A Column Generation Approach for the Electric Autonomous Dial-A-Ride Problem

Yue Su[1], Nicolas Dupin[2], Sophie N. Parragh[3], Jakob Puchinger[4,1]

[1] Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, 91190, Gif-sur-Yvette, France
{yue.su,jakob.puchinger}@centralesupelec.fr
[2] Univ Angers, LERIA, SFR MATHSTIC, F-49000 Angers, France.
nicolas.dupin@univ-angers.fr
[3] Institute of Production and Logistics Management, Johannes Kepler University Linz, 4040, Linz
sophie.parragh@jku.at
[4] EM Normandie Business School, Métis Lab, 92110, Clichy, France
jpuchinger@em-normandie.fr

Mots-clés : *Dial-A-Ride Problem, Electric Autonomous Vehicles, Labeling Algorithm.*

1 Introduction

The Electric Autonomous Dial-A-Ride Problem (E-ADARP) consists in scheduling a fleet of Electric Autonomous Vehicles (EAVs) to provide ride-sharing services for customers that specify their origins and destinations. The E-ADARP was first introduced by [1] as a problem variant of the classical Dial-A-Ride Problem (DARP) defined in [2]. Although the E-ADARP shares some constraints of the classical DARP, the E-ADARP differentiates in two aspects : (i) a weighted-sum objective that minimizes both total travel time and total excess user ride time, and (ii) the employment of EAVs and a partial recharging policy ; The first aspects (ii) requires determining minimal-excess-time schedules for a feasible E-ADARP route, while the second aspect complicates the feasibility checking process as en-route recharging needs to be considered. Other E-ADARP features further increase the complexity of solving the problem. These features include (a) the elimination of maximum route duration constraints due to the autonomy of vehicles, (b) a minimum battery level that must be maintained at the end of the route, and (c) limited visits to each recharging station.

This paper proposes a highly-efficient labeling algorithm that can be embedded in the Column Generation (CG) approach to solve the E-ADARP. The original mixed integer programming (MIP) model is decomposed by the Dantzig–Wolfe decomposition into a master problem and pricing subproblems, which are formulated as Elementary Shortest Path Problems with Minimizing Excess Ride Time (hereafter ESPP-MERT). To solve the problem, we first introduce a fragment-based representation. Based on this representation, A novel approach is invoked to abstract fragments to arcs while ensuring excess-time optimality. We then construct a new graph that possesses all feasible routes of the original one, by enumerating all feasible fragments and connecting them with depots and recharging stations in a feasible way. Routes with negative reduced costs are generated by a customized labeling algorithm with strong dominance rules and complex REFs. To demonstrate the performance of the proposed labeling algorithm, we perform our labeling algorithm in CG on all existing E-ADARP instances and compare CG results to best-reported results in [1, 5].

2 The ESPP-MERT Description

The problem is defined on a complete directed graph $G = (V, A)$, where V represents the set of vertices, and A denotes the set of arcs. V can be further partitioned into several subsets, i.e.,

$V = N \cup S \cup O \cup F$, where N represents the set of all customers, S is the set of recharging stations, O and F denote the set of origin depots and destination depots, respectively. The set of all pickup vertices is denoted as $P = \{1, \cdots, i, \cdots, n\}$ and the set of all drop-off vertices is denoted as $D = \{n+1, \cdots, n+i, \cdots, 2n\}$. The union of P and D is N, i.e., $N = P \cup D$. Each customer request is a pair $(i, n+i)$ for $i \in P$ and the maximum ride time for users associated with request i is assumed to be m_i. A time window is defined on each node $i \in V$, denoted as $[e_i, l_i]$, in which e_i and l_i represent the earliest and latest time at which the vehicle starts its service, respectively. A load q_i and a service duration s_i are also associated with each node $i \in V$. For pickup node $i \in P$, q_i is positive. For the corresponding drop-off node $n+i$, we have $q_{n+i} = -q_i$. For other nodes $j \in O \cup F \cup S$, q_j and s_j are equal to zero. Vehicles are assumed to be heterogeneous in terms of their maximum vehicle capacities (denoted as C_k) and homogeneous in terms of battery capacities (denoted as Q). The travel time on each arc $(i, j) \in A$ is denoted as $t_{i,j}$, and the battery consumption is denoted as $b_{i,j}$. The recharging rate at charging facilities is α. To avoid the numerical problem when calculating time and energy, we define $h_{i,j} = b_{i,j}/\alpha$ to convert the battery consumption $b_{i,j}$ on arc (i, j) to the time needed for recharging this amount of energy. Similarly, we can also convert the current energy level to the time needed to recharge to this energy level. Let H denote the time required to recharge from zero to full battery capacity Q. Partial recharging is allowed while a vehicle visits recharging stations, and a minimum battery level γQ must be respected at destination depots, where we analyze three different γ values, namely, $\gamma \in \{0.1, 0.4, 0.7\}$. Higher values of γ result in more tightly constrained instances, which are harder to solve. The triangle inequality is assumed to hold for travel times and battery consumption.

