

A Heuristic Approach for Fixed-Route Vehicle Charging with Nonlinear Time-Dependent Charging Function

Yangshuai Ke^{1, a}, Rui Luo^{2, b}, Jun Yang^{3, c *}

¹ Jilin University, China

² HKUST(GZ), China

³ UMass Amherst, MA, USA

^a keys1723@mails.jlu.edu.cn, ^b rluo041@connect.hkust-gz.edu.cn, ^c junyang@umass.edu

Abstract. This paper investigates the Fixed-Route Vehicle Charging Problem based on Time (FRVCP-T), considering the nonlinear characteristics of electric vehicle (EV) charging functions. A new linearization method using secant lines is introduced to convert the nonlinear charging curve into a Mixed-Integer Linear Programming (MILP) model with significantly fewer variables. To solve large-scale instances efficiently, we propose a heuristic algorithm—Dynamic Iterative Partial Optimization with Tabu Search (DIPOTS). Computational experiments on various datasets demonstrate that DIPOTS achieves high-quality solutions with average deviation below 0.6%, while reducing computational time by over 95% compared to commercial solver on large instances.

Keywords: Electric vehicle, nonlinear charging, mixed-integer linear programming, linearization, tabu search

1. Introduction

Electric vehicle (EV) routing is complicated by limited battery capacity and nonlinear charging behavior. While traditional vehicle routing problems focus on distance or cost minimization, charging constraints in EVs necessitate a different modeling approach.

Some research with simplified assumptions has been devoted to the fixed-tour vehicle-charging or fueling problems. Lin (Lin et al.[1]; Lin[2]) explored a fixed-tour vehicle-refueling problem and presented a linear-time algorithm to obtain the optimal solution. In the study conducted by Lin[2], different fuel prices at different vertices were considered. Lin's[2] research only considered the refueling amount and fuel price and did not consider customer nodes and detours, and Lin's[2] model requires the vehicle to travel with a lower limit and an upper limit. Liao et al. [3] considered on-site and off-site station fixed-tour EV touring and assumed that an EV swaps its battery at the battery switching station and continues the tour with the swapped full battery. Compared to the study conducted by Liao[3], our study extends the fixed-tour vehicle-charging problem (FRVCP) with partial nonlinear charging. According to Montoya et al. [4], Froger et al. [5] and Kullman et al. [6], FRVCP is a subproblem of electric vehicle routing problem (EVRP) and gets attention for potential to model complicated second-stage scene (e.g. partial nonlinear charging) Montoya et al.[4] raised an EVRP with partial charging and a nonlinear charging function that allow an EV to be partially charged and used a similar nonlinear charging function. Similarly linearization method is used by Froger et al. [5] and Kullman et al. [6].

We use a new linearization method for the nonlinear charging function and it use less variables compare to their work, which could significantly decrease the mixed-integer linear programming (MILP) model's complexity. The Fixed-Route Vehicle Charging Problem based on Time (FRVCP-T) seeks to minimize total travel and charging time along a predetermined route. Existing models often assume linear or full recharging, which does not align with real-world charging dynamics. This study addresses this gap through a novel MILP formulation and a heuristic solution method for practical efficiency.

2. Problem Description and Modeling

2.1 Nonlinear Charging Function Linearization

EV charging time increases nonlinearly as the battery approaches full capacity. To accurately capture this behavior, we use a piecewise linear approximation via a set of secant lines. For each charging event, binary variables determine which secant segment is active. This reduces the model's complexity and improves solver performance.

Let R denote the set of secant lines, and let K_r and B_r be the slope and intercept of each line $r \in R$. The charging time ct_i at station i is approximated as:

$$ct_i = \sum_{r \in R} (K_r pc_i + B_r) s_{ir} \quad (1)$$

Subject to constraints ensuring that only one segment s_{ir} is active at a time.

