POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

The Electric Vehicle Shortest Path Problem With Hard Time Windows And Prize Collection

Tesi di Laurea Magistrale in
Mathematical Engineering - Ingegneria Matematica

Author: **Antonio Cassia**

Student ID: 928509
Advisor: Prof. Ola Jabali
Co-advisors: Prof. Federico Malucelli
Academic Year: 2021-2022

Abstract

The Shortest Path Electric Vehicle Problem (SPEVP) aims at finding the shortest path for an electric vehicle (EV) departing from a given origin and arriving at a given destination. The limited autonomy of the EV is considered, and recharging its battery at charging stations (CS) is permitted. Due to the scarcity of CSs, compared to gas stations, finding an EV shortest path is difficult. The EV is allowed to partially recharge its battery at CSs, which may have different charging technologies. Furthermore, the charging time follows a nonlinear charging function. In particular, we consider long EV trips, e.g., when the unrestricted travel time between the origin and the destination is at least six hours. During such long trips. several charging stops may be necessary. The driver may have certain preferences to stop at CSs that match her interest, e.g., visiting cultural sites. Moreover, the user may also want to perform some particular activity during specific time windows, like having lunch or sleeping.

Therefore, the overall goal of the thesis is to model and solve a version of the SPEVP in which the charging decisions along the path are harmonized with user preferences and requirements. To achieve this goal, we attribute a score for each CS, which represents how much it suits the preferences of the user. We then address the problem of finding a route that maximizes the total gained score, respects all the time windows and never violates the EV autonomy constraints. In particular, we impose a temporal tolerance on the deviation of such a path from the shortest EV path in time. We propose a MILP formulation for this setting which we denote with Maximum Discounted Profit Model, and we develop a heuristic for it. The latter is based on a A* search algorithm, which works with modified arc weights of the graph in order to account for the CS scores. We evaluate our models on several realistic instances, with CSs located in Central Europe. In particular, we demonstrate the effectiveness of the proposed heuristic, when compared to the exact solutions obtained by the MILP.

Keywords: electric vehicle, shortest path, hard time windows, prize collection, A star algorithm

Abstract in lingua italiana

Il Problema dei Cammini Minimi per Veicoli Elettrici (SPEVP) cerca di trovare il cammino più veloce per un veicolo elettrico che percorre la strada da una data origine ad una data destinazione. Viene considerata l'autonomia limitata dei veicoli elettrici, ed è ammessa la ricarica, anche parziale, della batteria nelle stazioni di ricarica, le quali possono avere tecnologie di ricarica differenti. A causa della scarsità di stazioni di ricarica, rispetto alle stazioni di benzina, trovare il cammino più veloce per un veicolo elettrico è complesso. In particolare, si considerano lunghi viaggi, ad esempio quando il tempo minimo tra origine e destinazione senza tappe di ricarica è superiore alle sei ore. Durante questi lunghi viaggi, potrebbero essere necessarie diverse tappe per la ricarica. Il guidatore potrebbe preferire di fermarsi in stazioni di ricarica che ben rappresenta i suoi interessi, ad esempio, visitando siti culturali. Inoltre, l'utente potrebbe anche voler svolgere determinate attività durante specifiche finestre temporali, come pranzare o dormire. L'obbiettivo di questa tesi è quello di modellare e risolvere una versione del SPEVP in cui le decisioni sulle fermate di ricarica lungo il percorso sono allineate con le preferenze e i vincoli imposti dall'utente. Per raggiungere questo scopo, si è attribuito ad ogni stazione di ricarica un punteggio che rappresenta quanto quella stazione è interessante per l'utente. Si è poi considerato il problema di trovare un percorso che massimizza il punteggio totale ottenuto, che rispetti tutte le finestre temporali e che non violi mai i vincoli di autonomia del veicolo elettrico. Inoltre, si è imposto una tolleranza temporale sulla deviazione del percorso ottenuto rispetto alla percorso più veloce. Si è poi proposta una formulazione MILP per questo tipo di problema denotata con il nome di Modello del Massimo Profitto Scontato, e si è sviluppata una euristica per risolverlo. Quest'ultima si basa sull'algortimo di ricerca A*, che lavora con dei pesi modificati per ogni arco del grafo, in modo da tener conto dei punteggi per ogni stazione di ricarica. Si sono poi valutati i nostri modelli su diverse tratte realistiche, con stazioni di ricarica posizionati in Europa Centrale. Infine si è dimostrata l'efficacia dell'euristica proposta, rispetto alla soluzione esatta ottenuta dal MILP.

Parole chiave: veicolo elettrico, cammino minimo, finestre temporali, massimizzazione premi, algoritmo A*

Contents

Abstract **i**

Abstract in lingua italiana **iii**

Contents **v**

Introduction **1**

1 State of the art **3**

2 Problem definition **7**
2.1 Problem description . . . 7
2.2 Charging function . . . 8
2.3 Time Windows . . . 9
2.4 Score . . . 12
2.5 Model . . . 13
2.6 Shortest Path Model . . . 16
2.7 Discounted weights . . . 17
2.8 Reduce the number of legs . . . 18

3 Heuristic algorithm **21**
3.1 Potentials . . . 21
3.2 Labels . . . 25
3.3 A* Search Algorithm . . . 27
3.3.1 An example . . . 32
3.4 Score . . . 35

4 Data description 37
4.1 Data 37
4.2 Data cleaning 37
4.3 Weights Matrix 40
4.4 Machine Performance 40
4.5 Time windows 40
4.6 Construction of $\mathcal{S}$ and $\mathcal{A}$ 41
4.7 Preprocessing 42

5 Computational experiments 45
5.1 Vehicle 45
5.2 Datasets and instances 45
5.2.1 Instances details 48
5.3 Comparison Phase 49
5.3.1 Shortest Path 49
5.3.2 Maximum Profit 50
5.3.3 Discounted Models 50
5.4 Medium Dataset 52

6 Conclusions 55
6.1 Future work 55

Bibliography 57

Numerical Results 63

List of Symbols 71

List of Figures 75

List of Tables 77

Acknowledgements 79

Introduction

In the last years, many governments around the world started encouraging the adoption of electric vehicles (EVs) in order to reduce greenhouse gas emissions. Many of the most important multinational companies and corporations around the globe are EV100 members, a global initiative proposed by The Climate Group under which each member undertakes to switch its fleet to EVs and installing charging infrastructure by 2030 (Coplon-Newfield and Park [2017]). The number of companies that each year adhere to this initiative is constantly increasing, with a total of over 5.5 million vehicles committed to electric by 2030 (The Climate Group [2022]). The average market share for EVs in Europe increased from 3.0% in 2019 to 10.5% in 2020 (Transport & Environment [2021]). With respect to internal combustion engine vehicle (ICEVs), EVs have obvious advantages such as lower maintenance costs (Harto [2020]) and lower operational costs thanks to the reduced price of electricity with respect of fossil fuels. On the other side they have higher purchasing costs with respect to ICEVs and also their autonomy is limited due to the battery capacity. This last problem leads to the creation of a well-planned network of charging stations (CSs) in order to satisfy the EVs energy demand.
The increasing number in EVs is reflected in an increasing number of public CSs along the existing road network. Currently in Europe there is an average ratio of 7.5 EVs per public charger point (PCP) (Transport & Environment [2020]), which is lower than the recommended ratio of 10.0 EVs per PCP, as indicated by the European Union directive in 2014 (European Union [2014]). In general, CSs are still too scarce compared to conventional gas stations. Moreover, the charging time can take from few minutes to several hours, thus CSs should be installed near some point of interest (POI) in order to entertain the user while EV is charging. Choosing a right CS during a long trip is crucial and it can avoid annoying waiting and so sometimes is better to take a small detour from the shortest route just to charge at CSs that have POIs which better match the preferences of the user. Also, counting all the charging stops, some trips can take very long and a pause for resting and eating is necessary. Therefore, long EV trip planning may be enhanced by accounting for user preferences in terms of restaurants or hotels. In addition, the user can impose custom stops along the path, maybe for visiting places of interest. For instance, a user going from Milan to München may want to visit Bolzano, while the EV is charging.

The aim of this thesis is to explore the concept of planning long multi-day EV trips, with hard time windows and charging constraints. In particular, we attribute a score to each CS with respect to the preferences of the user. Using those scores we consider the problem of optimizing the shortest path for a single EV between a given origin and a given destination respecting all energy feasibility constraints, while maximizing the total score that the user can achieve with route duration limits. The model that solves this problem is the Maximum Discounted Profit Model (**MDPM**), while the duration limits are established by solving the Shortest Path Model (**SPM**), which is the shortest EV path in time. Then we compare the profit obtained with **MDPM** with the maximum possible score obtainable using the Maximum Profit Model (**MPM**). Furthermore, we develop a heuristic algorithm based on the A* Search Algorithm which handles larger instances with respect to **MDPM**. We formulate the A* Shortest Path Model (**AsM**) that is a heuristic approach to solve the **SPM**. Then we formulate the A* Maximum Discounted Profit Model (**AsDM**) which aim to heuristically solve the **MDPM**. We formulate **MDPM**, **SPM** and **MPM** as mixed integer linear problems (MILPs). We first solve the shortest path model **SPM** and the objective is then used as an upper bound to compute the **MDPM**. After that, we compare the profit gained with the maximum profit model **MPM**. To solve the **MDPM** we create a new weight for each arc that takes into account the driving time, the charging time in the starting node and the score in the arriving node. The same weight is then used to solve the **AsDM**, while the **AsM** is instead used as a comparison for the performance of the A* approach with respect to **SPM**. All models are then compared with the same set of 20 instances.
The contribution of this thesis is threefold: first we develop an EV shortest path model that accounts for user preferences; secondly we develop a heuristic based on A* algorithm that solve those types of problems; third we verify the performance of both the MILP models and the heuristic models on realistic test instances.

The structure of the thesis is as follow: Chapter 1 describes the state of the art of the scientific literature related to the electric vehicle shortest path problems. In Chapter 2 we formally define **MDPM** description, the definition of the hard time windows constraints and the mathematical model with MILP formulation for **SPM** and **MPM**. Then, the heuristic approach, with the A* search algorithm is proposed in Chapter 3, and it is used to formulate the **AsM** and **AsDM**. In Chapter 4 we describe the dataset that we use for the computational experiments. We propose several pruning techniques to scale the graph dimensions. In Chapter 5 we evaluate our models and algorithms with a set of 20 instances representing three main long trips, each one with different setting parameters. Finally, in Chapter 6 we state our conclusions and research perspectives.

1 | State of the art

The Shortest Path Problem (SPP) consist in finding a shortest path in a network, from a given origin to a given destination. A simple approach to solve this type of problem is to apply the Dijkstra's algorithm to a graph representation of the road network, using a fixed scalar weight for each arc, that represent the driving time. This approach, however, does not take in account for charging stops, so while it is useful for ICEVs, it is not necessarily feasible for EVs. It is possible to speed up the search by using other techniques like the A* Search Algorithm (Hart et al. [1968]) which uses potential functions to estimates the minimal cost to reach the target and guides the search towards it. For shortest path problems, this potentials can be computed using the ALT algorithm (A*, Landmarks, Triangle Inequality) proposed by Goldberg and Harrelson [2005]. This algorithm precomputes the distances between a set of landmarks and all the other points. Then it uses these landmarks and the triangle inequality to compute a lower bound for the distance to the target. Another approach uses the Contraction Hierarchies (CH) method, introduced by Geisberger et al. [2012], it removes unimportant vertices without changing the minimal distances between all the other vertices, inserting a new edge if the distance between to other vertices would otherwise increase. Combining both approaches it is possible to obtain better results, like for the Core-ALT algorithm (Bauer et al. [2008]).

The Electric Vehicle Shortest Path Problem (EVSPP) is instead a SPP where we take into account EVs, that are subject to battery limitation and charging constraints. Due to their limited autonomy, EVs may need to detour to CSs in order to recharge their battery (Adler et al. [2016]). This is particularly true in medium and long range routes, like in Schiffer et al. [2018]. A key decision in this context is where and how much charge the EVs. The problem of minimizing the overall trip time for EVs in road networks was studied by Baum et al. [2015]. Most recent works take in account also speed planning among the arcs, balancing driving times and energy consumption (Hartmann and Funke [2014], Baum et al. [2020]). These works achieve good results in both exact and heuristic approaches, but they do not include charging stops in route optimization. Bauer et al. [2016] and Schoenberg and Dressler [2022] applied the CH method to the EVSPP, while

Zündorf [2014] instead used the Core-ALT algorithm in developing a heuristic algorithm that solves the EVSPP in continental graph in few minutes. Rajan et al. [2021] propose instead two generalizations of the EVSPP in which they compute the shortest path for any initial state of charge and for every possible minimum energy threshold. Baum et al. [2020] introduced a functional representation of the optimal energy consumption between two locations, that led to the development of a heuristic algorithm based on the Core-ALT, which computes energy optimal paths within milliseconds after preprocessing the whole graph.

One of the main modeling decision is how the EV recharges its batteries. For instance, some works assume that the EV must recharge completely its battery before leaving a CS. Problems of this type were introduced by Erdoğan and Miller-Hooks [2012], and later studied by Montoya et al. [2016] and Bruglieri et al. [2019]. Other works use the state of charge (SoC) of the EV as a decision variable, letting the model to decides how much energy to recharge in each CS. This was studied in Montoya et al. [2017], in Froger et al. [2019] and in Kullman et al. [2021]. Another approach is to use battery swapping stations, like in Li et al. [2020] and Adler et al. [2016], or to take a combination of all those approach, like in Zündorf [2014]. In general, most studies assume that the energy consumption is directly and exclusively related to the traveled distance, however in reality it depends also on other factors (Goeke and Schneider [2015], Lin et al. [2016]) like the EV parameters, its speed and loads.

Another important model decision is to consider only public CSs, only private or both of them. Only a few number of public charging network owners allows for charging time reservations (Bruglieri et al. [2019]). A general assumption in the EV routing is that CSs are uncapacitated (Erdoğan and Miller-Hooks [2012], Montoya et al. [2016] or Montoya et al. [2017]), meaning that in every node there is at least a CS always available. The long charging times and the small number of CSs may generate congestion, leading to consider also the possibility of waiting in a queue or detouring to another CS nearby. This setting was studied by Kullman et al. [2021] who introduced dynamic optimization policies based on the state of the current CS.

The charging function is in general non linear with respect to time because voltage and current change during the charge process. Bruglieri et al. [2014] use a linear approximation that goes from 0 to $0.8Q$, where Q is the battery capacity of the EV, and so working with only the linear part of the charging process. Montoya et al. [2017] and Froger et al. [2019] introduced the non linear charging function modeled as a piecewise linear concave function. Uhrig et al. [2015], confirm that the piecewise non-linear approximation fits well the real charging process, for multiple combinations of charging speed and battery capacity.

Time windows constraints is another aspect well studied. They are used to force the EV to arrive in predetermined CSs before or during a particular time interval. Time windows can be classified using two main categories, hard time windows and soft time windows. Soft time windows mean that the EV can arrive before or after respectively of the initial and the ending time of the time window, but is penalized for doing so (Calvete et al. [2004], Calvete et al. [2007]). Hard time windows entails that the EV cannot arrive after the end of the time window, but it could arrive before it and wait (Schneider et al. [2014], Bruglieri et al. [2015]).

In this thesis we want to maximize the total gained score by selecting a path using CSs that better match the preferences of the user. The Prize Collection Traveling Salesman Problem was introduced by Balas [1989]. It consists of searching for a path in a graph, visiting a subset of customer. A profit is associated to each customer, and the aim of the model is to find the path that maximize the total gained profit.

In our work we consider a single EV, with partial recharge approach and non linear charging process. We consider only public CSs that are located near POIs, like hotels or restaurants, without taking in account the reservation or the possibility that a charger is not available. We also consider hard time windows, with an additional constraint that there is a limit of the leading arrival time. Each CS has a score associated to it that represent how much that particular CS is important for the user.

We first solve **SPM** and then use its objective as an upper bound to compute the **MDPM**. Then, we compare the results of profit gained between **MPM** and **MDPM**. In the latter we create a new weight for each arc that takes into account the driving time, the charging time in the starting node and the score in the arriving node. This weight is then used to solve the MILP formulation of **MDPM** and **AsDM**, a heuristic implementation of the A* search algorithm.

2 | Problem definition

In this chapter we define all the variables, sets and parameters which will then be used to construct the shortest path model **SPM** and the profit model **MPM**. Using a MILP formulation we are ready to solve the **SPM** model. Its objective value, appropriately resized, is then used as an upper bound for the duration of the trip in the **MPM** model. Finally we formulate another MILP model, **MDPM**, that accounts for discounted weights on the arcs.

2.1. Problem description

We consider an EVSPP with hard time windows constraints. There is only a single EV that can be partially recharged during stops in the CSs. The EV must stops during all the time windows in their given order. Each time window represents a moment of the day in which the user must do some particular activity, like lunching, visiting new places or sleeping. Each CS has a different charging speed and a different score associated. The scores are user-dependent. We formulate a MILP decision problem that tries to find the fastest path from an origin point $\mathcal{O}$ to a destination point $\mathcal{D}$ that satisfies all the charging and time windows constraints, while maximizing the total score of the optimal path.
Let $\mathcal{G} := \langle \mathcal{S}_{\mathcal{O},\mathcal{D}}, \mathcal{A} \rangle$ be a directed graph, where $\mathcal{S}_{\mathcal{O},\mathcal{D}}$ is the set of CSs including also $\mathcal{O}$ and $\mathcal{D}$ as nodes. $\mathcal{A}$ is the set of arcs that connects each pair of nodes in $\mathcal{S}_{\mathcal{O},\mathcal{D}}$. With each arc $(i, j) \in \mathcal{A}$ is associated a driving time t_{ij} and an energy consumption e_{ij}, both satisfying the triangular inequality.
Let t_{start} be the starting time of the trip, and t_{end} be the ending time. They are both parameters of the model and are defined as relative time with respect to the first day of the trip. Time 0 is associated to the midnight before t_{start}. So, for instance, if the trip starts at 10:00 of day 0 and it must ends before 18:30 of day 1, then $t_{\text{start}} = 10.0$ and $t_{\text{end}} = 42.5$. In this way it is easy to construct time windows for lunch breaks and rests. The EV starts in $\mathcal{O}$ fully charged, with the SoC of the battery equal to Q, and it must arrive in each CS with an amount of energy which is greater or equal to $q_{\min}$. This last value force all the models to have always a minimum amount of energy stored

in the battery, so that the EV can never be without energy. The EV has also a maximum average consumption of η and a maximum power charge P. The EV must respect all the ordered time windows and each of them can be satisfied only in a subset of the CSs $\mathcal{S}_{\mathcal{O},\mathcal{D}}$. We considered only lunch, tourism and nights time windows. For lunch, the associated POI that is searched by the model is "Restaurant" while the associated POI for nights is "Hotels". For tourism stops, the associated POI are computed dynamically as described in section 4.5.

2.2. Charging function

Let $\mathcal{S}$ be the set of charging stations at which the EV can fully or partially recharge its battery. The CSs network consists only for public station, each one with a different charging speed.
Each CSs $i \in \mathcal{S}$ has a charging speed associated with a piecewise linear concave charging function $\Phi_i(\Delta)$, where Δ is the time spent waiting while the EV is charging. The non-linear charging function was introduced by Montoya et al. [2017] and was shown to be a good approximation of actual behavior of the EV. They also demonstrate, in the same article, that using a simple linear approximation maybe can lead to expensive or infeasible solutions. This approach was lately used in many other works, like Zündorf [2014], Froger et al. [2019] and Kullman et al. [2021].
Let q be the SoC of the EV when arrives at the charging station i, then the SoC when it leaves is $\Phi_i(\Delta + \Phi_i^{-1}(q))$. Let $\mathcal{B}_i = \{0, b_1, \dots, b_{m_i}\}$ be the ordered set of breakpoints of the piecewise linear approximation of the charging curve of CS i. Let c_{ik} and a_{ik} be the charging time and SoC of breakpoint $k \in \mathcal{B}_i$. Each breakpoint connects $(c_{i,k-1}, a_{i,k-1})$ and (c_{ik}, a_{ik}) with a line with coefficient ρ_{ik}, with $k \in \mathcal{B}_i \setminus \{0\}$.
In each CS $i \in \mathcal{S}$ a minimum charging time can be imposed due to the minimum stopping time of each time windows (see section 2.3). To do that a fictitious breakpoint is added to $\mathcal{B}_i$ right before the last breakpoint b_{im_i}, taking its place while m_i is increased by 1. This is done in order to simulate a constant value of SoC, that represent the case in which the EV is fully charged but it cannot yet leave the CS (see Figure 2.1). This fictitious point has a SoC value $a_{i,m_i} = 0.999$ so that is possible to express also every point of the last piece of Φ_i as a convex combination of $(c_{i,m_i-1}, a_{i,m_i-1})$ and (c_{i,m_i}, a_{i,m_i}).

If the charging speed is greater than the maximum power P for the EV, the charging profile that is used is the one with P as the charging speed. The charging profile of a CS depends also on the EV that will use it (Montoya et al. [2017]), on battery degradation (Pelletier et al. [2017]) or external data like temperature, day of the year, timestamp

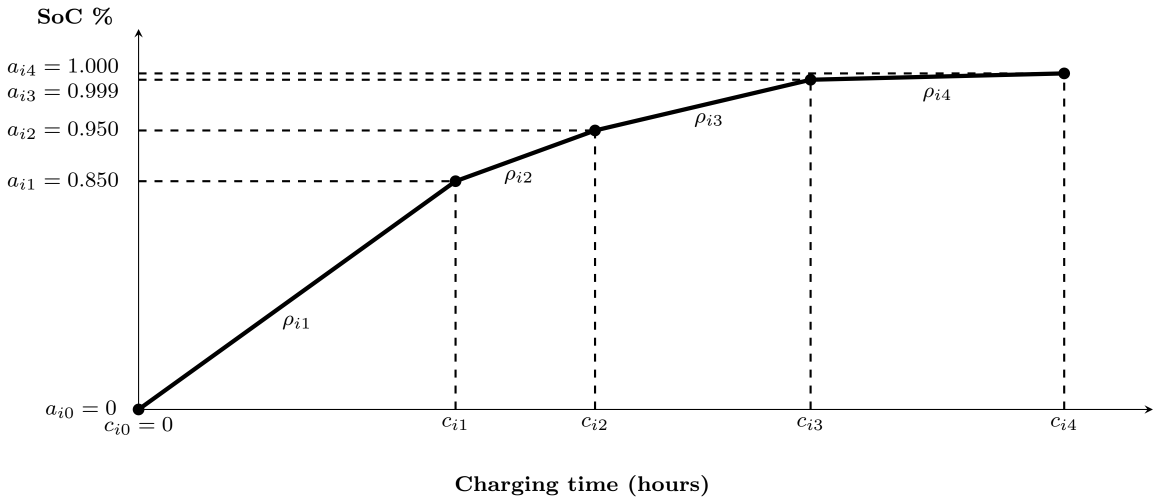


Figure 2.1: Example of a piecewise linear approximation for a CS $i \in \mathcal{S}$ with a power of 22 kWh adapted from Montoya et al. [2017]. The fictitious point is added to this charging function with the point $(c_{i,3}, a_{i,3})$, creating a new slope between $(c_{i,2}, a_{i,2})$ and $(c_{i,4}, a_{i,4})$.

(Mies et al. [2018]). We decided to simplify the model considering only the charging speed, the capacity Q and the maximum power P, using adapted charging function from Montoya et al. [2017] and ChargePrice.com. Moreover, we will assume that the EV can also partially recharge its battery, as in Froger et al. [2019].

