A Column-generation-based heuristic for the Electric Autonomous Dial-a-Ride Problem

Yue Su[1], Nicolas Dupin[2], Jakob Puchinger[1,3]

[1] Laboratoire Genie Industriel, CentraleSupélec, Université Paris-Saclay, France
{yue.su, jakob.puchinger}@centralesupelec.fr

[2] Laboratoire Interdisciplinaire des Sciences du Numérique, CNRS, Université Paris-Saclay, France
nicolas.dupin@lisn.upsaclay.fr

[3] IRT SystemX, Palaiseau, France
jakob.puchinger@irt-systemx.fr

Mots-clés : *dial-a-ride problem, electric autonomous vehicles, column generation, heuristic.*

1 Introduction

The dial-a-ride problem (DARP) involves scheduling a fleet of vehicles that provide ride-sharing services to users specifying their origins and destinations while respecting several types of constraints (e.g., time window, capacity, and user ride time constraints). It shares several features of vehicle routing problem (VRP) with time window and pickup and delivery vehicle routing problem ([3]). However, DARP is different from vehicle routing problems from several perspectives :(1) users should be picked up from specific origins and delivered to specific destinations, (2) user ride time is considered as constraints/objective to guarantee service quality. Accounting user ride time is the most important characteristic of DARP, making it more complicated than VRP as the delay of service may happen on customer nodes to eliminate unnecessary waiting time and reduce user ride time. Recently, many researchers have proposed various approaches for tackling DARPs. Detailed literature review can be found in [3].

Our study investigates the Electric Autonomous Dial-a-Ride Problem (E-ADARP) introduced in [1] which consists of scheduling electric autonomous vehicles (EAVs) to transport users from specific origins to specific destinations within predefined time windows. The E-ADARP has the following features that are different from typical DARPs : (1) detour to recharging station on the route, (2) partial recharging at recharging station, (3) vehicles can depart from different origin depots and select from a set of optional destination depots, (5) no restriction for route duration as we consider autonomous vehicles, (6) a minimum battery level (presented by a minimum battery level ratio γ) should be maintained at the end of the route.

The literature that investigated E-ADARP is scarce. Only two researches have studied the efficient approaches for tackling E-ADARP (i.e., [1] and [6]), which the first one employed an exact method (i.e., branch-and-cut), and the second one proposed a meta-heuristic method (i.e., Deterministic Annealing with local search, hereafter DA+LS). However, their limitations are that the proposed exact method in [1] cannot be applied in solving large-scale instances, and the proposed meta-heuristic method in [6] cannot guarantee the solution quality. In this study, we integrate the heuristic (DA+LS) with the exact method (i.e., column generation) to provide near-optimum solutions in reasonable computational time. In the proposed hybrid algorithm, heuristic operators (including constructive and local search operators) are embedded in column generation to identify negative reduced-cost columns aggressively. The solution obtained from column generation is further improved by DA. To accelerate the program, we also implement a column management strategy to clean useless columns and speed up the calculation (as presented in [4]). Then, we evaluate the effectiveness of hybridization. Two benchmark instance sets, namely the adapted instances from standard instances set of literature [2] and the real ride-sharing dataset of Uber obtained from https://github.com/dima42/uber-gps-analysis/tree/master/gpsdata) are considered for numerical experiments. For the sake of simplicity, we call the adapted standard instance set as "Cordeau instance set" and the real-data instance set as "Uber instance set". We consider three scenarios representing three different energy restriction levels (i.e., low, medium, high energy restriction). The obtained solutions are compared to the solutions from the literature (i.e., [1],[6]), which report the best results by using exact and meta-heuristic methods, respectively.

