes. At the beginning of each period a decision $u$ is to be made whether to not charge ($u = 0$) or charge ($u = 1$) the car. The prices $p_t$ are dependent and $p_t \in [4, 5, 6]$. The prices are distributed according to an unknown conditional distribution. At $t = T + 1$ the total amount of $L = 8$ kW should be charged, if not then the corresponding penalty costs are infinite.

The decision tree with dependent prices for the above situation is shown in Figure 2. The state is described by $(x_t, p_t, h_t)$, with $x_1 = 8$. The background colour from the larger rectangles, represent the price of the last stage. So a green background at time $t$ implies that at time $t-1$ we had a price value of 4, orange represents price value 5, and purple represents price value 6. It is unknown which value we had at $t = 0$, therefore is the colour neutral for $t = 1$. The colours of the smaller rectangles denote the current price value, with the same colour and price combinations, only these colours are more saturated. The costs resulting from the decisions are shown on the lines connecting the different periods.

It can be seen that the problem becomes rapidly computationally intractable. This is due to the fact that we need a lot more 'space' in the decision tree to describe all the possible histories. This could be approached with an approximation algorithm, but this can be tricky as the result may not be the optimal solution. Therefore we describe an alternative method in Section 6.

## 5.2 Weather dependence

One of the factors causing the price changes is the generation of renewable energy, such as solar power and wind power. Therefore, it can be of interest to include weather dependence in our model. We know that the wind power production and solar power generation are dependent on the weather [10]. Since there is no solar production at night, we neglect the solar power in the model and focus on the wind power. Studies, such as [10] have shown that there is a correlation between electricity prices and wind power production. Therefore, we are interested in forecasting the wind power production, so that we can use the correlation with the prices to determine whether we expect the prices to drop tomorrow. We want to forecast the weather using a Markov chain. Then we can use the Markov chain to predict whether we expect more wind power production tomorrow.

We already test the correlation between the wind production and short term fluctuations of the electricity price in Section 4. Therefore, we now want to find the possible correlation between the total wind power production overnight (8 pm - 8 am) and the average electricity price overnight.

First, we need to find the correlation coefficient $r$, see Equation (11), to verify a possible negative correlation between the total wind power production overnight and the mean electricity price overnight. Then we want to forecast the weather by using a simple Markov chain. We describe a Markov chain with two states: 0 and 1. State 0 corresponds with windless weather, so a low wind power production. State 1 corresponds with windy weather, so a high wind power production. We need criteria to determine whether the state is 0 or 1. Therefore, we say that we are in state 1, when the total wind power generation is equal or exceeds a certain value $M$. This value $M$ is determined by the relation between electricity price and amount of wind energy generated, where if the amount exceeds $M$ then the prices are low. So the relation can be described as follows:

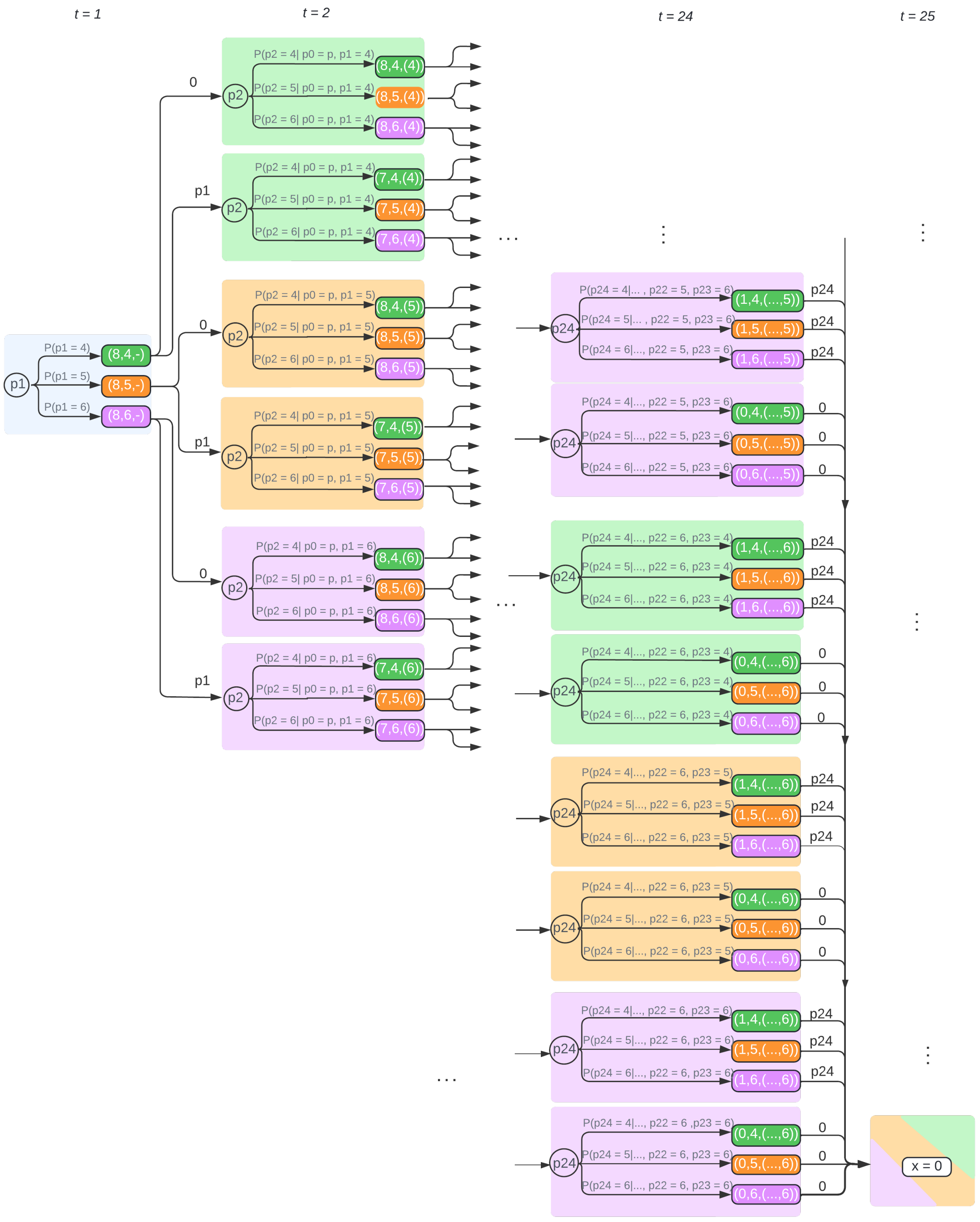

t = 1
t = 2
t = 24
t = 25
p1
P(p1 = 4)
P(p1 = 5)
P(p1 = 6)
(8,4,-)
(8,5,-)
(8,6,-)
0
p1
p2
P(p2 = 4| p0 = p, p1 = 4)
P(p2 = 5| p0 = p, p1 = 4)
P(p2 = 6| p0 = p, p1 = 4)
(8,4,(4))
(8,5,(4))
(8,6,(4))
(7,4,(4))
(7,5,(4))
(7,6,(4))
(8,4,(5))
(8,5,(5))
(8,6,(5))
(7,4,(5))
(7,5,(5))
(7,6,(5))
(8,4,(6))
(8,5,(6))
(8,6,(6))
(7,4,(6))
(7,5,(6))
(7,6,(6))
p24
P(p24 = 4|... , p22 = 5, p23 = 6)
P(p24 = 5|... , p22 = 5, p23 = 6)
P(p24 = 6|..., p22 = 5, p23 = 6)
(1,4,(...,5))
(1,5,(...,5))
(1,6,(...,5))
(0,4,(...,5))
(0,5,(...,5))
(0,6,(...,5))
P(p24 = 4|..., p22 = 6, p23 = 4)
P(p24 = 5|..., p22 = 6, p23 = 4)
P(p24 = 6|..., p22 = 6, p23 = 4)
(1,4,(...,6))
(1,5,(...,6))
(1,6,(...,6))
(0,4,(...,6))
(0,5,(...,6))
(0,6,(...,6))
P(p24 = 4|..., p22 = 6, p23 = 5)
P(p24 = 5|..., p22 = 6, p23 = 5)
P(p24 = 6|..., p22 = 6, p23 = 5)
P(p24 = 4|..., p22 = 6, p23 = 6)
P(p24 = 5|..., p22 = 6, p23 = 6)
P(p24 = 6|..., p22 = 6, p23 = 6)
x = 0

FIGURE 2: Decision tree of the SDP described by Equation (18) that considers the past price values. The bright colours represent the current price value, and the pastel colours represent the last price value. The green colour denotes price value 4, orange denotes 5, and purple denotes 6.

| State | Wind power production | Price value |
|---|---|---|
| 0 | low (< M) | high |
| 1 | high ($\geq$ M) | low |

TABLE 1: Relation between wind production and the price values, with the corresponding Markov chain state

A new strategy can be introduced: If given today we have a low production of wind energy and it is expected that tomorrow we will have a high production of wind energy (which will lower the prices significantly), then today the minimal amount of energy will be charged and tomorrow the car will be fully charged. Further, we would charge fully if today we have a high production of wind energy, regardless which state we will expect tomorrow. This strategy contradicts the constraint that the car is fully charged at the end of the night. Instead we make the constraint that at day $d$ the car needs to be charged an amount $L_d$ at the end of the last period. We describe the Markov chain with a transition probability matrix $\mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} a_{00} & a_{01} \\ a_{10} & a_{11} \end{bmatrix}$$

**Example**

To illustrate how this is combined with the SDP, we consider again the fictitious situation from before. Suppose this car owner has an electric car that has a total capacity $C$ of $C = 60$ kWh. Each day he commutes to work and back, which is a distance of around 40 km that uses 8 kWh [14], so there is 60 - 8 = 52 kWh left in the battery if the car was fully charged at the beginning of the day. The battery still has enough energy left for another commute to work.

Further suppose we have 6 nights of historical data, with the corresponding states 0, 0, 1, 1, 1, 0, Then we can determine the transition probabilities. We have that one out of two times state 0 stays state 0, and one out of two times state 0 transitions to state 1. Hence, $a_{00} = 1/2$, $a_{01} = 1/2$. Further, two out of three times state 1 transitions to state 0 and once stays state 1. Hence $a_{10} = 2/3$ and $a_{11} = 1/3$. Then the transition probability matrix is given by

$$\mathbf{A} = \begin{bmatrix} 1/2 & 1/2 \\ 2/3 & 1/3 \end{bmatrix}.$$

It can be of interest to make some constraints. For example, we always want to charge the battery, when its capacity $C$ is below 30 kWh (as we don't want the car to run out of energy in case the driver needs to make a detour), or when $C$ is below 40 kWh (else we need to charge during most periods and we will be unable to only choose the lowest periods, which is not more optimal then if we would have charged in the lowest periods during the night in state 0).

Suppose we use the constraint that we will always charge the car when the remaining capacity is below $C = 40$ kW, let

- $L_d$ denote the total amount we will charge at day $d$,
- $C_d$ denote the remaining energy in battery at day $d$,
- $s_d$ denote the state of the Markov chain at day $d$.

So, we first look at the weather conditions and accordingly determine the amount $L$ we want to charge. We know that yesterday we were in state 0. Today we will either be in state 0 or 1. The yellow colour denotes state 0 and the colour blue represents state 1. Following our strategy, we charge the car fully whenever we are in state 1, and in state 0 we only charge to $C = 40$ if needed.

Further, everyday the driver uses 8 kWh. We get the following decision tree to determine how much $L_d$ to charge during day $d$, when the state is $(C_d, s_d)$.