2.3. Time Windows

The user may personalize her trip. She can decide how many days it will last and which stops perform during the trip. To model this aspect, we implemented in the model time windows that represents moments in which the EV is forced to stop at a CS. This object allows us to create multi-day routes from an origin to a destination. Time windows are largely studied in EVRP, mostly to represent customer constraints, which are important constraints in real world application (Schneider et al. [2014] and Hiermann et al. [2016]). Let $\mathcal{W}$ be the set of possible time windows. A time window $k \in \mathcal{W}$ is defined as follow:

$$k := (\gamma_k^L, \gamma_k^U, t_k^{\min}, o_k, \nu_k) \tag{2.1}$$

where:

- the interval $[\gamma_k^L, \gamma_k^U]$, with $\gamma_k^L < \gamma_k^U$, depict its initial and ending time. Like t_{start} and t_{end} they are represented as a relative value of time with respect the first day of the trip. So, for instance, the interval $[\gamma^L = 36.0, \gamma^U = 38.0]$ goes from 12:00 of day 1 to 14:00 of the same day. Note that the value γ_k^U is intended as the maximum time that the EV must arrive in time window k. So, if the EV perform time windows k in node i, then it is not obliged to leave i before γ_k^U but, on the contrary it must arrive before γ_k^U.

- $t_k^{\min}$ is the minimum time that the EV needs to stop during k
- o_k binary value: 1 if k is an optional time windows, 0 otherwise (see below)
- ν_k is a label that identifies which type of POI is needed during k (see below).

$\mathcal{W}$ is an ordered set, that means that all the time slots need to be visited following this order, so $\forall k, h \in \mathcal{W}$

$$k \prec h \qquad \Rightarrow \qquad \gamma_k^L < \gamma_h^L.$$

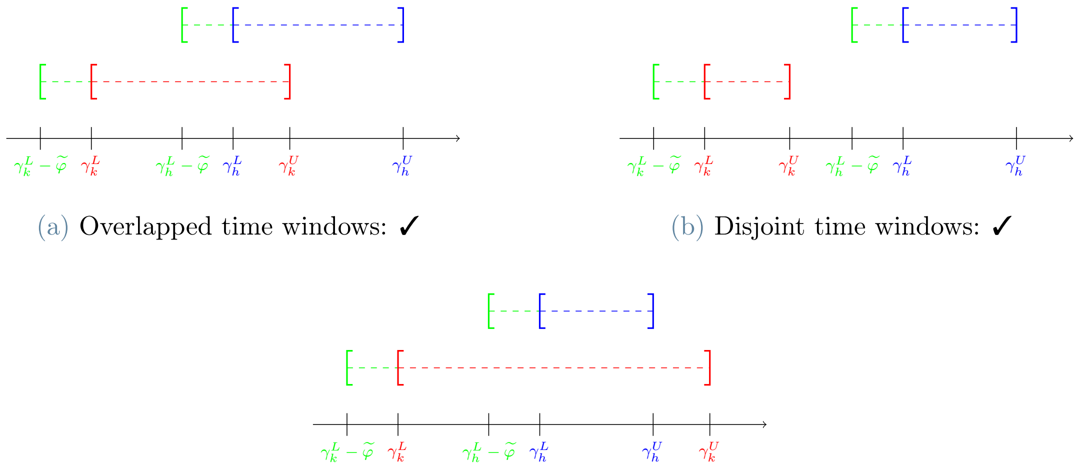
The EV must stop during each time window k for a minimum amount of time given by $t_k^{\min}$. It is allowed to arrive in a node with a maximum anticipation time $\widetilde{\varphi}$, but the minimum stopping time will starts however at γ_k^L. So, if the EV arrives in node i in the interval $\left[\gamma_k^L - \widetilde{\varphi}, \gamma_k^U\right]$, then it may decide to stop in i for $t_k^{\min}$ or instead perform k in the next node. The time windows that we use are hard time windows: this means that is not possible to perform time window k before $\gamma_k^L - \widetilde{\varphi}$ or after γ_k^U.
Time windows can overlap each others, but they can't be one inside the other (see fig. 2.2), formally:

$$\left[\gamma_k^L, \gamma_k^U\right] \cap \left[\gamma_h^L, \gamma_h^U\right] := \begin{cases} \emptyset & \text{if } \gamma_k^U < \gamma_h^L \\ \left[\gamma_h^L, \gamma_k^U\right] & \text{if } \gamma_k^U \geq \gamma_h^L \end{cases} \qquad \forall k, h \in \mathcal{W}, \text{with } k \prec h.$$

Let $\mathcal{W}^R \subseteq \mathcal{W}$ be set of required time windows, and $\mathcal{W}^O \subseteq \mathcal{W}$ be the set of optional time windows. They form a partition of $\mathcal{W}$, indeed if $o_k = 1$ then $k \in \mathcal{W}^O$, else $k \in \mathcal{W}^R$. The EV is forced to stop in each $k_R \in \mathcal{W}^R$, but it must stop in $k_O \in \mathcal{W}^O$ only if there exists at least one k_R such that $k_O \prec k_R$, otherwise k_O can be skipped. For this reason, and for convenience, let $\mathcal{W}^{\mathcal{NA}}$ be the ordered set of not avoidable time windows as

$$\mathcal{W}^{\mathcal{NA}} := \left\{ k \in W : \exists h \in \mathcal{W}^R \text{ with } k \neq h \text{ s.t. } k \prec h \right\} \subseteq \mathcal{W} \tag{2.2}$$

and let $\mathcal{W}^{\mathcal{A}} := (\mathcal{W}^{\mathcal{NA}})^c$, the complement of $\mathcal{W}^{\mathcal{NA}}$, be the ordered set of avoidable time windows.
In $\mathcal{W}^R$, for instance, are placed nocturnal time windows and tourism stops. Lunch breaks are instead inserted in $\mathcal{W}^O$ since it is possible that the model finds an optimal path that



(a) Overlapped time windows: ✓

(b) Disjoint time windows: ✓

(c) One inside the other: ✗

Figure 2.2: Possible relative positions of time windows

arrives in destination node after the last night in hotel but before lunch. As mentioned before, the number of nights is a decision of the user, they can't be totally removed. If instead, also lunch breaks were considered as required, the model can decide to enlarge the charging time in previous CS just to ensure that the user must stop at some CS even in the lunch of the last day of the trip. It make sense to ensure that this behavior is forbidden, since the last lunch is not a proper part of the journey but it can be added only if it strictly necessary. Indeed, the aim of this thesis is to find a shortest path to arrive at destination: every unnecessary stops must be avoided (fig. 2.3).

Each CSs has multiple POIs associated to it, and each time window requires a specific POI, so the model must select CSs that have that specific POI associated. This information is written in the label ν and is different for each time window. For instance, let $k \in \mathcal{W}$ refers to the first night, then $\nu_k =$ "Hotels" and the EV is forced to stops at a CSs near a hotel. So, given a time slot k, it is possible to construct the set of chargers $\mathcal{S}_k \subseteq \mathcal{S}$ that have in their neighborhood the POI stated in ν_k. Finally, let $\widetilde{\mathcal{S}} = \cup_{k \in \mathcal{W}} \mathcal{S}_k$ and $\mathcal{W}_i \subseteq \mathcal{W}$ be the set of time windows for which the EV can stop in CS $i \in \mathcal{S}$. For instance, if a charger i has in the neighborhood a hotel and a restaurant, then $\mathcal{W}_i$ contains all the time windows $k \in \mathcal{W}$ that have $\nu_k =$ "Hotels" or $\nu_k =$ "Restaurants".

Due to the minimum stopping time $t_k^{\min}$ for each $k \in \mathcal{W}$, it can happen that the EV is forced to stay and charging in the same place for more time than it actually needs to completely recharge its battery. This is why we introduced the fictitious point in the charging function in section 2.2. In this way we can simply model a MILP formulation without the necessity to include also a variable that indicate whether or not the EV is stationary without charging.

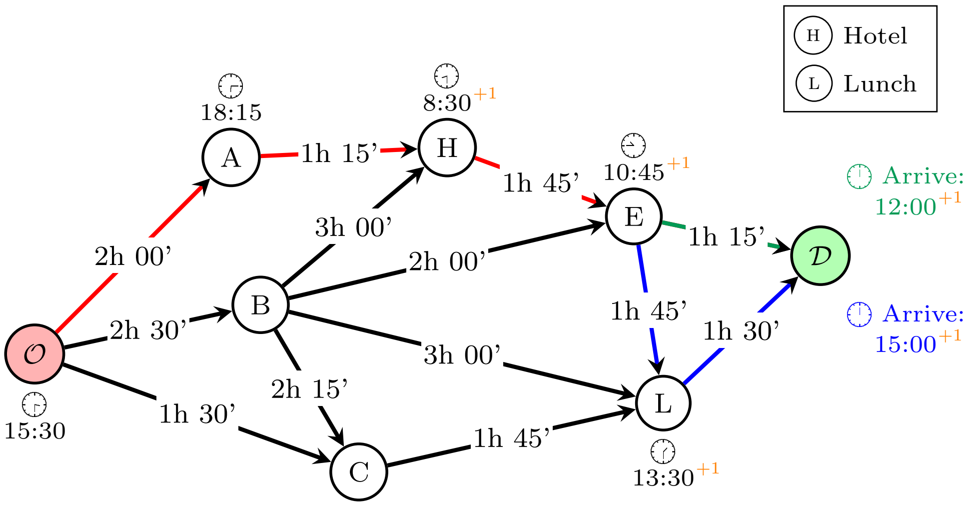


Figure 2.3: Example of a path from $\mathcal{O}$ to $\mathcal{D}$ with a night in hotel and lunch breaks. The red path is the optimal one, stopping in the hotel as required and reaching node E. If also the lunch break is required, then the EV is forced to arrive to $\mathcal{D}$ with stopping at L (blue path). Instead, with the optional flag, from E the EV can go directly to $\mathcal{D}$ with three hours in advance (green path). The timestamps near the nodes represents the departure time from that node, including also the recharging time (not reported in this figure). The "+1" over the timestamps indicates that it refers to the next day. For $\mathcal{D}$, instead, the timestamps represent the arrival time. The number on each arc symbolize the travel time.

2.4. Score

When the EV needs to be charged, the user will spend some time in the neighborhood of the selected CS. Moreover, if it is almost time for lunch, dinner or is late night, our user would like to select a CS that has restaurants or hotels. Most of the time CSs are placed strategically near those type of POIs. Sometimes in those places there are also special offers for EV users, like discounts or, in some hotels, even free usage of their swimming pool. To avoid annoying waiting periods, it is important that the user selects a CS that best suits her preferences.

To account for this aspect, we create a model that tries to maximize those preferences which are implemented as a score given to each CS. Different users may have different scores for the same CS. For instance, suppose that user A prefers to stops near city centers, while user B likes staying in a shopping mall. Then a CS placed near Castello Sforzesco, in Milan, will have a higher score for A with respect to B.

Given a CS, to compute the score other aspects may be accounted for like the charging speed, the cost per kWh charged or how many other chargers are in the neighborhood. Each score is given as an input and it is not the purpose of this research to find ways of how computing it. For this reason we decided to give to each CS a random generated score, from 0 to 5, as given in most websites.

2.5. Model

We define as follows the **MPM** problem. In the model there is only one vehicle that has a maximum capacity Q that leaves the origin point $\mathcal{O}$ fully charged. The full route starts at time t_{start} in position $\mathcal{O}$ and it must arrive in destination point $\mathcal{D}$ before t_{end}.
Let $\mathcal{S}_{\mathcal{O}} = \mathcal{S} \cup \{\mathcal{O}\}$, $\mathcal{S}_{\mathcal{D}} = \mathcal{S} \cup \{\mathcal{D}\}$, $\mathcal{S}_{\mathcal{O},\mathcal{D}} = \mathcal{S} \cup \{\mathcal{O}, \mathcal{D}\}$. Let $\mathcal{A}$ be the set of arcs (i, j), with $i \in \mathcal{S}_{\mathcal{O}}$ and $j \in \mathcal{S}_{\mathcal{D}}$. The time and energy from i to j is $t_{ij} \geq 0$ and $e_{ij} \geq 0$ respectively. We assume that the triangle inequality holds for both driving times and energy consumption. Given a CS $i \in \mathcal{S}$, let Δ_i be the time spent waiting while the EV is charging. Let $\underline{q}_i$ and $\overline{q}_i$ be the SoC when the EV arrives and depart from CS i. The variables $\underline{c}_i$ and $\overline{c}_i$ are respectively the start and end time for charging an EV. The variables $\underline{\lambda}_{ik}$ and $\overline{\lambda}_{ik}$ represents the coefficients associated with the breakpoint (c_{ik}, a_{ik}) in the linear approximation, when the EV enters and leaves CS i. Let $\underline{w}_{ik}$ and $\overline{w}_{ik}$ be binary variables equal to 1 when the SoC is in the interval $[a_{i,k-1}, a_{ik}]$, when respectively the EV enters and leaves the CS i, 0 otherwise.
Variables $\underline{\tau}_i$ and $\overline{\tau}_i$ tracks respectively the time when the EV arrives and leaves the CS $i \in \mathcal{S}$. There is also a tolerance φ_i, for each $i \in \mathcal{S}_{\mathcal{D}}$, that represents how much time in advance, with respect to γ_k^L, the EV can arrive in i. The maximum anticipation time is set to $\widetilde{\varphi}$, but even if the EV arrives in advance, the minimum stopping time $t_k^{\min}$ starts at γ_k^L and not before. For instance, suppose that the EV arrives in a node at 11:50, the lunch break start at 12:00 and last for minimum 1 hour. However, in this case, the EV may charge for at least one hour and ten minutes. The maximum lead time $\widetilde{\varphi}$ is not strictly necessary for the MILP problem, indeed since it is a minimization problem the solver will tend to reduce the variable ϕ_i because otherwise it can lead to great values of the variable Δ_i. Instead, an upper bound for the lead time is useful to have a fair comparison with the heuristic presented in Chapter 3.
Let $\mathcal{W}^R \subseteq \mathcal{W}$, $\mathcal{W}^O \subseteq \mathcal{W}$, $\mathcal{W}_i \subseteq \mathcal{W}$, $\mathcal{S}_k \subseteq \mathcal{S}$ and $\widetilde{\mathcal{S}}$ be defined as mentioned in section 2.3. The binary variable x_{ij} is equals to 1 if the EV arrives in node j, starting from i, 0 otherwise. The variable y_{jk} is also binary and it is 1 if the EV stops in j in time window k, 0 otherwise. Parameter σ_j represent the score for CS $j \in \mathcal{S}$. The maximum amount of time that the journey must last is $T^{\max}$ (we will see in section 2.6 a way to compute an upper bound for this value).
The variable z_k for all $k \in \mathcal{W}^{\mathcal{A}}$ is a binary variable that is equal to 1 if the EV arrives in $\mathcal{D}$ after time window k, 0 otherwise. It is used to link arrival time in destination node and avoidable time windows.

The profit model **MPM** is defined as follows:

$$
\begin{aligned}
[\textbf{MPM}] \quad & \max \quad \sum_{(i,j)\in\mathcal{A}} \sigma_j x_{ij} && && (2.3)\\
& \text{s.t.} \quad \sum_{(i,j)\in\mathcal{A}} x_{ij} \leq 1 && \forall i \in \mathcal{S}_{\mathcal{O}} && (2.4)\\
& \sum_{(i,j)\in\mathcal{A}} x_{ij} - \sum_{(j,i)\in\mathcal{A}} x_{ji} = \begin{cases} 1 & \text{if } i = \mathcal{O} \\ -1 & \text{if } i = \mathcal{D} \\ 0 & \text{otherwise} \end{cases} && \forall i \in \mathcal{S}_{\mathcal{O},\mathcal{D}} && (2.5)\\
& e_{ij}x_{ij} - (1 - x_{ij})\, Q \leq \overline{q}_i - \underline{q}_j \leq e_{ij}x_{ij} + (1 - x_{ij})\, Q && \forall (i,j) \in \mathcal{A} && (2.6)\\
& \overline{q}_{\mathcal{O}} = Q && && (2.7)\\
& \underline{q}_{\mathcal{D}} \geq q_{\min} && && (2.8)\\
& q_{\min} \sum_{(i,j)\in\mathcal{A}} x_{ij} \leq \underline{q}_i \leq \overline{q}_i \leq Q \sum_{(i,j)\in\mathcal{A}} x_{ij} && \forall i \in \mathcal{S} && (2.9)\\
& \underline{q}_i = \sum_{k\in B_i} \underline{\lambda}_{ik} a_{ik} && \forall i \in \mathcal{S} && (2.10)\\
& \underline{c}_i = \sum_{k\in B_i} \underline{\lambda}_{ik} c_{ik} && \forall i \in \mathcal{S} && (2.11)\\
& \sum_{k\in B_i} \underline{\lambda}_{ik} = \sum_{k\in B_i\setminus\{0\}} \underline{w}_{ik} && \forall i \in \mathcal{S} && (2.12)\\
& \sum_{k\in B_i\setminus\{0\}} \underline{w}_{ik} = \sum_{(i,j)\in\mathcal{A}} x_{ij} && \forall i \in \mathcal{S} && (2.13)\\
& \underline{\lambda}_{i0} \leq \underline{w}_{i1} && \forall i \in \mathcal{S} && (2.14)\\
& \underline{\lambda}_{ik} \leq \underline{w}_{ik} + \underline{w}_{i,k+1} && \forall i \in \mathcal{S}, \forall k \in \mathcal{B}_i\setminus\{0, b_{m_i}\} && (2.15)\\
& \underline{\lambda}_{i,b_i} \leq \underline{w}_{i,b_i} && \forall i \in \mathcal{S} && (2.16)\\
& \overline{q}_i = \sum_{k\in B_i} \overline{\lambda}_{ik} a_{ik} && \forall i \in \mathcal{S} && (2.17)\\
& \overline{c}_i = \sum_{k\in B_i} \overline{\lambda}_{ik} c_{ik} && \forall i \in \mathcal{S} && (2.18)\\
& \sum_{k\in B_i} \overline{\lambda}_{ik} = \sum_{k\in B_i\setminus\{0\}} \overline{w}_{ik} && \forall i \in \mathcal{S} && (2.19)\\
& \sum_{k\in B_i\setminus\{0\}} \overline{w}_{ik} = \sum_{(i,j)\in\mathcal{A}} x_{ij} && \forall i \in \mathcal{S} && (2.20)\\
& \overline{\lambda}_{i0} \leq \overline{w}_{i1} && \forall i \in \mathcal{S} && (2.21)\\
& \overline{\lambda}_{ik} \leq \overline{w}_{ik} + \overline{w}_{i,k+1} && \forall i \in \mathcal{S}, \forall k \in \mathcal{B}_i\setminus\{0, b_{m_i}\} && (2.22)\\
& \overline{\lambda}_{i,b_i} \leq \overline{w}_{i,b_i} && \forall i \in \mathcal{S} && (2.23)\\
& \Delta_i = \overline{c}_i - \underline{c}_i && \forall i \in \mathcal{S} && (2.24)
\end{aligned}
$$

$$
\begin{aligned}
\text{s.t.}\quad & \overline{\tau}_{\mathcal{O}} = t_{\text{start}} && && (2.25)\\
& \underline{\tau}_{\mathcal{D}} \le t_{\text{end}} && && (2.26)\\
& \underline{\tau}_{\mathcal{D}} - \overline{\tau}_{\mathcal{O}} \le T^{\max} && && (2.27)\\
& t_{\text{start}} \sum_{(i,j)\in\mathcal{A}} x_{ij} \le \underline{\tau}_i \le \overline{\tau}_i \le t_{\text{end}} \sum_{(i,j)\in\mathcal{A}} x_{ij} && \forall i \in \mathcal{S} && (2.28)\\
& t_{ij}x_{ij} - (1-x_{ij})\, t_{\text{end}} \le \underline{\tau}_j - \overline{\tau}_i \le t_{ij}x_{ij} + (1-x_{ij})\, t_{\text{end}} && \forall (i,j) \in \mathcal{A} && (2.29)\\
& \underline{\tau}_i + \Delta_i = \overline{\tau}_i && \forall i \in \mathcal{S} && (2.30)\\
& \sum_{k\in\mathcal{W}_j} y_{jk} \le 1 && \forall j \in \widetilde{\mathcal{S}} && (2.31)\\
& y_{jk} \le \sum_{(i,j)\in\mathcal{A}} x_{ij} && \forall k \in \mathcal{W}, \forall j \in \mathcal{S}_k && (2.32)\\
& \underline{\tau}_j \ge \gamma_k^L y_{jk} - \varphi_j && \forall k \in \mathcal{W}, \forall j \in \mathcal{S}_k && (2.33)\\
& \underline{\tau}_j \le \gamma_k^U y_{jk} + (1-y_{jk}) t_{\text{end}} && \forall k \in \mathcal{W}, \forall j \in \mathcal{S}_k && (2.34)\\
& \varphi_j \le \widetilde{\varphi} \sum_{k\in\mathcal{W}_j} y_{jk} && \forall j \in \widetilde{\mathcal{S}} && (2.35)\\
& \sum_{j\in\mathcal{S}_k} y_{jk} = 1 && \forall k \in \mathcal{W}^{\mathcal{NA}} && (2.36)\\
& \sum_{j\in\mathcal{S}_k} y_{jk} \le 1 && \forall k \in \mathcal{W}^{\mathcal{A}} && (2.37)\\
& \Delta_i \ge \varphi_i + \sum_{k\in\mathcal{W}_j} t_k^{\min} y_{ik} && \forall i \in \widetilde{\mathcal{S}} && (2.38)\\
& \sum_{j\in\mathcal{S}_{k+1}} y_{j,k+1} \le \sum_{j\in\mathcal{S}_k} y_{jk} && \forall k \in \mathcal{W} \setminus \{\, k_{last} \,\} && (2.39)\\
& \underline{\tau}_{\mathcal{D}} - \gamma_k^U \le z_k T^{\max} && \forall k \in \mathcal{W}^{\mathcal{A}} && (2.40)\\
& \gamma_k^U - \underline{\tau}_{\mathcal{D}} \le (1-z_k) T^{\max} && \forall k \in \mathcal{W}^{\mathcal{A}} && (2.41)\\
& \sum_{h\in\mathcal{S}_k} y_{hk} \ge z_k && \forall k \in \mathcal{W}^{\mathcal{A}} && (2.42)\\
& x_{ij} \in \{0,1\} && \forall (i,j) \in \mathcal{A} && (2.43)\\
& y_{jk} \in \{0,1\} && \forall j \in \mathcal{S}_k, \forall k \in \mathcal{W} && (2.44)\\
& z_k \in \{0,1\} && \forall k \in \mathcal{W}^{\mathcal{A}} && (2.45)\\
& \underline{q}_i \ge 0,\ \underline{\tau}_i \ge 0,\ \varphi_i \ge 0 && \forall i \in \mathcal{S}_{\mathcal{D}} && (2.46)\\
& \overline{q}_i \ge 0,\ \overline{\tau}_i \ge 0 && \forall i \in \mathcal{S}_{\mathcal{O}} && (2.47)\\
& \underline{\lambda}_{ik} \ge 0,\ \overline{\lambda}_{ik} \ge 0 && \forall i \in \mathcal{S}, \forall k \in \mathcal{B}_i && (2.48)\\
& \underline{w}_{ik},\ \overline{w}_{ik} \in \{0,1\} && \forall i \in \mathcal{S}, \forall k \in B_i \setminus \{0\} && (2.49)\\
& \underline{c}_i \ge 0,\ \overline{c}_i \ge 0,\ \Delta_i \ge 0 && \forall i \in \mathcal{S} && (2.50)
\end{aligned}
$$

The objective function (2.3) minimizes the total time. With constraints (2.4) every CS can be visited at most once. Constraints (2.5) impose the flow conservation conditions. Constraints (2.6) track the SoC of the EV for each pair of nodes. Constraint (2.7) impose that, at beginning, the EV is fully charged, while (2.8) impose a minimum charge at destination. Constraints (2.9) impose that the SoC of a leaving EV is greater than SoC when the EV is arrived at that CS. Also the EV can't arrive at CS i with no residual energy, and the maximum value of SoC must be Q. Constraints (2.10) to (2.16) define the SoC and the charging time, based on linear approximation of the charging function, upon arrival at CS, while constraints (2.17) to (2.23) define the same thing upon departure from CS. Constraints (2.24) define the time spent waiting on CS i. Constraint (2.25) impose the starting time, while (2.26) impose that the arrival at the destination cannot exceed t_{end}. Constraint (2.27) ensure that the journey last less then $T^{\max}$. Constraints (2.28) impose that the arrival time has to be lower than the departure, and both must be greater than $t^{\min}$ and less than t_{end}. Constraints (2.29) impose that the difference between arrival time in j and departure time from i is equal to t_{ij}. Constraints (2.30) link arrival, departure and waiting times. Constraints (2.31) assure that every CS $j \in \widetilde{\mathcal{S}}$ must be used for at most one time slot $k \in \mathcal{W}_j$, while (2.32) link x and y variables. Constraints (2.33) and (2.34) impose that, for every $k \in \mathcal{W}$, the arrival time is forced to be between γ_k^L and γ_k^U, considering also the tolerance φ_i. Constraints (2.35) links φ and y variables, imposing a maximum lead time of $\widetilde{\varphi}$. Constraints (2.36) assure that every required time slot is served, while (2.37) impose that optional time slot can also be unused. Constraints (2.38) impose a minimum waiting time if the EV is obliged to stop there. Constraints (2.39) describe the order in which the time slots must be used. Constraints (2.40) to (2.42) links the arrival time in $\mathcal{D}$ with the y variables and the avoidable time windows. Finally, (2.43) to (2.50) create the domains of the variables used in the formulation.