2 Problem Formulation

The problem is considered on a complete directed graph, denoted as $G = (V, A)$, where V is the set of vertices and A is the set of arcs. Set V can be further partitioned into the set of customers $N = \{1, 2, \cdots, i, \cdots, 2n\}$ and the set of recharging stations $F = \{2n+1, \cdots, 2n+f\}$. The customer set N is divided into the customer pick-up node set $P^u = \{1, \cdots, i, \cdots, n\}$ and the customer drop-off set $D^u = \{n+1, \cdots, n+i, \cdots, 2n\}$. The nodes in the customer set N can be visited exactly once, and the nodes in the recharging stations set F can be visited at most once (as defined in our baseline work [1]). Each user request is a couple $(i, n+i)$, where $i \in P^u$ and $n+i \in D^u$, n is the number of requests. As we have multiple origin depots and end depots, each vehicle can depart from different origin depot and end at different destination depot. Specifically, the number of end depots is larger than the number of vehicles. Consequently, the vehicle can select the end depot from a set of destination depots. In other words, the destination depot will not be predefined at the beginning of the time horizon, and it will be decided during the service. It can be visited, therefore, at most once. The total number of vehicles is denoted as K.

The MIP formulation of the problem is borrowed from [1], where several decision variables are defined as shown in Table 1. The objective function is a weighted sum of total travel time and excess user ride time. The excess user ride time of user i is denoted as R_i, and is calculated as the actual ride time of user i minus the direct travel time $t_{i,n+i}$.

Decision Variables	Definitions
$x_{i,j}^k$	1 if vehicle k travels from location i to $j \in V$, 0 otherwise
T_i^k	Service begin time of vehicle k at location $i \in V$
L_i^k	Load of vehicle k at location $i \in V$
B_i^k	Battery load of vehicle k at location $i \in V$
E_f^k	Charging time of vehicle k at charging station $f \in F$
R_i	Excess user ride time of passenger $i \in P^u$

TAB. 1 – Decision variables for E-ADARP

E-ADARP is an NP-hard problem and is more complicated than classical VRP and DARP as recharging needs to be considered. It is even much harder to be solved when taking into account the minimum-battery-level constraint. Last but not least, we should consider user ride time in both constraint and objective function. From the preliminary experiments of solving the instances with Gurobi, even the small-sized instance cannot be solved within two hours on our computer.

2.1 Column generation framework

We obtain our column generation formulation through the Dantzig-Wolfe decomposition associated with the compact MIP model presented in [1], in which we decompose the MIP model of [1] into a "master problem" and a set of sub-problems (identical in our case). The master problem is defined as a set covering problem in which a set of feasible routes cover the set of transportation requests.

The objective function is formulated as the sum of cost (shown in Equation 1). We improve formulation of set covering problem by adding in the objective function (equation 1) a high penalty P_i for request i not realized. We denote a_i as a binary variable to present whether request i is visited or not. If $a_i = 1$, the request i is omitted, otherwise, the request i is visited. Let Ω denote the set of all feasible routes concerning the constraints defined in the MIP model of [1]. For each route $\omega \in \Omega$, we define c_ω as the cost for route ω. Additionally, we define $\theta_{i\omega}$ as a binary coefficient that equals one if the request i is visited by the route ω (zero otherwise). Let y_ω denote a binary variable that equals one if and only if the route $\omega \in \Omega$ is included in the solution (0 otherwise). The number of requests to be served is n, and the total vehicle number is K. To restrict the visit of the recharging station and the destination depot (as the destination set is optional), we define another binary coefficient $\phi_{f\omega}$, determining whether the recharging station f is visited in route ω. The set covering problem is formulated as :

$$\min \sum_{\omega \in \Omega'} c_\omega * y_\omega + \sum_{i \in P^u} P_i * a_i \tag{1}$$

subject to :

$$\sum_{\omega \in \Omega'} \theta_{i\omega} y_\omega \geqslant 1 - a_i, \forall i \in P^u \tag{2}$$

$$\sum_{\omega \in \Omega'} \phi_{f\omega} y_\omega \leqslant 1, \forall f \in F \cup D^d \tag{3}$$

$$\sum_{\omega \in \Omega} y_\omega \leqslant |K| \tag{4}$$

$$y_\omega \in \{0, 1\}, \forall \omega \in \Omega \tag{5}$$

In the constraint (2), if request i is not served, $a_i = 1$ and it is obviously valid (in this case $\theta_{i\omega} = 0$). The benefit of adding penalties in the continuous RMP formulation is that we can avoid a large number of iteration times of DA+LS to have the feasible solution, especially in the case of $\gamma = 0.4, 0.7$. If there exist request i is not served in the solution obtained by continuous RMP, then the solution cost will be very high, which means a high value for the upper bound. Correspondingly, the dual values for the requests not severed will be also high. Therefore, the generation of columns will favor containing these omitted requests, which can lead to very negative columns. Then, those columns that contain the omitted requests will be added into the column pool, and the continuous RMP will be solved again. Usually, with the penalization in the continuous RMP formulation, column generation will find feasible solutions quickly.