The ESPP-MERT consists in finding the minimum-reduced-cost route while satisfying the following E-ADARP constraints :

1) Each route starts at an origin depot and ends at a destination depot ;
2) Pairing and precedence constraints for pickup and drop-off nodes ;
3) Time window constraints ;
4) Maximum user ride time constraints ;
5) Maximum vehicle capacity constraints ;
6) Maximum battery capacity constraints ;
7) Minimum battery level constraints at the destination depot ;
8) At-most-one visit to each recharging station ;
9) No passenger is onboard when the vehicle visits a recharging station ;

The reduced cost $\bar{c}_\omega$ for a route ω is formulated as :

$$c_\omega - \sum_{i \in V} \theta_{i,w} \lambda_i \tag{1}$$

where c_ω is the weighted-sum objective function value for a route ω, $\theta_{i,w}$ is the binary coefficient that denotes whether node i is on ω, and λ_i is the associated dual variable values.

3 Forward Labeling Algorithm for the ESPP-MERT

The most challenging point of solving the ESPP-MERT is maintaining excess-time optimality in the extension of labels. To handle this issue, we design a labeling algorithm by introducing a fragment-based representation in Section 3.1, which ensures the minimum excess user ride time being non-decreasing during the label extension. Our labeling algorithm includes two steps : From Section 3.2 to 3.3, we construct a new sparser network G_{sp} by abstracting fragments to arcs. In G_{sp}, we guarantee excess-time optimality for all arcs, and we prove that G_{sp} possesses all feasible routes over the original graph G. Section 3.4 generates routes with negative reduced costs by an efficient labeling algorithm with strong dominance rules over G_{sp}.

3.1 Representation of partial paths

To minimize the total excess user ride time for a partial path $\mathcal{P}$, we first introduce the battery-restricted fragment as in Definition 1 :

Definition 1. Assuming that $\mathcal{F} = (i_1, i_2, \cdots, i_k)$ is a sequence of pickup and drop-off nodes, where the vehicle arrives empty at i_1 and leaves empty at i_k and has passenger(s) on board at other nodes. Then, we call $\mathcal{F}$ a battery-restricted fragment if there exists a feasible route $\mathcal{R}$ of the form :

$$(o, s_{i_1}, \cdots, s_{i_v}, \overbrace{i_1, i_2, \cdots, i_k}^{\mathcal{F}}, s_{i_{v+1}}, \cdots, s_{i_m}, f),$$

where $s_{i_1}, \cdots, s_{i_v}, s_{i_{v+1}}, \cdots, s_{i_m} (v, m \geqslant 0)$ are recharging stations and $o \in O$, $f \in F$. In the case of route $\mathcal{R}$ is feasible without a recharging station (i.e., $v = m = 0$), the battery-restricted fragment is equivalent to the one defined in [4].

Based on Definition 1, each E-ADARP route can be regarded as the concatenation of battery-restricted fragments, recharging stations (if required), origin depot, and destination depot. Clearly, this representation guarantees the total excess user ride time for $\mathcal{P}$ being non-decreasing. Meanwhile, it eliminates the need to determine the reduced cost at every node along $\mathcal{P}$ as we can generalize a sequence of nodes to an arc when extending $\mathcal{P}$ by a fragment.