2.6. Shortest Path Model

The maximum amount of time of the journey can be computed as $T^{\max} = t_{\text{end}} - t_{\text{start}}$. However, using a large $T^{\max}$ may potentially lead to a trip that last too long just because is the solution that will maximize the profit. To handle this issue, we need to tune this parameter and tightening as much as possible.

What we did in this thesis is to solve initially an EV Shortest Path Problem **SPM**, defined as follows

$$
\begin{aligned}
[\mathbf{SPM}] \quad \min \quad & \sum_{(i,j)\in\mathcal{A}} t_{ij}x_{ij} + \sum_{i\in\mathcal{S}} \Delta_i \\
\text{s.t.} \quad & \text{constraints (2.4) to (2.26)} \\
& \text{constraints (2.28) to (2.50).}
\end{aligned}
\tag{2.51}
$$

The optimal solution T^{opt} is then a lower bound for T^{max}, so $T^{\text{opt}} \leq T^{\text{max}}$. Then let T^{add} be the total additional time that the user defines for detouring from the fastest path just to stops in node with higher scores. It is used to relax T^{opt} in order to find a feasible solution for the profit model **MPM**. Taking in account this change, we can now compute the maximum duration of the journey of the profit score model **MPM** with

$$
T^{\text{max}} = T^{\text{opt}} + T^{\text{add}}. \tag{2.52}
$$

2.7. Discounted weights

With the model **SPM** it is possible to find a path that minimize the total travel time. Instead with the model **MPM** is possible to find a path that maximize the total score obtained by visiting each CS of the path, that is not necessarily the shortest one, since the objective in this model is to maximize the profit. For this reason, we create a model that searches for a shortest path while maximizing the total score, denoting it with **MDPM**. In the evaluation part, in Chapter 5, we compare the scores obtained with **MPM** with the ones computed with **MDPM**.
Let T^{max} be the maximum duration of the journey computed in (2.52). For each arc $(i,j) \in \mathcal{A}$ with $j \in \mathcal{S}$, we create a new weight $\tilde{s}_{ij}$ defined as

$$
\tilde{s}_{ij} := t_{ij} + \Delta_j - \mu\sigma_j \tag{2.53}
$$

where μ is a coefficient that indicates how much importance we want to give to the score with respect to the needed time from i to j, charging time included.
The objective function of the **MDPM** model is then

$$
\min \sum_{\substack{(i,j)\in\mathcal{A} \\ j\in\mathcal{S}}} \tilde{s}_{ij}x_{ij}. \tag{2.54}
$$

Note that this quantity is non linear, since $\tilde{s}_{ij}$ includes the variable Δ_j in is definition, and so it becomes the product of two decision variables. To solve this issue, we introduce a new decision variable s_{ij} and we add some constraints that remove the non-linearity.
The **MDPM** is then defined as

$$[\mathbf{MDPM}] \quad \min \sum_{\substack{(i,j)\in\mathcal{A} \\ j\in\mathcal{S}}} [(t_{ij} - \mu\sigma_j)x_{ij} + s_{ij}] \tag{2.55}$$

$$\text{s.t.} \quad \text{constraints (2.4) to (2.50)}$$

$$s_{ij} \leq \Delta_j \qquad \forall (i,j) \in \mathcal{A} \text{ s.t. } j \in \mathcal{S} \tag{2.56}$$

$$s_{ij} \leq T^{\max} x_{ij} \qquad \forall (i,j) \in \mathcal{A} \text{ s.t. } j \in \mathcal{S} \tag{2.57}$$

$$s_{ij} \geq \Delta_j - T^{\max}(1 - x_{ij}) \qquad \forall (i,j) \in \mathcal{A} \text{ s.t. } j \in \mathcal{S} \tag{2.58}$$

$$s_{ij} \geq 0 \qquad \forall (i,j) \in \mathcal{A} \text{ s.t. } j \in \mathcal{S} \tag{2.59}$$

We define this model to compare its performance with the heuristic algorithm developed in section 3.4.

2.8. Reduce the number of legs

All the MILP models presented are exponentially large in the number of CSs, due to the huge number of arcs that are created for each pair of CSs. So the solver may have difficulties to find the optimal solution in a reasonable amount of time. To solve this issue, some action can be performed to drastically reduce the number of arcs and to speed up the computation.
The aim of the research is to find an optimal path for a single EV for a user that wants to perform a long trip, with some stops along the road for eating, sleeping and visit new places. With this in mind, and considering the fact that stopping too many times could be stressful, we want the number of charging stops as low as possible.
A new parameter so is introduced, $r^{\min}$, defined as

$$r^{\min} := \left\lfloor 0.4 \frac{Q - q_{\min}}{\eta} \right\rfloor \tag{2.60}$$

where η is the average energy consumption per kilometer (expressed in kWh/km), and is different for each EV that is taken in consideration. The quantity $\frac{Q-q_{\min}}{\eta}$ represent the maximum autonomy of the vehicle, excluding in the computation the minimal amount of energy that is always required. Then, $r^{\min}$ represent the 40% of the vehicle autonomy. With this arrangement is possible to prune all the arcs associated with a distance less

than $r^{\min}$. It also make sense since in this way the EV can't stops for charging after a short distance from the previous one, reducing in this way the total number of charging stops.
Let now be ξ the lower bound distance for the trip from $\mathcal{O}$ to $\mathcal{D}$. The description of how it is computed is in equation (4.2) in section 4.6. We can define the maximum number of legs N in a path as

$$N := \left\lceil \frac{3}{2} \left\lceil \frac{\xi}{r^{\min}} \right\rceil \right\rceil$$

Therefore, the following constraint is then added to the model **MPM**:

$$\sum_{(i,j)\in\mathcal{A}} x_{ij} \leq N \tag{2.61}$$

and, as a consequence, also to **SPM** and **MDPM**.

3 | Heuristic algorithm

In this chapter we propose an heuristic algorithm for the **SPM**. The heuristic is based on the A* Search, that find a path from an origin $\mathcal{O}$ to a destination $\mathcal{D}$ with the smallest cost. To do that, it maintains the tree of all the originated path from $\mathcal{O}$ and extends each path one arc at the time until $\mathcal{D}$ is reached. It uses a best-first search, meaning that it needs some sort of weight to decide from which node to continue the search. A* selects the node that minimizes the quantity

$$f(n) \coloneqq g(n) + h(n) \tag{3.1}$$

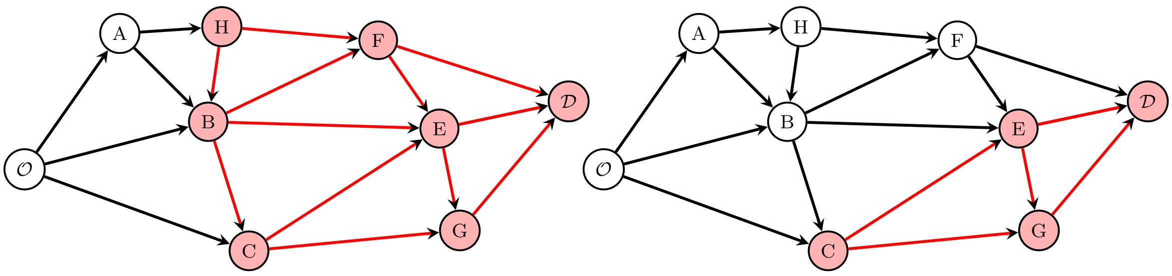
where n is the current node, $g(n)$ is the cost of the path from $\mathcal{O}$ to n, $h(n)$ is a heuristic function that estimates the cheapest cost from n to $\mathcal{D}$. If the $h(n)$ function never overestimates the real cost $h^*(n)$ to reach $\mathcal{D}$ from n, for all n, then the solution founded by the A* algorithm is the optimal.
First we compute the potentials for each node that will be used to estimate the heuristic function h; then we incorporate the time windows in the heuristic; later we try to add also the score in h; finally we solve the problem using the A* search algorithm. In all this chapter we use the same notation for sets and parameters that is presented in Chapter 2.

3.1. Potentials

We need to find an initial estimate of the total time from any CS i to $\mathcal{D}$. To do that, we on some techniques used in Zündorf [2014]. Dropping some constraints we obtain a simple problem that can be solved using the Dijkstra's algorithm. Let $\mathcal{G} \coloneqq \langle \mathcal{S}_{\mathcal{O},\mathcal{D}}, \mathcal{A} \rangle$ be the directed graph from $\mathcal{O}$ to $\mathcal{D}$, where $\mathcal{S}_{\mathcal{O},\mathcal{D}}$ is the set of nodes and $\mathcal{A} \coloneqq \mathcal{S}_{\mathcal{O}} \times \mathcal{S}_{\mathcal{D}}$ the set of arcs. The backward Dijkstra's algorithm applied to $\mathcal{G}$ is the Dijkstra applied to the reverted graph $\mathcal{G}' \coloneqq \langle \mathcal{S}_{\mathcal{O},\mathcal{D}}, \mathcal{A}' \rangle$ where

$$\mathcal{A}' \coloneqq \mathcal{S}_{\mathcal{D}} \times \mathcal{S}_{\mathcal{O}} \qquad \text{s.t. } (i,j) \in \mathcal{A} \Rightarrow (j,i) \in \mathcal{A}' \tag{3.2}$$

and so in $\mathcal{G}'$ we have that $\mathcal{O}$ became the target and $\mathcal{D}$ is considered as the starting point.



(a) Subtree of node H (b) Subtree of node C

Figure 3.1: In red is depicted $\mathcal{T}_H$ in the first figure, while $\mathcal{T}_C$ in the second one.

We start by dropping all the charging constraints and all time windows. The result is a simple problem that represents the minimization of the driving time from $\mathcal{O}$ to $\mathcal{D}$. Applying the backward Dijkstra algorithm is possible to find the unconstrained minimal driving time from every CS i to $\mathcal{D}$, and since adding battery constraints reduces the number of feasible paths, it will only increase the driving time. Also, time windows constraints reduce the number of feasible paths, so we obtain the lower bound of the minimal driving time $\boldsymbol{\pi}_{\text{dr}}(i)$ for each $i \in \mathcal{S}_{\mathcal{O},\mathcal{D}}$.
Now we want to add some information about the energy consumption. So we apply again the backward Dijkstra but this time with the energy consumption as a weight for the arcs. The result is that for each CS i we are now able to know the minimum amount of energy required from i to $\mathcal{D}$. We call this lower bound $\boldsymbol{\pi}_{\text{cons}}(i)$. In each node the EV arrives partially charged, with an amount of energy equal to $\text{SoC}(i)$. Moreover the minimal amount $q_{\min}$ needs to be respected in every i, so we need to slightly modify $\boldsymbol{\pi}_{\text{cons}}(i)$ to take in account those aspects. In each node the available energy is computed as $\text{SoC}(i) - q_{\min}$. To compute then the minimal amount of energy from i to $\mathcal{D}$ we need to subtract the available energy in i from $\boldsymbol{\pi}_{\text{cons}}(i)$, so let define

$$\tilde{\boldsymbol{\pi}}_{\text{cons}}(i) := \boldsymbol{\pi}_{\text{cons}}(i) - (\text{SoC}(i) - q_{\min}). \tag{3.3}$$

We now need to convert the minimal required energy $\tilde{\boldsymbol{\pi}}_{\text{cons}}(i)$ in amount of time in order to compute a lower bound for the charging time. We first define the subtree $\mathcal{T}_i$ of the graph $\mathcal{G}$ for a node i as the set of nodes in the directed graph $\mathcal{G}_i := \langle \mathcal{T}_i, \mathcal{A}_i \rangle$, where $\mathcal{A}_i \subseteq \mathcal{A}$. The set of nodes $\mathcal{T}_i$ contains all the nodes $j \in \mathcal{S}_{\mathcal{O},\mathcal{D}}$ such that there exists a sequence $\{\,(i, h_1), (h_1, h_2), \dots, (h_n, j)\,\}$ of arcs all contained in $\mathcal{A}_i$ (see fig. 3.1 and algorithm 3.1). We now define $s_{\max}(i)$ as the maximum charging rate of all the CSs $j \in \mathcal{T}_i$, defined as

$$s_{\max}(i) := \max\left\{\, \rho_{jk} : \forall j \in \mathcal{T}_i, \forall k \in \mathcal{B}_j = \{0, b_1, \dots, b_{m_j}\}\,\right\} \tag{3.4}$$

Algorithm 3.1 SubTree function. This function returns the list of the nodes that belongs to the subtree of the graph $\mathcal{G}$ generated from node i. The subtree is the directed subgraph of the directed graph $\mathcal{G}$, so it contains all the nodes that are reachable from i.

```
function SubTree(𝒢, i)
    𝒩 := { i }                              // Set of selected nodes
    𝒬 := { i }                              // Queue
    𝒫 := { i }                              // Processed
    while 𝒬 do
        c := Pop(Q)                         // Current node
        if c ∈ 𝒫 then
            go to 5
        end if
        𝒫 := 𝒫 ∪ { c }                      // Update processed
        ℋ := Star(𝒢, c)                     // See A
        𝒩 := 𝒩 ∪ ℋ                          // Update selected nodes
        𝒬 := 𝒬 ∪ ℋ                          // Update queue
    end while
    return 𝒩
end function
```

[A] Star($\mathcal{G}$, c) returns all the nodes $j \in \mathcal{S}_\mathcal{D}$ s.t. $(c, j) \in \mathcal{A}$, in descending order with respect to t_{cj}.

where ρ_{jk} is the slope of the charging function of node j for piecewise k (as defined in section 2.2). So $s_{\max}(i)$ is the maximal slope between the charging functions of all the CSs in $\mathcal{T}_i$, and it represents un upper bound for the charging speed for all the nodes from i to $\mathcal{D}$. This can be seen as a small improvement of the computation of the charging potential with respect to Zündorf [2014], where $s_{\max}$ is constant and it does not depend on the possible nodes that are actually reachable from i. Note that $\tilde{\boldsymbol{\pi}}_{\text{cons}}$ can be negative if the available energy is greater then the remaining energy needed to reach $\mathcal{D}$. So in the computation of a lower bound for charging time we need to consider two separate cases:

$$\boldsymbol{\pi}_{\text{ch}}(i) := \begin{cases} \dfrac{\tilde{\boldsymbol{\pi}}_{\text{cons}}(i)}{s_{\max}(i)} & \text{SoC}(i) - q_{\min} \leq \boldsymbol{\pi}_{\text{cons}}(i) \\ 0 & \text{otherwise} \end{cases} . \tag{3.5}$$

We now have a potential that returns the minimal charging time from any node i to $\mathcal{D}$. A lower bound for the total trip time can be computed simply as the sum of the minimal driving time and the minimal charging time

$$\widetilde{\boldsymbol{\pi}}_{\text{tt}}(i) := \boldsymbol{\pi}_{\text{dr}}(i) + \boldsymbol{\pi}_{\text{ch}}(i) \qquad \forall i \in \mathcal{S}_{\mathcal{O},\mathcal{D}}. \tag{3.6}$$

Again, note that also $\widetilde{\boldsymbol{\pi}}_{\mathrm{tt}}$ always underestimates the total trip time, since from the shortest path problem with charging constraints if we add the time windows constraints we only reduce the number of feasible paths, increasing the total trip time.
Using only $\widetilde{\boldsymbol{\pi}}_{\mathrm{tt}}$ as a potential for the total trip time can lead to a considerably large search space for the A* algorithm. This problem arise from the minimum stopping time of each time windows, especially if some of them are related to nights. Indeed, $\widetilde{\boldsymbol{\pi}}_{\mathrm{tt}}$ is an unconstrained lower bound for the total trip time and it does not take in account all the minimum stopping times. Lets now construct a better approximation by incorporating also the time windows, considering the obliged stopping time for each of them and let $\boldsymbol{\pi}_{\mathrm{tw}}(i)$ be the minimal stopping time that the EV must perform from i to $\mathcal{D}$. Recall that $\mathcal{W}^{\mathcal{NA}}$ is the ordered set of non avoidable time windows (see section 2.3), thus the minimum amount of time that the trip has to last must consider also the sum of all the minimum stopping times. Let $\tilde{k}$ be the last time window in the ordered set $\mathcal{W}^{\mathcal{NA}}$. Suppose that the EV when it is in node i has not performed all time windows in $\mathcal{W}^{\mathcal{NA}}$, then $g(i) \leq \gamma^L_{\tilde{k}} + t^{\min}_{\tilde{k}}$, where $g(i)$ is the arrival time in node i and $\gamma^L_{\tilde{k}}$ is the starting time of time windows $\tilde{k}$. In this case, we can compute a lower bound for the time windows potential as the sum of all the stopping times that are not yet performed by the EV at the time of $g(i)$. If instead, in node i, the EV has already done all the time windows in $\mathcal{W}^{\mathcal{NA}}$, then we have $g(i) > \gamma^L_{\tilde{k}} + t^{\min}_{\tilde{k}}$ and so the potential for the time windows must be zero. Therefore, let $\boldsymbol{\pi}_{\mathrm{tw}}(i)$ be the time windows potential for node i, we have

$$\boldsymbol{\pi}_{\mathrm{tw}}(i) \coloneqq \begin{cases} \displaystyle\sum_{\substack{k \in \mathcal{W}^{\mathcal{NA}}: \\ g(i) < \gamma^L_k}} t^{\min}_k & g(i) \leq \gamma^L_{\tilde{k}} + t^{\min}_{\tilde{k}} \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

For instance, suppose that $\mathcal{W}^{\mathcal{NA}}$ contains a stop for lunch for 1 hour, one tourism stop for 2 hours and one for sleeping for 11 hours. Then before lunch we have $\boldsymbol{\pi}_{\mathrm{tw}}(i) = 1+2+11 = 14$, after lunch we have $\boldsymbol{\pi}_{\mathrm{tw}}(i) = 2 + 11 = 13$, and the next day $\boldsymbol{\pi}_{\mathrm{tw}}(i) = 0$.
We now need to incorporate $\boldsymbol{\pi}_{\mathrm{tw}}$ in $\widetilde{\boldsymbol{\pi}}_{\mathrm{tt}}$. When the EV stops for lunch, it is also charging. Thus we can't simply sum $\boldsymbol{\pi}_{\mathrm{tw}}$ and $\boldsymbol{\pi}_{\mathrm{ch}}$, since they do not have completely distinct meanings. Instead, we can obtain a better lower bound considering the maximum between $\boldsymbol{\pi}_{\mathrm{ch}}$ and $\boldsymbol{\pi}_{\mathrm{tw}}$, and then add the driving potential $\boldsymbol{\pi}_{\mathrm{dr}}$: so

$$\boldsymbol{\pi}^1_{\mathrm{tt}}(i) \coloneqq \boldsymbol{\pi}_{\mathrm{dr}}(i) + \max\{\, \boldsymbol{\pi}_{\mathrm{ch}}(i), \boldsymbol{\pi}_{\mathrm{tw}}(i)\,\} \tag{3.8}$$

defines the lower bound for the total stopping time from i to $\mathcal{D}$. We could also take the minimum of them, but using the maximum we obtain a much more precise heuristic, and this will lead the A* algorithm to explore less labels. Note that we are not overestimating the real cost, since taking the maximum we are considering for each node the best possible charging scenario that at least the EV has to perform.

3.2. Labels

To apply the A* search algorithm we need to keep track of the $g(i)$ and $h(i)$ values for each node $i \in \mathcal{S}_{\mathcal{O},\mathcal{D}}$. Considering only the value of $f(i)$, however, is not enough, indeed, is possible that the EV arrives in the same node from different paths and so with different SoC or at different arrival times. So we need to keep track of those values when arriving at node i. We call these states labels. A label L_{j_m} represent the state of the EV when arrives in the m-th copy of node j. Each node j has M_j dynamically allocated copies, so $j_1, j_2, \ldots, j_m, \ldots, j_{M_j}$, indexed with $m = 1, \ldots, M_j$. Since it is possible to arrive in j from different nodes i, with different arrival times or with various SoC, we keep track of which state the EV is with this label. To reduce the notation, instead of writing all the functions with the parenthesis, we will write the node to which the function is applied as a subscript, so for instance we have $g_j^m := g(j_m)$. Each label is structured in a way that it includes the total time needed to reach j_m, so it includes both driving and charging time. For instance, suppose that the EV goes from the n-th copy of i to the m-th copy of j, namely from i_n to j_m. Then L_{j_m} is the label that considers the driving from i to j and the charge in i that is needed to reach j for the m-th time. Note that in L_{j_m} we include the charging time in i to reach j and not how much time the EV spent charging in j. To define the label L of node j_m, with i_n as its direct predecessor. Then, we have

$$L_{j_m} := (i,\ g_j^m,\ h_j^m,\ f_j^m,\ p_j^m,\ \beta_j^m,\ q_j^m,\ \underline{q}_j^m,\ \lambda_j^m,\ \Delta_j^m,\ \omega_j^m) \tag{3.9}$$

where

- i is the node from which the EV arrives to j_m;
- g_j^m is the actual total time traveled from $\mathcal{O}$ to j_m;
- h_j^m is the estimated remaining time from j_m to $\mathcal{D}$;
- f_j^m is the estimated arrival time from $\mathcal{O}$ to $\mathcal{D}$ if the path from $\mathcal{O}$ to j_m is performed. It is computed as $f_j^m := g_j^m + h_j^m$;
- p_j^m is the label from which the EV arrives, so it is equal to L_{i_n}, that is the label of the n-th copy of node i, with $n \in \{1, \ldots, M_i\}$;

- β_j^m is a positive real value that represents an additional time used to increase the amount of time spent charging, instead of relying only on the charging time needed to reach node j from i. Since it will be difficult to manipulate the real value of β, it is chosen each time from an ordered set $\boldsymbol{\beta} := \{\beta_1, \beta_2, \dots \beta_s\}$, with s finite, of discrete values.

- q_j^m is the amount of energy that is charged in the predecessor node i_n. It is computed to at least respect the consumption e_{ij}. It is defined as $q_j^m := \overline{q}_j^m - \underline{q}_j^m$, where

$$\overline{q}_j^m := \begin{cases} \underline{q}_i^n + e_{ij} & \text{if } e_{ij} < Q \\ Q & \text{otherwise} \end{cases} = \max\left\{ \underline{q}_i^n + e_{ij},\ Q \right\} \tag{3.10}$$

  so, it is the difference between the energy at departure and the energy of the EV when it arrives.