Constraint (3) restricts the number of visits to recharging stations. Constraint (4) guarantees that the number of vehicles used cannot exceed the maximum number of vehicles. Constraint 5 restricts the value of variable y_ω. In order to retrieve the dual information, we solve the linear relaxation of the master problem, that is $y_\omega \geqslant 0$. The continuous master problem is called continuous MP.

As the size of Ω grows exponentially with the number of customers, making the set covering problem intractable, we restrict the master problem to a subset of Ω (denoted as Ω'), and we solve the continuous Restricted MP (abbreviated as continuous RMP). New columns that have negative reduced cost (i.e., have the chance to improve objective 1) are generated by solving the pricing sub-problem in which the objective is to minimize the reduced cost (presented in Equation 6). In the expression, the dual variable of constraint of constraint 2 and constraint 3 are denoted as $\pi_i (i \in P^u)$ and $\tau_f (f \in F \cup D^d)$. Another dual variable associated with constraint (4) is δ. The new columns with negative reduced costs are added into Ω', and the master problem is resolved using the updated column pool. The continuous RMP and its sub-problem are solved iteratively until no more negative reduced cost column can be found, the column generation process converges.

$$\bar{c_w} = c_w - \sum_{i \in P^u} \theta_{i\omega} \pi_i - \sum_{f \in F \cup D^d} \phi_{f\omega} \tau_f - \delta \tag{6}$$

3 Hybrid Column Generation Heuristic

We propose a hybrid column generation algorithm to solve the E-ADARP. The initial column pool is first populated by applying DA+LS described in our previous work [6]. The subproblem is first solved heuristically on the reduced graph through a constructive algorithm designed in the fashion of the greedy randomized search procedure (hereafter GRASP). This method is applied as long as columns with negative reduced costs can be generated. The column generated by GRASP is then improved by a DA-based column generator. Different from the mentioned algorithm for initializing the column pool, the DA-based column generator minimizes the reduced cost of the column rather than the actual route cost. Several operators have been adjusted in order to fit the context. If the constructive algorithm cannot find negative columns on the reduced graph, then the subproblem is solved on the complete graph and the columns found are further improved by the DA algorithm. Every N_{iter}, we manage the size of the column pool by cleaning probably "useless" columns as in [4, 5]. The "useless" columns include the columns not used in the continuous RMP and integer RMP and the columns that the reduced costs are higher than a threshold value D_{max}. The column generation is iterated until no more columns can be identified by either of the heuristics or the maximum time limit is reached. Then, with the obtained solution from the final resolution of integer RMP, we apply deterministic annealing, which aims to minimize the actual cost to further improve the solution. The hybrid algorithm returns the best solution of iterating DA algorithm N^{DA} times. The hybridization scheme is outlined in Algorithm 1.

Algorithm 1 Hybrid column generation framework

```
Input: Initial columns pool population, initial solution from running DA+LS for $N^{init}$ times;
Output: Primal heuristic solution;
  Repeat
  Step1 : Constructing columns by GRASP
  Generate new negative reduced cost columns by solving reduced graph with GRASP;
  if No column is found by reduced graph heuristic then
      Generate new negative reduced cost columns by GRASP with complete graph;
  end if
  Step2 : Improving constructed columns by local search
  if Columns found by the previous step then
      for Each negative reduced cost column do
          Applying adapted DA algorithm to improve column;
      end for
  else
      for Each route in basic solution do
          Applying adapted DA algorithm to identify columns;
      end for
  end if
  Step3 : Continuous/integer RMP resolution
  Solve continuous RMP on $\Omega'$, update dual information for dual variables $\pi_i$,$\tau_f$,$\delta$;
  if $iter \% N_{iter} = 0$ then
      Solve integer RMP on $\Omega'$;
      Record the column used in continuous RMP and integer RMP resolution;
      Column management : remove useless columns in the column pool;
  end if
  Until no more column can be found by heuristic method or time limit is reached;
  Step4 : DA re-optimization
  Apply DA+LS for $N^{DA}$ times to improve the solution obtained from CG.
  Return the best solution obtained;
```