2.2 MILP Formulation

We define the objective as minimizing total time (travel + charging):

Parameters:

N	set of nodes, $N = \{1, \dots, n\}$.
N'	set of arcs, connecting consecutive nodes such as node 1 to node 2 until node n , in a fixed order.
i	index of arcs, $i = 1, 2, \dots, n$.
d_i	distance from node i to $i+1$.
t_i	travel time from node i to $i+1$.
d'_i	the shortest distance from node i to its nearest CS.
t'_i	travel time from node i to its nearest CS.
d''_i	the shortest distance from the nearest CS of node i to node $i+1$.
t''_i	the travel time from the nearest CS of node i to node $i+1$.
L	the maximum range of a fully charged EV.
R	set of secant lines from SOC = 0 to SOC = 100%, $R = \{1, \dots, r\}$.
K_r	slope of secant lines.
B_r	intercept of secant lines.
S_r	secant lines dots' ordinate value.
M	a large number, let $M = 9999$.
λ	safe threshold for arriving at a CS or final node.

Non-negative continuous variables:

ct_i	charging time at the nearest CS to node i .
cp_i	charged energy (in SDU) at the nearest CS to node i .
p_i	the remaining energy (in SDU) when arriving at or leaving from node i .
pc_i	the remaining energy (in SDU) when arriving at the nearest CS to node i .
tc_i	charging time required to charge a battery from SOC = 0 to pc_i .
D_i	actual travel distance from node i to node $i+1$.
T_i	actual travel time from node i to node $i+1$.

Binary variables:

y_i	if the nearest CS is visited after leaving node i . ($Y = \{y_i \mid i \in N\}$).
s_{ir}	indicating which secant line is used to obtain the relationship between pc_i and tc_i .

$$\text{Minimize } \sum_{i=1}^n T_i \quad (2)$$

$$\begin{cases} pc_i / L \geq S_r - 2(2 - s_{ir} - y_i) & \forall i \in N', \forall r \in R \\ pc_i / L \leq S_{r+1} + 2(2 - s_{ir} - y_i) & \forall i \in N', \forall r \in R \\ \sum_{r \in R} s_{ir} = y_i & \forall i \in N' \end{cases} \quad (3)$$

$$\begin{cases} pc_i / L \geq K_r tc_i + B_r - 2(2 - s_{ir} - y_i) & \forall i \in N', \forall r \in R \\ pc_i / L \leq K_r tc_i + B_r + 2(2 - s_{ir} - y_i) & \forall i \in N', \forall r \in R \end{cases} \quad (4)$$

$$\begin{cases} (pc_i + cp_i) / L \leq K_r (tc_i + ct_i) + B_r + 2(1 - y_i) & \forall i \in N', \forall r \in R \\ pc_i + cp_i \leq Ly_i & \forall i \in N' \\ tc_i + ct_i \leq My_i & \forall i \in N' \end{cases} \quad (5)$$

$$\begin{cases} p_1 = L \\ p_i - D_i + cp_i = p_{i+1} & \forall i \in N' \\ p_n - D_i + cp_i \geq \lambda \\ D_i = (1 - y_i)d_i + y_i(d'_i + d''_i) & \forall i \in N' \end{cases} \quad (6)$$

$$\begin{cases} pc_i \geq p_i - d'_i + L(1 - y_i) & \forall i \in N' \\ pc_i \leq p_i - d'_i + L(1 - y_i) & \forall i \in N' \\ pc_i \geq \lambda - L(1 - y_i) & \forall i \in N' \end{cases} \quad (7)$$

$$\begin{cases} D_i = (1 - y_i)d_i + y_i(d'_i + d''_i) & \forall i \in N' \\ T_i = (1 - y_i)t_i + y_i(t'_i + t''_i) + ct_i & \forall i \in N' \end{cases} \quad (8)$$

In the abovementioned model, objective function (2) minimizes the total time cost by summing real time cost in each arc. Constraints (3), (4) and (5) are piecewise linear charging constraints. Constraints (6) regulate the starting power, balances electric charge and consumption, and last visit with safe threshold. Constraints (7) regulate the remaining power in each section to reach the CS. Constraints (8) regulate the actual travel distance and time.