- $\underline{q}_j^m$ is the amount of energy that the EV has when it arrives at j_m. It is computed as $\underline{q}_j^m := \underline{q}_i^n + q_j^m - e_{ij}$, where e_{ij} is the energy weight in the arc (i, j);

- λ_j^m is the minimum amount of time that the EV must stay for charging in j_m. This amount of time is considered in the next label and not in L_{j_m}. This is coherent with the definition that is given for the labels, that is composed by the charging time in node i plus the driving time to reach node j. It is defined as

$$\lambda_j^m := \begin{cases} \max\left\{ 0,\ \gamma_k^L - g_j^m \right\} + t_k^{\min} & \text{if time window } k \text{ is performed in } i_n \\ 0 & \text{otherwise.} \end{cases} \tag{3.11}$$

  where γ_k^L is the starting time of time window k while $t_k^{\min}$ is its minimum stopping time. The time window k is retrieved using ω_j^m (see below).

- Δ_j^m is the charging time in i_n. It is computed as the maximum between λ_i^n and the difference $\overline{c}_j^m - \underline{c}_j^m$ where $\overline{c}_j^m := \Phi_j^{-1}(\overline{q}_j^m)$ and $\underline{c}_j^m := \Phi_j^{-1}(\underline{q}_i^n)$, with Φ_j^{-1} the inverse of the charging function in node j. In addition, to consider also the cases in which the EV charges more than it really needs, we add also the term β_j^m so

$$\Delta_j^m := \max\left\{ \lambda_i^n,\ \overline{c}_j^m - \underline{c}_j^m \right\} + \beta_j^m. \tag{3.12}$$

- ω_j^m represents the index of the last time window k that has been performed until node j_m.

In this way we fully keep track of the EV data during the entire path from $\mathcal{O}$ to $\mathcal{D}$.

Note that some parameters like Δ, q and $\underline{q}$ are strictly related to β. As a consequences, also g, h and f depends on β. The value of λ instead depends strictly on ω. This means that in reality the tags that properly characterize labels are p, β and ω, while all the others are computed using those three values. To avoid complex description, index and functions in pseudocode, we directly point out all the elements of L even if they are not strictly necessary.

3.3. A* Search Algorithm

Using the labeling system described in section 3.2 we are now able to implement the A* search algorithm. We start implementing a heuristic approach to find the fastest path from $\mathcal{O}$ to $\mathcal{D}$, referring to this as **AsM**.
The origin node $\mathcal{O}$ is initialized as follows:

$$\begin{aligned} L^1_{\mathcal{O}} := (i = \mathcal{O},\ g^1_{\mathcal{O}} = t_{\text{start}},\ h^1_{\mathcal{O}} = h^1(\mathcal{O}),\ f^1_{\mathcal{O}} = g^1_{\mathcal{O}} + h^1_{\mathcal{O}}, \\ p^1_{\mathcal{O}} = \textsc{None},\ \beta^1_{\mathcal{O}} = 0,\ q^1_{\mathcal{O}} = 0,\ \underline{q}^1_{\mathcal{O}} = Q,\ \lambda^1_{\mathcal{O}} = 0,\ \Delta^1_{\mathcal{O}} = 0,\ \omega^1_{\mathcal{O}} = 0) \end{aligned} \tag{3.13}$$

where with None we intend the absence of value. Note that $g^1_{\mathcal{O}}$ is initialized with the starting time of the trip. This means that the whole search is shifted, and so $f^m_{\mathcal{D}}$ represent the arrival time in $\mathcal{D}$ and not the how much time it will take to arrive to it.
Let $\mathcal{L}$ be the set of all labels, and M_j for all $j \in \mathcal{S}_{\mathcal{O},\mathcal{D}}$ a counter that keeps track in every iteration of the maximum index reached for the copies of each node j.
The algorithm keeps track of the open labels using a priority queue $\mathcal{Q}$. Every time a new label is created, it is added to $\mathcal{Q}$ in a way that the first element of $\mathcal{Q}$ is always the one which have the lowest $f^{\overline{m}}_{\overline{j}}$, between all labels L^m_j, so

$$\overline{j}_{\overline{m}} := \underset{j_m : j \in \mathcal{S}_{\mathcal{O},\mathcal{D}},\ m=1,\ldots,M_j}{\arg\min} \left\{ f^m_j \right\}.$$

We now introduce functions that are used but not written here in pseudocode. The function $\textsc{Pop}(\mathcal{Q})$ returns the label with the lowest f, while function $\textsc{Push}(\mathcal{Q}, L_{j_m})$ add the label L_{j_m} to the queue. Function $\textsc{Star}(\mathcal{G}, j)$ returns the set of nodes $h \in \mathcal{S}_{\mathcal{D}}$ such that $(j, h) \in \mathcal{A}$, in descending order with respect to t_{jh}. The function $\textsc{NextTW}(\mathcal{W}^{\mathcal{NA}}, \omega^m_j)$ returns, from $\mathcal{W}^{\mathcal{NA}}$, the next not visited time window when the EV arrives at node j_m. The maximum advance for time windows is set to φ for each node. The function $\textsc{NodeHasPoi}(i, \nu)$ returns a boolean value that represents if node i has or not in the neighborhood at least one POI of the given category ν. Function $\textsc{MaxSlope}(\dot{\mathcal{S}})$ applied to a generic subset $\dot{\mathcal{S}}$ of CS $\mathcal{S}$, returns the maximum slope between all the charging functions of nodes in $\dot{\mathcal{S}}$. The array FOREST$[i]$ returns $\mathcal{T}_i$, the subtree of node i: to speed up the algorithm, all the subtrees are previously computed and stored in the FOREST variable. The function ROUTING(i, j) returns the pair (t_{ij}, e_{ij}), that are respectively the time and the energy required to go from i to j, for all $i \in \mathcal{S}_{\mathcal{O}}$ and $j \in \mathcal{S}_{\mathcal{D}}$. The function MINSTOP$(g_i)$ returns $\boldsymbol{\pi}_{\text{tw}}(i)$. Some variables are defined globally, like $\mathcal{G}$, Q, $q_{\min}$, $\mathcal{W}^{\mathcal{NA}}$, the last time window $\tilde{k}$ in $\mathcal{W}^{\mathcal{NA}}$ and FOREST.

The A* algorithm is described in algorithm 3.2. It begins initializing the counters for all the copies and storing the subtree of each node in the graph $\mathcal{G}$. Then the label associated with the origin point is created and is added to the queue and to the set of all the labels. The search starts: the current label L_i^n with the lowest value of f is selected and then the algorithm finds the next time window k that must be performed. A check is performed to verify if the current label entails the arrival to the destination point: if it so, another control verifies that the index ω_i^n of the last visited time windows is at least equal to the cardinality of the set $\mathcal{W}^{\mathcal{NA}}$. If it is not the case, it means that the EV hasn't visited yet all the non avoidable time windows, and so the current label must be discarded. If instead it pass also this control, then the current label and the set of all the generated labels are returned and the search is finished.
At this point, the algorithm check if k is not NONE , and in the affermative case it checks if in the subtree $\mathcal{T}_i$ of the current node i there exists at least one node in which the POI constraint of time window k is satisfied. Then it perform the same check for all the time windows in $\mathcal{W}^{\mathcal{NA}}$ that must be performed after k. For instance, suppose that the next time window requires a lunch stop and after a tourism stop in Rome, Then the algorithm, using the subtree $\mathcal{T}_i$, checks if there exists at least one node associated with restaurants, and if it is true checks also if Rome is reachable from i. If this check fails, the current label will not satisfy all the constraints, so it is discarded and a new label is popped from the queue $\mathcal{Q}$.
We are now in the core of the algorithm. For each node j such that $(i, j) \in \mathcal{A}$, a sequence of operations is performed to assure that the trip from i_n to j_m is feasible, where m is the index of the m-th copy of j that will be created if all the checks are passed. First the energy constraints. If the energy required on arc (i, j) is grater than the maximum amount the EV can have, we discard this label, and the algorithm passes to the next node in STAR$(\mathcal{G}, i)$. Suppose now that $e_{ij} < Q - q_{\min}$. To add the possibility that a greater amount of energy with respect to e_{ij} is charged, all the instructions from now on are included in a for loop with increasing values of β.

Algorithm 3.2 ASTARSEARCH Algorithm. It returns the last label visited and the set of all the generated labels. In input it requires a graph $\mathcal{G}$, the maximum and the minimum EV energy Q and $q_{\min}$, the starting and ending times t_{start} and t_{end}, the set of non avoidable time windows $\mathcal{W}^{\mathcal{NA}}$ and the set of additional charging times $\boldsymbol{\beta}$

1: **function** ASTARSEARCH($\mathcal{G}$, Q, $q_{\min}$, t_{start}, t_{end}, $\mathcal{W}^{\mathcal{NA}}$, $\boldsymbol{\beta}$)
2:   **for each** node $h \in \mathcal{S}_{\mathcal{O},\mathcal{D}}$ **do** // $\mathcal{G} := \langle \mathcal{S}_{\mathcal{O},\mathcal{D}}, \mathcal{A} \rangle$
3:     FOREST$[h]$:= SUBTREE$(\mathcal{G}, h)$, $M_h := 0$
4:   **end for**
5:   $M_{\mathcal{O}} := 1$, Initialize $L^1_{\mathcal{O}}$, $\mathcal{L} := \{ L^1_{\mathcal{O}} \}$, $\mathcal{Q} := \{ \mathcal{O} \}$ // Initialize $L^1_{\mathcal{O}}$ as in (3.13)
6:   **while** $\mathcal{Q}$ **do**
7:     $L^n_i :=$ POP$(\mathcal{Q}) = (\bar{i}, g^n_i, h^n_i, f^n_i, p^n_i, \beta^n_i, q^n_i, \underline{q}^n_i, \lambda^n_i, \Delta^n_i, \omega^n_i)$ // Select current label
8:     $k :=$ NEXTTW$(\mathcal{W}^{\mathcal{NA}}, \omega^n_i)$ // See [A]
9:     **if** $i == \mathcal{D}$ **then** // $\mathcal{D}$ node reached
10:       **if** $\omega^n_i < |\mathcal{W}^{\mathcal{NA}}|$ **then go to** 6 // See [B]
11:       **return** L^n_i, $\mathcal{L}$
12:     **end if**
13:     **if** k **is not** NONE **then** // See [C]
14:       IDX $:= \omega^n_i$, C $:= k$
15:       **while** C **is not** NONE **do**
16:         **if** FOREST$[i] \cap \mathcal{S}_C == \emptyset$ **then go to** 6
17:         IDX := IDX $+ 1$, C := NEXTTW$(\mathcal{W}^{\mathcal{NA}}$, IDX$)$
18:       **end while**
19:     **end if**
20:     NEIGHBORS := STAR$(\mathcal{G}, i)$
21:     **for each** $j \in$ NEIGHBORS **do**
22:       $t_{ij}, e_{ij} :=$ ROUTING(i, j)
23:       **if** $e_{ij} > Q - q_{\min}$ **then go to** 21 // See [D]
24:       **for each** $\beta \in \boldsymbol{\beta}$ **do**
25:         $\Delta,\ q :=$ CHARGINGENERGY$(i, e_{ij}, \underline{q}^n_i)$
26:         $\Delta := \max\{ \lambda^n_i,\ \Delta \} + \beta$, $q :=$ CHARGINGFORTIME$(i, \underline{q}^n_i, \Delta)$
27:         $g_{\text{temp}} := g^n_i + \Delta + t_{ij}$ // Temporary g value
28:         **if** $g_{\text{temp}} > t_{\text{end}}$ **then go to** 21 // See [E]
29:         **if** k **is not** NONE **then**
30:           **if** $g_{\text{temp}} > \gamma^U_k$ **then go to** 21 // See [F]
31:           **if** $\gamma^L_k - \varphi \leq g_{\text{temp}} \leq \gamma^U_k$ **then** // See [G]
32:             **if** $j == \mathcal{D}$ **or not** NODEHASPOI(j, ν_k) **then go to** 38
33:             $M_j := M_j + 1,\ m := M_j,\ \lambda := \max\{0,\ \gamma^U_k - g_{\text{temp}}\} + t^{\min}_k$
34:             $L^m_j :=$ CREATELABEL$(i, j, g^n_i, \underline{q}^n_i, \Delta, q, t_{ij}, e_{ij}, \lambda, \omega^n_i + 1, \beta, L^n_i)$
35:             $\mathcal{L} := \mathcal{L} \cup \left\{ L^m_j \right\}$, PUSH$(\mathcal{Q}, L^m_j)$
36:           **end if**
37:         **end if**
38:         $M_j := M_j + 1,\ m := M_j$
39:         $L^m_j :=$ CREATELABEL$(i, j, g^n_i, \underline{q}^n_i, \Delta, q, t_{ij}, e_{ij}, 0, \omega^n_i, \beta, L^n_i)$
40:         $\mathcal{L} := \mathcal{L} \cup \left\{ L^m_j \right\}$, PUSH$(\mathcal{Q}, L^m_j)$
41:       **end for**
42:     **end for**
43:   **end while**
44:   **return** NONE , NONE // Node not found
45: **end function**

[A] Select next time window for i_n.
[B] Not visited all time windows in $\mathcal{W}^{\mathcal{NA}}$.
[C] Check if the subtree of current node contains the category of POI that is needed for the next time window k and for the subsequents ones; otherwise goes to the next element of the queue.
[D] Energy required is greater than the maximum available.
[E] Out of maximum time for the model.
[F] Out of maximum time for the current time window.
[G] Arriving in node j_m during time window k.

Algorithm 3.3 ChargingEnergy function. It returns a pair of values that represents respectively the charging time and the charged energy. In input it requires the node i in which the EV needs to charge, the energy e that is necessary to reach the next node from i, the current SoC $\underline{q}$.

---

function ChargingEnergy(i, e, $\underline{q}$)
    if $\underline{q} == Q$ **then** // See [A]
        return $0, 0$
    end if
    if $\underline{q} - e \geq q_{\min}$ **then** // See [B]
        return $0, 0$
    end if
    $s_2 = \max\{\, \underline{q} + e,\ Q \,\}$ // Final SoC
    $c_1 = \Phi_i^{-1}(\underline{q}), \quad c_2 = \Phi_i^{-1}(s_2)$
    return $c_2 - c_1, s_2 - \underline{q}$
end function

[A] Current state of charge is equal to Q, the maximum possible.
[B] If not necessary, return the minimal amount of energy needed to arrive in next node.

Algorithm 3.4 ChargingForTime function. It returns the amount of energy that will be charged if the EV arrive in node i with a SoC $\underline{q}$ and it will stop there for an amount of time Δ.

---

function ChargingForTime(i, $\underline{q}$, Δ)
    if $\underline{q} \geq Q$ **then** // See [A]
        return 0
    end if
    $c_1 = \Phi_i^{-1}(\underline{q}), \quad c_2 = c_1 + \Delta$
    $s_2 = \Phi_i(c_1)$
    return $s_2 - s_1$
end function

[A] Current state of charge is equal to Q, the maximum possible.

First, the function ChargingEnergy returns the charging time and the charged energy given the current SoC $\underline{q}_i^n$ and the amount of energy required e_{ij}. This function requires the current node i, e_{ij} and $\underline{q}_i^n$, and is used (see algorithm 3.3) to retrieve the charging time and the charged energy at i. The charging time Δ is then updated: it becomes the sum of the current additional time β and the maximum between the current value of Δ and the minimum stopping time λ_i^n. The charging energy q is then recomputed using ChargingForTime function (see algorithm 3.4).

Algorithm 3.5 HEURISTIC function. It returns the estimated remaining time from current node i and destination $\mathcal{D}$ in the graph $\mathcal{G}$, considering the current SoC $\underline{q}$ and the current arrival time g.

```
function HEURISTIC(i, q, g)
    π_tw(i) := MINSTOP(g)                                  // Time windows stops
    if q − q_min ≥ π_cons(i) then                           // See A
        π_ch(i) := 0
    else
        tree := SUBTREE(G, i)
        s_max := MAXSLOPE(tree)
        π_ch(i) := [π_cons(i) − (q − q_min)]/s_max
    end if
    return π_dr(i) + max { π_tw(i), π_ch(i) }
end function
```

A Available energy is potentially sufficient to reach $\mathcal{D}$.

The algorithm now computes a temporary value g_{temp} of the arrival time in j. In the case that $g_{\text{temp}} > t_{\text{end}}$, the current label is discarded since the arrival time will be greater than the limit imposed by the problem. We now need to verify if the time window k selected before is in $\mathcal{W}^{\mathcal{NA}}$ or not. If this control passes, then the algorithm must satisfy the constraints associated with k. First, if $g_{\text{temp}} > \gamma_k^U$ it means that the arrival time at j will be after the ending time of k, so its not feasible. If instead, g_{temp} is included in the range $\left[\gamma_k^L - \varphi, \gamma_k^U\right]$ then the EV arrives at j exactly during the time window k. In this situation, a user can decides to stop in j, and respect time window k's constraints, or drive again to the next node h and see if it is possible to respect k in node h. This scenario models the case in which, for instance, instead of respecting in node j the lunch constraints, the user wants to drive more and eats in an another place. In the case that we stay in j to respect time window k, lets then see if node j has the POI that is required for k. If it is the case, a label L_j^m is created, imposing that $\omega_j^m := \omega_i^n + 1$ (from j_m on, time window k is respected) and $\lambda_j^m = \max\left\{\, 0,\ \gamma_k^L - g_{\text{temp}} \,\right\} + t_k^{\min}$. The max function is used to compute how much time in advance the EV arrived in j, so the minimum stopping time that will be imposed from any arc from j_m will be λ_j^m. If instead node j does not have any POI of the given category ν_k, we can step over and create a label that goes from i_n to j_m without imposing a minimum stopping time λ_j^m. In this case $\lambda_j^m := 0$ and $\omega_j^m := \omega_i^n$. In both cases, the newly created label L_{j_m} is added to the set of labels $\mathcal{L}$ and pushed to the queue $\mathcal{Q}$. Finally, the loop on β restarts with the next value of β.

The end of the main while loop is met: if the queue $\mathcal{Q}$ has others elements, all the iteration is repeated, otherwise the destination node $\mathcal{D}$ it was not possible to reach $\mathcal{D}$ respecting all the imposed constraints and the function returns NONE .

Algorithm 3.6 CreateLabel function. It creates the label from node a to node b, giving the current arrival time g in a, the current SoC $\underline{q}$, the charging time Δ, the charged energy q, the driving time t and the driving energy e, the minimum amount of time that is needed to charge in node b in the next label, the last time windows index that has been performed ω, the charger additional time β and the previous label L.

```
function CreateLabel(a, b, g, q̲, Δ, q, t, e, λ, ω, β, L)
    g := g + Δ + t                              // Arrival time
    q̲ := q̲ + q − e                              // Energy at arrival
    h := Heuristic(b, q̲, g)                     // Heuristic value
    f := g + f                                  // Estimated arrival time
    L̃ := (a, g, h, f, L, β, q, q̲, λ, Δ, ω)       // See A
    return L̃
end function
```

A New label is created. Compare the order with (3.9).

3.3.1. An example

We will now analyze an example of the A* Search Algorithm described in section 3.3. Consider the Figure 3.2: it depicts the final path founded from $\mathcal{O}$ to $\mathcal{D}$. To better understand the example, all the times are not converted in absolute value with respect the midnight before the departure time, as described in chapter 2. So, only in this subsection, time variables are expressed as a timestamp: for instance t_{start} =10:00 and t_{end} =16:30 and not, respectively, $t_{\text{start}} = 10.0$ and $t_{\text{end}} = 16.5$, as in the rest of the thesis. For the same reason, also duration variables are expressed as timestamps, so for instance $\widetilde{\varphi}$ =0h 45' and not $\widetilde{\varphi} = 0.75$.

Suppose that the EV starts in $\mathcal{O}$ at t_{start} =10:00 o'clock fully charged, with $Q = 60\,\text{kWh}$ and $q_{\min} = 15\,\text{kWh}$. We want to find a path from $\mathcal{O}$ arriving at $\mathcal{D}$ before t_{end} =16:30 considering only one time windows related to lunch, with γ^L =12:00, γ^U =14:00 and a minimum stopping time $\widetilde{\varphi}$ =1h 00'. The maximum lead time is set to $\widetilde{\varphi}$ =0h 45'.
Let assume that the current label is L_A^1, the first copy of A. The current time is g_A^1 =11:50 and the next time window k that is possible to perform is the one related to lunch. Since k it is not None , the algorithm verifies in the SubTree(A) if there is at least one node that contains a restaurant. Then loops for all the nodes that are linked to A: node B, node C and node E. Since g_A^1 =11:50, the user can decides to stops in A to have lunch or to continue and have lunch in the next node. We first explain node B. From L_A^1, if the EV decides to continue then the labels L_B^1 and L_B^2 are generated and added to the queue. They differ in terms of the charging time in A before reaching B: Δ_B^1 =0h 25' and

$\Delta_B^2 = 0\text{h}\,40$'. If instead the EV decides to stop in A for lunch, then the labels L_B^3 and L_B^4, that differ again in terms of the charging time, are also generated and added to the queue. Those labels have respectively $\lambda_B^3 = 1\text{h}\,10$' and $\lambda_B^4 = 1\text{h}\,10$' (since the EV would arrive 10 minutes in advance in A with respect to γ^L), and for both the lunch time windows is denoted as $\omega_B^3 = \omega_B^4 = 1$. We now continue to C, the next node which is linked from A. If the EV does not stop in A for lunch, then it will arrive in C at $g_C^1 = 14{:}15$ that is after the maximal arrival time for lunch γ^U. So label L_C^1 will not be created. Instead, if the EV stops in A for lunch, then the label L_C^2 is generated, with $\omega_C^2 = 1$ and $\lambda_C^2 = 1\text{h}\,10$'. Finally, we check the node E. If the algorithm decides to go over instead to have lunch in A, then it will find out that the subtree of node E does not contain any restaurant. If instead it will stops in A for lunch, then L_E^2 is created, with $\omega_E^2 = 1$ and $\lambda_E^2 = 1\text{h}\,10$'.
All nodes linked to A are visited, so the algorithm passes to the next element of the queue. Suppose it is L_E^2. Among all the arcs outgoing from E there is one linked to $\mathcal{D}$. The arrival time in E is $g_E^2 = 14{:}05$ and it includes the lunch time in A and the driving time to E. From L_E^2 it is possible to reach $\mathcal{D}$, but the arrival time will be $g_{\mathcal{D}}^3 = 17{:}05$ which is greater than t_{end} so $L_{\mathcal{D}}^3$ must be discarded.
We now move to L_B^2. The arrival time is $g_B^2 = 13{:}40$. Among the outgoing arcs of B there are F and $\mathcal{D}$. If the EV decides that lunch will be performed on the next node, suppose $\mathcal{D}$, the label $L_{\mathcal{D}}^1$ will be generated, but it is then discarded since $\mathcal{D}$ is reached without having all the time windows required. The next node now is F, and suppose that lunch will be performed on L_B^2. The arrival time in L_F^1 is then $g_F^1 = 15{:}25$ and $\omega_F^1 = 1$.
Finally, suppose that the next element in the queue is L_F^1. From here it is possible to reach $\mathcal{D}$ at $g_{\mathcal{D}}^2 = 16{:}15$. This arrival time is coherent with the requirements of the problem, so it can be a possible candidate for terminating the algorithm. If there are labels with an f value less than 16:15, then the algorithm will continue with the next label in the queue, otherwise $L_{\mathcal{D}}^2$ is returned and the algorithm stops, returning the last current label $L_{\mathcal{D}}^2$ and the set $\mathcal{L}$ of all the generated labels. The final path is then computed by taking the predecessor of $L_{\mathcal{D}}^2$, then its predecessor and so on, until NONE is retrieved, meaning that we reach the label associated with the origin.