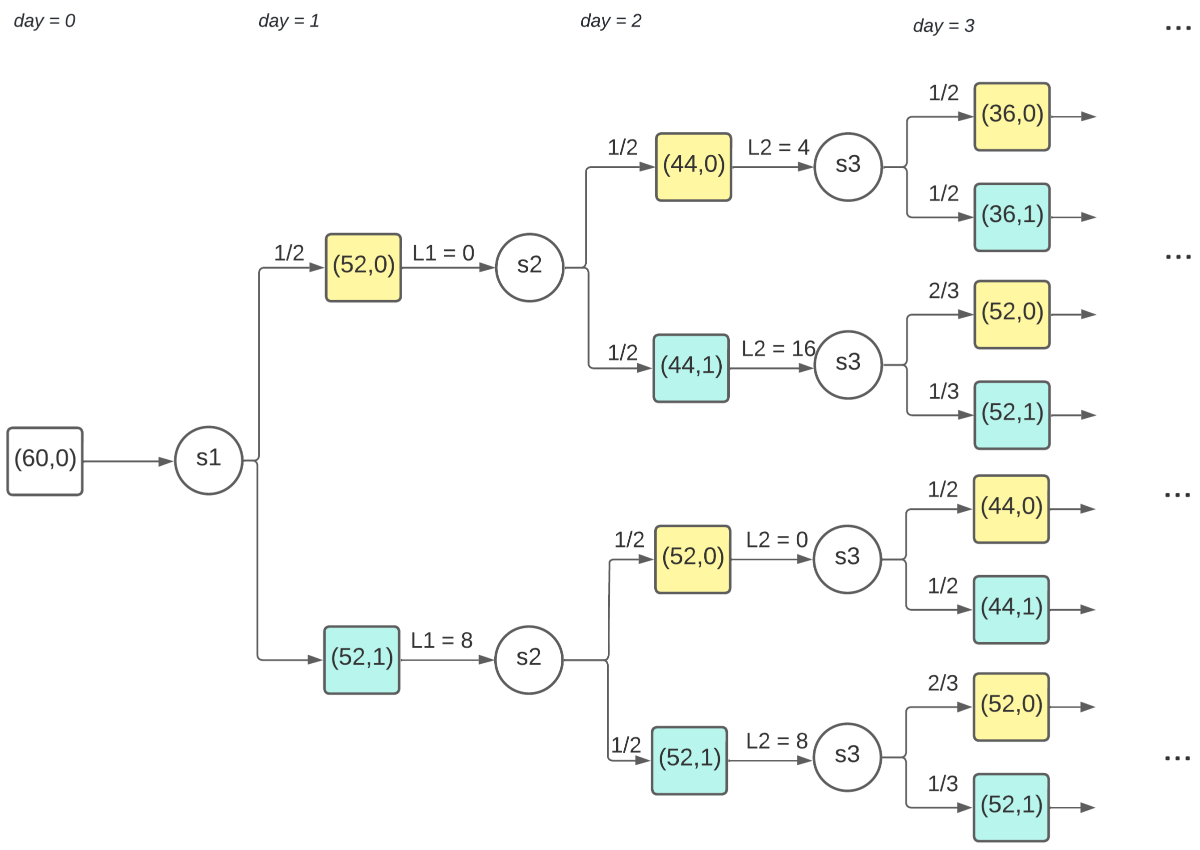


FIGURE 3: Decision tree of the Markov chain that considers the wind production and shows corresponding decisions, when current state is $(C_d, s_d)$, with $d = 0, 1, \ldots$. The yellow blocks denote state 0 and blue denotes state 1.

## 6 Stochastic dynamic program with Markov chain

A second method to implement the price dependency between two periods, is by forecasting the prices using a Markov chain. We define a Markov chain with, for example 4 states. Note that we do not want to have a Markov chain with as many states as we have price values. Since the transition probabilities are based on past data, and most likely not all price values appear in the data. This would result in a Markov chain with many zero entries, which leads to a distorted distribution. We will categorize the prices $p$ in one of the 4 states, using the following constraints with constants $a$, $b$, $c$

$$\text{State 0} \quad \text{if} \quad p < a \tag{19}$$
$$\text{State 1} \quad \text{if} \quad a \le p < b \tag{20}$$
$$\text{State 2} \quad \text{if} \quad b \le p < c \tag{21}$$
$$\text{State 3} \quad \text{if} \quad c \le p \tag{22}$$

We define the constant $b$ by the long run mean $\mu$. Then the $a$ and $c$ are determined by using the long run standard deviation $\sigma$. We get $b = \mu$, $a = \mu - \sigma$, $c = \mu + \sigma$. Now that we have categorized our prices in 4 states, we define the transition probability matrix by $\mathbf{P}$, with $\sum_{j=0}^{3} p_{ij} = 1$ for all $i = 0, \ldots, 3$;

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \\ p_{30} & p_{31} & p_{32} & p_{33} \end{bmatrix}$$

Let the state of $p$ and $p'$ be $i$ and $j$, with $i, j \in [0, 1, 2, 3]$, respectively. The probability that next period we will be in state $j$ given that we are now in state $i$, is $\mathbf{P}_{ij} = p_{ij}$. Note that the probability from state $i$ to $j$ is $p_{ij}$, which has a double subscript, while the price value at time $t$ is denoted by $p_t$. Since each category represents more than one value, we need to give each category its own distribution for its values. So we let the prices be conditional uniformly distributed within categories, that change by a Markov chain. So suppose category $j$ represents $v_j$ values, then the probability $\mathbb{P}(P_{t+1} = p'|P_t = p)$ is given by $p_{ij}/v_j$. Further, let

- $L$ be the total amount of energy to be charged,
- $x_t$ be the amount of energy that is still left to be charged at time $t$,
- $u_t \in [0, u_{max}]$ be the amount of energy charged in period $t$,
- $p_t$ : price value at time $t$ per unit energy,

We obtain the value function $V_t(x, p)$

$$V_{T+1}(x) = \begin{cases} 0 & \text{if} \quad x = 0 \\ \infty & \text{otherwise,} \end{cases} \tag{23}$$

and for $t = T, \dots, 1$,

$$V_t(x, p) = \min_u \left[ up + \sum_{p'} \mathbb{P}(P_{t+1} = p'|P_t = p) V_{t+1}(x - u, p') \right] \tag{24}$$

$$= \min_u \left[ up + \sum_{p'} \frac{p_{ij}}{v_j} V_{t+1}(x - u, p') \right]. \tag{25}$$

**Example**

We use a fictitious situation to describe the situation. We want to charge our car a total amount $L = 8$ kWh. Between 8 pm and 8 am we have a total of 24 periods of 30 minutes. Our decision variable is again $u$. At the beginning of each period we decide whether we want to charge the car ($u = 1$ kWh) or not ($u = 0$)kWh. Suppose we have 3 possible prices; $p \in [4, 5, 6]$, then we define a Markov chain with 3 states. State 1 represents price value 4, state 2 represents price value 5, and state 3 represents price value 6. Let the transition probability matrix be given by

$$\mathbf{P} = \begin{bmatrix} 3/5 & 1/5 & 1/5 \\ 1/4 & 1/2 & 1/4 \\ 1/5 & 3/10 & 1/2 \end{bmatrix}.$$

Here, each state represents one value, so when $i$ is the state of $p_t$ and $j$ is the state of $p'$, then

$$\mathbb{P}(P_{t+1} = p'|P_t = p) = p_{ij}$$

The state of the value function is described by $(x_t, p_t)$. Suppose $p_0 = 6$, which corresponds with state 2. The corresponding decision tree is shown in Figure 4. The large blocks represent one period and the background has the color corresponding to the state of the last price value. The green color represents state 0, orange represents state 1, and purple represents state 2. So, since $p_0 = 6$, the block at $t = 1$ is purple. The smaller blocks have the color of the current state of the price value, with the same state and color combinations as before, only now the colors are more saturated. The costs resulting from the decisions are shown on the lines connecting the different periods.

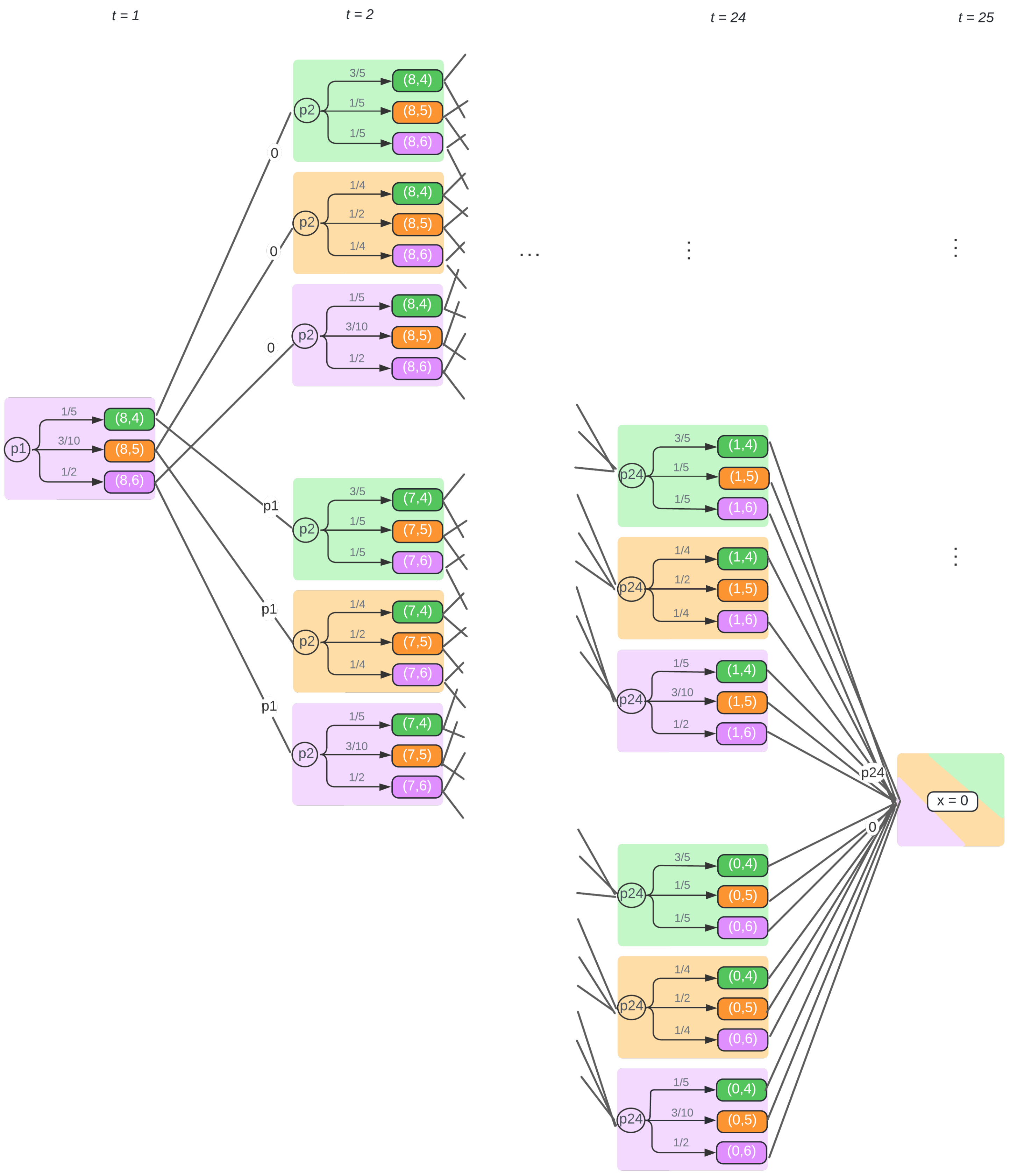


FIGURE 4: Decision tree of the fictitious situation with the SDP with Markov chain described by Equation (25). The bright colors represent the current price value, and the pastel colors represent the last price value. Here, green denotes price value 4, orange denotes 5, and purple denotes 6.

