Dynamic Programming for the Fixed Route Electric Vehicle Charging Problem with NonLinear Energy Management

Anthony Deschênes[1], Jonathan Gaudreault[1], Claude-Guy Quimper[1]

***Abstract*— The Fixed Route electric Vehicle Charging Problem with NonLinear Energy Management (FRVCP-NLEM) consists of finding the optimal charging and vehicle speed decisions given a fixed route for an electric vehicle. The objective is to minimize the total route duration including charging duration. The problem takes into consideration the non-linearity of the charging and energy consumption functions. We propose a new dynamic programming model to solve this problem that uses Akima interpolation to take into consideration the continuous state space over the state of charge. Experimental results show that the proposed model solves optimally all instances in less than 0.2 seconds, outperforming the current state of the art model by several orders of magnitude. The complexity of the proposed model has a pseudo-linear increase with the number of charging stations on the fixed route, meaning that it is easily scalable.**

I. Introduction

The number of electric vehicles (EV) has been significantly increasing in the recent years [1]. Governments see electric vehicles as a way to reduce their greenhouse emission [2] and increase subventions and tax reductions in order to facilitate their penetration [3]. As a result, popularity of electric vehicles is increasing in both urban and rural regions [4]. However, the adoption of electric vehicles leads to new challenges [5] such as range anxiety. It is related to the incertitude of the range of an EV under cold temperatures [6] and the fear of running out of energy. It is therefore a necessity to develop efficient solutions in order to facilitate EV usage. One difficulty that quickly arises for EV users is how to plan a long trip. The user needs to decide where to stop, for how long and driving speed along the road. Minimizing total time is complex since the time needed to charge an electric vehicle and its energy consumption functions are nonlinear [7]. Moreover, the number of charging stations is still limited [8] and a decision to charge or not at a charging station may result in running out of energy or unnecessary charging time if the wrong decision is made. This problem is called the Fixed Route electric Vehicle Charging Problem with NonLinear Energy Management (FRVCP-NLEM) [9].

We propose a new method based on dynamic programming (DP) to solve the FRVCP-NLEM. In Section II we present preliminary concepts. Section III presents the proposed dynamic programming model. Section IV presents the dataset and the experiments to compare the proposed model with the state of the art for this specific problem. Section IV-C presents the results of our experiments.

[1]CRISI Research Consortium for Industry 4.0 System Engineering, *Université Laval*, Québec, Canada `anthony.deschenes.1@ulaval.ca`, `jonathan.gaudreault@ift.ulaval.ca`, `claude-guy.quimper@ift.ulaval.ca`

II. The Fixed Route electric Vehicle Charging Problem with NonLinear Energy Management (FRVCP-NLEM)

The Fixed Route electric Vehicle Charging Problem with NonLinear Energy Management has first been introduced by [9]. It consists of finding the optimal charging and driving speed decisions in order to minimize total route duration given a fixed route for an electric vehicle. It takes into consideration speed limits and charging detour energy and time costs. It is a variant of the FRVCP introduced by Montoya *et al.* [10] that adds driving speed decisions on each segment of the fixed route.

FRVCP-NLEM takes into consideration the non-linearity of energy consumption functions as well as the non-linearity of the charging functions. Each charging station close to the fixed route implies to make a detour that has a cost in time and energy. These costs change depending on the distance of the charging station from the fixed route and the speeds driven on the segments. Each charging stations have a *unique* charging function that is more or less efficient depending on the type of the charging station (fast charger, etc.).

The FRVCP is a subproblem of a more general problem called the electric vehicle routing problem with nonlinear charging time (EVRP-NL) [11]. FRVCP is NP-Hard [10] and since the FRVCP-NLEM is a general case of FRVCP, FRVCP-NLEM is also NP-Hard.

Given a route with n charging stations, the objective is to go from an origin (r_0) to a destination (r_n). The route is thus split in $n+1$ segments. For each of these segment we need to decide if we want to charge at the charging station associated with the segment or not. Moreover, driving speed decisions are also needed. Figure 1 presents a given segment $s \in \{1, \cdots, n\}$ where c_s is a charging station. Charging stations are either on the fixed route or in the vicinity of the fixed route. r_s is a decision node for each charging station where a decision to charge or not at the next charging station must be taken. The pattern presented in Figure 1 is then repeated for each charging stations from the origin to the destination. We make the assumption that the time needed to travel a segment is deterministic and that the factors that affect the time required to travel the segment are known prior to the departure. It is a common assumption in the literature [7] [9] [10].