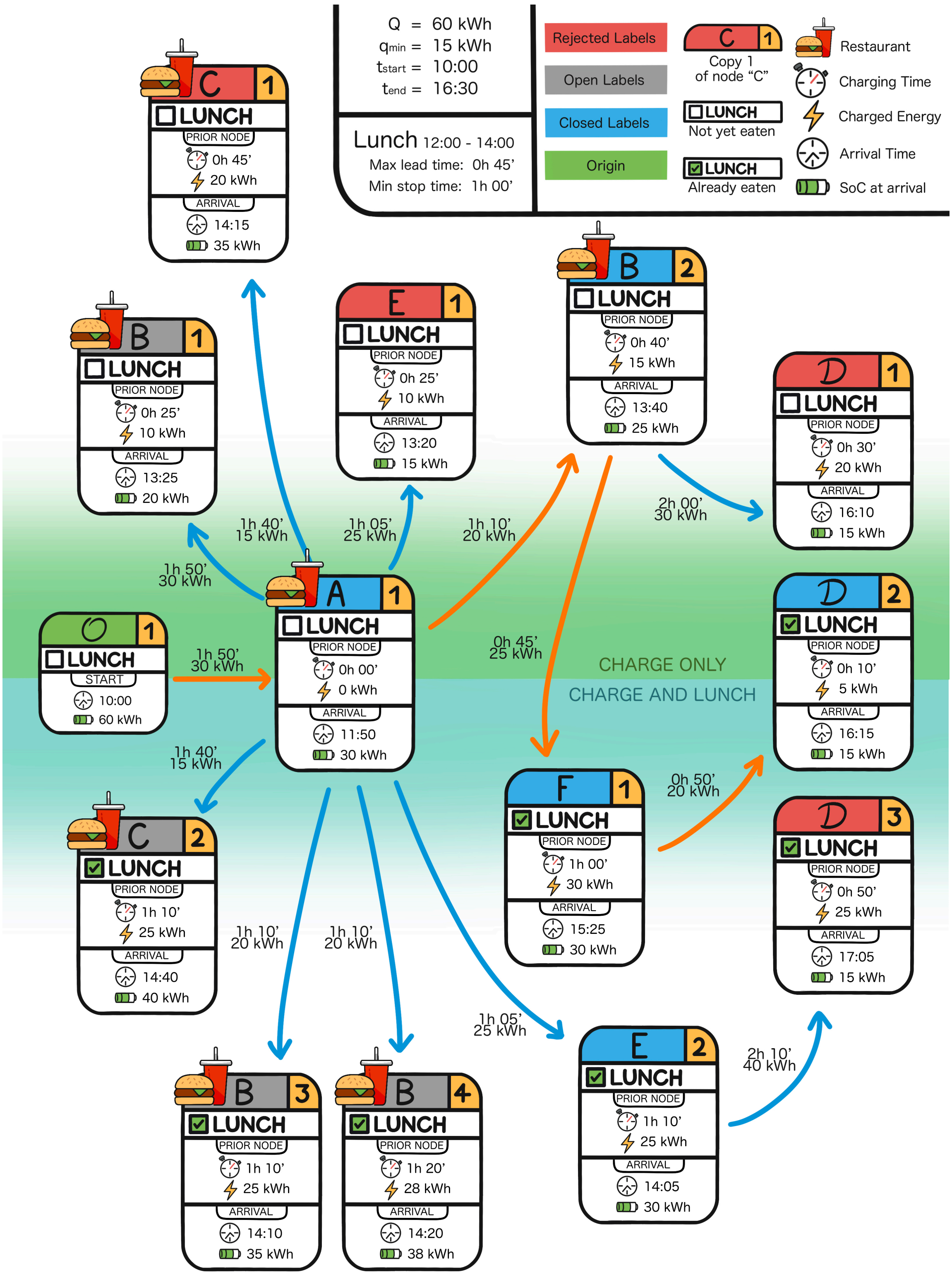


Figure 3.2: Schematic example of the A* search algorithm presented in algorithm 3.2.

3.4. Score

We now want to use the A* search algorithm to find a fastest path from $\mathcal{O}$ to $\mathcal{D}$ while maximize the total profit gained by visiting each node, as this is the objective of **MDPM**. We refer to this algorithm with **AsDM**. Let μ be a coefficient that describes how important a score must be with respect to the actual needed route time, as in section 2.7. Parameter μ is defined globally.
It uses a slightly modified version of algorithm 3.2. The first difference is in the definition of the labels. In particular, two new parameters are added to each label L:

$$L_{j_m} := (i,\ g_j^m,\ h_j^m,\ f_j^m,\ p_j^m,\ \beta_j^m,\ q_j^m,\ \underline{q}_j^m,\ \lambda_j^m,\ \Delta_j^m,\ \omega_j^m, \Sigma_j^m, \underline{\tau}_j^m) \tag{3.14}$$

The first one is Σ_j^m and it represents the total score gained from $\mathcal{O}$ to the current label, formally for copy j_m

$$\Sigma_j^m := \Sigma_i^n + \sigma_j \tag{3.15}$$

and $\Sigma_{\mathcal{O}}^1 := 0$. The second parameter is $\underline{\tau}_j^m$ and, as in chapter 2, it represents the arrival time in copy j_m. All the time windows constraints are now satisfied using this parameter instead of g. This means, for instance, that $\tau_{\text{temp}} = \underline{\tau}_j^m + \Delta - t_{ij}$, and every g_{temp} is replaced with τ_{temp}. The minimum waiting time becomes $\lambda_j^m := \max\left\{0,\ \gamma_k^L - \underline{\tau}_j^m\right\} + t_k^{\text{min}}$. For the origin node we have $\underline{\tau}_{\mathcal{O}}^1 := t_{\text{start}}$ and $g_{\mathcal{O}}^1 := 0$.
Another difference with algorithm 3.2 is given by the CreateLabel function that becames CreateLabelDiscounted (see algorithm 3.8). It is an obvious change due to the different definition of the labels.
The next big difference is in the heuristic function: Heuristic becomes HeuristicDiscounted and is described in algorithm 3.7. With respect to algorithm 3.5 it takes in input the additional parameter Σ and it returns the following discounted estimated time to reach $\mathcal{D}$:

$$\boldsymbol{\pi}_{\text{tt}}^2(i) := \boldsymbol{\pi}_{\text{tt}}^1(i) - \mu\Sigma. \tag{3.16}$$

In this way, the final value of f computed for node $\mathcal{D}$ will be comparable with the one computed with **MDPM**.

Algorithm 3.7 HEURISTICDISCOUNTED function. It returns the estimated remaining time from current node i and destination $\mathcal{D}$ in the graph $\mathcal{G}$, considering the current SoC $\underline{q}$, the current arrival time g and the current score gained Σ

function HEURISTICDISCOUNTED$(i,\ \underline{q},\ g,\ \Sigma)$
  $\boldsymbol{\pi}_{\text{tw}}(i) \coloneqq \max\left\{0,\ \gamma_{\tilde{k}}^{L} - g + t_{\tilde{k}}^{\min}\right\}$ // Time windows stops
  if $\underline{q} - q_{\min} \geq \boldsymbol{\pi}_{\text{cons}}(i)$ **then** // See [A]
    $\widetilde{\boldsymbol{\pi}}_{\text{cons}}(i) \coloneqq 0$
  else
    tree $\coloneqq$ SUBTREE$(\mathcal{G}, i)$
    $s_{\max} \coloneqq$ MAXSLOPE(tree)
    $\boldsymbol{\pi}_{\text{ch}}(i) \coloneqq \dfrac{\boldsymbol{\pi}_{\text{cons}}(i) - (\underline{q} - q_{\min})}{s_{\max}}$
  end if
  return $\boldsymbol{\pi}_{\text{dr}}(i) + \max\{\boldsymbol{\pi}_{\text{tw}}(i),\ \boldsymbol{\pi}_{\text{ch}}\}(i) - \mu\Sigma$
end function

[A] Available energy is potentially sufficient to reach $\mathcal{D}$.

Algorithm 3.8 CREATELABELDISCOUNTED function. It creates the label from node a to node b, giving the current discounted cost g to a, the current SoC $\underline{q}$, the charging time Δ, the charged energy q, the driving time t and the driving energy e, the minimum amount of time that is needed to charge in node b in the next label, the last time windows index that has been performed ω, the charger additional time β, the previous label L, the arrival time $\underline{\tau}$ and the current gained profit Σ.

function CREATELABELDISCOUNTED$(a,\ b,\ g,\ \underline{q},\ \Delta,\ q,\ t,\ e,\ \lambda,\ \omega,\ \beta,\ L,\ \underline{\tau},\ \Sigma)$
  $\underline{\tau} \coloneqq \underline{\tau} + \Delta + t$ // Arrival time
  $\underline{q} \coloneqq \underline{q} + q - e$ // Energy at arrival
  $\sigma \coloneqq$ SCORE(b) // See [A]
  $g \coloneqq g + \Delta + t - \mu\sigma$ // Discounted time
  $h \coloneqq$ HEURISTICDISCOUNTED$(B, \underline{q}, \underline{\tau}, \Sigma)$ // Heuristic value
  $f \coloneqq g + f$ // Estimated arrival time
  $\widetilde{L} \coloneqq (A, g, h, f, L, \beta, q, \underline{q}, \lambda, \Delta, \omega, \Sigma, \underline{\tau})$ // See [B]
  return $\widetilde{L}$
end function

[A] SCORE(b) returns the score of node b, namely σ_b.
[B] New label is created. Compare the order with (3.14).

4 | Data description

In this chapter we describe the dataset and the preprocessing approach that is used to clean and reduce the number of CS that are then used to perform our analysis. Moreover, we describe a simple process that leads to considering only CSs that are effectively useful to the trip that the user wants to fulfill. Some preprocess is executed also on the set of arcs $\mathcal{A}$ in order to discard useless arcs as much as possible.

4.1. Data

Data about charging station were given by a company in a SQL file. Each row contains information about every single station such as: single and group identifiers (`external_id` and `station_id`), charge data (`cp_type`, `connector_speed` and `status`), GPS position (`latitude`, `longitude`), address (`street`, `street_number`, `zip`, `city`, `district`, `state`, `country`), with a total amount of 83 526 entries. All this data were extracted from a bounding box that goes from 43.55 to 49.05 latitude and from 8.68 to 13.11 longitude (see Figure 4.1), and covers parts of Italy, Germany, Austria, Switzerland and the totality of Liechtenstein and San Marino (Table 4.1). The density of CSs changes according to the proximity with big cities. For instance, they are less distributed along the Alps and Apennines, and more near big cities like Stuttgart, München and Milan with respectively 450, 438 and 230 stations. The great number of CSs in cities can lead to problems due to the huge number of possible arcs that are created to connect each pair of them. For this reason, a first step is to reduce the size of $\mathcal{S}$, without loosing information for retrieving an optimal path.

4.2. Data cleaning

The first step is to clean up the database of the CSs that was given in input. Each row describes a single charger, but in some cases there were multiple CSs all in the same location (i.e., same parking lot or shopping mall), and so with the same identifier. We decide to group all the CSs that have the same `station_id`, and from each group taking

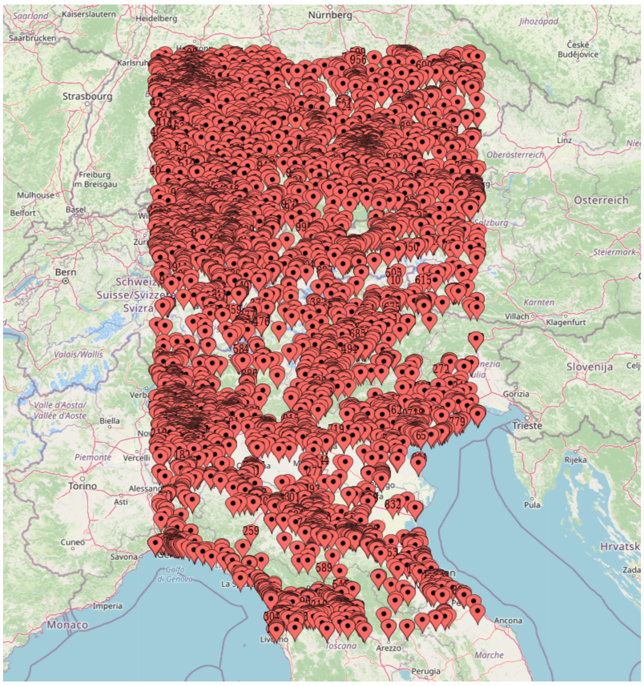


Figure 4.1: Distribution of the given CSs. Map created with mapcustomizer.com (Kaeding)

the one with the highest speed. So for the purpose of this thesis, we suppose that the station with the maximum speed is always available. This procedure reduced the number of CSs to 14 789. It can happen that a user arrives at a CS and finds it already occupied. Techniques that manage also this aspect are described in Keskin et al. [2019] and Kullman et al. [2021]. To take in consideration also this possibility, we suggest instead, without exploring it in this thesis, to include this information into the score, maybe increasing or decreasing it with respect to the probability that each charger is occupied or not.
Each CS has a different charging speed, that is used to categorize charging technologies in slow chargers, medium chargers and fast chargers. Let p_i be the charging speed of CS i. We categorize all the CSs as follows

- slow: $p_i \leq 11.0\,\mathrm{kWh}$
- medium: $11.0\,\mathrm{kWh} < p_i \leq 30.0\,\mathrm{kWh}$
- fast: $p_i > 30.0\,\mathrm{kWh}$.

Sometimes, stations near to each others are not correctly inserted in this database, so it may happen that even if they are in the same area they have different `station_id`. To solve this problem, all CS which have other CSs within a merging radius of $r_M = 100\,\mathrm{m}$ will form a cluster and if one CS belongs to two or more clusters, then all of this cluster will be merged.
To perform this preprocess we needed to compute the distance between each pair of nodes.

Country	# CSs before pre process	# CSs after grouping same station id	# CSs after grouping same area
Germany	59 339	5 955	4 135
Italy	15 009	5 713	4 911
Switzerland	5 400	1 583	1 334
Austria	3 679	1 488	1 219
Liechtenstein	65	33	27
San Marino	34	17	15
Total	**83 526**	**14 789**	**11 641**

Table 4.1: Distribution of chargers for each country.

Since the merging distance is quite small, we can compute an approximation supposing that distances were symmetric and using only the straight line distance with the Haversine formula. The Haversine distance is used as an initial filter since it is computationally faster than retrieving the real route distance from online map services. Moreover, it can also capture the curvature of the Earth. Suppose to have two points A and B with GPS coordinates (ϕ_1', ϕ_2') and (λ_1', λ_2'). Those value are initially converted in radians, (ϕ_1, ϕ_2) and (λ_1, λ_2), then let $\Delta\phi = \phi_2 - \phi_1$ and $\Delta\lambda = \lambda_2 - \lambda_1$. The Haversine distance between A and B is computed as

$$\text{hav}(A, B) := 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\lambda_1 \cos\lambda_2 \cos^2\left(\frac{\Delta\lambda}{2}\right)}\right) \tag{4.1}$$

where $R = 6\,371.009$ km is the Earth's average radius. Since the Earth is not a perfect sphere, this distance is not perfectly accurate, but for the scope of this thesis is sufficient. The distance matrix computation is quite fast indeed, thanks to symmetry.

Merging all the CSs in a cluster produces a new CS that is added to the database, while the CSs of the cluster were deleted. The cluster position is computed as an average of the GPS coordinates of the original CSs, while the charging speed is the maximum among the original CSs. Using this procedure, 3 148 CSs were deleted, while 1 226 new ones were created, bringing the total number of CSs to 11 641. We denote this final dataset with Γ. The final number of CSs that are taking into consideration for each country is described in Table 4.1.

4.3. Weights Matrix

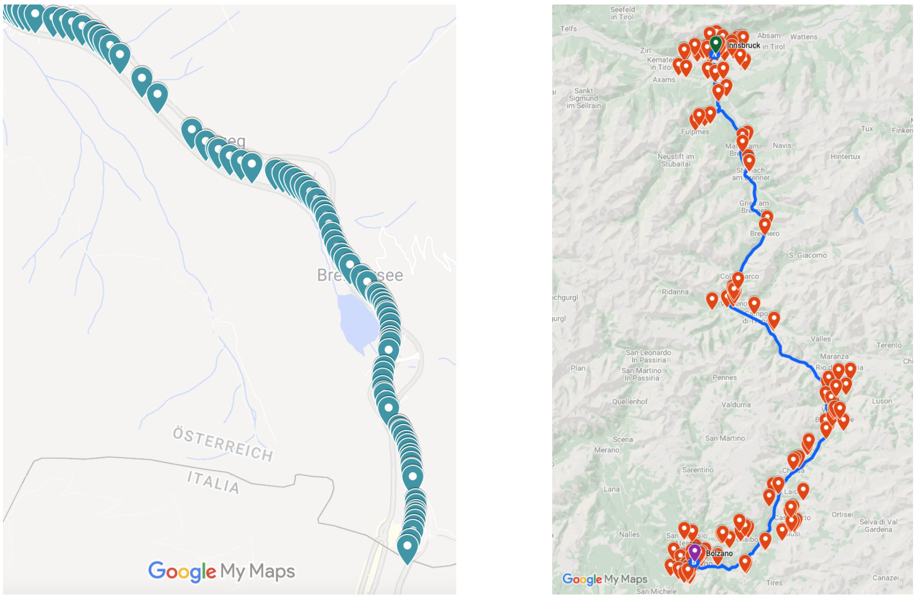
Using the post processed database, we then computed the distance and timing matrix for each pair of CSs. To do that we performed multiple requests to the Open Source Routing Machine API (OSRM, Luxen and Vetter [2011]) server and stored the result in a JSON file. In this way all the weights were ready to use, without affecting the computation of the models presented in this thesis. The energy required for each arc is instead computed as the product of the arc length and the average consumption per kilometer, as done also in Montoya et al. [2017].

4.4. Machine Performance

The entire model has been implemented in *Python* and solved using *IBM ILOG CPLEX Optimization Studio 20.1.0* (IBM ILOG CPLEX [2009]). The computation was performed on a single core of an Apple MacBook Pro with 8 core Apple M1 processor clocked up to 3.20 GHz, with 8 GB of LPDDR4 RAM.

4.5. Time windows

Time windows are given as input in the form described in section 2.3 for both required $\mathcal{W}^R$ and optional $\mathcal{W}^O$ time windows sets. Let $\mathcal{W}^T$ be the set of time windows related to a tourism stop. For each $k \in \mathcal{W}^T$ the additional parameter $\mathcal{P}_k$ is given, that is the pair of GPS coordinates of that place. To construct the set $\mathcal{S}_k$, the haversine distance is computed by searching for CSs in a radius r_T centered in $\mathcal{P}_k$. The CSs that have a distance less then r_T are included in $\mathcal{S}_k$. If no eligible nodes are presents, the radius is iteratively increased by a quantity δ_T and the search is repeated. The maximum radius imposed for this search is $\widetilde{r}_T$: if $\mathcal{S}_k$ is still empty, the entire computation is blocked and an error is raised informing that is not possible to reach $\mathcal{P}_k$ unless the EV is left more than $\widetilde{r}_T$ away. This process simulates the approach that a user needs to do to search for a CS near the location she wants to visit. If no CSs are available in a reasonable distance from $\mathcal{P}_k$, than the user can decide to remove or change that tourism stop. For our experiments, we imposed $r_T = 2.0\,\text{km}$, $\delta_T = 200\,\text{m}$ and $\widetilde{r}_T = 4.0\,\text{km}$.



(a) (b)

Figure 4.2: On the left, polyline representing part of the Brennero's highway (fig. 4.2a). On the right, instead, the optimal path from Innsbruck to Bolzano (blue line) computed via OSRM. The pinpoints in red are the position of the CS selected from Γ (fig. 4.2b).

4.6. Construction of $\mathcal{S}$ and $\mathcal{A}$

Given the origin point $\mathcal{O}$, the destination point $\mathcal{D}$ and the set of tourism stops $\mathcal{W}^T$, we now analyze how to construct the set of chargers $\mathcal{S}$ and the set of arcs $\mathcal{A}$.
Using OSRM, the optimal path without charging stops is computed and it will be used as a lower bound for **SPM**. This makes sense since the fastest path from $\mathcal{O}$ to $\mathcal{D}$ can't be along slower routes for ICEVs. The server returns up to three alternatives in a JSON file that contains the entire routes from $\mathcal{O}$ to $\mathcal{D}$, with all the navigation details. The interesting elements that will be used are:

- `distance`: is the total length in meters of the optimal path computed by OSRM
- `steps`: it contains a list of the single steps that the vehicle must perform (turns, roundabout, etc.)
- `geometry`: each step contains a string composed by ASCII characters.

The lower bound on the optimal solution is defined as

$$\xi := \min \{ d_i | \text{ where } d_i \text{ is the distance of alternative } i \} . \tag{4.2}$$

Each step is composed by its starting point and its ending point which are connected by

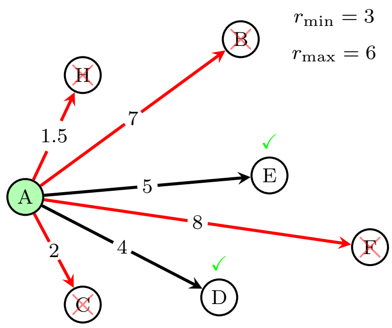


Figure 4.3: Example of the range preprocessing that is performed to reduce the size of $\mathcal{A}$.

a polyline. The GPS coordination of the points that form the polyline are encoded in the `geometry` string (fig. 4.2a). Taking the union of all these points, is possible to reconstruct on the map the optimal path that OSRM has computed. Now, all the points for all the alternatives are placed in a unique set of points. For each point, all the CSs that have a haversine distance within a radius of 5 km are collected. This is due to the fact that sometimes to reach a CS a small detour is needed. Since those points are very dense, to speed up the process, and without loosing too much data, the search has been performed only in one point every six. At the end, all the CSs that are in a bandwidth of 5 km centered on the OSRM optimal route will form the set $\mathcal{S}$. Finally we construct $\mathcal{A}$ as the set of all possible arcs (i, j) with $i \in \mathcal{S}_{\mathcal{O}}$ and $j \in \mathcal{S}_{\mathcal{D}}$, with $i \neq j$.

4.7. Preprocessing

Before starting to solve the **MDPM** we preprocess the set of chargers $\mathcal{S}$ and the set of arcs $\mathcal{A}$. We start by removing from $\mathcal{S}$ all the nodes that have a distance from $\mathcal{O}$ less then $r^{\min}$. This is due to the fact that we want the solution to have a small number of stops, since the objective is to perform a long trip on the road. We suppose that deleting them does not change the optimal solution since selecting one of those will cause the EV to stop for charging, increasing the total travel time just after few kilometers from the origin point. The same reason can be applied, using again $r^{\min}$, for $\mathcal{D}$ node, indeed is better to arrive at a destination instead stopping again for charging near $\mathcal{D}$. Consequently, all arcs in $\mathcal{A}$ related to those points can be deleted.
Is possible also remove all arcs that are going to the opposite side with respect $\mathcal{D}$. More precisely, let $d[i]$ be the distance from CS i to $\mathcal{D}$. For each arc $(i, j) \in \mathcal{A}$, if by going from i to j the distance to $\mathcal{D}$ is reduced, that means $d[i] > d[j]$, then (i, j) is kept, otherwise it is deleted. This process substantially halves the amount of edges included in $\mathcal{A}$.