## 7 Deterministic Knapsack

Another approach to finding a charging strategy is to forecast the price values for a whole timeseries ahead. Since the price values follow daily patterns, we use the Holt-Winter's exponential smoothing from Section 4. Then we can use this forecast and consider it as a knapsack problem as described in Section 2. We obtain the strategy that is optimal for the forecasting. Then we will use the same strategy on the true timeseries.

## 8 Numerical Results

### 8.1 Parameters

We let the car charge overnight between 8pm and 8am. We use periods of 15 minutes, so the total amount of periods is $T = 48$.

A typical electric car has a battery capacity of 60 kWh [14]. Home charging stations usually have a charging speed between 3.5 kw and 7 kw [14]. We take a charging station with a speed of 4 kW per hour, so the maximum amount of energy that can be charged in 15 minutes is $u_{max} = 1$ kWh.

In [16] a study has been done on driving behaviour. Most people use the car for commuting between work and home. They found that people drive on average 40 km per day. The most efficient electric vehicles have an efficiency between 0.15 kWh/km and 100 kWh/km, where the average is 0.20 kWh/km [7]. Therefore, we need to charge $L = 40 \cdot 0.20 = 8$ kWh.

For our simulations, we use data from different days in the period between 28 April and 8 June 2022 from the Belgian network. The data containing the measured and upscaled wind power production is available at Elia [5] The electricity prices for the Belgium network are available at Epex Spot [6], but only for a limited time, therefore we only have a small collection of data. Both data sources provide data for 15-minute periods. Elia also provides solar production data, but we focus on wind production as there is no solar production at night. See Appendix 12.4 for an example of how the price and wind data are formatted. The simulations are performed in Python.

### 8.2 Knapsack Problem

For the knapsack problem, we used the electricity prices from Epex Spot [6]. We take

- $u_{max} = 1$ kWh,
- $L = 8$ kWh.

In Table 2, we show the minimized costs for some dates. Later in Section 9, we can use the knapsack problem to determine the performance of the other models, as the result of the knapsack problem is the optimal solution. The policy for 6- 7 June is later shown in Figure 12 or in tabular form in Appendix 12.6.

Table 2: Minimized costs of the Knapsack strategy

| Date | Minimized costs (€) |
|---|---|
| 28 - 29 April | 1.45698 |
| 3 - 4 May | 1.52983 |
| 18 - 19 May | 1.00775 |
| 6 - 7 June | 0.88145 |
| 7 - 8 June | 1.19141 |
| 8 - 9 June | 1.08151 |

## 8.3 Stochastic dynamic program with independent prices

Similar to the Knapsack problem, we take $u_{max} = 1$ kWh, $L = 8$ kWh. Further, we use a normal distribution for the random variables denoting the prices. We use the data from 22 May to 5 June as the training data to determine the long- run average and standard deviation for the Normal distribution. We obtain expectation $\mu_1 \approx 0.164$ and standard deviation $\sigma_1 \approx 0.0470$. In theory, prices can take any value, but in practice they usually don't exceed €0.40000/kWh, see also [6]. Therefore, we assume that prices can range from 0 to 0.40000 and don't have negative values. However, this resulted in an unwanted long duration for the computation. Therefore we decided to round the price values to 3 decimals (the third decimal is a 1/10 part of a cent, hence four or five decimals could also be seen as redundant). We obtain a range $R_1$ from 0 to 0.400 with steps of 0.001. We also try the range $R_2$ from 0 to 0.350, since the maximum observed value in our training data is only €0.33329/kWh. So

- $L = 8$ kWh
- $u_{max} = 1$ kWh
- $\mu_1 = 0.164$
- $\sigma_1 = 0.0470$
- $R_1$ : range of price values from 0 to 0.400 with stepsize 0.001
- $R_2$ : range of price values from 0 to 0.350 with stepsize 0.001

The result is a lookup table, where for each possible price and state the optimal decision is stated and the expected minimized cost. For a number of dates the resulting costs are shown in Table 3 under "$I(\mu_1, R_1)$" and "$I(\mu_1, R_2)$". We see that the results of $I(\mu_1, R_1)$ and $I(\mu_1, R_2)$ are equal. So apparently, here the range does not matter. In the next sections we continue with both ranges, to see whether it does make a difference in the other model.

Not all of the policies are not as optimal as we would like; Most policies charge during the last few periods. This is due to the fact that the test data did not have the low values that the model expected based on the training data. Each date in our test data had an average higher than $\mu_1 = 0.164$, see also Table 12 in Appendix 12.5. Therefore, we inspected our training data and found three nights, namely 25 - 26 May, 26 - 27 May, 27 - 28 May, with a low average price (see Figure 5, note that the dates in the figure are not continuous). We decided to consider these as outliers, and to remove these from our training data to see whether this improved our results.

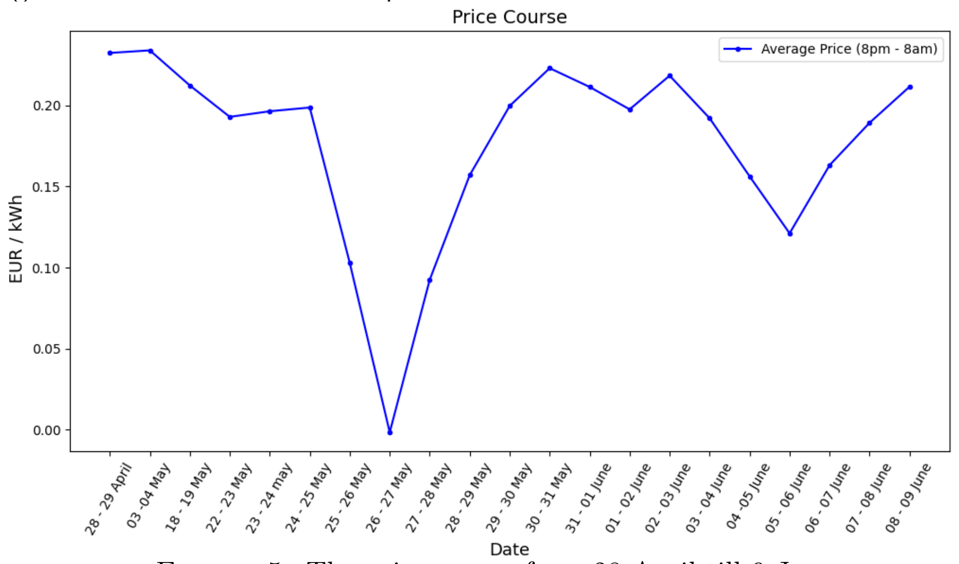


Figure 5: The price course from 28 April till 9 June

We obtain a new mean $\mu_2$ and standard deviation $\sigma_2$

- $\mu_2 = 0.193$
- $\sigma_2 = 0.038$

We use these values in the normal distribution and the results can be found in Table 3 under "$I(\mu_2, R_1)$", and "$I_(\mu_2, R_2)$". Now, there is a small difference between the two ranges $R_1$ and $R_2$. The costs from with the mean $\mu_2$ from the filtered data are overall lower than the mean $\mu_1$ from the unfiltered data. Hence, the new mean appear to give better results.

To compare the overall results, we find the cost increase for $I(\mu_1, R_1)$, $I(\mu_1, R_2)$, $I(\mu_2, R_1)$ and $I(\mu_2, R_2)$ with respect to the optimal strategy (the minimized costs from the knapsack problem), The policy from $I(\mu_2, R_1)$ has lowest cost increase and performs on average best. Therefore, we take $I(\mu_2, R_1)$ as the "Independent" strategy.

TABLE 3: Minimized costs for different dates of the $I(\mu, R)$ strategies, with $\mu = \mu_1, \mu_2$ and $R = R_1, R_2$

| | **Minimized costs (€)** | | | |
|---|---|---|---|---|
| **Date of Night** | $I(\mu_1, R_2)$ | $I(\mu_1, R_1)$ | $I(\mu_2, R_1)$ | $I(\mu_2, R_2)$ |
| 28 - 29 April | 2.09731 | 2.09731 | 1.95484 | 1.95484 |
| 03 - 04 May | 2.08959 | 2.08959 | 1.87519 | 1.89097 |
| 18 - 19 May | 1.27988 | 1.27988 | 1.087 | 1.08286 |
| 06 - 07 June | 0.89699 | 0.89699 | 1.10755 | 1.06955 |
| 07 - 08 June | 1.36178 | 1.36178 | 1.22317 | 1.20694 |
| 08 - 09 June | 1.37132 | 1.37132 | 1.15291 | 1.24087 |
| Cost increase | 26% | 26% | 18% (17.7%) | 18% (18.2 %) |

## 8.4 Forecasting

Data from the electricity prices was again used from EpexSpot [6], and data for the measured and upscaled wind production from Elia [5]. For all forecasting methods we considered the dates 28 - 29 April, 3 - 4 May, 18 - 19 May and 22 - 23 May.

**Moving-Average Method**

We used the moving-average method (MA) on multiple nights and an overview of the optimal window size and corresponding MAD is given in Table 4. Notably, the optimal window size equals 5 for all dates. In Figure 6 the forecast is plotted together with the true price for 3 - 4 May. It can be seen that the true price fluctuates a lot over the intervals, and the moving-average somewhat follows the overall course.

TABLE 4: Results of the moving-average for different dates

| Date of Night | Optimal Window Size | MAD |
|---|---|---|
| 28 - 29 April | 5 | 0.018149... |
| 03 - 04 May | 5 | 0.021821... |
| 18 - 19 May | 5 | 0.018179... |
| 22 - 23 May | 5 | 0.035986... |

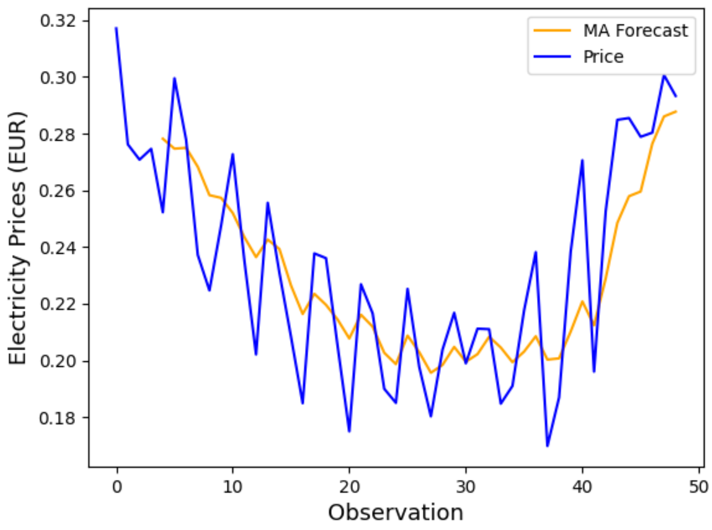


FIGURE 6: Moving-average (MA) forecast of the time series of May 3 - 4, from 8 pm - 8 am with 15-minute periods (resulting in a total of 48 periods)

**Simple Exponential smoothing**

For the simple exponential smoothing method are the optimal smoothing constants $\alpha$ and corresponding MADs shown in Table 5. Recall that $0 < \alpha < 1$, and if the optimal $\alpha$ exceeds 0.5, then most likely either trend, seasonality, or cyclical variation occurs. Here, all smoothing constants in Table 5 are below 0.5. Therefore, we don't need to use the Holt-Winter's seasonality components. In Figure 7 the forecast is plotted with the true price.