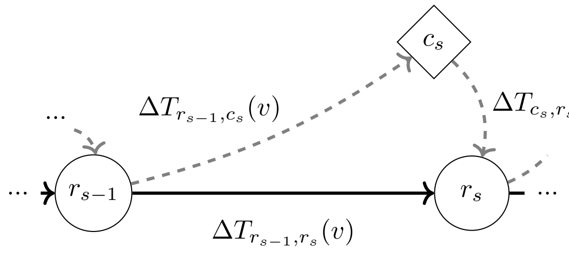

Fig. 1. Graph representation of a segment s of an instance of the FRVCP-NLEM [9]

Froger *et al.* [7] propose multiple methods to solve the FRVCP such as exact labelling algorithm and MIP models. Lee *et al.* [12] use branch and price to solve the EVRP-NL. These methods have not yet been extended to the FRVCP-NLEM. The use of dynamic programming to solve this problem is new in the literature. Velimirovic *et al.* [13] show that dynamic programming has a potential to solve the EVRP. Rama *et al.* [14] use dynamic programming in order to find energy optimal routes. Dynamic programming has also been used by Pourazarm *et al.* [15] to find time optimal route in electric vehicles. They do not, however, allow to vary the speed on the different segments.

The FRVCP-NLEM can be solved using a Mixed-Integer Programming model (MIP) proposed by [9]. It uses multiple linear approximations in order to approximate the charging and energy consumption functions. The number of linear functions used can be considered as hyperparameters. The MIP model is able to solve to optimality routes with less than 24 charging stations. The charging and energy consumption functions must be convex and monotonically increasing in order for the model to find the optimal solution. Some charging and energy consumption functions follow these prerequisites, but some functions such as empirical functions learned using, for example neural networks trained from empirical data, cannot ensure these properties [16].

We therefore propose a new model based on dynamic programming to solve the FRVCP-NLEM to optimality that does not require prerequisites for these functions. It also does not require to approximate these functions, but instead require discretization of the state of charge and vehicle speed. Having a pseudo-polynomial computational time, this approach is expected to compute solutions much faster than the MIP model.

III. Proposed Dynamic Programming Model

Charging stations are indexed from the origin to the destination. We define $\mathcal{S} = \{0, \ldots, n\}$ as the set of indexes for the segments of the fixed route. Let $s \in \mathcal{S}$ be the segment from r_{s-1} to r_s (see Figure 1). The vehicle can either pass by charging at station c_s, if a charge decision is taken, or go directly to r_s.

The problem has a number of parameters defined by the user. Note that we use the term state of charge to represent a percentage of the usable battery (from 0% to 100%):

- p_0 is the initial state of charge at origin.
- $p_s^{\text{min arrival}}$ is the minimal acceptable arrival state of charge at charging station c_s.
- $p_{c_s}^{\text{max charge}}$ is the maximal authorized final state of charge when charging at charging station c_s.
- v^{min} is the minimum driving speed as a ratio of the recommended driving speed. This ratio is used to express the minimum driving speed on a segment with respect to a recommended driving speed. As an example, a value of 0.9 represents driving at 90 km/h on a highway with a recommended speed of 100 km/h (the recommended speed is provided by the web service used to obtain the fixed route and takes into consideration speed limitations, projected traffic conditions, presence of intersections, merging roadways, etc.). Note that the ratio is fixed on a given segment, but the actual speed can vary. Thus once the ratio is chosen, one can compute the actual speed profile on a given segment with any desired level of granularity.
- v^{max} is the maximum driving speed as a ratio of the recommended driving speed. This ratio is used to express the maximum driving speed on a segment with respect to a recommended driving speed.

It also requires three functions to be defined:

- $D_{c_s}(p_1, p_2)$ = Charging function that returns the time to charge from a state of charge p_1 to p_2 at charging station c_s.
- $\Delta P_{a,b}(v)$ = Energy consumption function that returns the decrease of state of charge when driving from point a to point b with a given speed ratio v.
- $\Delta T_{a,b}(v)$ = Time required to travel a specific segment from point a to point b with a given speed ratio v.

A. Sampling of the state spaces

In order to use dynamic programming, we need to discretize both the state of charge and the driving speed. As such, the number of points to discretize the state of charge (k) and the driving speed (m) are two hyperparameters for the model.