Another type of preprocessing is given again by using $r^{\min}$. Let $r^{\max}$ be

$$r^{\max} := \left\lfloor \frac{Q - q_{\min}}{\eta} \right\rfloor. \tag{4.3}$$

Since the EV needs to respect the battery capacity Q and we want the total travel time to be low, we have decided to forbid the EV to go through arcs that are too small or too long. In particular, we remove from $\mathcal{A}$ all arcs (i, j) that have a distance $d_{ij} < r^{\min}$ or $d_{ij} > r^{\max}$ (Figure 4.3).

Finally, after all preprocessing, we found some nodes that are not reachable from any other CS, and some recurrent nodes (CSs that don't have any outgoing arc). Those nodes and their relative arcs are safely deleted.

5 | Computational experiments

In this chapter we create some instances that are used to test the performance of the MILP formulations (**SPM**, **MPM** and **MDPM**) and the A* search heuristics (**AsM** and **AsDM**).

5.1. Vehicle

The EV that we choose for our analysis is a Škoda Enyaq iV 60, with a net battery capacity of $Q \coloneqq 58\,\mathrm{kWh}$ and a maximum average consumption of $\eta \coloneqq 0.187\,\mathrm{kWh/km}$ (ChargePrice.com). It has a maximum power charge $P \coloneqq 40\,\mathrm{kW}$, and we set a minimum required energy $q_{\min} \coloneqq 15\,\mathrm{kWh}$. The breakpoints used for the charging functions related to the selected vehicle are reported, for each CS technology, in fig. 5.1.

5.2. Datasets and instances

We initially created two subsets of the CSs instead of using all the 11 641 CSs. The first one, denoted with Γ_1, is very small with only 650 CSs ($\sim$5.5%) and it is used in order to obtain an exact solution with the MILP models and to properly compare then with the heuristic approach. The second one, Γ_2, instead contains 5 813 CSs ($\sim$50.0%) used to test the A* search with larger datasets. Since the two datasets are extracted with uniform probability from Γ, we need to guarantee that in each tourism stop there is at least one CS. We created a square of $5\,\mathrm{km}\times 5\,\mathrm{km}$ centered in the coordinates of each tourism stop and uniformly extracted 4 CSs. Then we selected uniformly from Γ a certain amount of CSs and we take the union with the ones selected before for the tourism stops, obtaining Γ_1 and Γ_2.

For both sets of CSs we use the same set of instances defined as trips. Since our CSs lay in a rectangular area that covers part of Central Europe, those instance are chosen so that they completely lay in the same geographical area. We generate three main trips, each one with some variations like starting time, presence of tourism stops and its mini-

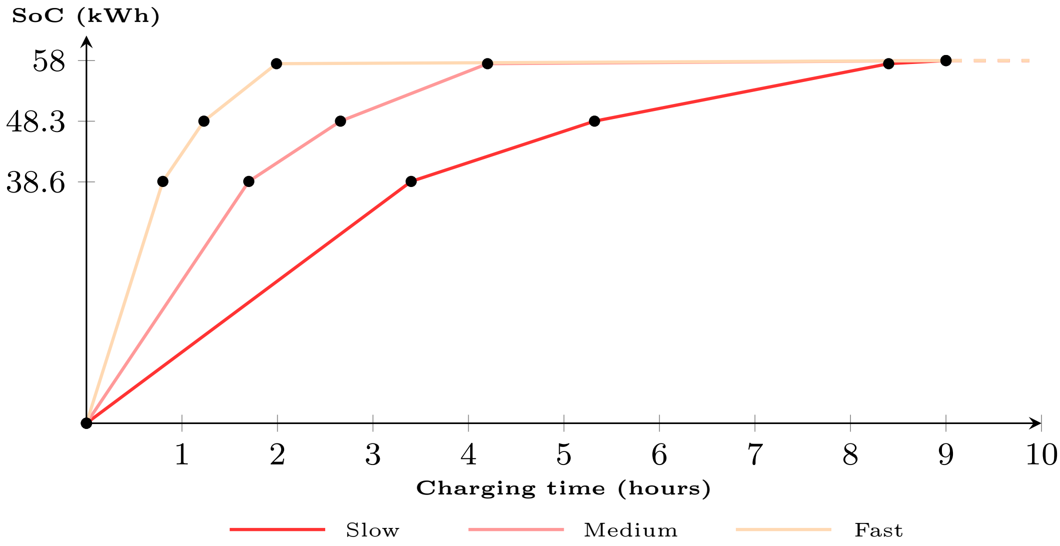


Figure 5.1: Piecewise linear approximation, including the fictitious breakpoint, for different CSs technologies for the selcted EV.

mum stopping time, presence or not of lunch and nights time windows. Those trips are from Genoa to Zürich, from Livorno to Regensburg and from Stuttgart to Ancona. The details of each instance are summarized in table 5.1.

We tested Γ_1 with the MILP models **SPM**, **MPM** and **MDPM**, and both the A* heuristics, **AsM** and **AsDM**, while Γ_2 with only **AsM** and **AsDM**. Then for all **AsDM** models we solve the **SPM** imposing that the EV must use the arcs selected by the heuristic. We note that CPLEX was unable to solve any instance in Γ_2 within one hour. We denote with TS_x and TT_x respectively the total score and the trip time of the model x. Those values are retrieved a posteriori on the path returned by the models.

Description of the instances

ID	Origin		Destination		Stop at			Lunch Break		Nights		
	From	When	To	By	Stop at	When	Min stop [h]	When	Min stop [h]	Nights	When	Min stop [h]
Ge_Zu_1	Genoa	10:00 Day 0	Zürich	18:30 Day 1	Lugano	14:00-17:30 Day 0	2	-	-	0	-	-
Ge_Zu_2	Genoa	10:00 Day 0	Zürich	18:30 Day 1	Lugano	14:00-17:30 Day 0	2	12:00-13:30 Every day	1	0	-	-
Li_Re_1	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	-	-	-	12:00-13:30 Every day	1	0	-	-
Li_Re_2	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	-	-	-	-	-	1	19:00-22:30 Every day	11
Li_Re_3	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	12:00-13:30 Every day	1	0	-	-
Li_Re_4	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	-	-	1	19:00-22:30 Every day	11
Li_Re_5	Livorno	04:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
Li_Re_6	Livorno	06:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
Li_Re_7	Livorno	08:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
Li_Re_8	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2.5	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
Li_Re_9	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	2	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
Li_Re_10	Livorno	10:00 Day 0	Regensburg	18:30 Day 1	Verona	14:00-17:30 Day 0	3	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
St_An_1	Stuttgart	10:00 Day 0	Ancona	18:30 Day 1	-	-	-	12:00-13:30 Every day	1	0	-	-
St_An_2	Stuttgart	10:00 Day 0	Ancona	18:30 Day 1	Vaduz	14:00-17:30 Day 0	2	12:00-13:30 Every day	1	0	-	-
St_An_3	Stuttgart	20:00 Day 0	Ancona	18:30 Day 1	Bologna	7:00-10:30 Day 1	2	12:00-13:30 Every day	1	0	-	-
St_An_4	Stuttgart	10:00 Day 0	Ancona	18:30 Day 1	Vaduz	14:00-17:30 Day 0	2	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
St_An_5	Stuttgart	10:00 Day 0	Ancona	18:30 Day 1	Bologna	7:00-10:30 Day 1	2	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11
St_An_6	Stuttgart	06:00 Day 0	Ancona	02:00 Day 1	Vaduz Bologna	09:30-13:00 Day 0 18:30-22:00 Day 0	2 2	-	-	0	-	-
St_An_7	Stuttgart	10:00 Day 0	Ancona	22:00 Day 1	Vaduz Bologna	14:00-17:30 Day 0 9:30-12:30 Day 1	2 2	-	-	1	19:00-22:30 Every day	11
St_An_8	Stuttgart	13:00 Day 0	Ancona	22:00 Day 1	Vaduz Bologna	14:00-17:30 Day 0 14:00-17:30 Day 1	2 2	12:00-13:30 Every day	1	1	19:00-22:30 Every day	11

Table 5.1: Description of the instances

5.2.1. Instances details

As we can see in table 5.1, some instances describe the same trip, just with different timings, and for this reason they lead to the same set of unconstrained shortest optimal path (see section 4.6), and so to the same set of CSs. In particular, we can subdivide the instances in the following manner:

- `Ge_Zu_1` and `Ge_Zu_2`
- `Li_Re_1` and `Li_Re_2`
- from `Li_Re_3` to `Li_Re_10`
- `St_An_1`
- `St_An_2` and `St_An_4`
- `St_An_3` and `St_An_5`
- `St_An_6`, `St_An_7` and `St_An_8`.

For this reason, we extract, for both Γ_1 and Γ_2, the set of CSs related to each subdivision and store those sets in memory. In this way, each instance that belongs to the same subdivision uses the same graph for all the models. Then the sets $\mathcal{S}_{\mathcal{O},\mathcal{D}}$ and $\mathcal{A}$ are computed and preprocessed, as described in section 4.6. In table table 5.2 we report the number of nodes and arcs for each subdivision, for both dataset Γ_1 and Γ_2, before and after the preprocessing phase.

Instances	Dataset Γ_1				Dataset Γ_2			
	Before		After		Before		After	
	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs	Nodes	Arcs
`Ge_Zu_1`, `Ge_Zu_2`	36	1 260	22	52	296	87 320	212	3 236
`Li_Re_1`, `Li_Re_2`	43	1 806	36	157	480	229 920	391	22 230
`Li_Re_3` to `Li_Re_10`	49	2 352	42	244	492	241 572	403	23 889
`St_An_1`	84	6 972	54	276	667	444 222	415	19 762
`St_An_2`, `St_An_4`	104	10 712	73	629	820	671 580	555	34 622
`St_An_3`, `St_An_5`	83	6 806	54	276	666	442 890	414	19 698
`St_An_6` to `St_An_8`	103	10 506	73	629	819	669 942	554	34 542

Table 5.2: Number of nodes and arcs for each instance.

ID	SPM			AsM			$\frac{TT_{\text{AsM}} - TT_{\text{SPM}}}{TT_{\text{SPM}}}$
	Trip Time [h]	Total Score	Comp. Time [s]	Trip Time [h]	Total Score	Comp. Time [s]	×100 [%]
Ge_Zu_1	8.629	3.735	0.092	8.629	3.735	0.002	0.000
Ge_Zu_2	8.966	4.060	0.057	8.966	4.060	0.005	0.000
Li_Re_1	14.130	6.942	6.994	14.545	6.942	0.017	2.937
Li_Re_2	23.594	8.301	4.267	24.627	5.365	0.138	4.378
Li_Re_3	15.749	18.322	9.827	16.047	15.375	0.125	1.892
Li_Re_4	24.574	11.740	3.413	25.530	11.740	0.141	3.890
Li_Re_5	30.558	14.250	7.139	31.412	13.820	0.283	2.795
Li_Re_6	28.558	13.199	8.899	30.246	16.904	0.158	5.911
Li_Re_7	26.558	11.320	7.917	27.412	11.320	0.387	3.216
Li_Re_8	25.313	13.924	5.423	26.271	12.339	0.329	3.785
Li_Re_9	25.088	18.077	10.057	25.943	13.976	0.369	3.408
Li_Re_10	25.709	7.797	6.559	26.749	11.664	0.308	4.045
St_An_1	15.767	13.395	24.409	16.306	6.884	0.056	3.419
St_An_2	18.182	15.952	7.870	18.778	15.952	0.057	3.278
St_An_3	16.861	9.593	6.661	17.360	12.616	0.225	2.959
St_An_4	28.674	10.134	6.572	28.813	10.134	0.095	0.485
St_An_5	26.341	13.450	3.182	26.387	13.450	0.082	0.175
St_An_6	18.971	9.396	0.917	19.689	13.055	0.124	3.785
St_An_7	29.037	11.955	1.029	29.258	13.055	0.199	0.761
St_An_8	29.813	11.955	0.488	30.258	13.055	0.044	1.493
Average			6.089			0.157	2.631

Table 5.3: Comparison between **SPM** and **AsM** using Γ_1

5.3. Comparison Phase

We analyze the performance of the A* search with respect to the exact solution achieved by the MILPs, so we use the small dataset Γ_1. Initially we find the shortest path objective value T^{opt} of **SPM** and **AsM**. Then we compute the maximum relaxed time $T^{\max} = T^{\text{opt}} + T^{\text{add}}$, and we solve the **MPM** model. For the evaluation, we fix the value of T^{add} to 1 h 30', for all the models, so $T^{\text{add}} := 1.5$. The result is then compared to the profit gained with **MDPM** and **AsDM** for different values of the score multiplier μ. We set μ to 0.75, 1.0 and 2.0. For the heuristic models we set $\boldsymbol{\beta} := \{\, 0.0,\ 1.0,\ 4.0 \,\}$.

5.3.1. Shortest Path

We start solving the shortest path problem for the **MPM** and **AsM** models. The results for each instance are presented respectively in table N.1 and table N.6. As a summary, we can see the comparison of this two models in table 5.3. Using Γ_1, for each instance we have a small post processed graph, therefore CPLEX solves **SPM** relatively quickly, with the majority of the instances solved in less than 10 seconds. The solution computed with **AsM** is instead retrieved in less than 0.5 seconds, with a percentage change in average less than 2.631 % with respect to the exact solution.

ID	MPM Comp. Time [s]	MPM Total Score	MDPM Comp. Time [s]	MDPM Total Score	AsDM Comp. Time [s]	AsDM Total Score	$\frac{TS_{\text{MPM}} - TS^{2.0}_{\text{MDPM}}}{TS_{\text{MPM}}} \times 100$ [%]	$\frac{TS_{\text{MPM}} - TS^{2.0}_{\text{AsDM}}}{TS_{\text{MPM}}} \times 100$ [%]	$\frac{TS^{2.0}_{\text{MDPM}} - TS^{2.0}_{\text{AsDM}}}{TS^{2.0}_{\text{MDPM}}} \times 100$ [%]
Ge_Zu_1	0.036	7.679	0.062	7.679	0.003	7.679	0.000	0.000	0.000
Ge_Zu_2	0.039	4.576	0.106	4.576	0.006	4.576	0.000	0.000	0.000
Li_Re_1	0.488	19.201	0.711	19.201	0.017	12.776	0.000	33.462	33.462
Li_Re_2	1.332	16.576	2.105	16.576	0.005	13.780	0.000	16.868	16.868
Li_Re_3	1.165	23.183	1.604	23.183	0.054	22.558	0.000	2.696	2.696
Li_Re_4	1.300	22.624	2.913	22.624	0.081	18.647	0.000	17.579	17.579
Li_Re_5	0.793	22.807	2.700	22.807	0.046	22.807	0.000	0.000	0.000
Li_Re_6	0.598	20.929	3.609	20.929	0.044	15.081	0.000	27.942	27.942
Li_Re_7	0.667	21.598	2.473	21.598	0.006	21.307	0.000	1.347	1.347
Li_Re_8	1.414	22.784	3.510	22.502	0.008	17.712	1.238	22.261	21.287
Li_Re_9	1.537	23.876	3.086	23.854	0.007	21.416	0.092	10.303	10.221
Li_Re_10	0.652	17.373	3.202	17.373	0.013	14.720	0.000	15.271	15.271
St_An_1	1.361	17.116	1.788	17.047	0.007	15.201	0.403	11.188	10.829
St_An_2	3.070	18.515	2.774	18.515	0.017	16.077	0.000	13.168	13.168
St_An_3	1.377	15.195	2.925	15.195	0.081	11.379	0.000	25.114	25.114
St_An_4	2.636	13.690	3.350	13.690	0.013	13.331	0.000	2.622	2.622
St_An_5	2.014	17.452	3.448	17.452	0.157	14.553	0.000	16.611	16.611
St_An_6	0.501	18.156	0.830	18.030	0.019	18.030	0.694	0.694	0.000
St_An_7	0.325	19.340	0.514	19.215	0.009	19.215	0.646	0.646	0.000
St_An_8	0.279	19.340	0.459	19.215	0.009	15.631	0.646	19.178	18.652
Average	1.079		2.108		0.030		0.186	11.847	11.683

Table 5.4: Comparison of total score and computational time between **MPM**, **MDPM** and **AsDM** with $\mu = 2.0$ in Γ_1

5.3.2. Maximum Profit

To measure the difference **MDPM** and **MPM**, we solve the same instances with the **MPM**. Since Γ_1 is pretty small, the computational time for the CPLEX is quite fast. The results are reported in table N.2. In table 5.4 we report the computational time and the maximum total score that is possible to gain.

5.3.3. Discounted Models

For each value of μ we solve the **MDPM** and **AsDM** models. The detailed results for the MILP formulations are reported in table N.3, table N.4 and table N.5, and as we can see they are almost identical, with the sole exceptions of `Li_Re_3` and `Li_Re_7`, in which higher values of μ find solutions with slightly better total scores. For the A* search approach, instead, the results are in table N.7, table N.8 and table N.9. Even in this case the total score increase with higher values of μ. As we expected, increasing the weight that the score must have in the decision process, both models return solutions with better scores, with respect to the models with lower μ.

ID	SPM Trip Time $[h]$	MDPM Trip Time $[h]$	AsDM Trip Time $[h]$	Ref. SPM Trip Time $[h]$	$\frac{TT^{2.0}_{\text{MDPM}} - TT_{\text{SPM}}}{TT_{\text{SPM}}}$ ×100 [%]	$\frac{TT^{2.0}_{\text{AsDM}} - TT_{\text{SPM}}}{TT_{\text{SPM}}}$ ×100 [%]	$\frac{TT^{2.0}_{\text{Ref. SPM}} - TT_{\text{SPM}}}{TT_{\text{SPM}}}$ ×100 [%]	$\frac{TT^{2.0}_{\text{Ref. SPM}} - TT^{2.0}_{\text{AsDM}}}{TT^{2.0}_{\text{AsDM}}}$ ×100 [%]
Ge_Zu_1	8.629	9.005	9.939	9.005	4.357	15.181	4.357	−9.397
Ge_Zu_2	8.966	8.987	8.987	8.987	0.234	0.234	0.234	0.000
Li_Re_1	14.130	15.524	15.943	15.655	9.866	12.831	10.793	−1.806
Li_Re_2	23.594	24.659	25.634	24.392	4.514	8.646	3.382	−4.845
Li_Re_3	15.749	16.629	16.783	16.061	5.588	6.565	1.981	−4.302
Li_Re_4	24.574	25.522	26.631	25.503	3.858	8.371	3.780	−4.236
Li_Re_5	30.558	30.943	32.274	30.943	1.260	5.616	1.260	−4.124
Li_Re_6	28.558	29.715	31.116	28.578	4.051	8.957	0.070	−8.157
Li_Re_7	26.558	27.183	27.575	26.666	2.353	3.829	0.407	−3.296
Li_Re_8	25.313	26.288	26.593	25.445	3.852	5.057	0.521	−4.317
Li_Re_9	25.088	26.045	27.045	26.180	3.815	7.801	4.353	−3.198
Li_Re_10	25.709	26.653	26.874	26.617	3.672	4.531	3.532	−0.956
St_An_1	15.767	16.865	17.451	17.389	6.964	10.681	10.287	−0.355
St_An_2	18.182	18.628	19.932	19.457	2.453	9.625	7.012	−2.383
St_An_3	16.861	17.460	17.441	17.266	3.553	3.440	2.402	−1.003
St_An_4	28.674	29.667	30.017	29.787	3.463	4.684	3.882	−0.766
St_An_5	26.341	27.242	27.403	27.349	3.421	4.032	3.827	−0.197
St_An_6	18.971	19.333	19.974	19.334	1.908	5.287	1.913	−3.204
St_An_7	29.037	29.153	29.596	29.153	0.399	1.925	0.399	−1.497
St_An_8	29.813	29.870	30.376	29.870	0.191	1.888	0.191	−1.666
Average					3.489	6.459	3.229	−2.985

Table 5.5: Comparison of total trip time between **SPM**, **MDPM**, **AsDM** and the refined **SPM** with $\mu = 2.0$ in Γ_1

In table table 5.4 we report a comparison between the total score gained and the computational time of **MPM**, **MDPM** and **AsDM** when $\mu = 2.0$. In average, the percentage change between the maximum possible score computed with **MPM** and the one computed with **MDPM** is 0.186%, while for **AsDM** there is an average of 11.847% worse scores with respect to the optimal solution computed by **MPM**. This last value is quite large, with some instances having a percentage change of over 25%. A possible approach to obtain better results maybe rely on different values of the parameter μ. The computational times is quite low for all the models, but we are considering the small graph created with the CSs in Γ_1.

Finally, the solutions obtained with the A* search are refined using a MILP formulation. In particular, we create another **SPM** in which we set $x_{ij} = 1$ for each arc (i, j) selected in the final solution of **AsDM**. The total trip time obtained by both the approaches are quite similar. For $\mu = 2.0$ the results are summarized in table 5.5 and compared with the shortest path model **SPM**. The total trip time computed via **AsDM** is in average 6.459% worse than the shortest path, but this average decrease to 3.229% when compared with the refined **SPM**. We can also see that from **AsDM** and the refined **SPM** there is an average total trip time reduction of -2.985%.

ID	AsM		AsDM, $\mu = 0.75$		AsDM, $\mu = 1.0$		AsDM, $\mu = 2.0$		$\frac{TS^{0.75}_{\text{AsDM}} - TS_{\text{AsM}}}{TS_{\text{AsM}}}$	$\frac{TS^{1.0}_{\text{AsDM}} - TS_{\text{AsM}}}{TS_{\text{AsM}}}$	$\frac{TS^{2.0}_{\text{AsDM}} - TS_{\text{AsM}}}{TS_{\text{AsM}}}$
	Comp. Time [s]	Total Score	Comp. Time [s]	Total Score	Comp. Time [s]	Total Score	Comp. Time [s]	Total Score	$\times$100 [%]	$\times$100 [%]	$\times$100 [%]
Ge_Zu_1	0.064	8.478	0.008	9.747	0.012	9.747	0.011	9.747	14.968	14.968	14.968
Ge_Zu_2	0.064	7.099	0.030	8.546	0.031	8.546	0.032	9.078	20.383	20.383	27.877
Li_Re_1	0.742	6.155	0.380	22.784	0.077	22.784	71.918	22.784	270.171	270.171	270.171
Li_Re_2	175.537	11.685	8.964	20.985	14.044	20.985	25.716	20.985	79.589	79.589	79.589
Li_Re_3	36.531	4.707	0.504	23.165	0.547	23.165	0.715	23.267	392.139	392.139	394.306
Li_Re_4	77.321	8.562	0.231	27.341	0.267	27.341	0.437	19.348	219.330	219.330	125.975
Li_Re_5	196.607	10.986	36.612	26.214	6.758	26.214	4.423	26.214	138.613	138.613	138.613
Li_Re_6	463.745	14.870	24.277	20.022	22.675	20.022	25.821	20.022	34.647	34.647	34.647
Li_Re_7	213.243	16.491	0.495	27.048	0.405	27.394	0.330	27.394	64.017	66.115	66.115
Li_Re_8	132.141	8.785	0.132	18.701	0.283	23.862	0.076	24.508	112.874	171.622	178.976
Li_Re_9	78.603	12.870	0.914	19.443	0.485	19.443	0.328	23.638	51.072	51.072	83.667
Li_Re_10	128.441	7.644	0.074	23.671	0.072	23.671	0.374	17.543	209.668	209.668	129.500
St_An_1	0.689	8.054	99.609	19.167	134.355	19.167	540.437	19.167	137.981	137.981	137.981
St_An_2	2.837	5.556	0.200	21.176	0.503	21.176	0.212	16.992	281.138	281.138	205.832
St_An_3	99.768	9.723	12.145	18.964	15.034	18.964	40.176	18.964	95.043	95.043	95.043
St_An_4	90.617	18.624	0.311	22.079	62.193	26.095	95.962	26.095	18.551	40.115	40.115
St_An_5	92.701	12.800	2.726	18.772	2.738	15.562	4.666	15.562	46.656	21.578	21.578
St_An_6	30.212	4.098	28.898	14.297	34.371	14.297	36.549	10.530	248.878	248.878	156.955
St_An_7	148.879	6.880	32.558	12.919	37.872	12.939	38.373	19.030	87.776	88.067	176.599
St_An_8	185.645	11.381	0.273	18.563	0.218	18.563	0.088	18.847	63.105	63.105	65.601
Average	107.719		12.467		16.647		44.332		129.330	132.211	122.205

Table 5.6: Comparison of total score and computational time between **AsM** and **AsDM** with $\mu = 0.75$, $\mu = 1.0$ and $\mu = 2.0$ in Γ_2

5.4. Medium Dataset

We want to test our A* search algorithm with the medium dataset Γ_2. We tested all the instances using only the heuristic approach, so with **AsM** and **AsDM**. The results are reported in table N.10, table N.11, table N.12 and table N.13.