3. Solution Methodology: DIPOTS Algorithm

To tackle computational complexity, we propose a heuristic: Dynamic Iterative Partial Optimization with Tabu Search (DIPOTS). This method iteratively improves charging decisions y_i by solving subproblems using a MILP solver.

The steps are:

1. Start with an initial feasible solution Y_0 .
2. In each iteration, fix most y_i and select a random subset to optimize.
3. Solve the partial MILP problem.
4. Update if a better solution is found.
5. Adjust the number of optimized variables based on solver time.
6. Repeat until convergence or reaching maximum attempts.

A. Algorithm Design

DIPOTS iteratively improves charging decisions by optimizing a subset of them using Gurobi while the rest are fixed. A dynamic mechanism adjusts the number of variables optimized at each iteration based on elapsed time to balance solution quality and speed.

Key steps:

1. Start with an initial feasible solution Y_{0Y_0}
2. In each iteration:
 - Select a random subset of y_{iy_i}
 - Solve the partial MILP using Gurobi
 - Update the solution if improved
 - Adjust subset size based on solver time
3. Terminate when no improvement is observed for $B=B_{max}$ iterations

B. Pseudocode

Procedure DIPOTS:

Input: Initial Y_0 , W , B_{max} , t_{min} , t_{max}

1. Set $Y \leftarrow Y_0$, $B \leftarrow 1$
2. Repeat:
 - a. Fix most y_i , unfix W variables
 - b. Solve partial MILP via Gurobi
 - c. If improved, update Y , reset B
 - d. Else, increase B
 - e. Adjust W based on solver time
3. Until $B > B_{\max}$

Output: Best Y found

4. Computational Experiments

We evaluate DIPOTS using test sets of 80, 160, and 320 customer nodes. Each instance was tested with different battery ranges ($L = 200, 300, 400$) and numbers of secant lines ($R = 2, 4, 8$). The solver performance was benchmarked against the Gurobi solver.

Results Summary:

- 80 Nodes: DIPOTS achieved $<0.2\%$ deviation in under 5 seconds.
- 160 Nodes: DIPOTS maintained $<0.4\%$ deviation with 10–15x faster computation.
- 320 Nodes: Gurobi often failed to finish; DIPOTS completed in 44s with $<0.6\%$ deviation.

Table 1 Effects of different L values on FRVCP-T

No.	Gurobi ($L=200$)				Gurobi ($L=300$)				Gurobi ($L=400$)		
	Obj.	Time	Gap		Obj.	Time	Gap		Obj.	Time	Gap
80_1	2906.90	25	0		2661.44	63	0		2540.09	44	0
80_2	3417.88	641	0		3093.06	36	0		2944.22	23	0
80_3	3160.18	37	0		2877.24	24	0		2741.73	26	0
80_4	3145.92	19	0		2860.73	75	0		2724.00	23	0
80_5	3033.40	134	0		2765.62	109	0		2638.23	24	0
80_6	3091.31	15	0		2786.91	19	0		2647.87	20	0
80_7	2795.66	14	0		2549.28	23	0		2428.05	22	0
80_8	3231.37	37	0		2898.14	10	0		2754.63	15	0
80_9	2863.34	18	0		2583.11	12	0		2460.08	12	0
80_10	3405.27	15	0		3059.34	11	0		2903.28	16	0
80_11	3171.32	15	0		2850.70	13	0		2714.01	14	0
80_12	3242.89	4	0		2901.42	5	0		2751.01	7	0
80_13	3379.32	8	0		3041.42	20	0		2885.43	19	0
80_14	3173.32	14	0		2860.50	10	0		2721.91	15	0
80_15	3075.73	17	0		2756.57	13	0		2619.93	20	0
AVG	3139.59	67	0		2836.37	30	0		2698.30	20	0