For a given driving speed decision, we consider m distinct speed ratios $v \in \mathcal{V}$ ranging uniformly from v^{min} to v^{max}. In the same way, we discretize the state of charge using k points in $\mathcal{K} = \{1, \ldots, k\}$. We first compute $P_{a_s}^{min}$ and $P_{a_s}^{max}$ for each vertex where we need to consider the state of charge and then discretize uniformly between these two values. The objective is to determine what is the minimal and maximal possible arrival states of charge at a given node a_s. For example, if the node is r_s, intuitively the minimal arrival state of charge is the minimal departure state of charge of the previous point (r_{s-1} which is stored in $P^{min}_{r_{s-1}}$) minus the maximal consumption on the segment between r_{s-1} and r_s. We must also ensure that the arrival state of charge is always greater or equal to the minimal authorized state of charge. The same intuition applies for defining the maximum state of charge. The base cases of these equations are trivial. The minimum departure state of charge at the origin is the initial state of charge. The maximal departure state of charge is the maximum charge at the origin (or the initial state of charge if it is not possible to charge at the origin).

$$P^{min}_{a_s} = \begin{cases} p_0 & \text{if} \quad s = 0 \quad (1a) \\ \max(p^{\text{min arrival}}_s, & (1b) \\ P^{min}_{r_{s-1}} - \max_{v \in \mathcal{V}}(\Delta P_{r_{s-1},a_s}(v)) & \text{otherwise} \quad (1c) \end{cases}$$

$$P^{max}_{a_s} = \begin{cases} p^{\text{max charge}}_{c_s} & \text{if} \quad s = 0 \quad (2a) \\ \max(p^{\text{max charge}}_{c_s} - \Delta P_{c_s,r_s}(v), & (2b) \\ P^{max}_{r_{s-1}} - \min_{v \in \mathcal{V}}(\Delta P_{r_{s-1},a_s}(v)) & \text{otherwise} \quad (2c) \end{cases}$$

B. Solving the FRVCP-NLEM

To solve the FRVCP-NLEM, we recursively compute the values of a function $T_{r_s}(p)$ (see Equation (3)) for each point r_s from the origin to the destination. This function returns the answer to this crucial question: what is the *minimal* time needed to go from the *origin* r_0 to point r_s, while having a state of charge of p when *leaving* point r_s given that we know the answer to this question for the previous point r_{s-1} and for all possible states of charge for the previous point? It takes into account the fact that we may (or not) have made the detour and charged at charging station c_s.

The dynamic programming model is built around Equation (3). It defines how to compute the answer to this question for a specific state of charge p for a point r_s. s represents a segment index on the fixed route with the origin being $s = 0$. Answering this question for each point r_s and for each possible departure state of charge for these points allows to compute the optimal solution from the origin to the destination. For a specific state of charge (p) and a given point r_s, it is equivalent to sampling a function $T_{r_s}(p)$ which returns the answer. This function is sampled for each value of state of charge defined using $k_{r_s} \in \mathcal{K}$ (with $p = P^{min}_{r_s} + k_{r_s} \cdot \frac{P^{max}_{r_s} - P^{min}_{r_s}}{|\mathcal{K}|}$). It is the same as sampling k states of charge from the minimal possible arrival state of charge to the maximal possible arrival state of charge for point r_s.

Case (3a) represents the initial condition of the problem which sets the value of the function to 0 if we want to leave at the initial state of charge at the origin. Case (3b) represents the case where the desired departure state of charge (p) is higher than the initial state of charge. In that case the only possibility is to charge at the origin in order to leave the origin with a state of charge of p. Case (3c) detects cases where it is impossible to reach point r_s while respecting the minimal authorized arrival state of charge. This case detects infeasible routes. When we are not at the origin, there is only two possibilities to answer the question. The first is to reach point r_s directly without charging at the associated charging station. In that case we need to evaluate all possible driving speed decisions and choose the one with the minimal total time (Case (3d)). The other possibility is to charge at the charging station associated with point r_s. In that case we need to consider all possible discretized arrival state of charge (p_{c_s}) to the charging station *and* the different driving speed decisions. These states of charge must be smaller than the desired departure state of charge at the charging station. We finally choose the desired arrival state of charge that has the minimal total time (Case (3e)). Moreover, when charging at a charging station, we need to consider the detours costs in time and energy to reach the point r_s *after* charging at the associated charging station. The optimal solution is the minimum of these two possibilities which decides if we need to charge at the charging station or not. It is not always possible to charge at the destination, in that case we simply do not evaluate this possibility (which correspond to not computing Case (3e)). The same logic applies to prevent charge at the origin.