3.2 Abstracting fragments as arcs

In the labeling algorithm, a partial path is not fixed until reaching the sink node, which introduces difficulty in ensuring excess-time optimality while maintaining feasibility. This section solves this issue from another perspective : assuming a fragment $\mathcal{F} = \{1, 2, \cdots, m\}$ and any excess-time optimal schedule $\mathcal{A}$ is presented as a set of service start times over $\mathcal{F}$, i.e., $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_m)$. Based on our representation of paths, we only need to determine all possible values of $\mathcal{A}_1, \mathcal{A}_m$ for any excess-time optimal schedule $\mathcal{A}$, such that we can maintain excess-time optimality by restricting time windows at node 1 and node m. To calculate all possible values of $\mathcal{A}_1, \mathcal{A}_m$, we introduce *vehicle-waiting-time optimal schedules* :

Definition 2. A vehicle-waiting-time optimal schedule $\mathcal{B}$ for a fragment $\mathcal{F}$ is defined as a set of service start times $\mathcal{B}_i$, $i \in \mathcal{F}$ that minimize the total waiting time of the vehicle waits at each node along $\mathcal{F}$ (i.e., $\sum_{i=2}^{m}[\mathcal{B}_i - (\mathcal{B}_{i-1} + t_{i-1,i} + s_{i-1})]$).

For a given fragment $\mathcal{F}$, we determine the following two vehicle-waiting-time optimal schedules. We show later in Theorem 1 that these schedules determine all possible values of $\mathcal{A}_1, \mathcal{A}_m$.

1. the "latest" vehicle-waiting-time optimal schedule $\mathcal{B}^l$;
2. the "earliest" vehicle-waiting-time optimal schedule $\mathcal{B}^e$.

Assuming that $\mathcal{B}_i^l$ is the service start time for schedule $\mathcal{B}^l$ at node i. Then, the arrival time at each node i is $Arr_i = \mathcal{B}_{i-1}^l + t_{i-1,i} + s_{i-1}, 2 \leqslant i \leqslant m$. The waiting time Δ_i at node i is calculated as $\Delta_i = \max\{0, \mathcal{B}_i^l - Arr_i\}$. $\mathcal{B}_i^l$ is defined inductively as follows :

1. $\mathcal{B}_1^l = l_1$;
2. assuming $\mathcal{B}_i^l$ has been defined for $i < v$, we define $\mathcal{B}_v^l$ by :
   (a) if the extension from node $(v-1)$ will not violate the time window constraint at node v (i.e., $Arr_v \leq l_v$), then we define $\mathcal{B}_v^l = \max\{e_v, Arr_v\}$;
   (b) Otherwise,
   — if $\min_{i<v}\{0, \mathcal{B}_i^l - e_i\} \geq Arr_v - l_v$, we can update the schedule at nodes $1, \cdots, v-1$ by moving forward $Arr_v - l_v$. Then the extension from node $(v-1)$ to node v will not violate the time window constraint. I.e., we update $\mathcal{B}_i^l$ as $\mathcal{B}_i^l - (Arr_v - l_v)$ for $i = 1, \cdots, v-1$ and define $\mathcal{B}_v^l = l_v$;
   — Otherwise, there is no feasible schedule for $\{1, 2, \cdots, v\}$, as the time window constraint at node v is always violated.

After determining $\mathcal{B}^l$ on $\mathcal{F}$, $\mathcal{B}^e$ is determined by moving forward $\mathcal{B}^l$ on $\mathcal{F}$ by the maximum amount of time that will not change the minimum vehicle waiting time. We denote the maximum amount of time that B^l can be moved forward as $\overleftarrow{\Delta}$, which is calculated by taking the minimum value among all the $\mathcal{B}_i^l - e_i$, that is :$\overleftarrow{\Delta} = \min_{i \in \mathcal{F}}\{\mathcal{B}_i^l - e_i\}$. Then, $\mathcal{B}_i^e = \mathcal{B}_i^l - \overleftarrow{\Delta}$, $i \in \mathcal{F}$.