4 Numerical Results

We implement the proposed algorithm in Julia 1.3.0 and Gurobi 9.1.1, and we conduct experiments on a Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 6 Cores, 12 threads computer with 32G RAM. We consider two benchmark instance sets from the literature [1]. For each instance, we conduct experiments under different energy restriction levels for vehicles returning the end depot (i.e., low/medium/high energy restriction), which is represented by a minimum energy level that vehicles should maintain at the end of the route. We define γ as the minimum energy level ratio. We set three different values for γ (i.e., $\gamma = 0.1, 0.4, 0.7$), describing low, medium, high energy restriction scenario, respectively. For example, when $\gamma = 0.1$, vehicles should maintain at least 10% energy when arriving end depot.

The main parameters used in the hybrid algorithm are : (1) Number of iterations of DA for column pool initialization N^{init} ; (2) Number of iteration of DA to further improve the solution from CG, denoted as N^{DA} ; (3) Maximum time for CG execution T^{CG} ; Based on the parameter tuning experiments, we select $N_{init} = 200$ for CG initialization, and 2500 as N_{DA}, 100 seconds for T^{CG}. Due to the limit of pages, we do not precise the parameter adjusting results here.

4.1 Comparing to existing results

We first evaluate the effectiveness of hybridization compared to the pure DA+LS iterating 2700 times. The results of the proposed hybrid CG algorithm are concluded in the line named "hybrid CG" in Table 2 and the results of pure DA+LS iterating 2700 times are concluded in the line named "DA+LS 2700". We sequentially conduct the experiments on adapted Cordeau instances and Uber instances under three energy scenarios ($\gamma = 0.1, 0.4, 0.7$). We use the exact results of [1] as our baseline results and denoted as "BKS" in the Table. Both "hybrid CG" and "DA+LS 2700" perform ten runs on each instance. The average gap between the best/average-obtained solution of the proposed CG algorithm with BKS (denoted as "BC%" and "AC%" respectively) are presented in the second and third column in Table 2. To better illustrate the algorithm performance, we also report the number of the best solution obtained and the number of the feasible solution obtained, denoted as "B/F". Finally, the average computational time (denoted as "AT") of instance is presented.

Compared to the pure DA+LS iterating 2700 times, our proposed hybrid CG algorithm outperforms DA+LS 2700 on both adapted Cordeau and Uber instances in all cases. In the case of $\gamma = 0.1, 0.4$, the hybrid CG finds an equal number of best/feasible solutions as DA+LS does but with less average gap to BKS (i.e., BC%, AC%). However, the hybrid CG algorithm is more robust than the pure DA+LS algorithm when $\gamma = 0.7$, as the proposed hybrid CG algorithm finds more feasible and best solutions in much less computational time.

Then we compare the results of the hybrid CG algorithm to "DA+LS 5000" ([6]) and BKS ([1]). From Table 2, the overall BC% (i.e., the average gap between best-obtained hybrid CG results with BKS) of hybrid algorithm results is 0.89%. For the solution quality, DA+LS iterating 5000 times performs better than the proposed algorithm with lower BC% and AC% on each case. However, its computational efficiency deteriorates when solving hard instances. We observe a significant improvement in the computational time compared to the DA+LS 5000, with up to 27.81% (236.00s v.s. 326.92s on Uber instance with $\gamma = 0.7$). On average, the computational time is decreased by 3.34% per instance. Compared to BKS, we find feasible solutions for all the instances when $\gamma = 0.1$. On Cordeau instance set, we find one new-best solution, so we mark the "B/F" for BKS [1] with 13/14. Two new-best solutions are found when $\gamma = 0.4$ and another two new best solutions are found in the case of $\gamma = 0.7$.

5 Conclusion and Perspectives

In this paper, we consider solving the E-ADARP with a hybrid CG algorithm. The proposed hybrid algorithm integrates exact and heuristic methods and is proven efficient in solving E-ADARP.