Comparing the results in table 5.6, we can see that we obtain an average increase of the total score with $\mu = 1.0$ with respect to $\mu = 2.0$. The difference is due to the fact that with $\mu = 2.0$ the shortest arcs became more important, even with a lower score in the arrival node. It might be possible that tuning μ, also dynamically for each arc, the final solution might improve. The computational time for the **AsM** model are quite high, with an average of 118.77 sec for the trip from Livorno to Regensburg and of 81.42 sec for the trip from Stuttgart to Ancona. If instead we analyze the **AsDM** models, we see a meaningful drop in the computational time, with some exceptions. In particular, the instances that continue to have higher computational times are the ones that have large time intervals without any time windows constraints. For instance, `St_An_1`, after the lunch break, has no other time window that constraints the problem, so the research for a best bound solutions takes more time.

ID	AsM	AsDM, $\mu = 0.75$	AsDM, $\mu = 1.0$	AsDM, $\mu = 2.0$	$\frac{TT^{0.75}_{\text{AsDM}} - TT_{\text{AsM}}}{TT_{\text{AsM}}}$	$\frac{TT^{1.0}_{\text{AsDM}} - TT_{\text{AsM}}}{TT_{\text{AsM}}}$	$\frac{TT^{2.0}_{\text{AsDM}} - TT_{\text{AsM}}}{TT_{\text{AsM}}}$
	Trip Time $[h]$	Trip Time $[h]$	Trip Time $[h]$	Trip Time $[h]$	$\times 100$ [%]	$\times 100$ [%]	$\times 100$ [%]
Ge_Zu_1	8.603	8.623	8.623	8.623	0.232	0.232	0.232
Ge_Zu_2	8.818	8.910	8.910	9.530	1.043	1.043	8.074
Li_Re_1	13.477	14.920	14.920	14.920	10.707	10.707	10.707
Li_Re_2	23.131	24.510	24.510	24.510	5.962	5.962	5.962
Li_Re_3	15.373	16.792	16.792	16.652	9.230	9.230	8.320
Li_Re_4	24.808	26.291	26.291	26.259	5.978	5.978	5.849
Li_Re_5	30.988	31.955	31.955	31.955	3.121	3.121	3.121
Li_Re_6	28.599	30.050	30.050	30.050	5.074	5.074	5.074
Li_Re_7	26.599	27.972	27.641	27.641	5.162	3.917	3.917
Li_Re_8	25.009	26.387	26.386	26.420	5.510	5.506	5.642
Li_Re_9	24.558	25.460	25.460	26.019	3.673	3.673	5.949
Li_Re_10	25.520	26.802	26.802	26.919	5.024	5.024	5.482
St_An_1	15.099	16.569	16.569	16.569	9.736	9.736	9.736
St_An_2	17.527	18.950	18.950	18.989	8.119	8.119	8.341
St_An_3	15.806	16.978	16.978	16.978	7.415	7.415	7.415
St_An_4	27.322	28.726	28.667	28.667	5.139	4.923	4.923
St_An_5	26.156	27.492	27.406	27.406	5.108	4.779	4.779
St_An_6	17.771	18.980	18.980	19.245	6.803	6.803	8.294
St_An_7	27.995	29.467	29.485	29.080	5.258	5.322	3.876
St_An_8	29.683	30.183	30.183	30.953	1.684	1.684	4.279
Average					5.499	5.412	5.999

Table 5.7: Comparison of total trip time between **AsM** and **AsDM** with $\mu = 0.75$, $\mu = 1.0$ and $\mu = 2.0$ in Γ_2

The same applies for `Li_Re_1`, `Li_Re_2` and `St_An_6`. Instead `St_An_8`, that is the instance with the most number of time windows, is solved in less than 1 second in each discounted model. When the time windows are balanced along the trip, with not too many uncovered time intervals, the computation with the A* search is very fast, even in medium sized graphs.

In table 5.7 we can see an average total trip time variation of less than 6.0% in the **AsDM** models with respect to the **AsM**.

6 | Conclusions

The aim of this thesis is to develop an algorithm that finds an approximate solution to the realistic situation in which a user wants to organize a long road trip with an EV. Selecting the CSs that better matches the preferences of the user can be difficult especially if the user wants to respect also time windows constraints.
As reported in the Chapter 5, the A* algorithm **AsM** performs very well with respect to the exact solution **SPM** for computing the shortest path with charging and time windows constraints. The heuristic approach implemented in this thesis is quite fast for small instances, but the computational time tends to increase as the graph increase in size. If instead, the trip chosen has many time windows, then the computation is very fast even in medium sized graph. The same applies also for the **AsDM**, that is quite fast in finding a shortest path solution with high total score.
The computational time of the MILP formulations increase exponentially in the number of nodes in the graph, so it can be computationally infeasible to solve even in medium sized graphs. The A* search algorithm solves this problem, keeping only the states of the EV that are promising. On the contrary, however, the heuristic search can't manage real values of the additional charging time, so we need to discretize those values using the set $\boldsymbol{\beta}$. Solving again the **SPM** model with the arcs selected by the heuristic helps to optimize the charging times on the final solution. However, very large graphs tend to reduce also the performances of the heuristic.

6.1. Future work

While we developed a heuristic approach that works for small and medium instances, there is still potential for further improvements. For instance, it is possible to integrate more information inside the score value, and not only the POIs preferences, in order to better capture user's predilections or CSs characteristics. Further research can also incorporate more efficient heuristic potentials, so that particular instances can benefit more from the A* search algorithm speedup techniques. In particular, a better lower bound for estimating the arrival time in $\mathcal{D}$ can takes in account the starting time of the last non avoidable time window with its minimum stopping time. Indeed, since the EV is obliged to stop in the last non avoidable time window, then at least it will arrive in $\mathcal{D}$ after completing this time window. Finally, to account also for larger instances, it might be possible to better preprocess the initial graph, removing useless nodes or arcs, maybe taking in account the natural time dependency of the problem with respect to the time windows or with the presence of the POIs.

Bibliography

[1] J. D. Adler, P. B. Mirchandani, G. Xue, and M. Xia. The electric vehicle shortest-walk problem with battery exchanges. *Networks and Spatial Economics*, 16(1):155–173, 2016.

[2] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[3] R. Bauer, D. Delling, P. Sanders, D. Schieferdecker, D. Schultes, and D. Wagner. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. In *International Workshop on Experimental and Efficient Algorithms*, pages 303–318. Springer, 2008.

[4] R. Bauer, T. Columbus, I. Rutter, and D. Wagner. Search-space size in contraction hierarchies. *Theoretical Computer Science*, 645:112–127, 2016.

[5] M. Baum, J. Dibbelt, A. Gemsa, D. Wagner, and T. Zündorf. Shortest feasible paths with charging stops for battery electric vehicles. In *Proceedings of the 23rd SIGSPATIAL international conference on advances in geographic information systems*, pages 1–10, 2015.

[6] M. Baum, J. Dibbelt, T. Pajor, J. Sauer, D. Wagner, and T. Zündorf. Energy-optimal routes for battery electric vehicles. *Algorithmica*, 82(5):1490–1546, 2020.

[7] M. Baum, J. Dibbelt, D. Wagner, and T. Zündorf. Modeling and engineering constrained shortest path algorithms for battery electric vehicles. *Transportation Science*, 54(6):1571–1600, 2020.

[8] M. Bruglieri, A. Colorni, and A. Lue. The vehicle relocation problem for the one-way electric vehicle sharing: an application to the milan case. *Procedia-Social and Behavioral Sciences*, 111:18–27, 2014.

[9] M. Bruglieri, F. Pezzella, O. Pisacane, and S. Suraci. A matheuristic for the electric vehicle routing problem with time windows. *arXiv preprint arXiv:1506.00211*, 2015.

[10] M. Bruglieri, S. Mancini, and O. Pisacane. The green vehicle routing problem with

capacitated alternative fuel stations. *Computers & Operations Research*, 112:104759, 2019.

[11] H. I. Calvete, G. Carmen, J. Mara, and S. Beln. Vehicle routing problems with soft time windows: An optimization based approach. *Journal of Monografas del Seminario Matemtico Garca de Galdeano*, 31:295–304, 2004.

[12] H. I. Calvete, C. Galé, M.-J. Oliveros, and B. Sánchez-Valverde. A goal programming approach to vehicle routing problems with soft time windows. *European Journal of Operational Research*, 177(3):1720–1733, 2007.

[13] A. ChargePrice.com. Open EV Data, Chargeprice.app API. `https://github.com/chargeprice/open-ev-data`. Accessed: 18-03-2022.

[14] G. Coplon-Newfield and S.-J. Park. Corporate fleets making the switch to electric vehicles. `https://www.ecowatch.com/corporate-fleets-electric-vehicles-2499038220.html`, 2017. Accessed: 26-03-2022.

[15] S. Erdoğan and E. Miller-Hooks. A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, 48(1):100–114, 2012.

[16] European Union. Directive 2014/94/EU of the European Parliament and of the Council of 22 October 2014 on the deployment of alternative fuels infrastructure Text with EEA relevance. `https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32014L0094`, 2014. Accessed: 26-03-2022.

[17] A. Froger, J. E. Mendoza, O. Jabali, and G. Laporte. Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104:256–294, 2019.

[18] R. Geisberger, P. Sanders, D. Schultes, and C. Vetter. Exact routing in large road networks using contraction hierarchies. *Transportation Science*, 46(3):388–404, 2012.

[19] D. Goeke and M. Schneider. Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81–99, 2015.

[20] A. V. Goldberg and C. Harrelson. Computing the shortest path: A search meets graph theory. In *SODA*, volume 5, pages 156–165. Citeseer, 2005.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[22] F. Hartmann and S. Funke. Energy-efficient routing: Taking speed into account. In *Joint German/Austrian Conference on Artificial Intelligence (Künstliche Intelligenz)*, pages 86–97. Springer, 2014.

[23] C. Harto. Electric vechicle ownership costs: today's electric vehicle offer big savings for consumers. `https://advocacy.consumerreports.org/wp-content/uploads/2020/10/EV-Ownership-Cost-Final-Report-1.pdf`, 2020. Accessed: 26-03-2022.

[24] G. Hiermann, J. Puchinger, S. Ropke, and R. F. Hartl. The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995–1018, 2016.

[25] IBM ILOG CPLEX. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009.

[26] P. Kaeding. MapCustomizer. `https://www.mapcustomizer.com`. Accessed: 12-03-2022.

[27] M. Keskin, G. Laporte, and B. Çatay. Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, 107:77–94, 2019.

[28] N. D. Kullman, J. C. Goodson, and J. E. Mendoza. Electric vehicle routing with public charging stations. *Transportation Science*, 55(3):637–659, 2021.

[29] J. Li, F. Wang, and Y. He. Electric vehicle routing problem with battery swapping considering energy consumption and carbon emissions. *Sustainability*, 12(24):10537, 2020.

[30] J. Lin, W. Zhou, and O. Wolfson. Electric vehicle routing problem. *Transportation research procedia*, 12:508–521, 2016.

[31] D. Luxen and C. Vetter. Real-time routing with openstreetmap data. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 513–516, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-1031-4. doi: 10.1145/2093973.2094062. URL `http://doi.acm.org/10.1145/2093973.2094062`.

[32] J. J. Mies, J. R. Helmus, and R. Van den Hoed. Estimating the charging profile of individual charge sessions of electric vehicles in the netherlands. *World Electric Vehicle Journal*, 9(2):17, 2018.

[33] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. A multi-space sampling

heuristic for the green vehicle routing problem. *Transportation Research Part C: Emerging Technologies*, 70:113–128, 2016.

[34] A. Montoya, C. Guéret, J. E. Mendoza, and J. G. Villegas. The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103:87–110, 2017. ISSN 0191-2615. doi: https://doi.org/10.1016/j.trb.2017.02.004. URL `https://www.sciencedirect.com/science/article/pii/S0191261516304556`. Green Urban Transportation.

[35] S. Pelletier, O. Jabali, G. Laporte, and M. Veneroni. Battery degradation and behaviour for electric vehicles: Review and numerical analyses of several models. *Transportation Research Part B: Methodological*, 103:158–187, 2017.

[36] P. Rajan, M. Baum, M. Wegner, T. Zündorf, C. J. West, D. Schieferdecker, and D. Delling. Robustness generalizations of the shortest feasible path problem for electric vehicles. In *21st Symposium on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems (ATMOS 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[37] M. Schiffer, S. Stütz, and G. Walther. Electric commercial vehicles in mid-haul logistics networks. In *Behaviour of Lithium-Ion Batteries in Electric Vehicles*, pages 153–173. Springer, 2018.

[38] M. Schneider, A. Stenger, and D. Goeke. The electric vehicle-routing problem with time windows and recharging stations. *Transportation science*, 48(4):500–520, 2014.

[39] S. Schoenberg and F. Dressler. Reducing waiting times at charging stations with adaptive electric vehicle route planning. *IEEE Transactions on Intelligent Vehicles*, 2022.

[40] The Climate Group. Progress and Insights Report: The road to 2030. `https://www.theclimategroup.org/sites/default/files/2022-03/EV100%20Progress%20and%20Insights%20Report%202022_0.pdf`, 2022. Accessed: 5-04-2022.

[41] Transport & Environment. Recharge EU: how many charger points will Europe and its Member States need in the 2020s. `https://www.transportenvironment.org/wp-content/uploads/2021/07/01%202020%20Draft%20TE%20Infrastructure%20Report%20Final.pdf`, 2020. Accessed: 26-03-2022.

[42] Transport & Environment. The great acceleration. `https://`

`www.transportenvironment.org/annual-report-2020/documents/TE-Annual-Report-2020.pdf`, 2021. Accessed: 26-03-2022.

[43] M. Uhrig, L. Weiß, M. Suriyah, and T. Leibfried. E-mobility in car parks–guidelines for charging infrastructure expansion planning and operation based on stochastic simulations. In *EVS28 International Electric Vehicle Symposium and Exhibition*, pages 1–12, 2015.

[44] T. Zündorf. Electric vehicle routing with realistic recharging models. *Unpublished Master's thesis, Karlsruhe Institute of Technology, Karlsruhe, Germany*, 2014.

Numerical Results

Table N.1: **SPM** Model Results: Shortest Time using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Best Bound [h]	Relative Gap [%]	Comp. Time [s]	Variables	Constraints
Ge_Zu_1	8.629	3.735	8.629	8.629	0.000	0.092	945	1 193
Ge_Zu_2	8.966	4.060	8.966	8.966	0.000	0.057	1 002	1 440
Li_Re_1	14.130	6.942	14.130	14.130	0.004	6.994	1 298	2 099
Li_Re_2	23.594	8.301	23.594	23.593	0.007	4.267	1 246	1 893
Li_Re_3	15.749	18.322	15.749	15.747	0.009	9.827	1 554	2 674
Li_Re_4	24.574	11.740	24.574	24.572	0.007	3.413	1 496	2 447
Li_Re_5	30.558	14.250	30.558	30.556	0.009	7.139	1 575	2 745
Li_Re_6	28.558	13.199	28.558	28.556	0.008	8.899	1 575	2 745
Li_Re_7	26.558	11.320	26.558	26.557	0.004	7.917	1 575	2 745
Li_Re_8	25.313	13.924	25.313	25.312	0.002	5.423	1 575	2 745
Li_Re_9	25.088	18.077	25.088	25.085	0.010	10.057	1 575	2 745
Li_Re_10	25.709	7.797	25.709	25.708	0.003	6.559	1 575	2 745
St_An_1	15.767	13.395	15.767	15.765	0.009	24.409	2 545	4 002
St_An_2	18.182	15.952	18.182	18.180	0.008	7.870	3 440	6 075
St_An_3	16.861	9.593	16.861	16.860	0.009	6.661	2 458	3 794
St_An_4	28.674	10.134	28.674	28.674	0.000	6.572	3 496	6 284
St_An_5	26.341	13.450	26.341	26.339	0.010	3.182	2 570	4 164
St_An_6	18.971	9.396	18.971	18.971	0.004	0.917	3 266	5 391
St_An_7	29.037	11.955	29.037	29.035	0.006	1.029	3 322	5 717
St_An_8	29.813	11.955	29.813	29.813	0.000	0.488	3 398	6 043

Table N.2: **MPM** Model Results: Maximum Profit using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Best Bound [h]	Relative Gap [%]	Comp. Time [s]	Variables	Constraints
Ge_Zu_1	9.488	7.679	7.679	7.679	0.000	0.036	945	1 194
Ge_Zu_2	9.195	4.576	4.576	4.576	0.000	0.039	1 002	1 441
Li_Re_1	15.630	19.201	19.201	19.201	0.000	0.488	1 298	2 100
Li_Re_2	25.094	16.576	16.576	16.576	0.000	1.332	1 246	1 894
Li_Re_3	17.249	23.183	23.183	23.183	0.000	1.165	1 554	2 675
Li_Re_4	26.074	22.624	22.624	22.624	0.000	1.300	1 496	2 448
Li_Re_5	32.058	22.807	22.807	22.807	0.000	0.793	1 575	2 746
Li_Re_6	30.058	20.929	20.929	20.929	0.000	0.598	1 575	2 746
Li_Re_7	28.058	21.598	21.598	21.598	0.000	0.667	1 575	2 746
Li_Re_8	26.813	22.784	22.784	22.784	0.000	1.414	1 575	2 746
Li_Re_9	26.588	23.876	23.876	23.876	0.000	1.537	1 575	2 746
Li_Re_10	27.209	17.373	17.373	17.373	0.000	0.652	1 575	2 746
St_An_1	17.267	17.116	17.116	17.116	0.000	1.361	2 545	4 003
St_An_2	19.682	18.515	18.515	18.515	0.000	3.070	3 440	6 076
St_An_3	17.500	15.195	15.195	15.195	0.000	1.377	2 458	3 795
St_An_4	30.174	13.690	13.690	13.690	0.001	2.636	3 496	6 285
St_An_5	27.500	17.452	17.452	17.452	0.000	2.014	2 570	4 165
St_An_6	20.000	18.156	18.156	18.156	0.000	0.501	3 266	5 392
St_An_7	30.537	19.340	19.340	19.340	0.000	0.325	3 322	5 718
St_An_8	31.313	19.340	19.340	19.340	0.000	0.279	3 398	6 044

Table N.3: **MDPM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Best Bound [h]	Relative Gap [%]	Comp. Time [s]	Variables	Constraints
Ge_Zu_1	9.005	7.679	0.604	0.604	0.000	0.063	985	1 314
Ge_Zu_2	8.987	4.576	2.887	2.887	0.000	0.080	1 042	1 561
Li_Re_1	15.524	19.201	−0.074	−0.074	0.000	2.446	1 449	2 553
Li_Re_2	24.659	16.576	10.596	10.596	0.000	2.407	1 397	2 347
Li_Re_3	16.061	22.558	−2.113	−2.113	0.002	4.030	1 792	3 389
Li_Re_4	25.522	22.624	7.299	7.299	0.000	6.356	1 734	3 162
Li_Re_5	30.943	22.807	12.237	12.237	0.004	4.456	1 813	3 460
Li_Re_6	29.519	20.185	12.780	12.780	0.000	4.134	1 813	3 460
Li_Re_7	26.666	21.307	9.431	9.431	0.000	5.664	1 813	3 460
Li_Re_8	26.288	22.502	8.157	8.156	0.006	9.322	1 813	3 460
Li_Re_9	26.045	23.854	6.900	6.900	0.000	4.422	1 813	3 460
Li_Re_10	26.653	17.373	12.022	12.021	0.005	7.265	1 813	3 460
St_An_1	16.865	17.047	1.639	1.639	0.000	6.585	2 810	4 798
St_An_2	18.628	18.515	2.240	2.240	0.000	3.260	4 058	7 930
St_An_3	17.460	15.195	3.533	3.533	0.000	2.995	2 723	4 590
St_An_4	29.667	13.690	16.869	16.867	0.008	5.603	4 114	8 139
St_An_5	27.242	17.452	11.672	11.671	0.009	3.275	2 835	4 960
St_An_6	19.333	18.030	3.330	3.330	0.000	1.031	3 884	7 246
St_An_7	29.153	19.215	12.261	12.261	0.000	0.664	3 940	7 572
St_An_8	29.870	19.215	12.978	12.978	0.000	0.566	4 016	7 898

Table N.4: **MDPM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Best Bound [h]	Relative Gap [%]	Comp. Time [s]	Variables	Constraints
Ge_Zu_1	9.005	7.679	−1.315	−1.315	0.000	0.066	985	1 314
Ge_Zu_2	8.987	4.576	1.743	1.743	0.000	0.090	1 042	1 561
Li_Re_1	15.524	19.201	−4.875	−4.875	0.000	1.378	1 449	2 553
Li_Re_2	24.659	16.576	6.452	6.452	0.007	2.576	1 397	2 347
Li_Re_3	16.629	23.183	−7.809	−7.810	0.007	2.923	1 792	3 389
Li_Re_4	25.522	22.624	1.643	1.643	0.002	5.210	1 734	3 162
Li_Re_5	30.943	22.807	6.535	6.535	0.000	3.702	1 813	3 460
Li_Re_6	29.715	20.929	7.588	7.588	0.000	4.717	1 813	3 460
Li_Re_7	26.666	21.307	4.104	4.104	0.009	3.114	1 813	3 460
Li_Re_8	26.288	22.502	2.531	2.531	0.000	4.239	1 813	3 460
Li_Re_9	26.045	23.854	0.936	0.936	0.000	3.107	1 813	3 460
Li_Re_10	26.653	17.373	7.679	7.679	0.000	3.820	1 813	3 460
St_An_1	16.865	17.047	−2.622	−2.622	0.007	3.861	2 810	4 798
St_An_2	18.628	18.515	−2.389	−2.389	0.006	2.975	4 058	7 930
St_An_3	17.460	15.195	−0.266	−0.266	0.000	3.061	2 723	4 590
St_An_4	29.667	13.690	13.445	13.444	0.007	5.996	4 114	8 139
St_An_5	27.242	17.452	7.309	7.309	0.008	4.317	2 835	4 960
St_An_6	19.333	18.030	−1.178	−1.178	0.000	0.987	3 884	7 246
St_An_7	29.153	19.215	7.457	7.457	0.000	0.609	3 940	7 572
St_An_8	29.870	19.215	8.174	8.174	0.000	0.462	4 016	7 898