C. Interpolation over the state of charge

Since the energy consumption is a real number, it is possible that we want to obtain the value of function $T_{r_s}(p')$ for a p' that is between two sampled states of charge. To obtain an approximation of this value, we use Akima interpolation [17] between the points sampled by $k_{r_s} \in \mathcal{K}$. Akima interpolation smooths the second derivative and, since the function can be non-continuous, it allows for a better estimation of the function [18]. The points used for the Akima interpolation are the following: $(p, T_{r_s}(p))$ with $p = P^{min}_{r_s} + k_{r_s} \cdot \frac{P^{max}_{r_s} - P^{min}_{r_s}}{|\mathcal{K}|} \forall k_{r_s} \in \mathcal{K}$.

D. Constructing the optimal solution

The solution is constructed from the destination to the origin. The optimal solution is the one for which $\min_{k_{r_k} \in \mathcal{K}} T_{r_k}(p)$ with $p = P^{min}_{r_k} + k_{r_k} \cdot \frac{P^{max}_{r_k} - P^{min}_{r_k}}{|\mathcal{K}|}$ where r_k represents the destination. Let p' be the state of charge that minimizes the time T_{r_k}, we check which case in Equation (3) leads to the computation of $T_{r_k}(p')$. By recursively computing back the function, one can retrieve the decision to charge or not, the travelling speed ratio v, and the amount of charging p_{c_s} (if any) that were taken for each road segment.

Figure 2 presents a visual representation of the constructed solutions for different desired departure state of charge at a given point r_s that is not the destination

$$T_{r_s}(p) = \begin{cases} 0 & \text{if } p = p_0 \land s = 0 \quad (3a) \\ D_{c_s}(p_0, p) & \text{if } p > p_0 \land s = 0 \quad (3b) \\ \infty & \text{if } P_{s-1}^{max} - \min\limits_{v \in \mathcal{V}}(\Delta P_{r_{s-1},r_s}(v)) < p_s^{\text{min arrival}} \quad (3c) \\ \min\limits_{v \in \mathcal{V}} \Big(T_{r_{s-1}}(p + \Delta P_{r_{s-1},r_s}(v)) + \Delta T_{r_{s-1},r_s}(v), & (3d) \\ \quad \min\limits_{p_{c_s} | p_{c_s} < p + \Delta P_{c_s,r_s}(v)} T_{r_{s-1}}(p_{c_s} + \Delta P_{r_{s-1},c_s}(v)) + & (3e) \\ \quad\quad \Delta T_{r_{s-1},c_s}(v) + D_{c_s}(p_{c_s}, p + \Delta P_{c_s,r_s}(v)) + & \\ \quad\quad \Delta T_{c_s,r_s}(v) \Big) & \text{if } s > 0 \\ & \text{where } p_{c_s} = P_{c_s}^{min} + k_{c_s} \cdot \frac{P_{c_s}^{max} - P_{c_s}^{min}}{|\mathcal{K}|} \quad (3f) \end{cases}$$

(since we can charge at the charging station) for a given route. The continuous blue line over the points is the Akima interpolation associated with the point r_s. It is an approximation of the value of the function $T_{r_s}(p')$ for any state of charge p' that is not sampled. The bars are the constructed solution for each sampled departure state of charge. They are ordered by charging station in the following pattern (from bottom to top): origin (segment 0) route time, origin detour time, origin charge duration; segment 1 route time, segment 1 detour time, segment 1 charge duration, etc. Each route time bar has a color representing the average speed driven on the segment. In this example the optimal solution reduces the speed on some segment in order to arrive faster at destination depending on the desired arrival state of charge at point r_s. In terms of charging decisions to reach this point, Figure 2 can be reduced to essentially 4 different charging scenarios depending on the desired departure state of charge (charging at charging stations 3, 6 and 9; 5, 8 and 11; 3, 6, 9 and 12; 3, 6, 9 and 13).

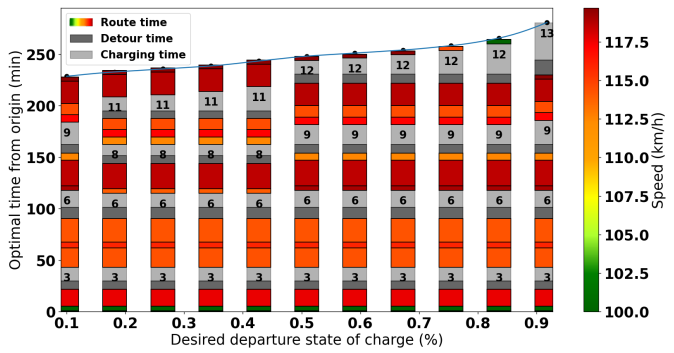

Fig. 2. Example of constructed solutions for different desired arrival state of charge at a given point r_s.