TABLE 5: Results SES for different dates

| Date of Night | Optimal $\alpha$ | MAD |
|---|---|---|
| 28- 29 April | 0.45379... | 0.018671... |
| 03 - 04 May | 0.40572... | 0.023955... |
| 18 - 19 May | 0.23627... | 0.034593... |
| 22 - 23 May | 0.45398... | 0.018761... |

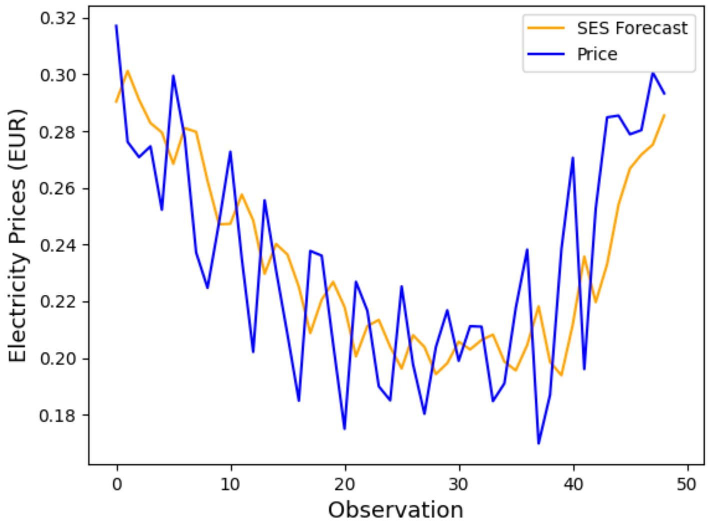


FIGURE 7: Simple exponential smoothing (SES) forecast for May 3 - 4, with $\alpha = 0.4057211\ldots$, from 8 pm - 8 am with a total of 48 periods

**Simple Linear Regression**

We use the wind power production as the independent variable. Therefore, we consider all wind power parks in Belgium on- and offshore. We first sum all the different locations, so that we have the total generated wind power in Belgium. The dependent variable is the electricity price. Now we are able to calculate the correlation coefficient $r$. This was found to be $r = 0.255\ldots$ for the night of 3 - 4 May. Recall that, the correlation coefficient ranges from zero to one, where a coefficient around zero implies a weak linear relationship. Hence, based on the correlation coefficient the linear relationship is low. Further, we would expect a negative correlation, as the generation of wind energy is supposed to drop the energy prices [10]. In Figure 8 the scatter plot of the wind production against the electricity prices is shown with the according least square regression line, which is described by

$$y = 4.544\ldots 10^{-5}x + 0.222\ldots.$$

It can be seen that the line has a very gentle slope. We validated this result with three other nights. The result can be found Table 6. Some of the resulting correlation coefficients do suggest a (weak) negative correlation. So the simple linear regression method is not fitting for every night.

TABLE 6: Results simple linear regression for different dates

| Date of Night | Correlation Coefficient $r$ |
|---|---|
| 28 - 29 April | -0.194... |
| 03 - 04 May | 0.255... |
| 18 - 19 May | -0.662... |
| 22 - 23 May | -0.305... |

It follows from our results of the simple linear regression that there was no (consistent) correlation between wind production and short term price fluctuations. This is in line with the research in [12] which showed that the wind production had little effect on the short term fluctuations in electricity prices. Therefore, we chose to not continue with this method.

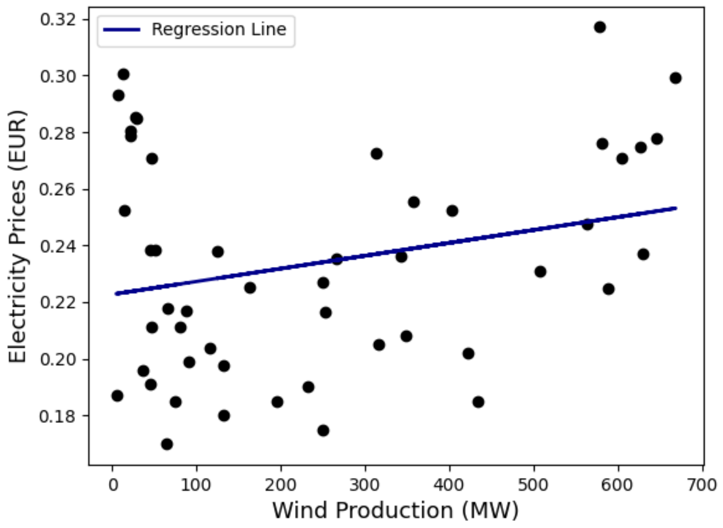


FIGURE 8: Relation between wind production and electricity prices with according least square regression line of the time series from 8 pm - 8 am with 15 minute periods (resulting in a total of 48 periods) from 3 - 4 May.

**Auto Regressive Integrated Moving Average**

The auto regressive integrated moving average (ARIMA) method is also used to forecast one period ahead. The optimal ARIMA parameters for $p, d, q$ are shown in Table 7.

TABLE 7: Optimal ARIMA-parameters for different dates.

| Date of Night | Optimal $(p, d, q)$ | MAD |
|---|---|---|
| 28 - 29 April | (0, 1, 1) | 0.018996... |
| 03 - 04 May | (0, 1, 1) | 0.024316... |
| 18 - 19 May | (0, 1, 1) | 0.034820... |
| 22 - 23 May | (0, 1, 1) | 0.019154... |

All time series have the same optimal ($p$, $d$, $q$) values, namely $(0, 1, 1)$. Again, the forecast series and the true timeseries is shown for May 3 - 4 in Figure 9.

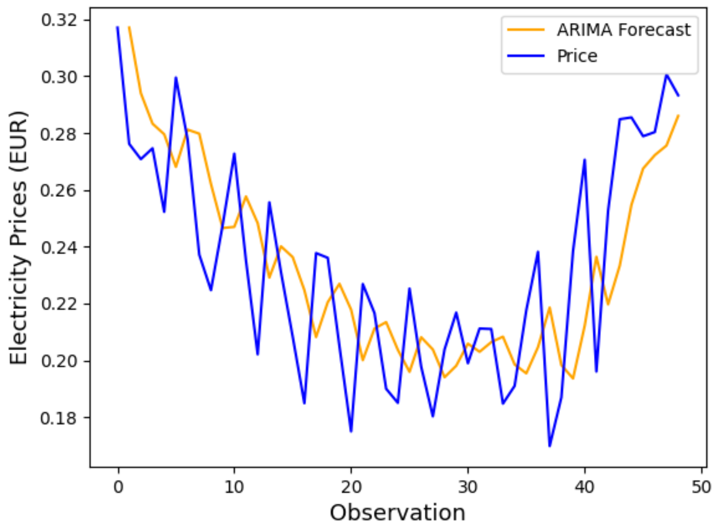


FIGURE 9: ARIMA forecast for May 3 - 4, from 8 pm - 8 am with 48 periods of 15 minutes

### 8.4.1 Comparison

Comparing the different forecasting methods, we find that the MAD for the moving-average, simple exponential smoothing and ARIMA are close together. The average MAD for the moving-average, simple exponential smoothing and ARIMA are $\overline{MAD} \approx 0.023533$, $\overline{MAD} \approx 0.023995$, and $\overline{MAD} \approx 0.024321$ respectively. Therefore, one forecasting is not performing significantly better.

## 8.5 Weather Dependence

To find a possible correlation between the wind production and the course of electricity prices, we use data from several days in the period between 28 April and 5 June (we used data of 17 nights). First, we select the period of time that we consider, which is 8 pm - 8 am, then we have to sum over all these periods to obtain the total wind production over one night. Similar for the prices, but we use the average, so we divide by the number of periods.

The resulting correlation coefficient is $r = -0.714\ldots$, implying a strong negative linear relationship. The corresponding least squares linear regression line is

$$y = -6.204\ldots 10^{-7}x + 0.231\ldots$$

and is also shown in Figure 10.

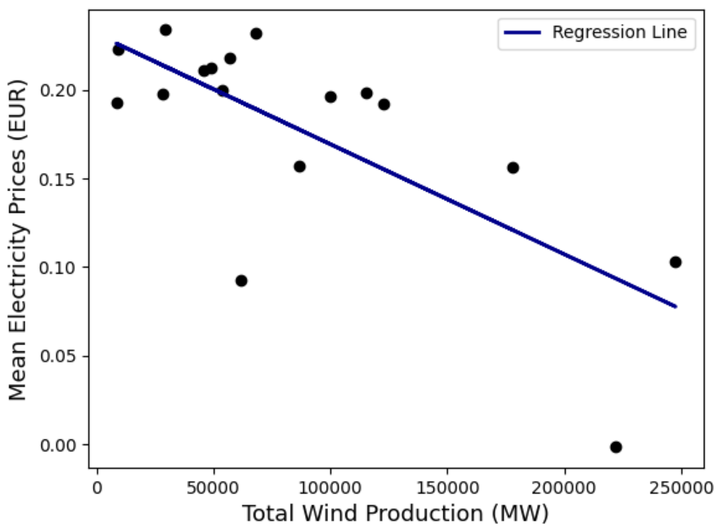


FIGURE 10: Relation between the total overnight wind production and mean electricity prices with the according least square regression line.

Then we need to find a suitable value for $M$ (see Section 5.2) to determine whether we are in state 0 or state 1.

We find the mean value of the prices, this is $\mu = 0.177\ldots$ The corresponding total wind production according to the regression line equals $M = 87160.594\ldots$.

The resulting transition probability matrix is

$$\mathbf{A} = \begin{bmatrix} 2/3 & 1/3 \\ 1/2 & 1/2 \end{bmatrix}.$$

## 8.6 Stochastic Dynamic Program with Markov chain

For the stochastic dynamic program with the Markov chain (which we now also refer to as the SDP with dependent prices), we use

- $L = 8$ kWh,
- $u_{max} = 1$ kWh,
- $\mu_1 = 0.164$ with $\sigma_1 = 0.0470$,
- $\mu_2 = 0.193$ with $\sigma_2 = 0.038$,
- $R_1$ : range of price values from 0 to 0.400 with stepsize 0.001,
- $R_2$ : range of price values from 0 to 0.350 with stepsize 0.001.

Now we determine the constraints for our transition probability matrix. For the Markov chain we use both $\mu_1 = 0.164$ with $\sigma_1 = 0.047$ and $\mu_2 = 0.193$ with $\sigma_2 = 0.038$. Then for the transition probability matrix with four states, we obtain the following parameters to use in the constraints (19)-(22).

$$\begin{aligned} a_1 &= 0.164\ \cdots - 0.047\cdots \approx 0.117 & a_2 &= 0.193\ \cdots - 0.038\cdots \approx 0.155 \\ b_1 &= 0.164 \qquad\qquad\qquad\qquad \text{and} & b_2 &= 0.193 \\ c_1 &= 0.164\ \cdots + 0.047\cdots \approx 0.211 & c_2 &= 0.193\ \cdots + 0.038\cdots \approx 0.231. \end{aligned}$$

We get the following four-state probability transition matrices $\mathbf{P_4}(\mu)$ with $\mu = \mu_1, \mu_2$

$$\mathbf{P_4}(\mu_1) = \begin{bmatrix} 0.889\ldots & 0.055\ldots & 0.027\ldots & 0.027\ldots \\ 0.063\ldots & 0.609\ldots & 0.262\ldots & 0.063\ldots \\ 0.013\ldots & 0.203\ldots & 0.615\ldots & 0.168\ldots \\ 0.015\ldots & 0.005\ldots & 0.25 & 0.729\ldots \end{bmatrix}$$

$$\mathbf{P_4}(\mu_2) = \begin{bmatrix} 0.592\ldots & 0.283\ldots & 0.061\ldots & 0.061\ldots \\ 0.144\ldots & 0.659\ldots & 0.175\ldots & 0.020\ldots \\ 0.020\ldots & 0.289\ldots & 0.455\ldots & 0.234\ldots \\ 0.027\ldots & 0.009\ldots & 0.379\ldots & 0.583\ldots \end{bmatrix}.$$

We also use both ranges $R_1$ and $R_2$ for the price values. The resulting costs can be found in Table 8 under "$D(\mu, R)$" for $\mu = \mu_1, \mu_2$ and $R = R_1, R_2$. We can conclude that $R_1$ with $\mu_2 = 0.193$ gives the lowest cost increase with respect to the optimal strategy and hence, on average has the lowest costs.