Clearly, for any excess-time optimal schedule $\mathcal{A}$ of a fragment $\mathcal{F}$, we can obtain $\mathcal{A}$ by moving forward $\mathcal{B}^l$ by a certain amount of time δ if $\mathcal{B}^e \neq \mathcal{B}^l$. In case of waiting time generated, we proved that the service start time of $\mathcal{A}$ is the same as $\mathcal{B}^l$ at the first and last node of the fragment (Theorem 1). The proof is omitted due to the limit of space.

Theorem 1. *Assuming that fragment $\mathcal{F} = \{1, 2, \cdots, m\}$, $\mathcal{A}$ is an **excess-time optimal** schedule and $\mathcal{B}^l$ is the constructed latest vehicle-waiting time optimal schedule over $\mathcal{F}$. Then there exists $\delta^e, \delta^l \geq 0$ such that :*

$$\mathcal{A}_1 = \mathcal{B}_1^e + \delta^e, \ \mathcal{A}_m = \mathcal{B}_m^e + \delta^e;$$
$$\mathcal{A}_1 = \mathcal{B}_1^l - \delta^l, \ \mathcal{A}_m = \mathcal{B}_m^l - \delta^l.$$

Based on Theorem 1, restricting time windows at node 1 and node m to $[\mathcal{B}_1^e, \mathcal{B}_1^l]$ and $[\mathcal{B}_m^e, \mathcal{B}_m^l]$ will include all excess-time optimal schedules on fragment $\mathcal{F} = \{1, \cdots, m\}$. Then we can abstract $\mathcal{F}$ to an arc $(1, m)$ such that :

1. the total travel time from 1 to m (denoted as $t'_{1,m}$) is $\mathcal{B}_m^l - \mathcal{B}_1^l$;
2. the time window of node 1 is $[\mathcal{B}_1^e, \mathcal{B}_1^l]$ and $[\mathcal{B}_m^e, \mathcal{B}_m^l]$ for node m ;
3. the energy consumption from 1 to m is $\sum_{i=1}^{m-1} h_{i,i+1}$;

3.3 Construction of a new sparser graph

In this section, we construct a new sparser graph G_{sp} by enumerating all feasible fragments and abstracting them to arcs. To generate all feasible fragments, we start from each pickup node and extend it in a feasible way, assuming that the initial battery inventory is Q. Then, these "arcs" are connected to depots and recharging stations in a feasible way. Figure 1 shows an example of constructing new arcs in G_{sp}. It should be noted that for two different fragments, even though they have the same start node $i+$ and end node $j-$, we must treat them as two different arcs in G_{sp} as they represent different fragments containing different sequences of nodes and have different restricted time windows.

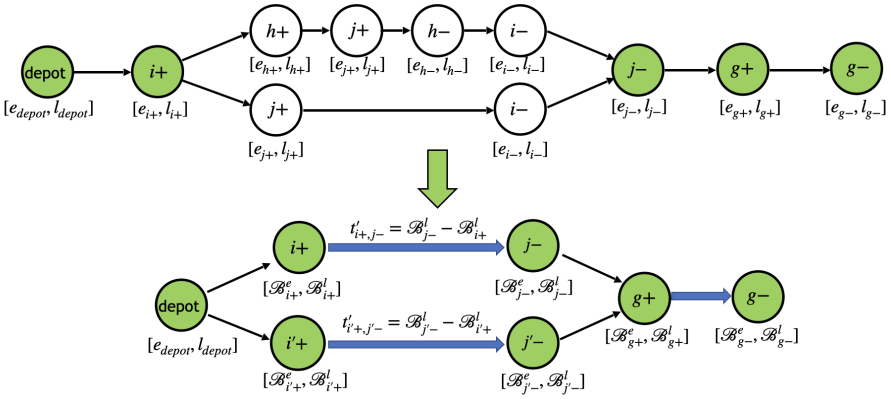

FIG. 1 – Example of constructing arcs in the new graph G_{sp}

Next, we work on the new sparser graph G_{sp} instead of G, as we show in Theorem 2 that G_{sp} possesses all feasible routes over G.