E. Complexity analysis

The proposed dynamic programming model is a pseudo-polynomial algorithm and thus proves that the FRVCP-NLEM is weakly NP-Hard.

We assume that the energy consumption function $\Delta P_{a,b}(v)$ and the charging functions $D_{c_s}(p_1, p_2)$ take constant time to compute. Given that $T_{r_{s-1}}(p)$ was already interpolated over p, computing a point on function $T_{r_s}(p)$ requires $\Theta(|\mathcal{V}||\mathcal{K}|)$ time. A sample of $|\mathcal{K}|$ points is required for each segment s. Computing Akima interpolation is done in linear time. Sampling and interpolating the function $T_{r_s}(p)$ for one segment thus takes $\Theta(|\mathcal{V}||\mathcal{K}|^2)$ time and for $|\mathcal{S}|$ segments it takes $\Theta(|\mathcal{S}||\mathcal{V}||\mathcal{K}|^2)$. This represents the total running time complexity as computing and interpolating the function $T_{r_s}(p)$ dominates the construction of the solution[2]. The model has a pseudo-linear increase with the number of segments, and thus with the number of charging stations.

IV. Experiments

Experiments are done using a similar framework as [9]. The main goal is to compare the MIP model from [9] with the dynamic programming model we propose. We compare the *real* route duration returned by each model, which is the total route duration after removal of the approximation errors induced by the hyperparameters. Both models ensure that the *real* arrival state of charge at charging stations is always greater or equal to the margin imposed by the user.

A. Dataset

All routes are generated for a Nissan LEAF with a battery capacity of 40 kWh. Our dataset contains 18 routes, each one duplicated to consider 7 different temperatures (in °C) $(-30, -20, -10, 0, 10, 20, 30)$. Table I presents a summary of the routes. Note that it is almost the same routes as from [9], but with a higher number of charging stations. It is caused by a large number of newly installed charging stations returned by the Open Charge Map API [19].

B. Hyperparameters optimization

Since both models have their own hyperparameters that impact the predicted and *real* total route duration, we need to find values for these hyperparameters that

[2]It is possible to improve this complexity to $\theta(\max(|\mathcal{S}||\mathcal{K}|^2, |\mathcal{S}||\mathcal{K}||\mathcal{V}|))$ if we consider that the optimal speed multiplier from points r_{s-1} to r_s is the same as for points r_{s-1} to c_s

TABLE I

Routes in the dataset

Route ID	Type	Number of CS	Distance (km)
1	Urban	9	51
2	Urban	18	92
3	Urban	26	122
4	Long route	18	261
5	Long route	33	439
6	Long route	10	149
7	Long route	42	642
8	Long route	52	879
9	Mix of long route and countryside	13	227
10	Countryside	12	333
11	Countryside	27	1007
12	Mix of countryside and highway	22	135
13	Highway	15	587
14	Highway	27	325
15	Highway	20	410
16	Extreme case (really long)	96	2047
17	Extreme case (loop)	46	546
18	Extreme case (round trip)	54	960

allow to obtain good quality solutions. For this purpose, we separated the dataset in a train and test set in order to do a grid search on the train set to find good values for hyperparameters (which affect discretization precision) for each model.

It has been shown that the number of charging station is a good indicator of the complexity of an instance [9]. The complexity analysis of the proposed algorithm (see Section III-E) enforces this conclusion. Thus, for the train set, we use the routes with the highest and lowest number of charging stations. By doing so, we make sure that the models have good hyperparameters for short and long-range routes. We also included the route closest to the average number of charging stations for all routes in the dataset. Therefore, our train set consists of routes 1, 11 and 16.

For the MIP from [9], there are three hyperparameters: (1) the number of linear approximations used to approximate the charging functions, (2) the number of linear approximations used to approximate the energy consumption functions and (3) the relative gap of the MIP model. The search space of 1. is defined over the interval $[2, 8]$ per increment of one. It has been shown that it is important to consider the non-linearity of the charging functions [7] [10] (thus removing 1 from the search space). The search space of 2. is defined over the interval $[1, 8]$ per increment of one. The search space of 3. is defined by three values (0.01, 0.03 and 0.05). The total number of combinations tested for the MIP model is 168.