TABLE 8: Minimized costs for different dates for the dependent SDP with Markov chain with 4 states $D_4(\mu, R)$ with $\mu = \mu_1, \mu_2$ and $R = R_1, R_2$

| | **Minimized costs (€)** | | | |
|---|---|---|---|---|
| **Date of Night** | $D_4(\mu_1, R_1)$ | $D_4(\mu_1, R_2)$ | $D_4(\mu_2, R_1)$ | $D_4(\mu_2, R_2)$ |
| 28- 29 April | 1.88938 | 1.97840 | 1.57501 | 1.92001 |
| 03- 04 May | 1.88545 | 2.0025 | 1.70824 | 1.88317 |
| 18- 19 May | 1.18339 | 1.20396 | 1.17526 | 1.10276 |
| 06- 07 June | 1.01760 | 1.04375 | 1.02163 | 0.98522 |
| 07- 08 June | 1.32338 | 1.46242 | 1.32163 | 1.3245 |
| 08- 09 June | 1.15392 | 1.35695 | 1.17485 | 1.24219 |
| Cost increase | 18% | 27% | 12% | 18% |

To possibly lower the costs even more, we consider a transition probability matrix with 6 states. We use the following constraints to determine whether a price value $p$ is in state 0, 1, 2, 3, 4, or 5.

$$\text{State 0 if } p < \mu - \sigma \tag{26}$$
$$\text{State 1 if } \mu - \sigma \leq p < \mu - \sigma/2 \tag{27}$$
$$\text{State 2 if } \mu - \sigma/2 \leq p < \mu \tag{28}$$
$$\text{State 3 if } \mu \leq p < \mu + \sigma/2 \tag{29}$$
$$\text{State 4 if } \mu + \sigma/2 \leq p < \mu + \sigma \tag{30}$$
$$\text{State 5 if } \mu + \sigma \leq p \tag{31}$$

We again take $\mu = \mu_1, \mu_2$ with $\sigma = \sigma_1, \sigma_2$, respectively. This gave the following six-state probability transition matrices $\mathbf{P_6}(\mu)$ with $\mu = \mu_1, \mu_2$

$$\mathbf{P_6}(\mu_1) = \begin{bmatrix} 0.889\ldots & 0.036\ldots & 0.018\ldots & 0.018\ldots & 0.009\ldots & 0.027\ldots \\ 0.127\ldots & 0.553\ldots & 0.148\ldots & 0.127\ldots & 0.042\ldots & 0.0 \\ 0.031\ldots & 0.117\ldots & 0.446\ldots & 0.244\ldots & 0.063\ldots & 0.095\ldots \\ 0.013\ldots & 0.034\ldots & 0.227\ldots & 0.289\ldots & 0.144\ldots & 0.089\ldots \\ 0.012\ldots & 0.0 & 0.098\ldots & 0.308\ldots & 0.271\ldots & 0.308\ldots \\ 0.015\ldots & 0.0 & 0.005\ldots & 0.096\ldots & 0.153\ldots & 0.729\ldots \end{bmatrix}$$

and

$$\mathbf{P_6}(\mu_2) = \begin{bmatrix} 0.592\ldots & 0.172\ldots & 0.111\ldots & 0.049\ldots & 0.012\ldots & 0.061\ldots \\ 0.151\ldots & 0.454\ldots & 0.212\ldots & 0.080\ldots & 0.090\ldots & 0.010\ldots \\ 0.136\ldots & 0.294\ldots & 0.357\ldots & 0.105\ldots & 0.073\ldots & 0.031\ldots \\ 0.044\ldots & 0.117\ldots & 0.205\ldots & 0.294\ldots & 0.132\ldots & 0.205\ldots \\ 0.0 & 0.038\ldots & 0.220\ldots & 0.233\ldots & 0.246\ldots & 0.259\ldots \\ 0.027\ldots & 0.0 & 0.009\ldots & 0.092\ldots & 0.287\ldots & 0.583\ldots \end{bmatrix}.$$

The resulting costs for the different policies are shown in Table 9. Some dates give a lower cost than $D_4(\mu_2, R_1)$. However, looking at the cost increase with respect to the optimal solution, we determine that none has a lower cost increase than $D_4(\mu_2, R_1)$. The strategy $D_4(\mu_2, R_1)$ has a cost increase of 12 %. Therefore, we take $D_4(\mu_2, R_1)$ as the "Dependent (MC)" strategy.

Table 9: Minimized costs for different dates for the dependent SDP with Markov chain with 6 states $D_6(\mu, R)$ with $\mu = \mu_1, \mu_2$ and $R = R_1, R_2$.

| | **Minimized costs (€)** | | | |
|---|---|---|---|---|
| **Date of Night** | $D_6(\mu_1, R_1)$ | $D_6(\mu_1, R_2)$ | $D_6(\mu_2, R_1)$ | $D_6(\mu_2, R_2)$ |
| 28- 29 April | 1.84984 | 1.94805 | 1.61646 | 1.91099 |
| 03- 04 May | 1.8617 | 1.88942 | 1.67789 | 1.88317 |
| 18- 19 May | 1.09287 | 1.01976 | 1.29537 | 1.09126 |
| 06- 07 June | 1.06567 | 1.03201 | 1.02163 | 0.98522 |
| 07- 08 June | 1.20733 | 1.21757 | 1.3273 | 1.32345 |
| 08- 09 June | 1.15392 | 1.21848 | 1.2424 | 1.24219 |
| Cost increase | 15% | 16% | 14% | 17% |

## 8.7 Deterministic Knapsack

We forecast the timeseries using Holt-Winter's simple exponential smoothing as in Section 4. We know that the length of the seasonal period equals the number of periods in one night, which is $T = 48$. We used the two preceding nights for the forecast. We experimented with different numbers of nights to include, but the most accurate results were returned with only using the two preceding nights. For 6 - 7 June we obtain the forecast shown in Figure 11.

Then we use the knapsack method as we did in Section 2 to determine the optimal policy. Next we perform this policy on the true values. These resulting costs are shown in Table 10. The dates 28 - 29 April, 3 - 4 May, 18 - 19 May are excluded, as the preceding data was not available.

Table 10: Results of the Deterministic Knapsack strategy

| Date of Night | MAD forecast | Minimized costs |
|---|---|---|
| 6 - 7 June | 0.0634… | 1.34008 |
| 7 - 8 June | 0.110… | 1.41895 |
| 8 - 9 June | 0.0538… | 1.32699 |
| Cost increase | - | 23% |

## 8.8 Comparison Charging Strategies

We illustrate the different strategies for 6 - 7 June in Figure 12. In this figure are the true price values shown, these run from 8 pm to 8 am in periods of 15 minutes. We show

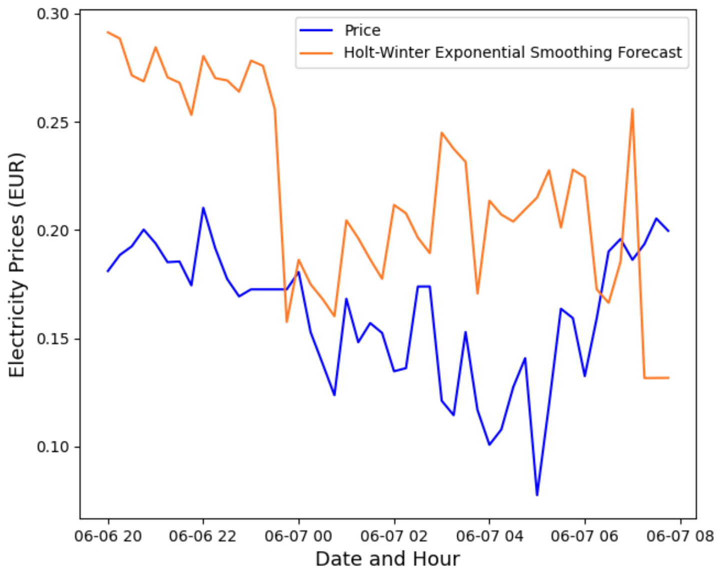


FIGURE 11: Holt-Winter Exponential Smoothing forecast for the night 6 - 7 June.

the Knapsack, Independent , Dependent (MC) and the Deterministic Knapsack strategy.

The charging times of the policies are shown in the graph by a $\star$. Each charging time we charge $u_{max} = 1$ kWh during the 15 minute period. In Figure 12, we see that the periods that are chosen by the Deterministic Knapsack strategy are not optimal, since the forecast timeseries peaks at slightly different times than the true price values. The dependent strategy somewhat follows the most optimal strategy, and the independent strategy deviates more.

The costs for this night can be found in the previous tables or below in Table 11. The total costs for all six nights are also shown and the cost increase compared to the optimal strategy. For the Deterministic Knapsack, we determined the cost increase for the last 3 dates. We see that the Dependent (MC) strategy has the lowest cost increase and therefore approaches the optimal strategy. The Dependent (MC) strategy is on average

$$\frac{(8.40066 - 7.97122)}{6} \approx €0.07,$$

lower per night, which is a decrease of

$$\frac{(8.40066 - 7.97122)}{8.40066} \cdot 100\% \approx 5\%. \tag{32}$$

Table 11: Minimized costs for different strategies

| | **Minimized costs (€)** | | | |
|---|---|---|---|---|
| Date | Knapsack | Det. Knapsack | Independent | Dependent (MC) |
| 28 - 29 April | 1.45698 | - | 1.95484 | 1.57501 |
| 3 - 4 May | 1.52983 | - | 1.87519 | 1.70284 |
| 18 - 19 May | 1.00775 | - | 1.087 | 1.17526 |
| 6 - 7 June | 0.88145 | 1.12161 | 1.10755 | 1.02163 |
| 7 - 8 June | 1.19141 | 1.41895 | 1.22317 | 1.32163 |
| 8 - 9 June | 1.08151 | 1.32699 | 1.15291 | 1.17485 |
| Total costs | 7.14264 | 3.86755 | 8.40066 | 7.97122 |
| Cost increase | 0% | 23% | 18% | 12% |

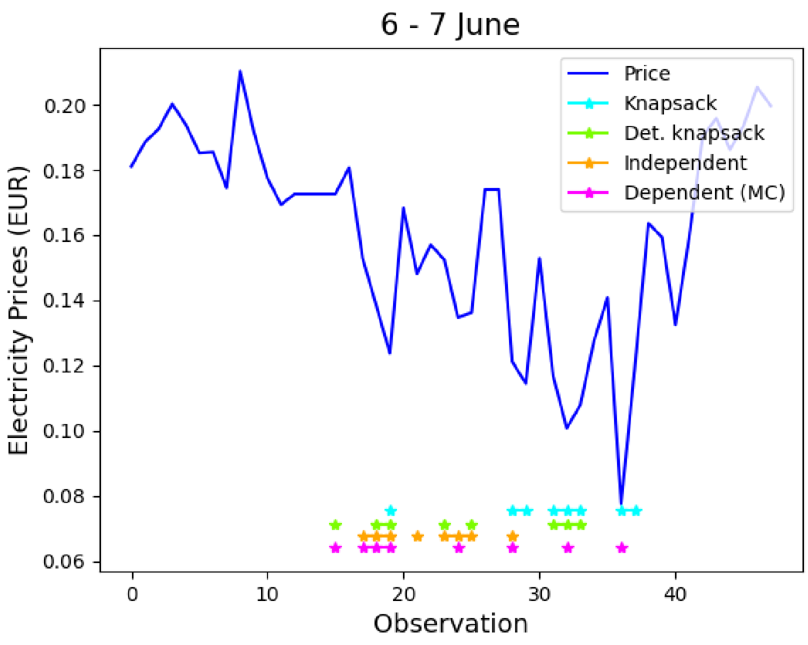


Figure 12: The policies for 6 - 7 June, the charging times are denoted by a $\star$

## 9 Discussion

The knapsack problem gives the most optimal solution and is used to classify the performance of the other models. Considering all past values makes the stochastic dynamic program computationally intractable. Therefore, to consider the dependency between periods, we used a Markov chain.