Theorem 2. *Let $\mathcal{R}$ be a route over graph G, and $\mathcal{R}_{sp}$ be the corresponding route over G_{sp}. Then $\mathcal{R}$ is feasible if and only if $\mathcal{R}_{sp}$ is feasible.*

3.4 Forward labeling algorithm

We develop a labeling algorithm on G_{sp} by extending the one proposed by [3] considering features of the E-ADARP. We denote L_i as the label associated with a partial path ends with node i. The forward labeling algorithm extends labels from a source node $o_k \in O$ to a sink node $f \in F$. Let a label associated with a partial path $\mathcal{P}$ from o_k to current vertex i be $L_i = \{T_i^{cost}, (T_i^{rch_s})_{s\in S}, T_i^{tMin}, T_i^{tMax}, T_i^{rtMax}, T_i^{req}\}$, the definition of each resource is described as follows :

1. T_i^{cost} : The reduced cost of the partial route until i ;
2. $T_i^{rch_s}$: The number of times recharging station $s \in S$ is visited along partial path $\mathcal{P}$;
3. T_i^{tMin} : The earliest service start time at vertex i assuming that, if a recharging station is visited prior to i along the partial path, a minimum recharge (ensuring the battery feasibility up to i) is performed ;
4. T_i^{tMax} : The earliest service start time at vertex i assuming that, if a recharging station is visited prior to i along the partial path, a maximum recharge (ensuring the time-window feasibility up to i) is performed ;
5. T_i^{rtMax} : In order to propagate the information along the path, we make the artificial assumption that vehicles can be recharged at all vertices. But in reality, the vehicle will never go to the recharging station when passengers are on board. With this assumption, T_i^{rtMax} denotes the maximum recharging time required to fully recharge at vertex i if a recharging station is visited prior to i along $\mathcal{P}$, a minimum recharge (ensuring the battery feasibility up to i) is performed ;
6. T_i^{req} : The set of visited and unreachable requests until i along partial path $\mathcal{P}$. A request is said to be unreachable if time window constraints are violated if visited.

We extend a label $L_i = \{T_i^{cost}, (T_j^{rch_s})_{s\in S}, T_i^{tMin}, T_i^{tMax}, T_i^{rtMax}, T_i^{req}\}$ along arc (i, j) of G_{sp} using Resource Extension Functions (REFs) in [3]. The label feasibility is checked via the following proposition :

Proposition 1. *The route $\mathcal{R}$ is feasible if and only of $\forall j \in R$, the label L_j satisfies :*

$$T_j^{tMin} \leqslant \mathcal{B}_j^l, \quad T_j^{tMin} \leqslant T_j^{tMax}, \quad T_j^{req_p} \leqslant 1, \forall p \in P, \quad T_j^{rtMax} \leqslant \begin{cases} (1-\gamma)H, & j \in F \\ H, & otherwise \end{cases}$$

If j is a recharging station, then $T_j^{rch_j} \leqslant 1$ must be satisfied. In case of a feasibility violation, the corresponding label will be discarded. The following dominance rules are applied for two labels associated with the same node.

Definition 3. Let $L^k = \{T_i^{cost}, (T_j^{rch_s})_{s\in S}, T_i^{tMin}, T_i^{tMax}, T_i^{rtMax}, T_i^{req}\}, k \in 1, 2$ be two labels. Assume that the partial path associated to L^1 and L^2 are $\mathcal{P}_1$ and $\mathcal{P}_2$, respectively, and $\mathcal{P}_1$, $\mathcal{P}_2$ end at the same node. L^1 dominates L^2 if and only if :

$$T_1^r \leqslant T_2^r, \forall r \in \{cost, rch, tMin\} \tag{2}$$

$$T_1^{req} \subseteq T_2^{req} \tag{3}$$

$$T_1^{rtMax} - (T_1^{tMax} - T_1^{tMin}) \leqslant T_2^{rtMax} - (T_2^{tMax} - T_2^{tMin}) \tag{4}$$

$$T_1^{rtMax} - (T_2^{tMax} - T_1^{tMin}) \leqslant T_2^{rtMax} \tag{5}$$

The last two conditions are equivalent to the requirement that : for every service start time $T_2 \in [T_2^{tMin}, T_2^{tMax}]$, there exists a service start time $T_1 \in [T_1^{tMin}, T_2]$ such that $T_1^{rtMax} - (T_1 - T_1^{tMin}) \leqslant T_2^{rtMax} - (T_2 - T_2^{tMin})$. In other words, we can always find a service start time $T_1 \leqslant T_2$ that consumes not more energy.