Table 2 Effects of different R values on FRVCP-T

No.	Gurobi ($R=\{1,2\}$)				Gurobi ($R=\{1,\dots,4\}$)				Gurobi ($R=\{1,\dots,8\}$)		
	Obj.	Time	Gap		Obj.	Time	Gap		Obj.	Time	Gap
80_1	3018.68	3	0		2920.18	31	0		2906.90	25	0
80_2	3513.84	6	0		3434.30	162	0		3417.88	641	0
80_3	3264.27	3	0		3173.61	35	0		3160.18	37	0
80_4	3256.98	15	0		3162.62	31	0		3145.92	19	0
80_5	3141.43	5	0		3047.90	68	0		3033.40	134	0
80_6	3185.25	4	0		3109.91	33	0		3091.31	15	0
80_7	2899.73	3	0		2808.92	14	0		2795.66	14	0
80_8	3301.78	2	0		3245.13	11	0		3231.37	37	0

80_9	2933.89	2	0		2875.91	13	0		2863.34	18	0
80_10	3484.35	2	0		3424.86	6	0		3405.27	15	0
80_11	3261.85	3	0		3188.25	7	0		3171.32	15	0
80_12	3334.25	2	0		3256.63	2	0		3242.89	4	0
80_13	3473.86	3	0		3395.63	4	0		3379.32	8	0
80_14	3257.47	2	0		3185.01	6	0		3173.32	14	0
80_15	3154.90	2	0		3091.96	10	0		3075.73	17	0
AVG	3232.17	4	0		3154.72	29	0		3139.59	68	0

Table 3 Computational results for the 80, 160, and 320 size instances of FRCVP-T ($L = 200$)

No.	Gurobi				DIPOTS			
	Obj.	Time	Gap		Best_Obj.	AVG_Obj.	AVG_Time	AVG_Dev
80_1	2906.90	25	0		2906.90	2910.04	16	0.11%
80_2	3417.88	641	0		3417.88	3418.20	18	0.01%
80_3	3160.18	37	0		3160.18	3160.35	16	0.01%
80_4	3145.92	19	0		3145.92	3146.07	15	0.01%
80_5	3033.40	134	0		3033.40	3033.40	16	0
80_6	3091.31	15	0		3091.31	3091.57	9	0.01%
80_7	2795.66	14	0		2795.66	2795.66	9	0
80_8	3231.37	37	0		3231.37	3231.37	8	0
80_9	2863.34	18	0		2863.34	2863.34	9	0
80_10	3405.27	15	0		3405.27	3405.27	8	0
80_11	3171.32	15	0		3171.32	3179.12	9	0.25%
80_12	3242.89	4	0		3242.89	3247.82	5	0.15%
80_13	3379.32	8	0		3379.32	3380.60	5	0.04%
80_14	3173.32	14	0		3173.32	3173.38	7	0.01%
80_15	3075.73	17	0		3075.73	3076.40	9	0.02%
AVG	3139.59	68	0		3139.59	3140.84	11	0.04%
160_1	6121.28	1000	0.28%		6121.28	6123.60	28	0.04%
160_2	5704.57	1000	0.13%		5704.57	5704.71	30	0.01%
160_3	5998.95	1000	0.16%		5998.95	6002.71	29	0.06%
160_4	6237.73	1000	0.20%		6237.73	6242.05	28	0.07%
160_5	6007.54	1000	0.09%		6007.54	6031.93	30	0.41%
160_6	6728.02	509	0		6728.02	6728.36	14	0.01%
160_7	6443.02	234	0		6443.02	6443.59	14	0.01%
160_8	6129.51	1000	0.11%		6129.51	6129.52	17	0.01%
160_9	6267.67	935	0		6267.67	6275.46	14	0.12%
160_10	6523.41	273	0		6523.41	6528.00	12	0.07%
160_11	6518.21	629	0		6518.21	6520.05	14	0.03%
160_12	6106.17	280	0		6106.17	6106.89	15	0.01%
160_13	6495.61	42	0		6495.61	6502.81	14	0.11%
160_14	6387.32	776	0		6390.79	6425.13	16	0.59%
160_15	6491.48	33	0		6491.48	6494.60	11	0.05%
AVG	6277.37	648	0.06%		6277.60	6283.96	19	0.11%
320_1	12178.35	1000	0.37%		12177.85	12184.00	70	0.05%
320_2	12793.35	1000	0.25%		12793.30	12793.50	69	0.01%
320_3	12730.54	1000	0.29%		12731.25	12736.66	77	0.05%
320_4	12130.26	1000	0.32%		12129.95	12147.77	60	0.14%
320_5	12576.23	1000	0.29%		12575.36	12576.41	63	0.01%
320_6	12586.13	1000	0.21%		12586.13	12587.71	34	0.01%