From the simulations in Section 8, we found that on some days the Independent strategy has lower costs than the Dependent (MC) strategy. Nevertheless, the dependent strategy performs better on average. We saw that the total costs increases with 18% for the Independent strategy, while the total costs for the Dependent strategy only increases 12%, compared to the optimal strategy. This is in line with our expectations as prices are generally not independent and identically distributed. Nevertheless, the Independent strategy does perform well. More precisely, on the days where the prices are closer to the long run mean. This makes sense, since the normal distribution then approaches the prices more.

We were unable to find literature that used a similar approach by implementing the price dependency with a Markov chain.

Further, we tried a different approach to possibly improve the results. Here, we used the Holt-Winter's exponential smoothing with seasonality to perform an out-of-sample forecast. We saw that the forecast was not very accurate. Hence, unsurprisingly the Deterministic Knapsack strategy returned did not return a more optimal strategy. This can be a subject of further improvement and we could also test the ARIMA-method with seasonality to possibly improve the forecast.

We also noticed that the training data is very important to the result, as a few outliers can already result in less optimal strategies. Therefore, it would be interesting to perform more simulations using a bigger sample size to test the results during a year and possibly also work with a moving window. Besides, we could also forecast the expected average price for today and use this value in the price distributions for the Independent and Dependent (MC) strategies. Further, it would be interesting to implement the conceptual model that considers the weather dependence. Another interesting subject for further research could be to consider a whole fleet of cars that needs to charge.

## 10 Conclusion

In this paper we compared different charging strategies to find whether price dependency between periods can be used to an advantage. This price dependency was attempted to be included by means of the wind power production. But, we found that there's no correlation between short term price fluctuations and the wind production. Since, we did find a correlation between the average energy prices and total wind production per night, we presented a conceptual model that adapts the amount of energy to charge per day, dependent on the amount of wind production. We included the price dependency in the stochastic dynamic program by use of a Markov chain.

Real-world data was used to determine the performance of the different strategies. We found that the stochastic dynamic program with price dependency approaches, on average the optimal strategy best. On average the total costs are €0.07 lower per night, which is an improvement of 5% compared to the strategy with independent prices.

Therefore, coming back to our research question: Is there a significant cost reduction when the price dependency is taken into account in determining a charging strategy? We conclude that there is a significant reduction in the minimized costs.

## 11 References

[1] ANWB. Hier staan de laadpalen. https://www.anwb.nl/auto/elektrisch-rijden/waar-staan-de-oplaadpunten, visited at 5-5-2022.

[2] R. Boucherie and N.M. van Dijk (Eds.). *Markov Decision Processes in Practice*. International Series in Operations Research; Management Science. Springer, 2017. 10.1007/978-3-319-47766-4.

[3] CBS. Markt voor energieprijzen. https://www.cbs.nl/nl-nl/dossier/energieprijzen/hoofdstukken/markt-voor-energieprijzen, visited at 29-4-2022.

[4] EIA. Electricity explained, factors affecting electricity prices, April 20, 2022. https://www.eia.gov/energyexplained/electricity/prices-and-factors-affecting-prices.php, visited at 2-5-2022.

[5] Elia. Wind power generation data. https://www.elia.be/en/grid-data/power-generation/wind-power-generation, (2022).

[6] Epexspot. Data. https://www.epexspot.com/en/market-data (2022).

[7] T. Fraser. What is a good energy consumption figure for electric vehicles? https://www.drive.com.au/caradvice/what-is-a-good-energy-consumption-figure-for-electric-vehicles/, visited at 9-6-2022.

[8] R.J. Hyndman and G. Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2 edition. https://otexts.com/fpp2/.

[9] Roland Irle. Global EV Sales for 2021. https://www.ev-volumes.com/country/total-world-plug-in-vehicle-volumes/, visited at 29-4-2022.

[10] D. Keles, M. Genoese, D. Möst, S. Ortlieb, and W. Fichtner. A combined modeling approach for wind power feed-in and electricity spot prices. *Energy Policy*, pages 213–225, 2013. 10.1016/j.enpol.2013.03.028.

[11] P. kemper, N. van Dijk, W. Scheinhardt, H. van den Berg, and J. Hurink. Optimization of Charging Strategies for Electric Vehicles in Powermatcher-Driven Smart Energy Grids. *Valuetools*, 2015. 10.4108/eai.4-1-2016.151091.

[12] Š. Lyócsa L. P. C. Do and P. Molnár. Impact of wind and solar production on electricity prices: quantile regression approach. *Journal of the Operational Research Society*, pages 1752–1768, 2019. 10.1080/01605682.2019.1634783.

[13] J. Laarakkers. Powermatcher, Matching Energy Supply and Demand To Expand Smart Energy Potential. 14-04-2016. https://www.tno.nl/media/1986/tno-powermatcher-jrv140416-01.pdf.

[14] Pod Point. How Long Does It Take to Charge an Electric car?, 11-11-2021. https://pod-point.com/guides/driver/how-long-to-charge-an-electric-car, visited at 5-5-2022.

[15] Love Energy Savings. Off-Peak Electricity Explained: Is Electricity Cheaper at Night? https://www.loveenergysavings.com/content-hub/energy-guides-and-advice/off-peak-electricity-explained-is-electricity-cheapest-at-night/, visited at 2-5-2022.

[16] V. Silva, L. Glorieux, C. Kieny, M. Ortega-Vazquez, B. Roussien, J. Laaraakers, C. Matrose, and M. Bolczek. Estimation of Innovative Operational Processes and Grid Management for the Integration of EV. 2011.

[17] W. L. Winston. *Operations Research; Applications and Algortithms.* Thomson Learning, 4th edition edition, 2004.

# 12 Appendix

## 12.1 List of symbols

| Symbol | Description |
|---|---|
| $\mathbf{A}$ | state transition matrix for wind production |
| $a, b, c$ | bounds to determine categories for prices for the probability transition matrix |
| $C_d$ | amount of energy left in car battery at day $d$ |
| $c$ | total costs |
| $I(\mu, R)$ | Independent strategy with $\mu$ and $R$ |
| $L$ | total amount of energy to be charged at the end of period $T$ |
| $L_d$ | total amount of energy that we want to charge on day $d$ |
| $M$ | parameter for state determination in wind production |
| MAD | Mean average deviation |
| $p, d, q$ | ARIMA parameters |
| $P_t$ | stochastic price variable for period $t$ |
| $p_t$ | price value at time $t$ per unit energy |
| $\mathbb{P}$ | probability distribution |
| $\mathbf{P}$ | probability transition matrix |
| $p_{ij}$ | probability of transition from state $i$ to state $j$ |
| $\tilde{p}$ | forecast price value |
| $R_1$ | range of price values from 0 to 0.400 with stepsize 0.001 |
| $R_2$ | range of price values from 0 to 0.350 with stepsize 0.001 |
| $r$ | correlation coefficient |
| $s_d$ | state of wind production at day $d$ |
| $t$ | current period |
| $T$ | the total amount of periods available to charge |
| $u_{max}$ | maximum amount of energy that can be charged during any period |
| $u_t$ | amount of energy charged during period $t$ |
| $V_t(x)$ | the expected costs when an amount of $x$ is still to charge at time $t$ |
| $V_t(x, p)$ | the expected costs when an amount of $x$ is still to charge at time $t$ and current price is $p$ |
| $x_t$ | amount of energy to still be charged at period $t$ |
| $\mu$ | mean |
| $\mu_1$ | $\approx 0.163$ mean of unfiltered data |
| $\mu_2$ | $\approx 0.193$ mean of filtered data |
| $\sigma_1$ | $\approx$, standard deviation of unfiltered data |
| $\sigma_2$ | $\approx$, standard deviation of filtered data |
| $\alpha$ | simple exponential smoothing parameter |

## 12.2 Stochastic dynamic program with independent prices; continued

To find explicit expressions for $V_t(x,p)$ and $u_t(x,p)$ in [11], they continue by using order statistics for $P_t$. We only know the long-run average price, hence the prices are assumed to be independent ánd identically distributed (i.i.d.). So we have the random variables $P_t$ that, describe the price per unit of energy in period $t$, for $t = 1, \dots, T$. The variable $P_t$ has a distribution with an expectation equal to the long run expected average price. Then an optimal decision rule is given by [11]

$$u_t(x,p) = \begin{cases} 0 \quad \text{if} \quad p > \mathbb{E}P^{t+1}_{(k+1)} \\ x - ku_{max} \quad \text{if} \quad \mathbb{E}P^{t+1}_{(}k) < p \leq \mathbb{E}P^{t+1}_{(k+1)} \\ \min(u_{max}, x) \quad \text{if} \quad p \leq \mathbb{E}P^{t+1}_{(}k), \end{cases} \tag{33}$$

with its corresponding minimal cost [11]:

$V_t(x,p) = \infty$ if $\frac{x}{u_{max}} > T - t + 1$, or else

$V_t(x,p) = u_{max} \sum_{l=1}^{k} \mathbb{E}[P^t_{(l)} | P_t = p] + (x - ku_{max})\mathbb{E}[P^t_{(k+1)} | P_t = p]$

Which can be computationally demanding, hence they introduce an heuristic for the optimal control law:

$$u_t(x,p) = \begin{cases} 0 \quad \text{if} \quad F(p) \leq \frac{k}{T-t+1} \\ x - ku_{max} \quad \text{if} \quad \frac{k}{T-t+1} < F(p) \leq \frac{k+1}{T-t+1} \\ \min(u_{max}, x) \quad \text{if} \quad F(p) > \frac{k}{T-t+1}, \end{cases} \tag{34}$$

## 12.3 Forecasting methods; continued

### Moving-Average Method

Moving-average method is an extrapolation forecasting method. It uses the average of the last $N$ observations of a series of past data $(x_1, x_2, \dots, x_t, \dots)$ to predict value of the next period. The variable $f_{t,1}$ denotes the forecast for period $t+1$. We get

$$f_{t,1} = \frac{x_t + x_{t-1}, \dots, x_{t-N+1}}{N}, \tag{35}$$

where $N$ is a chosen parameter. The forecast accuracy depends on $N$. The MAD is again given by Equation (9), with forecast error $e_t$;

$$e_t = x_t - f_{t,1} \quad [17]. \tag{36}$$

The moving-average method works well for a time series that fluctuates around a certain value, the base value, but it is not very accurate for sudden fluctuations. Hence, the moving-average method is less suitable for time series with a trend or seasonality [17].