Table N.5: **MDPM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Best Bound [h]	Relative Gap [%]	Comp. Time [s]	Variables	Constraints
Ge_Zu_1	9.005	7.679	−8.994	−8.994	0.000	0.062	985	1 314
Ge_Zu_2	8.987	4.576	−2.834	−2.834	0.000	0.106	1 042	1 561
Li_Re_1	15.524	19.201	−24.076	−24.076	0.003	0.711	1 449	2 553
Li_Re_2	24.659	16.576	−10.123	−10.123	0.000	2.105	1 397	2 347
Li_Re_3	16.629	23.183	−30.993	−30.993	0.002	1.604	1 792	3 389
Li_Re_4	25.522	22.624	−20.981	−20.981	0.000	2.913	1 734	3 162
Li_Re_5	30.943	22.807	−16.271	−16.271	0.000	2.700	1 813	3 460
Li_Re_6	29.715	20.929	−13.340	−13.340	0.000	3.609	1 813	3 460
Li_Re_7	27.183	21.598	−17.210	−17.211	0.004	2.473	1 813	3 460
Li_Re_8	26.288	22.502	−19.971	−19.971	0.000	3.510	1 813	3 460
Li_Re_9	26.045	23.854	−22.918	−22.919	0.004	3.086	1 813	3 460
Li_Re_10	26.653	17.373	−9.694	−9.695	0.006	3.202	1 813	3 460
St_An_1	16.865	17.047	−19.669	−19.670	0.005	1.788	2 810	4 798
St_An_2	18.628	18.515	−20.905	−20.907	0.009	2.774	4 058	7 930
St_An_3	17.460	15.195	−15.460	−15.460	0.000	2.925	2 723	4 590
St_An_4	29.667	13.690	−0.245	−0.245	0.000	3.350	4 114	8 139
St_An_5	27.242	17.452	−10.143	−10.143	0.006	3.448	2 835	4 960
St_An_6	19.333	18.030	−19.208	−19.209	0.004	0.830	3 884	7 246
St_An_7	29.153	19.215	−11.757	−11.757	0.000	0.514	3 940	7 572
St_An_8	29.870	19.215	−11.040	−11.040	0.000	0.459	4 016	7 898

Table N.6: **AsM** Model Results: Shortest Time using Γ_1

ID	Trip Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	8.629	3.735	8.629	21	50	0.002
Ge_Zu_2	8.966	4.060	8.966	33	66	0.005
Li_Re_1	14.545	6.942	14.545	33	341	0.017
Li_Re_2	24.627	5.365	24.627	583	4 721	0.138
Li_Re_3	16.047	15.375	16.047	426	3 934	0.125
Li_Re_4	25.530	11.740	25.530	599	4 350	0.141
Li_Re_5	31.412	13.820	31.412	2 934	7 861	0.283
Li_Re_6	30.246	16.904	30.246	1 677	3 599	0.158
Li_Re_7	27.412	11.320	27.412	1 707	10 073	0.387
Li_Re_8	26.271	12.339	26.271	1 453	8 362	0.329
Li_Re_9	25.943	13.976	25.943	1 577	10 753	0.369
Li_Re_10	26.749	11.664	26.749	1 373	7 389	0.308
St_An_1	16.306	6.884	16.306	93	1 669	0.056
St_An_2	18.778	15.952	18.778	130	1 816	0.057
St_An_3	17.360	12.616	17.360	663	8 073	0.225
St_An_4	28.813	10.134	28.813	309	1 725	0.095
St_An_5	26.387	13.450	26.387	158	2 085	0.082
St_An_6	19.689	13.055	19.689	757	1 175	0.124
St_An_7	29.258	13.055	29.258	822	5 705	0.199
St_An_8	30.258	13.055	30.258	156	692	0.044

Table N.7: **AsDM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_1

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	9.939	9.005	7.679	−5.820	11	48	0.001
Ge_Zu_2	8.987	8.987	4.576	−4.445	37	65	0.003
Li_Re_1	15.943	15.655	12.776	−3.639	61	138	0.008
Li_Re_2	25.325	24.824	15.877	3.417	12	90	0.002
Li_Re_3	16.783	16.061	22.558	−10.136	182	342	0.011
Li_Re_4	26.631	25.503	18.647	2.646	179	304	0.013
Li_Re_5	32.274	30.943	22.807	11.169	1 447	1 698	0.059
Li_Re_6	31.156	29.138	18.112	11.572	993	1 134	0.051
Li_Re_7	27.575	26.666	21.307	3.594	35	156	0.005
Li_Re_8	26.593	25.445	17.712	3.309	12	85	0.003
Li_Re_9	26.167	25.145	21.108	0.336	10	93	0.003
Li_Re_10	26.874	26.617	14.720	5.834	25	142	0.004
St_An_1	17.492	17.232	15.161	−3.879	13	106	0.003
St_An_2	19.846	19.317	17.783	−3.492	58	221	0.006
St_An_3	17.441	17.266	11.379	−11.093	823	1 083	0.040
St_An_4	29.967	29.667	13.690	9.699	56	161	0.006
St_An_5	27.403	27.349	14.553	6.488	1 519	1 867	0.095
St_An_6	19.974	19.334	18.030	0.451	36	173	0.007
St_An_7	29.596	29.153	19.215	5.185	16	140	0.004
St_An_8	30.376	29.870	15.631	5.653	34	106	0.005

Table N.8: **AsDM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_1

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	9.939	9.005	7.679	−7.740	11	48	0.002
Ge_Zu_2	8.987	8.987	4.576	−5.589	39	65	0.006
Li_Re_1	15.943	15.655	12.776	−6.833	73	142	0.012
Li_Re_2	25.634	24.392	13.780	1.854	11	77	0.005
Li_Re_3	16.783	16.061	22.558	−15.775	254	396	0.034
Li_Re_4	26.631	25.503	18.647	−2.016	454	576	0.058
Li_Re_5	32.274	30.943	22.807	5.467	1 339	1 539	0.080
Li_Re_6	31.156	29.138	18.112	7.044	885	998	0.065
Li_Re_7	27.575	26.666	21.307	−1.732	44	166	0.013
Li_Re_8	26.593	25.445	17.712	−1.119	14	85	0.008
Li_Re_9	26.167	25.145	21.108	−4.941	11	94	0.007
Li_Re_10	26.874	26.617	14.720	2.155	39	154	0.011
St_An_1	17.492	17.232	15.161	−7.669	13	106	0.007
St_An_2	19.818	19.388	17.067	−7.249	75	196	0.012
St_An_3	17.441	17.266	11.379	−13.938	962	1 095	0.072
St_An_4	29.967	29.667	13.690	6.277	66	163	0.013
St_An_5	27.403	27.349	14.553	2.850	1 873	2 127	0.138
St_An_6	19.974	19.334	18.030	−4.056	48	172	0.017
St_An_7	29.596	29.153	19.215	0.382	19	140	0.009
St_An_8	30.376	29.870	15.631	1.745	32	106	0.006

Table N.9: **AsDM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_1

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	9.939	9.005	7.679	−15.418	11	48	0.003
Ge_Zu_2	8.987	8.987	4.576	−10.166	39	65	0.006
Li_Re_1	15.943	15.655	12.776	−19.610	117	176	0.017
Li_Re_2	25.634	24.392	13.780	−11.926	10	75	0.005
Li_Re_3	16.783	16.061	22.558	−38.333	465	571	0.054
Li_Re_4	26.631	25.503	18.647	−20.663	811	877	0.081
Li_Re_5	32.274	30.943	22.807	−17.340	567	684	0.046
Li_Re_6	31.116	28.578	15.081	−5.046	505	607	0.044
Li_Re_7	27.575	26.666	21.307	−23.039	47	166	0.006
Li_Re_8	26.593	25.445	17.712	−18.830	14	85	0.008
Li_Re_9	27.045	26.180	21.416	−25.787	12	85	0.007
Li_Re_10	26.874	26.617	14.720	−12.565	71	166	0.013
St_An_1	17.451	17.389	15.201	−22.950	22	118	0.007
St_An_2	19.932	19.457	16.077	−22.222	95	210	0.017
St_An_3	17.441	17.266	11.379	−25.317	1 130	1 199	0.081
St_An_4	30.017	29.787	13.331	−6.644	83	144	0.013
St_An_5	27.403	27.349	14.553	−11.703	2 299	2 404	0.157
St_An_6	19.974	19.334	18.030	−22.086	64	172	0.019
St_An_7	29.596	29.153	19.215	−18.833	21	140	0.009
St_An_8	30.376	29.870	15.631	−13.886	33	106	0.009

Table N.10: **AsM** Model Results: Shortest Time using Γ_2

ID	Trip Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	8.603	8.478	8.603	1 096	3 820	0.064
Ge_Zu_2	8.818	7.099	8.818	1 109	2 213	0.064
Li_Re_1	13.477	6.155	13.477	111	23 241	0.742
Li_Re_2	23.131	11.685	23.131	12 891	2 682 796	175.537
Li_Re_3	15.373	4.707	15.373	15 283	683 354	36.531
Li_Re_4	24.808	8.562	24.808	16 654	1 636 135	77.321
Li_Re_5	30.988	10.986	30.988	589 411	2 844 617	196.607
Li_Re_6	28.599	14.870	28.599	294 053	5 482 346	463.745
Li_Re_7	26.599	16.491	26.599	39 320	3 323 800	213.243
Li_Re_8	25.009	8.785	25.009	20 025	2 277 509	132.141
Li_Re_9	24.558	12.870	24.558	15 113	1 460 876	78.603
Li_Re_10	25.520	7.644	25.520	22 954	2 244 378	128.441
St_An_1	15.099	8.054	15.099	68	11 389	0.689
St_An_2	17.527	5.556	17.527	1 750	46 400	2.837
St_An_3	15.806	9.723	15.806	16 528	2 557 258	99.768
St_An_4	27.322	18.624	27.322	20 791	1 561 876	90.617
St_An_5	26.156	12.800	26.156	11 508	1 432 008	92.701
St_An_6	17.771	4.098	17.771	10 134	880 119	30.212
St_An_7	27.995	6.880	27.995	28 039	2 871 256	148.879
St_An_8	29.683	11.381	29.683	599 612	2 999 655	185.645

Table N.11: **AsDM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_2

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	8.623	8.623	9.747	−8.687	15	408	0.008
Ge_Zu_2	8.910	8.910	8.546	−7.499	442	893	0.030
Li_Re_1	14.920	14.809	22.784	−12.167	5 445	7 840	0.380
Li_Re_2	24.510	27.462	20.985	−1.229	69 611	81 060	8.964
Li_Re_3	16.792	16.217	23.165	−10.582	7 139	9 795	0.504
Li_Re_4	26.291	25.806	27.341	−4.214	2 325	3 807	0.231
Li_Re_5	31.955	31.408	26.214	8.294	229 519	264 465	36.612
Li_Re_6	30.050	30.201	20.022	9.033	112 552	129 444	24.277
Li_Re_7	27.972	27.003	27.048	−0.314	2 762	3 954	0.495
Li_Re_8	26.387	25.876	18.701	2.362	1 100	2 619	0.132
Li_Re_9	25.460	25.373	19.443	0.878	14 470	19 629	0.914
Li_Re_10	26.802	26.645	23.671	−0.951	8	840	0.074
St_An_1	16.569	16.425	19.167	−7.807	1 012 641	1 099 271	99.609
St_An_2	18.950	18.479	21.176	−6.932	1 489	3 519	0.200
St_An_3	16.978	16.726	18.964	−17.245	14 443	16 652	12.145
St_An_4	28.726	28.486	22.079	2.167	2 861	5 086	0.311
St_An_5	27.492	26.887	18.772	3.413	23 821	26 485	2.726
St_An_6	18.980	17.862	14.297	2.257	50 440	56 875	28.898
St_An_7	29.467	36.000	12.919	9.778	189 154	194 746	32.558
St_An_8	30.183	29.975	18.563	3.260	2 661	4 290	0.273

Table N.12: **AsDM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_2

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	8.623	8.623	9.747	−11.124	13	408	0.012
Ge_Zu_2	8.910	8.910	8.546	−9.636	429	893	0.031
Li_Re_1	14.920	14.809	22.784	−17.863	25	878	0.077
Li_Re_2	24.510	27.462	20.985	−6.475	98 888	106 735	14.044
Li_Re_3	16.792	16.217	23.165	−16.374	8 435	10 624	0.547
Li_Re_4	26.291	25.806	27.341	−11.049	3 464	4 843	0.267
Li_Re_5	31.955	31.408	26.214	1.740	61 726	68 391	6.758
Li_Re_6	30.050	30.201	20.022	4.027	91 256	99 314	22.675
Li_Re_7	27.641	27.506	27.394	−7.753	2 302	3 491	0.405
Li_Re_8	26.386	26.377	23.862	−7.476	3 904	5 656	0.283
Li_Re_9	25.460	25.373	19.443	−3.983	7 229	9 700	0.485
Li_Re_10	26.802	26.645	23.671	−6.869	8	840	0.072
St_An_1	16.569	16.425	19.167	−12.599	1 204 009	1 245 442	134.355
St_An_2	18.950	18.479	21.176	−12.226	1 800	3 577	0.503
St_An_3	16.978	16.726	18.964	−21.985	21 480	23 423	15.034
St_An_4	28.667	28.486	26.095	−7.428	555 259	594 772	62.193
St_An_5	27.406	27.052	15.562	1.844	17 519	18 690	2.738
St_An_6	18.980	17.862	14.297	−1.317	57 544	61 121	34.371
St_An_7	29.485	36.000	12.939	6.546	198 199	199 751	37.872
St_An_8	30.183	29.975	18.563	−1.380	2 186	3 501	0.218

Table N.13: **AsDM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_2

ID	Trip Time [h]	Refined Time [h]	Total Score	Objective [h]	Opened Labels	Created Labels	Comp. Time [s]
Ge_Zu_1	8.623	8.623	9.747	−20.872	11	408	0.011
Ge_Zu_2	9.530	9.530	9.078	−18.625	362	974	0.032
Li_Re_1	14.920	14.809	22.784	−40.647	556 267	562 594	71.918
Li_Re_2	24.510	27.462	20.985	−27.461	186 604	189 761	25.716
Li_Re_3	16.652	16.360	23.267	−39.882	12 180	13 591	0.715
Li_Re_4	26.259	27.694	19.348	−22.437	6 808	7 726	0.437
Li_Re_5	31.955	31.408	26.214	−24.474	43 105	45 422	4.423
Li_Re_6	30.050	30.201	20.022	−15.995	119 386	121 901	25.821
Li_Re_7	27.641	27.506	27.394	−35.146	1 846	2 915	0.330
Li_Re_8	26.420	26.072	24.508	−32.596	14	1 029	0.076
Li_Re_9	26.019	25.940	23.638	−31.257	4 854	6 136	0.328
Li_Re_10	26.919	26.693	17.543	−18.166	6 781	7 825	0.374
St_An_1	16.569	16.425	19.167	−31.766	1 872 349	1 882 024	540.437
St_An_2	18.989	18.759	16.992	−24.995	2 100	3 262	0.212
St_An_3	16.978	16.726	18.964	−40.949	45 383	46 559	40.176
St_An_4	28.667	28.486	26.095	−33.522	855 678	862 122	95.962
St_An_5	27.406	27.052	15.562	−13.719	20 552	21 313	4.666
St_An_6	19.245	17.707	10.530	−7.816	55 613	56 454	36.549
St_An_7	29.080	28.809	19.030	−18.980	195 287	196 346	38.373
St_An_8	30.953	30.861	18.847	−19.742	103	717	0.088

List of Symbols

Description of the Sets

Sets	
Set	**Description**
$\mathcal{S}$	set of charging stations
$\mathcal{S}_{\mathcal{O}}$	$\mathcal{S} \cup \{\mathcal{O}\}$
$\mathcal{S}_{\mathcal{D}}$	$\mathcal{S} \cup \{\mathcal{D}\}$
$\mathcal{S}_{\mathcal{O},\mathcal{D}}$	$\mathcal{S} \cup \{\mathcal{O}, \mathcal{D}\}$
$\mathcal{S}_k$	set of chargers that can be used during time window $k \in \mathcal{W}$; is a subset of $\mathcal{S}$
$\widetilde{\mathcal{S}}$	$\cup_{k\in\mathcal{W}} \mathcal{S}_k$
$\mathcal{A}$	set of arcs (i,j), with $i \in \mathcal{S}_{\mathcal{O}}$ and $j \in \mathcal{S}_{\mathcal{D}}$
$\mathcal{B}_i$	set of breakpoints of the charging function of CS i
$\mathcal{W}$	set of time slots
$\mathcal{W}^R$, $\mathcal{W}^O$	set of required and optional time slots; form a partition of $\mathcal{W}$
$\mathcal{W}^{\mathcal{NA}}$, $\mathcal{W}^{\mathcal{A}}$	set of non avoidable and avoidable time windows; form a partition of $\mathcal{W}$
$\mathcal{W}_i$	set of time windows that can be used for CS i; is a subset of $\mathcal{W}$

Table S.1: Description of the Sets

Description of the Routing Parameters

Routing Parameters	
Parameter	**Description**
$\mathcal{O}$	origin point
$\mathcal{D}$	destination point
Q	maximum battery capacity of the EV (in kWh)
q_{min}	minimum amount of energy always required for the EV (in kWh)
η	average consumption rate (in kWh/km)
P	maximum power charge of the EV (in kW)
t_{start}	relative starting time with respect to the midnight of day 0
t_{end}	relative ending time with respect to the midnight of day 0
$T^{\max}$	maximum total duration of the trip
t_{ij}	driving time from CS i to CS j
e_{ij}	energy consumption to go from CS i to CS j
(c_{ik}, a_{ik})	breakpoint $k \in \mathcal{B}_i$ of the charging function related to the CS i
γ_k^L , γ_k^U	starting and ending time of time window k
$t_k^{\min}$	minimum stopping time that the EV must respect during time window $k \in \mathcal{W}$
o_k	binary value: 1 if $k\mathcal{W}$ is an optional time windows, 0 otherwise
ν_k	label that identifies which type of POI is needed during $k \in \mathcal{W}$
σ_i	score of CS i
$\widetilde{\varphi}$	maximum anticipation time for time windows. It is set to 45 min
$r^{\min}$, $r^{\max}$	minimum and maximum distance reachable for the EV; used to prune the graph
ξ	worse optimal distance retrieved by OSRM's server
N	maximum number of legs
μ	coefficient that indicates how much importance is given to the score

Table S.2: Description of the Routing Parameters

Description of the Variables

Variables	
Variable	**Description**
x_{ij}	binary value: 1 if arc (i, j) is used, 0 otherwise
y_{jk}	binary value: 1 if the EV is obliged to stops in j in time slot k, 0 otherwise
z_k	binary value: 1 if the EV arrives in $\mathcal{D}$ after time window $k \in \mathcal{W}^{\mathcal{A}}$, 0 otherwise
$\underline{q}_i$, $\overline{q}_i$	SoC of the EV when respectively arrive and leaves CS i
$\underline{c}_i$, $\overline{c}_i$	in the charging function, corresponding time of a particular SoC of the EV when respectively arrive and leaves CS i
$\underline{w}_i$, $\overline{w}_i$	binary value: 1 if the SoC is in the interval $[a_{i,k-1}, a_{ik}]$ when the EV respectively arrive and leaves CS i during time frame k, 0 otherwise
$\underline{\lambda}_i$, $\overline{\lambda}_i$	coefficients associated with the breakpoint (c_{ik}, a_{ik}) in the linear approximation of the charging function when the EV respectively arrive and leaves CS i during time frame k
Δ_i	charging time at CS i
$\underline{\tau}_i$, $\overline{\tau}_i$	relative time when the EV respectively arrive and leaves CS i
φ_i	amount of time for which the EV can stop in advance with respect to a given time slot
s_{ij}	real variable used to linearize the objective function of **MDPM**

Table S.3: Description of the Variables

Description of the Model Design Parameters

Model Design Parameters	
Parameter	**Description**
r_M	merging radius for CSs. It is set to 100 m
r_T	initial radius for searching tourism CSs. It is set to 2.0 km
δ_T	step increasing value for r_T. It is set to 200 m
$\widetilde{r}_T$	maximum value of r_T. It is set to 4.0 km

Table S.4: Description of the Model Design Parameters

List of Figures

2.1 Example of a piecewise linear approximation for a CS $i \in \mathcal{S}$ with a power of 22 kWh adapted from [34]. The fictitious point is added to this charging function with the point $(c_{i,3}, a_{i,3})$, creating a new slope between $(c_{i,2}, a_{i,2})$ and $(c_{i,4}, a_{i,4})$. 9
2.2 Possible relative positions of time windows 11
2.3 Example of a path from $\mathcal{O}$ to $\mathcal{D}$ with a night in hotel and lunch breaks. The red path is the optimal one, stopping in the hotel as required and reaching node E. If also the lunch break is required, then the EV is forced to arrive to $\mathcal{D}$ with stopping at L (blue path). Instead, with the optional flag, from E the EV can go directly to $\mathcal{D}$ with three hours in advance (green path). The timestamps near the nodes represents the departure time from that node, including also the recharging time (not reported in this figure). The "+1" over the timestamps indicates that it refers to the next day. For $\mathcal{D}$, instead, the timestamps represent the arrival time. The number on each arc symbolize the travel time. 12

3.1 In red is depicted $\mathcal{T}_H$ in the first figure, while $\mathcal{T}_C$ in the second one. 22
3.2 Schematic example of the A* search algorithm presented in algorithm 3.2. . 34

4.1 Distribution of the given CSs. Map created with mapcustomizer.com ([26]) 38
4.2 On the left, polyline representing part of the Brennero's highway (fig. 4.2a). On the right, instead, the optimal path from Innsbruck to Bolzano (blue line) computed via OSRM. The pinpoints in red are the position of the CS selected from Γ (fig. 4.2b). 41
4.3 Example of the range preprocessing that is performed to reduce the size of $\mathcal{A}$. 42

5.1 Piecewise linear approximation, including the fictitious breakpoint, for different CSs technologies for the selcted EV. 46

List of Tables

4.1 Distribution of chargers for each country. . . . 39

5.1 Description of the instances 47
5.2 Number of nodes and arcs for each instance. 48
5.3 Comparison between **SPM** and **AsM** using Γ_1 49
5.4 Comparison of total score and computational time between **MPM**, **MDPM** and **AsDM** with $\mu = 2.0$ in Γ_1 50
5.5 Comparison of total trip time between **SPM**, **MDPM**, **AsDM** and the refined **SPM** with $\mu = 2.0$ in Γ_1 51
5.6 Comparison of total score and computational time between **AsM** and **AsDM** with $\mu = 0.75$, $\mu = 1.0$ and $\mu = 2.0$ in Γ_2 52
5.7 Comparison of total trip time between **AsM** and **AsDM** with $\mu = 0.75$, $\mu = 1.0$ and $\mu = 2.0$ in Γ_2 53

N.1 **SPM** Model Results: Shortest Time using Γ_1 63
N.2 **MPM** Model Results: Maximum Profit using Γ_1 64
N.3 **MDPM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_1 64
N.4 **MDPM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_1 65
N.5 **MDPM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_1 65
N.6 **AsM** Model Results: Shortest Time using Γ_1 66
N.7 **AsDM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_1 66
N.8 **AsDM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_1 67
N.9 **AsDM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_1 67
N.10 **AsM** Model Results: Shortest Time using Γ_2 68
N.11 **AsDM** Model Results with $\mu = 0.75$: Discounted Shortest Path using Γ_2 68
N.12 **AsDM** Model Results with $\mu = 1.00$: Discounted Shortest Time using Γ_2 69
N.13 **AsDM** Model Results with $\mu = 2.00$: Discounted Shortest Time using Γ_2 69
S.1 Description of the Sets 71
S.2 Description of the Routing Parameters 72
S.3 Description of the Variables 73
S.4 Description of the Model Design Parameters 73

Acknowledgements

I would like to express my gratitude to my advisor, Ola Jabali, who guided me throughout this project and made me appreciate the optimization research. Also thanks to Federico Malucelli, co-advisor of this thesis, for his constant support.
A deepest gratitude to Stephen, who is my love, that helped and supported me during all the realization of this thesis.
Thanks to my sister, Alessia, who offered invaluable support and humor over the years.
Most importantly, I am grateful for my parents whose constant love and support keep me motivated and confident. My accomplishments and success are because they believed in me.
Special thanks also to Sally, our family's electric vehicle, without which this thesis would not exist. Thank you for accompanying me on this journey.
Last but not the least I would also like to thank all of my friends and family members for encouraging and supporting me whenever I needed them.