320_7	12501.17	1000	0.25%		12500.89	12500.89	39	0
320_8	12556.32	1000	0.14%		12556.32	12556.42	30	0.01%
320_9	12681.50	1000	0.20%		12681.22	12684.54	38	0.02%
320_10	13265.31	1000	0.14%		13265.31	13272.45	32	0.05%
320_11	12613.75	1000	0.17%		12613.75	12619.87	33	0.05%
320_12	12597.20	1000	0.31%		12597.20	12598.56	34	0.01%
320_13	12948.81	1000	0.11%		12948.81	12973.20	28	0.19%
320_14	12777.27	1000	0.23%		12777.27	12789.03	31	0.09%
320_15	13199.92	1000	0.10%		13199.64	13206.15	28	0.05%
AVG	12675.74	1000	0.23%		12675.62	12681.81	44	0.05%

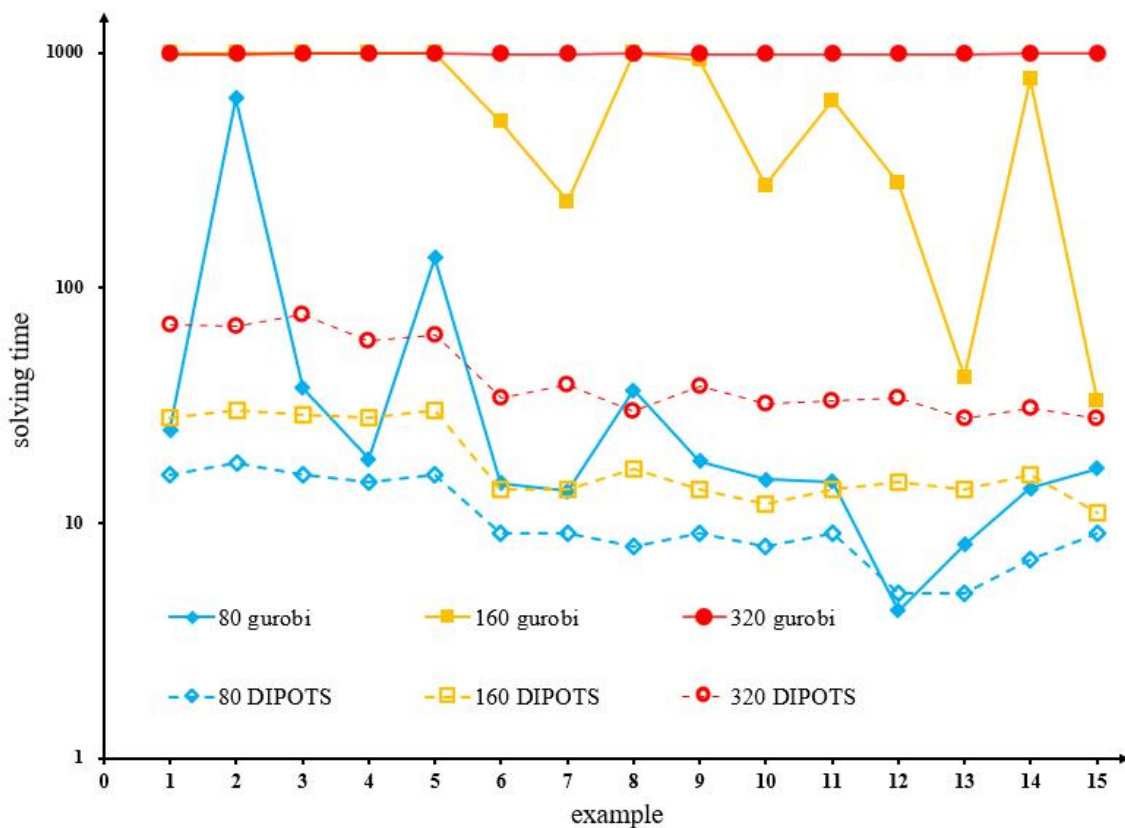


Fig. 1 Solving time comparison curves of the Gurobi solver and DIPOTS algorithm for different sizes

First, in terms of solution time, it can be observed that the DIPOTS algorithm requires less average time for almost all examples of different sizes, except for 80_12. In larger case problems, the DIPOTS algorithm shows a higher potential. In 160-size problem, the DIPOTS only consumes 2.9% of the CPU time compared to that consumed when the Gurobi solver is directly invoked. Furthermore, in 320-size problem, the DIPOTS required an average CPU time of 44.4 s, whereas the Gurobi solver requires more than 1000 s. A time comparison curve of these two methods is shown in Fig. 1. Solving time comparison curves of the Gurobi solver and DIPOTS algorithm for different size, wherein the logarithmic axis and each curve show the huge time savings.

Second, in terms of solution quality, almost all optimal solutions are obtained after 10 runs of the DIPOTS algorithm in three size problems, except for 160_14 and 320_3.

Moreover, the maximum average deviation was only 0.59% in all the three size problems, which are acceptable in real-life scenarios.