### Auto Regression Integrated Moving Average

An autoregressive model predicts future values based on past values, as extrapolation methods. Autoregressive models assume that the future will resemble the past, it examines the differences between the values in the time series instead of through actual values. An Auto Regression Integrated Moving Average (ARIMA) model combines the autoregressive features with that of moving averages. It makes use of lagged moving averages to smooth the time series. Hence, ARIMA models can take into account trends, cycles and seasonality when making forecasts.Time series regression predicts the behaviour of dynamic systems from current and past observations. [8]

ARIMA uses three parameters $p, d, q$, where each parameter has an integer value that indicate the type of ARIMA model. The parameter $p$ is the "AR"-part of ARIMA. It denotes the lag order, so the number of autoregressive terms in the model. The parameter $d$ is the "I"-part and denotes the degree of differencing needed to make the time series stationary. Finally, $q$ is the "MA"-part and represents the order of the moving average (size of the moving window) [8].

The ARIMA model is then described by:

$$\left(1 - \sum_{i=1}^{p} \phi_i B^i\right)(1 - B)^d y_t = c + \left(1 + \sum_{i=1}^{q} \theta_i B^i\right) \varepsilon_t, \tag{37}$$

where $B$ is the lag operator, $\phi_i$ are the parameters of the autoregressive part, the parameters $\theta_i$ are from the moving average part, $\epsilon_i$ are the error terms and $c$ is a constant [8].

In case the timeseries is not stationary, it should be converted to one that is in order to apply the ARIMA model. A time series is stationary if it does not have trend of seasonal effects. Hence the data may need to be prepped. It can be made stationary by differencing a $d$ number of times. Differencing is done by computing the difference between consecutive observations:

$$x'_t = x_t - x_{t-1} \tag{38}$$

This stabilizes the mean of the time series. It may be possible that the series needs to be differenced more than once, before it is stationary The series is stationary when the p-value is below the treshold of 0.05. Selecting the right values for $p$, $d$, and $q$ can be tricky. Fortunately, the function auto_arima will pick the most optimal values automatically [8].

## 12.4 Example of Data

This appendix is purposely excluded for licensing reasons.

## 12.5 Test Data Summary

Table 12: Average price, minimum price, and maximum price of one night

| Date of Night | Mean | Min | Max |
|---|---|---|---|
| 28- 29 April | 0.23219... | 0.18703 | 0.30127 |
| 03- 04 May | 0.23379... | 0.16986 | 0.31712 |
| 18- 19 May | 0.21218... | 0.05008 | 0.3412 |
| 06- 07 June | 0.16134229... | 0.07754 | 0.21028 |
| 07- 08 June | 0.188202... | 0.13594 | 0.25314 |
| 08- 09 June | 0.2103108... | 0.07991 | 0.31971 |

## 12.6 Policy 6 - 7 June

| | | **Amount to charge (kW)** | | | |
|---|---|---|---|---|---|
| **Time** | **Price** | **Knapsack** | **Det. knapsack** | **Independent** | **Dependent** |
| 20:00:00 | 0.18106 | 0 | 0 | 0 | 0 |
| 20:15:00 | 0.18849 | 0 | 0 | 0 | 0 |
| 20:30:00 | 0.19247 | 0 | 0 | 0 | 0 |

|  |  |  |  |  |  |
|---|---|---|---|---|---|
| 20:45:00 | 0.20021 | 0 | 0 | 0 | 0 |
| 21:00:00 | 0.19382 | 0 | 0 | 0 | 0 |
| 21:15:00 | 0.18515 | 0 | 0 | 0 | 0 |
| 21:30:00 | 0.18547 | 0 | 0 | 0 | 0 |
| 21:45:00 | 0.17442 | 0 | 0 | 0 | 0 |
| 22:00:00 | 0.21028 | 0 | 0 | 0 | 0 |
| 22:15:00 | 0.19164 | 0 | 0 | 0 | 0 |
| 22:30:00 | 0.17742 | 0 | 0 | 0 | 0 |
| 22:45:00 | 0.16933 | 0 | 0 | 0 | 0 |
| 23:00:00 | 0.17259 | 0 | 0 | 0 | 0 |
| 23:15:00 | 0.17259 | 0 | 0 | 0 | 0 |
| 23:30:00 | 0.17259 | 0 | 0 | 0 | 0 |
| 23:45:00 | 0.17259 | 0 | 1 | 0 | 1 |
| 00:00:00 | 0.18062 | 0 | 0 | 0 | 0 |
| 00:15:00 | 0.15282 | 0 | 0 | 1 | 1 |
| 00:30:00 | 0.13838 | 0 | 1 | 1 | 1 |
| 00:45:00 | 0.12368 | 1 | 1 | 1 | 1 |
| 01:00:00 | 0.16827 | 0 | 0 | 0 | 0 |
| 01:15:00 | 0.14816 | 0 | 0 | 1 | 0 |
| 01:30:00 | 0.15702 | 0 | 0 | 0 | 0 |
| 01:45:00 | 0.15245 | 0 | 0 | 1 | 0 |
| 02:00:00 | 0.13475 | 0 | 0 | 1 | 1 |
| 02:15:00 | 0.13618 | 0 | 0 | 1 | 0 |
| 02:30:00 | 0.1739 | 0 | 0 | 0 | 0 |
| 02:45:00 | 0.17392 | 0 | 0 | 0 | 0 |
| 03:00:00 | 0.12113 | 1 | 0 | 1 | 1 |
| 03:15:00 | 0.11445 | 1 | 0 | 0 | 0 |
| 03:30:00 | 0.15287 | 0 | 0 | 0 | 0 |
| 03:45:00 | 0.11685 | 1 | 1 | 0 | 0 |
| 04:00:00 | 0.10077 | 1 | 0 | 0 | 1 |
| 04:15:00 | 0.10785 | 1 | 0 | 0 | 0 |
| 04:30:00 | 0.12742 | 0 | 0 | 0 | 0 |
| 04:45:00 | 0.14081 | 0 | 0 | 0 | 0 |
| 05:00:00 | 0.07751 | 1 | 0 | 0 | 1 |
| 05:15:00 | 0.11921 | 1 | 0 | 0 | 0 |
| 05:30:00 | 0.16363 | 0 | 0 | 0 | 0 |
| 05:45:00 | 0.15932 | 0 | 0 | 0 | 0 |
| 06:00:00 | 0.13251 | 0 | 0 | 0 | 0 |
| 06:15:00 | 0.15916 | 0 | 0 | 0 | 0 |
| 06:30:00 | 0.19012 | 0 | 1 | 0 | 0 |
| 06:45:00 | 0.19586 | 0 | 0 | 0 | 0 |
| 07:00:00 | 0.18623 | 0 | 0 | 0 | 0 |
| 07:15:00 | 0.1935 | 0 | 1 | 0 | 0 |
| 07:30:00 | 0.20533 | 0 | 1 | 0 | 0 |
| 07:45:00 | 0.19963 | 0 | 1 | 0 | 0 |

## 12.7 Python Code: Knapsack Model

```python
import pandas as pd
import numpy as np
import math

def valuefunction(T, x0, umax, df):
    """
    :param T: Total amount of available periods
    :param x0: Total amount of charging needed
    :param umax: Maximum charging speed
    :param df: Dataframe containing the timeseries of the prices
    :return: Results of the optimal value function
    :return: The corresponding optimal decisions
    """
    prices_list = df['Price'].values.tolist()  # Timeseries of prices in list
    states = [i / 10 for i in range(0, 10 * x0 + 1, int(10 * umax))]
    D = [0, umax] # Decisions
    if x0 <= 0:
        return 'no charging needed'
    elif x0 > umax * T:
        return 'car can not be fully charged during one night'
    else:
        V = np.full(shape=(T, math.ceil(x0/umax) + 1), fill_value=np.inf)
        choice = np.full(shape=(T, math.ceil(x0/umax) + 1), fill_value=np.inf)
        for x in states: # Last state
            if x == 0: # No charging needed
                u = 0
                V[T - 1, states.index(x)] = 0
                choice[T - 1, states.index(x)] = u
            elif 0 < x <= umax: # Only one (or partial) charging period needed
                u = x
                cost = u * prices_list[-1]
                V[T - 1, states.index(x)] = cost
                choice[T - 1, states.index(x)] = u
            else: # Charge is not be fully charged at T + 1
                u = np.inf
                cost = np.inf
                V[T - 1, states.index(x)] = cost
                choice[T - 1, states.index(x)] = u

        for t in range(T - 2, -1, -1):
            for x in states:
                V_all = [] # Empty list to store all values of all decisions
                if x == 0: # No charging needed
                    u = 0
                    V[t, states.index(x)] = 0
                    choice[t, states.index(x)] = u
                elif 0 < x <= umax: # Only one (or partial) charging period needed
                    for u in [0, x]:
                        cost = 0
                        cost += u * prices_list[t] + V[t + 1, states.index(x - u)]
                        V_all.append(cost)
                    V[t, states.index(x)] = min(V_all)
                    choice[t, states.index(x)] = [0,x][V_all.index(min(V_all))]
                elif umax < x <= umax * (T-t): # More than one charging period needed and at least this many periods left
```

```python
                    for u in D: # Either charge on maximum speed or not
                        cost = 0
                        cost += u * prices_list[t] + V[t + 1, states.index(x - u)]
                        V_all.append(cost)
                    V[t, states.index(x)] = min(V_all)
                    choice[t, states.index(x)] = D[V_all.index(min(V_all))]
                else: # Car can not be fully charged at T + 1
                    u = np.inf
                    cost = np.inf
                    V[T - 1, states.index(x)] = cost
                    choice[T - 1, states.index(x)] = u

        return V, choice

### 8pm - 8am with regular intervals of 15 min -> 48 stages (first period starts at 8:00 last at 7:45) ###
stages = 48
start = 8
umaximum = 1

"""  UNCOMMENT THE DATE YOU WANT TO USE """
data = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot.xlsx', sheet_name='28-29April', usecols=[1,2],header=0, skiprows=[1]) #, skiprows=[1])
#data = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElelctricityPrices_EpexSpot_34.xlsx', sheet_name='03-04May', usecols=[1, 2], header=0, skiprows=[1])
#data = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElelctricityPrices_EpexSpot_1819.xlsx', sheet_name='18-19May', usecols=[1,2],header=0, skiprows=[1]) #, skiprows=[1])
#data = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElelctricityPrices_EpexSpot_2223.xlsx', sheet_name='22-23May', usecols=[1,2],header=0, skiprows=[1]) #, skiprows=[1])

""" EXECUTE THE VALUEFUNCTION """
V, choices = valuefunction(stages, start, umaximum, data)


""" PRINT RESULTS TO AN EXCEL FILE """

with pd.ExcelWriter("Output_Knapsack.xlsx") as writer:
    for t in range(stages):
        df1 = pd.DataFrame(V[:,:])
        df2 = pd.DataFrame(choices[:,:])
        df1.to_excel(excel_writer=writer, sheet_name=f'E(c)')
        df2.to_excel(excel_writer=writer, sheet_name=f'choices')
```