4 Results and Discussion

The proposed labeling algorithm has been embedded in the CG approach to solving the E-ADARP. The obtained CG results are compared to the best-reported results in [1] and [5]. We summarize all results in Table 1, where *#opt*, *#bestlb*, and *#bestub* denote the number of times the respective algorithm provides optimal solutions, the best lower bound/integer solution of all considered algorithms, respectively. B/F denotes the ratio between the number of best solutions obtained and the number of feasible solutions obtained. $LB\%$ is the gap between the obtained lower bound and the best-obtained integer solution among all considered algorithms.

On small-to-medium-sized instances (type-a and -u instances), Our CG algorithm optimally solves 50 out of 84 instances and generates high-quality lower bounds with a 0.31% average deviation to the best-known solutions. We obtain 40 equal lower bounds and 24 better lower bounds than those reported in [1]. The overall quality of the lower bound is improved by 1.33% on average. In addition, we provide 10 new optimal solutions on previously solved and unsolved instances. As [1] does not test type-r instances (large-scale instances with up to 8 vehicles and 96 requests), we compare CG results to the best-reported heuristic results of [5]. We report 14 better solutions (5 of them are optimal) as well as 17 lower bounds. Hence, we prove the performance of our labeling algorithm.

type-a and -u instances results summary

	CG with labeling algorithm					Best-reported results [1]				
scenario	*#opt*	*#bestlb*	$LB\%$	B/F	avg.time(s)	*#opt*	*#bestlb*	$LB\%$	B/F	avg.time(s)
$\gamma = 0.1$	21	23	0.10%	25/28	1154.12	23	23	1.17%	26/28	1502.83
$\gamma = 0.4$	20	22	0.13%	26/28	1382.65	19	19	1.84%	21/27	1701.36
$\gamma = 0.7$	9	20	0.71%	20/26	2454.04	11	15	1.93%	14/19	3965.85
Overall	50	64	0.31%	71/82	1663.60	53	57	1.64%	61/74	2390.01

type-r instances results summary

	CG with labeling algorithm					Best-reported results [5]				
scenario	*#opt*	*#bestlb*	*#bestub*	B/F	avg.time(s)	*#opt*	*#bestlb*	*#bestub*	B/F	avg.time(s)
$\gamma = 0.1$	3	10	8	8/10	9423.78	0	NA	2	2/10	981.36
$\gamma = 0.4$	2	7	6	6/8	9974.08	0	NA	3	3/9	1368.13
Overall	5	17	14	14/18	9698.93	0	NA	5	5/19	1174.74

TAB. 1 – Results summary of CG algorithm on all benchmark instances

Our labeling algorithm offers new insights into designing an exact algorithm for solving a practical version of the electric DARP. One important "by-product" of our labeling algorithm is an exact and efficient schedule optimization method that can determine the excess-time optimal schedule for a given E-ADARP route. This is the first time that excess user ride time minimization is handled exactly in the E-ADARP. This schedule optimization method can also be applied to DARPs with multiple objectives, where total excess user ride time is minimized.

Références

[1] Claudia Bongiovanni, Mor Kaspi et Nikolas Geroliminis. "The electric autonomous dial-a-ride problem". In : *Transportation Research Part B : Methodological* 122 (2019), p. 436-456.

[2] Jean-François Cordeau et Gilbert Laporte. "The dial-a-ride problem : models and algorithms". In : *Annals of operations research* 153.1 (2007), p. 29-46.

[3] Guy Desaulniers et al. "Exact algorithms for electric vehicle-routing problems with time windows". In : *Operations Research* 64.6 (2016), p. 1388-1405.

[4] Yannik Rist et Michael A Forbes. "A new formulation for the dial-a-ride problem". In : *Transportation Science* 55.5 (2021), p. 1113-1135.

[5] Yue Su, Jakob Puchinger et Nicolas Dupin. "A Deterministic Annealing Local Search for the Electric Autonomous Dial-a-Ride Problem". In : (2021).