For the proposed dynamic programming model, there are two hyperparameters: (1) the number of points used to sample the state of charge (k) and (2) the number of points used to sample the vehicle speed ratios (m). The search space of 1. is defined over the interval $[11, 101]$ per increment of 10. The search space of 2. is defined over the interval $[2, 11]$ per increment of one. The interval starts at 2 to ensure that we have at least the minimal and maximal speed ratios considered for each segment. The total number of combinations tested for the DP model is 100.

Increasing the values of the hyperparameters should lead to a better solution, but also increases the computation time. A good value for an hyperparameter is thus a compromise between computation time and *real* total route duration. We use a grid search in order to compute the performance of a set of combinations of the hyperparameters for each model in the train set. Each route of the train set is tested at 7 different temperatures. A grid search can cover the search space in a reasonable time [20] and there is no need for more complex methods [21]. We then compute a Pareto Front [22] to visualize all possible compromise between solution quality and computation time.

Figures 3 and 4 present the Pareto front for each of the models. The label of each point represents the value of each hyperparameters that yield to a given point (in the order presented above). For both models, multiple combinations yield acceptable total route duration. We use the minimal distance from the ideal point algorithm to determine which combination to use [23] for each model.

For the MIP model, the best hyperparameters selected are 3 for the number of linear approximations of the charging functions, 4 for the number of linear approximations for the energy consumption functions and a relative gap of 1% (0.01).

For the DP model, the best parameters found by the grid search are 11 for the number of points used to sample the state of charge and 11 for the number of points used to sample the vehicle speed ratios for each segment.

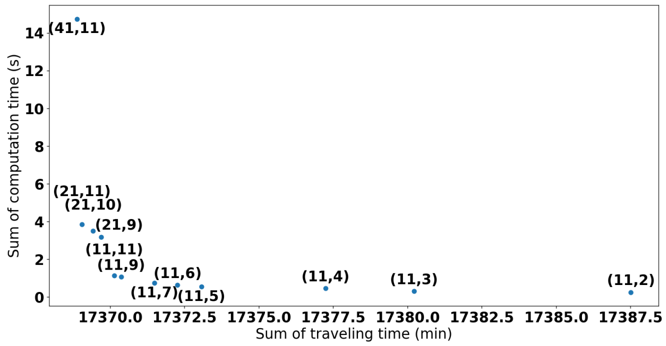

Fig. 3. Pareto Front of the dynamic programming model for all tested combinations of hyperparameters.

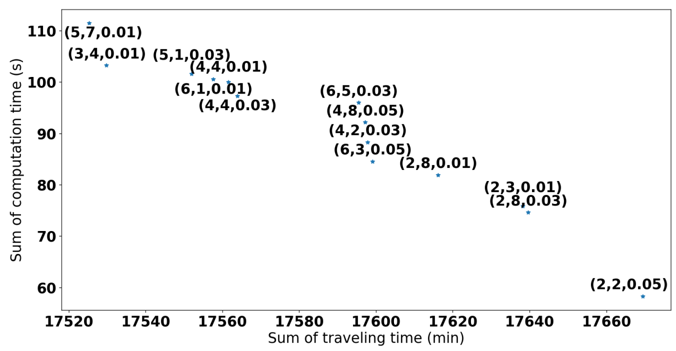

Fig. 4. Pareto Front of the MIP model for all tested combinations of hyperparameters.

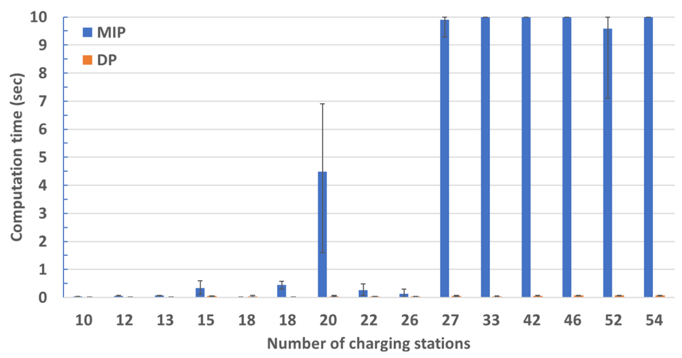

Fig. 5. Average computation for all temperatures with respect to the number of charging station on each route for each model. Computation time limit is set to 10 seconds and the error bars represent the maximum and minimum computation time for each instance.

C. *Results*

Both models are run on a Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, 2208 Mhz, 6 Cores with 8Gb of RAM. Computation time limit is set to 10 seconds since we want to use the model in a real-life context where the user wants to have the solution within seconds. When the MIP reaches the computation time limit of 10 seconds, we take its best solution found as the returned solution. As a results the solutions of the MIP when reaching the 10-seconds limit may not be optimal.