## 12.8 Python Code: Stochastic Dynamic Program with Independent Prices

```python
import pandas as pd
import numpy as np
import math


def prob(x):
    """
    :param x: Random variable
    :return: Probability of random variable x using normal distribution
    """
    mean = 1932     # Long run mean times 10000
    stdev = 458     # Long run standard deviation times 10000
    probability = 10*(1/ (stdev * (2 * math.pi)**(1/2))) * math.exp(-(x-mean)**2/(2*stdev**2))
    return probability

def valuefunction(T, x0, umax):
    """
    :param T: Total amount of available periods
    :param x0: Total amount of charging needed
    :param umax: Maximum charging speed
    :return: Results of the optimal value function
    :return: The corresponding optimal decisions
    """
    prices_list = [k/10000 for k in range(0, 4000 + 1, 10)]
    states = [i / 10 for i in range(0, 10 * x0 + 1, int(10 * umax))]
    D = [0, umax]     # Either charge on maximum speed or not

    if x0 <= 0:
        return 'no charging needed'
    else:
        V = np.full(shape=(T, math.ceil(x0/umax) + 1,len(prices_list)), fill_value=np.inf)
        choice = np.full(shape=(T, math.ceil(x0/umax) + 1, len(prices_list)), fill_value=np.inf)
        for x in states:
            if x == 0:
                u = 0
                V[T - 1, states.index(x), :] = 0
                choice[T - 1, states.index(x), :] = u
            elif 0 < x <= umax:
                for p in prices_list:
                    u = min(umax, x)
                    cost = u * p
                    V[T - 1, states.index(x), prices_list.index(p)] = cost
                    choice[T - 1, states.index(x), prices_list.index(p)] = u
            else:
                u = np.inf
                cost = np.inf
                V[T - 1, states.index(x), :] = cost
                choice[T - 1, states.index(x), :] = u


        for t in range(T - 2, -1, -1):
            print(t)
            for x in states:
                for p in prices_list:
                    V_all = []  # Empty list to store all values of all decisions
                    if x == 0:  # No charging needed
                        u = 0
                        V[t, states.index(x), prices_list.index(p)] = 0
                        choice[t, states.index(x), prices_list.index(p)] = u
                    elif 0 < x <= umax:  # Only one (or partial) charging period needed
                        for u in [0, x]:
                            cost = 0
                            for l in prices_list:
                                cost += prob(l*10000) * V[t + 1, states.index(x - u), prices_list.index(l)]
                            cost += u * p
                            V_all.append(cost)
                        V[t, states.index(x), prices_list.index(p)] = min(V_all)
                        choice[t, states.index(x), prices_list.index(p)] = [0, x][V_all.index(min(V_all))]
                    elif umax < x <= umax * (
                            T - t):  # More than one charging period needed and at least this many periods left
                        V_all = []
                        for u in D:
                            cost = 0
                            for l in prices_list:
                                cost += prob(l*10000) * V[t + 1, states.index(x - u), prices_list.index(l)]
                            cost += u * p
                            V_all.append(cost)
                        V[t, states.index(x), prices_list.index(p)] = min(V_all)
                        choice[t, states.index(x), prices_list.index(p)] = D[V_all.index(min(V_all))]

                    else:  # Charge can not be fully charged at T + 1
                        u = np.inf
                        cost = np.inf
                        V[T - 1, states.index(x), prices_list.index(p)] = cost
                        choice[T - 1, states.index(x), prices_list.index(p)] = u

        return V, choice

""" EXECUTE THE VALUEFUNCTION """
### 8pm - 8 am with regular intervals of 15 min gives a total of 48 stages ###
stages = 48
start = 8
umaximum = 1
V, choices = valuefunction(stages, start, umaximum)

""" PRINT RESULTS TO AN EXCEL FILE """
with pd.ExcelWriter("output_indep400.xlsx") as writer:
    for t in range(stages):
        df1 = pd.DataFrame(V[t,:,:])
        df2 = pd.DataFrame(choices[t, :,:])
        df1.to_excel(excel_writer=writer, sheet_name=f't={t} E(c)')
        df2.to_excel(excel_writer=writer, sheet_name=f't={t} choices')
```

## 12.9 Python Code: Stochastic Dynamic Program with Dependent Prices

```
import pandas as pd
import numpy as np
import math

"""  COLLECT PRICEDATA FOR TRANSITION PROBABILITY MATRIX """

data2223 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\
```

```python
ElelctricityPrices_EpexSpot_2223.xlsx', sheet_name='22-23May', usecols=[1,2],header=0)
data2324 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2324.xlsx', sheet_name='23-24May', usecols=[1,2],header=0)
data2425 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2425.xlsx', sheet_name='24-25May', usecols=[1,2],header=0)
data2526 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2526.xlsx', sheet_name='25-26May', usecols=[1,2],header=0)
data2627 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2627.xlsx', sheet_name='26-27May', usecols=[1,2],header=0)
data2728 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2728.xlsx', sheet_name='27-28May', usecols=[1,2],header=0)
data2829 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2829.xlsx', sheet_name='28-29May', usecols=[1,2],header=0)
data2930 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_2930.xlsx', sheet_name='29-30May', usecols=[1,2],header=0)
data3031 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_3031.xlsx', sheet_name='30-31May', usecols=[1,2],header=0)
data3101 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_3101.xlsx', sheet_name='31-1June', usecols=[1,2],header=0)
data60102 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_60102.xlsx', sheet_name='01-02June', usecols=[1,2],header=0)
data60203 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_60203.xlsx', sheet_name='02-03June', usecols=[1,2],header=0)
data60304 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_60304.xlsx', sheet_name='03-04June', usecols=[1,2],header=0)
data60405 = pd.read_excel(r'C:\Users\femke\PycharmProjects\BA1\ElectricityPrices_EpexSpot_60405.xlsx', sheet_name='04-05June', usecols=[1,2],header=0)

def state(df):
    """
    :param df: Dataframe with timeseries of price values
    :return: Series of prices converted to states
    """
    timeseries = df['Price'].values.tolist()
    a = 0.155
    b = 0.193
    c = 0.231

    series = []
    for t in timeseries:
        if t < a:
            series.append(0)
        elif a <= t < b:
            series.append(1)
        elif b <= t < c:
            series.append(2)
        else:
            series.append(3)
    return series

def onestate(t):
    """
    :param t: One price value whose state needs to be determined
    :return: The corresponding state of price value t
    """
    a = 0.155
    b = 0.193
    c = 0.231

    if t < a:
        number = len(range(0,int(a*1000)))           # The number of price
    values in this category
        return 0, number
    elif a <= t < b:
        number = len(range(int(a*1000),int(b*1000)))
        return 1, number
    elif b <= t < c:
        number = len(range(int(b*1000), int(c*1000)))
        return 2, number
    else:
        number = len(range(int(c*1000), 400+1))     # Price ranges from 0
    to 0.400
        return 3, number

def transprob(list_timeseries):
    """
    :param list_timeseries: list with states of a timeseries
    :return: Transition probability matrix
    """
    maxstates = []
    for timeseries in list_timeseries:
        m = max(timeseries)
        maxstates.append(m)
    n = 1 + max(maxstates)
    M = [[0] * n for _ in range(n)]
    for timeseries in list_timeseries:
        for (i, j) in zip(timeseries, timeseries[1:]):
            M[i][j] += 1
    for r in M:
        s = sum(r)
        if s > 0:
            r[:] = [elem / s for elem in r]
    return M

def valuefunction(T, x0, umax, P):
    """
    :param T: Total amount of available periods
    :param x0: total amount to be charged at the end of period T
    :param umax: Maximum charging speed
    :param P: Transition probability matrix
    :return: Results of the optimal value function
    :return: The corresponding optimal decisions
    """
    states = [i / 10 for i in range(0, 10 * x0 + 1, int(10 * umax))]
    D = [0, umax]  # Decisions
    prices_list = [k / 1000 for k in range(0, 400+1, 1)]

    if x0 <= 0:
```

```python
        return 'no charging needed'
    elif x0 > umax * T:
        return 'car can not be fully charged during one night'
    else:
        V = np.full(shape=(T, math.ceil(x0 / umax) + 1, len(prices_list)), fill_value=np.inf)
        choice = np.full(shape=(T, math.ceil(x0 / umax) + 1, len(prices_list)), fill_value=np.inf)
        for x in states:          # Last state
            if x == 0:            # No charging needed
                u = 0
                V[T - 1, states.index(x), :] = 0
                choice[T - 1, states.index(x), :] = u
            elif 0 < x <= umax:  # Only one (or partial) charging period needed
                u = x
                cost = u * prices_list[-1]
                V[T - 1, states.index(x), :] = cost
                choice[T - 1, states.index(x), :] = u
            else:                 # Charge is not be fully charged at T + 1
                u = np.inf
                cost = np.inf
                V[T - 1, states.index(x), :] = cost
                choice[T - 1, states.index(x), :] = u

        for t in range(T - 2, -1, -1):
            print('t=',t)       # To keep track where the program is running
            for x in states:
                for p in prices_list:
                    m, a = onestate(p)
                    V_all = []  # Empty list to store all values of all decisions
                    if x == 0:  # No charging needed
                        u = 0
                        V[t, states.index(x), prices_list.index(p)] = 0
                        choice[t, states.index(x), prices_list.index(p)] = u
                    elif 0 < x <= umax:  # Only one (or partial) charging period needed
                        for u in [0, x]:
                            cost = 0
                            for l in prices_list:
                                n, b = onestate(l)
                                if P[m][n] > 0:
                                    pass
                                else:
                                    P[m][n] = 0.000001
                                cost += P[m][n] * V[t + 1, states.index(x - u), prices_list.index(l)] / b
                            cost += u * p
                            V_all.append(cost)
                        V[t, states.index(x), prices_list.index(p)] = min(V_all)
                        choice[t, states.index(x), prices_list.index(p)] = [0, x][V_all.index(min(V_all))]
                    elif umax < x <= umax * (
                            T - t):    # More than one charging period needed and at least this many periods left
                        V_all=[]
                        for u in D:  # Either charge on maximum speed or
```

```python
not
                            cost = 0
                            for l in prices_list:
                                n, b = onestate(l)
                                if P[m][n] > 0:
                                    pass
                                else:
                                    P[m][n] = 0.000001
                                cost += P[m][n] * V[t + 1, states.index(x - u), prices_list.index(l)]/b
                            cost += u * p
                            V_all.append(cost)
                        V[t, states.index(x), prices_list.index(p)] = min(V_all)
                        choice[t, states.index(x), prices_list.index(p)] = D[V_all.index(min(V_all))]

                    else:              # Charge can not be fully charged at T + 1
                        u = np.inf
                        cost = np.inf
                        V[T - 1, states.index(x), prices_list.index(p)] = cost
                        choice[T - 1, states.index(x), prices_list.index(p)] = u

        return V, choice

""" EXECUTE THE VALUEFUNCTION """
# 8pm - 8 am with regular intervals of 30 min gives a total of 24 stages
stages = 48
start = 8
umaximum = 1
liststates_data = [state(data2223), state(data2324), state(data2425), state(data2829), state(data2930), state(data3031),
                   state(data3101), state(data60102), state(data60203), state(data60304), state(data60405)]
transmatrix = transprob(liststates_data)

V, choices = valuefunction(stages, start, umaximum, transmatrix)

""" PRINT RESULTS TO AN EXCEL FILE """
with pd.ExcelWriter("Output_SDP&MC.xlsx") as writer:
    for t in range(stages):
        df1 = pd.DataFrame(V[t,:,:])
        df2 = pd.DataFrame(choices[t, :,:])
        df1.to_excel(excel_writer=writer, sheet_name=f't={t} E(c)')
        df2.to_excel(excel_writer=writer, sheet_name=f't={t} choices')
```