The main aim of this experiment was to demonstrate that the proposed mathematical MILP model could be solved by the Gurobi solver and could be used to develop the heuristic DIPOTS algorithm to solve the problem by consuming less CPU time and with acceptable deviation.

5. Conclusion

This paper presents a new MILP formulation for the FRVCP-T, incorporating a novel linearization method that reduces model complexity. To address scalability, we propose DIPOTS, a heuristic algorithm that combines dynamic partial optimization and tabu search. Computational results demonstrate that DIPOTS is both efficient and accurate, making it suitable for real-time or large-scale applications.

References

- [1] Lin, S. H., Gertsch, N., & Russell, J. R. (2007). A linear-time algorithm for finding optimal vehicle refueling policies. *Operations Research Letters*, 35(3), 290-296.
- [2] Lin, S. H. (2008). Finding optimal refueling policies: a dynamic programming approach. *Journal of Computing Sciences in Colleges*, 23(6), 272-279.
- [3] Liao, C. S., Lu, S. H., & Shen, Z. J. M. (2016). The electric vehicle touring problem. *Transportation Research Part B: Methodological*, 86, 163-180.
- [4] Montoya, A., Guéret, C., Mendoza, J. E., & Villegas, J. G. (2017). The electric vehicle routing problem with nonlinear charging function. *Transportation Research Part B: Methodological*, 103, 87-110.
- [5] Froger, A., Mendoza, J. E., Jabali, O., & Laporte, G. (2019). Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104, 256-294.
- [6] Kullman, N. D., Froger, A., Mendoza, J. E., & Goodson, J. C. (2021). frvcpy: An open-source solver for the fixed route vehicle charging problem. *INFORMS Journal on Computing*, 33(4), 1277-1283.