1) Computation time: Figure 5 presents the average computation time over all temperatures of the MIP and DP models with respect to the number of charging stations on the fixed route. The error bars represent the maximum and minimum computation time (for the different temperatures). We observe that both models optimally solve routes with fewer than 27 charging stations (the MIP model proves the optimality with the gap and the DP model always return optimal solutions). For these routes, the DP model has on average a significantly faster computation time than the MIP (on average 16.1 times smaller). Moreover, we observe that the MIP model takes more time to solve the 20 charging station instance than the 26. This is because the density of the charging stations can also have an impact on the hardness of an instance for the MIP model (highly condensed charging stations makes the problem harder to solve for the MIP model). This is not the case for the DP model, since its computational time is only affected by the number of charging stations and the choice of hyperparameters (see Section III-E). It makes the computational time of the DP model more stable than the MIP model. For routes with 27 charging stations or more, the MIP model reaches the computation time limit of 10 seconds. Deschênes *et al.* [9] show that for these instances the computation time *exceeds* 300 seconds, which makes the MIP unable to solve theses instances in a reasonable time. In comparison, the DP model never exceeds 0.1 seconds of computation time for all test instances. The DP model is thus several orders of magnitude faster than the MIP model for large instances.

2) Predicted route duration as returned by the model: We also compare the quality of the predictions of both models. Since the Akima interpolation presented in Section III-C can not ensure that the DP model predictions are always greater than the *real* route duration, we use the difference test [24] to verify if it is the case. The DP model predictions are on average 0.46 % ± 0.24 bigger than the *real* route duration. This represents a prediction for a 500-minute route that is 2.3 minutes bigger. The MIP model has predictions that are on average 0.18 % ± 0.26 closer to the *real* route duration than the DP model. Since 0 is in the interval, we *can not* conclude that there is a significant difference.

3) Real route duration: We show in Section IV-C.1 that the DP model dominates the MIP model in computation time, but it also needs to have similar *real* route duration. Since both models are tested on the same data, we compute the difference test [24] with 95 % confidence for the instances that did not reach the computation time limit of 10 seconds and for the instances that reached this limit. Results for the instances that did not reach the computation time limit show that we can not conclude on average a statistical difference between both models (the difference is 0.14 % ± 0.20 in favor of the MIP model). This is expected since both models proved the optimality of their solutions. For the instances that reached the time limit, results show that the DP model has on average statistically smaller *real* route duration. It is also expected since the solutions of the DP model are optimal, but it is not always the case for the solutions of the MIP model (due to the time limit). *Real* route duration is on average 0.30 % ± 0.13 lower for the DP model than for the MIP model. This represents a 90-second difference on a 500-minute route. The maximal difference is 1.87 % (a 9-minute difference on a 500-minute route). We can conclude that both models yield similar *real* route duration when the optimal solution is proven by both models. For large instances the solutions of the DP model has on average a smaller *real* route duration than the MIP model.

V. Conclusion

We showed that the proposed dynamic programming model solves the FRVCP-NLEM with a significantly lower computation time than the current state of the art for this specific problem (a MIP model). Experiments show that the dynamic programming model solves all instances (up to 96 charging stations) in less than 0.2 second. In comparison, the MIP model can only solve instances of up to 27 charging stations in a reasonable time and reaches the computation time limit of 10 seconds for all of the other instances. The MIP model takes over 300 seconds to solve these instances. This makes the DP model several orders of magnitude faster than the MIP model. For the instances that did not reach the computation time limit, the dynamic programming model has on average a computation time 16.1 times smaller than the MIP model. The DP and MIP models have a similar optimal *real* route duration when optimality is proven, but the DP model has on average a smaller *real* route duration for instances when the MIP model did not prove the optimality of the solutions. The dynamic programming model has a pseudo-linear increase in computation time as the number of charging stations increases, which makes it more scalable than the MIP model. As for future work, the dynamic programming model opens interesting possibilities to consider the stochastic aspect of the problem where, for example, segment duration can be affected by prior uncertain traffic conditions. It would also be interesting to see how this new model performs for solving the EVRP-NL. Other possible extensions include calculating a bound on the error as a function of the number of discretization of the state of charge in order to choose this hyperparameter more effectively. Finally, the proposed algorithm can be generalized into a path-finding algorithm that does not require a fixed route.