$\gamma = 0.1$					$\gamma = 0.1$				
Cordeau instances	BC%	AC%	B/F	AT(s)	Uber instances	BC%	AC%	B/F	AT(s)
hybrid CG	0.68%	1.89%	8/14	226.02	hybrid CG	0.90%	2.57%	8/14	315.86
DA+LS 2700	0.71%	1.83%	8/14	120.46	DA+LS 2700	2.16%	2.87%	4/14	333.76
DA+LS 5000 [6]	0.37%	0.74%	9/14	203.43	DA+LS 5000 [6]	0.69%	1.52%	6/14	281.71
BKS^a [1]	-	-	13/14	1210.54	BKS^a [1]	-	-	14/14	1280.83
$\gamma = 0.4$					$\gamma = 0.4$				
Cordeau instances	BC%	AC%	B/F	AT(s)	Uber instances	BC%	AC%	B/F	AT(s)
hybrid CG	1.18%	2.78%	6/14	230.92	hybrid CG	1.50%	3.65%	5/14	340.15
DA+LS 2700	2.09%	3.43%	6/14	174.52	DA+LS 2700	2.09%	4.26%	4/14	342.10
DA+LS 5000 [6]	0.64%	1.38%	6/14	265.58	DA+LS 5000 [6]	0.57%	1.43%	7/14	324.84
BKS^a [1]	-	-	13/14	1233.47	BKS^a [1]	-	-	13/13	2169.25
$\gamma = 0.7$					$\gamma = 0.7$				
Cordeau instances	BC%	AC%	B/F	AT(s)	Uber instances	BC%	AC%	B/F	AT(s)
hybrid CG	NA^b	NA^b	6/7	229.94	hybrid CG	NA^b	NA^b	4/9	236.00
DA+LS 2700	NA^b	NA^b	2/4	352.58	DA+LS 2700	NA^b	NA^b	1/8	338.89
DA+LS 5000 [6]	NA^b	NA^b	6/9	247.57	DA+LS 5000 [6]	0.67%	3.72%	4/10	326.92
BKS^a [1]	-	-	7/9	4333.5	BKS^a [1]	-	-	10/10	3598.2

BKS^a presents the results on a 3.6 GHz Intel(R) Core(TM) with 16 Gb of RAM.
NA^b indicates the gap cannot be calculated due to the unequal number of solved instances.

TAB. 2 – Results comparison on Cordeau and Uber instances under different γ

Its efficiency does not deteriorate when solving medium-to-large-sized instances under high energy restriction levels while the existing heuristic (DA+LS 5000) and exact method are impacted. Compared to the state-of-art heuristic results, the proposed algorithm improves the computational efficiency by up to 27.81%. Compared to exact results, the average gap between best-obtained solutions and BKS is 0.89%, and several new best solutions are found on the previously solved and unsolved instances.

The results offer several research possibilities. The current study can be improved by considering adaptive-ordered operators in column generation and applying stabilization technique for dual variables (e.g., [5]). Another interest is to combine the exact method in the column generation to solve the pricing sub-problem into optimality. This topic is currently being addressed in our next work, and the proposed algorithm provides tighter lower bounds for E-ADARP than the best-reported lower bound. Additionally, the proposed column generation scheme can be applied in solving larger instances in the real world.

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. “A branch-and-cut algorithm for the dial-a-ride problem”. In : *Operations Research* 54.3 (2006), p. 573-586.

[3] 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.

[4] Jacques Desrosiers et Marco E Lübbecke. “A primer in column generation”. In : *Column generation*. Springer, 2005, p. 1-32.

[5] Nicolas Dupin, Rémi Parize et El-Ghazali Talbi. “Matheuristics and Column Generation for a Basic Technician Routing Problem”. In : *Algorithms* 14.11 (2021), p. 313.

[6] Yue Su, Jakob Puchinger et Nicolas Dupin. “A Deterministic Annealing Local Search for the Electric Autonomous Dial-a-Ride Problem”. In : *ROADEF 2020, full preprint available at https://hal.archives-ouvertes.fr/hal-03211499/document* (2021).