References

[1] IEA. (2020) Global ev outlook 2020. [Online]. Available: https://www.iea.org/reports/global-ev-outlook-2020

[2] W. Choi, E. Yoo, E. Seol, M. Kim, and H. H. Song, "Greenhouse gas emissions of conventional and alternative vehicles: Predictions based on energy policy analysis in south korea," *Applied Energy*, vol. 265, p. 114754, 2020.

[3] D. Hall, M. Moultak, and N. Lutsey, "Electric vehicle capitals of the world," *ICCT White Paper*, 2017.

[4] J. Kester, B. K. Sovacool, L. Noel, and G. Z. de Rubens, "Rethinking the spatiality of nordic electric vehicles and their popularity in urban environments: Moving beyond the city?" *Journal of Transport Geography*, vol. 82, p. 102557, 2020.

[5] T. Capuder, D. M. Sprčić, D. Zoričić, and H. Pandžić, "Review of challenges and assessment of electric vehicles integration policy goals: Integrated risk analysis approach," *International Journal of Electrical Power & Energy Systems*, vol. 119, p. 105894, 2020.

[6] X. Hao, H. Wang, Z. Lin, and M. Ouyang, "Seasonal effects on electric vehicle energy consumption and driving range: A case study on personal, taxi, and ridesharing vehicles," *Journal of Cleaner Production*, vol. 249, p. 119403, 2020.

[7] 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*, vol. 104, pp. 256–294, 2019.

[8] J. Liu, G. Lin, S. Huang, Y. Zhou, Y. Li, and C. Rehtanz, "Optimal logistics ev charging scheduling by considering the limited number of chargers," *IEEE Transactions on Transportation Electrification*, 2020.

[9] A. Deschênes, J. Gaudreault, L.-P. Vignault, F. Bernard, and C.-G. Quimper, "The fixed route electric vehicle charging problem with nonlinear energy management and variable vehicle speed," in *IEEE International Conference on Systems, Man and Cybernetics 2020*. IEEE, 2020, pp. 1451–1458.

[10] 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*, vol. 103, pp. 87–110, 2017.

[11] C. Lee, "An exact algorithm for the electric-vehicle routing problem with nonlinear charging time," *Journal of the Operational Research Society*, pp. 1–24, 2020.

[12] ——, "An exact algorithm for the electric-vehicle routing problem with nonlinear charging time," *Journal of the Operational Research Society*, pp. 1–24, 2020.

[13] L. Z. Velimirovic, A. Janjic, P. Vranic, I. Petkovski, and J. D. Velimirovic, "Dynamic electric vehicle routing problem," 2021.

[14] N. Rama, H. Wang, J. Orlando, D. Robinette, and B. Chen, "Route-optimized energy management of connected and automated multi-mode plug-in hybrid electric vehicle using dynamic programming," *Society of Automotive Engineers Technical Paper Series*, vol. 1, 2019.

[15] S. Pourazarm, C. G. Cassandras, and A. Malikopoulos, "Optimal routing of electric vehicles in networks with charging nodes: A dynamic programming approach," in *2014 IEEE International Electric Vehicle Conference (IEVC)*. IEEE, 2014, pp. 1–7.

[16] Y. Bengio, N. L. Roux, P. Vincent, O. Delalleau, and P. Marcotte, "Convex neural networks," in *Advances in neural information processing systems*, 2006, pp. 123–130.

[17] H. Akima, "A new method of interpolation and smooth curve fitting based on local procedures," *Journal of the ACM (JACM)*, vol. 17, no. 4, pp. 589–602, 1970.

[18] H. Ozdemir, "Comparison of linear, cubic spline and akima interpolation methods," *Hüseyin Özdemir.—2007*, 2007.

[19] M. LLC. (1999) Open charge map api. [Online]. Available: https://openchargemap.org/site/develop/api

[20] M. Feurer and F. Hutter, "Hyperparameter optimization," in *Automated Machine Learning*. Springer, Cham, 2019, pp. 3–33.

[21] P. Liashchynskyi and P. Liashchynskyi, "Grid search, random search, genetic algorithm: A big comparison for nas," *arXiv preprint arXiv:1912.06059*, 2019.

[22] T. Tušar and B. Filipič, "Visualization of pareto front approximations in evolutionary multiobjective optimization: A critical review and the prosection method," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 2, pp. 225–245, 2014.

[23] D. de la Fuente, M. A. Vega-Rodríguez, and C. J. Pérez, "Automatic selection of a single solution from the pareto front to identify key players in social networks," *Knowledge-Based Systems*, vol. 160, pp. 228–236, 2018.

[24] M. F. Triola, *Elementary statistics*. Addison Wesley Publishing Company, 1992.