Yue SU
Jakob PUCHINGER
Nicolas DUPIN
Laboratoire Génie Industriel, CentraleSupélec
Gif-sur-Yvette, France
Contact : yue.su@centralesupelec.fr

A Deterministic Annealing Local Search for the Electric Autonomous Dial-a-Ride Problem

Introduction

- With an astounding growth in automobile ownership, a series of transport-related problems appear in urban cities worldwide.
- Traffic congestion and air pollution severely impacted both the economy and the environment.
- Two effective axes have been introduced: using clean energy powered vehicles and providing ride-sharing services.
- DARP is a fleet of vehicles provides shared-ride services to users specifying their origin, destination, and preferred arrival time [1].
- No significant has been done on the DARP with electric autonomous vehicles

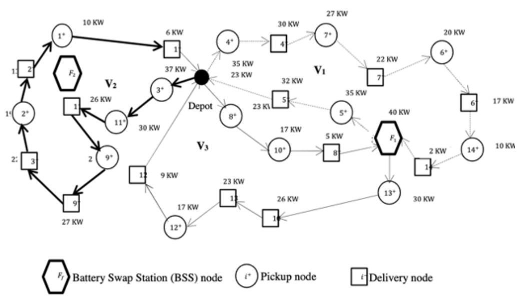
Modelling

Objective:
Minimizing total travel time and excess user ride time: $w_1 = 0.75, w_2 = 0.25$

$$w_1 * \sum_{k \in K} \sum_{j \in V \setminus P^d} \sum_{i \in V} x_{i,j}^{k,r} \cdot t_{i,j}^r + w_2 * \sum_{i \in P^u} R_i$$

Constraints
- Constraints on arcs and nodes
- Time window on pickup and drop-off nodes
- Capacity of vehicle
- User maximum ride time
- Battery and charging limitation

Given data :
- Pickup and drop-off locations for each request
- Recharging station locations
- Average energy consumption rate (KWh/min)
- Planning horizon



Illustrative example of E-ADARP

Features of problem:
Detours to recharging stations
Partial charging at the recharging stations
Vehicle can locate at different origin depots
Vehicle select from optimal destination depots
No restriction for route duration

Meta-heuristic

Deterministic Annealing local search algorithm

Algorithm structure

Input:
- Obtained solution from a parallel insertion heuristic
- Solution cost $c(x)$, number of requests inserted Nb_{req}

Local search:
- *AddNewRequest:* a special operator to insert "rejected" requests.
- If all the requests have been inserted into the initial solution, *AddNewRequest* is deactivated.
- Otherwise, *AddNewRequest is activated* to insert uninserted requests.

Threshold updated:
- When no global best solution is found, the threshold value is reduced by T_{max}/T_{red}
- If T is negative, the threshold is reset

Algorithm 1 Deterministic Annealing Algorithm
Input: Initial solution generated by parallel heuristic
Parameter initialization: $T = T_{max}$, n_{imp}, $Nb_{iteration}$, $i_{imp} = 0$, $iter = 0$;
$x = x_b = x_{init}$;
$c(x_b) = c(x) = c(x_{init})$;
$Nb_{req} = \sum_{k \in K} \sum_{i \in P^u} v_{i,k}$;
Output: Improved feasible solution x_b;
while $iter \leqslant Nb_{iteration}$ **do**
  $i_{imp} \leftarrow i_{imp} + 1$;
  for $j = 1 \rightarrow n_{oper}$ **do**
    Apply local search operator on x to obtain neighboring solution x';
    if x is battery-infeasible solution **then**
      Backward insertion algorithm to repair in-feasibility;
    end if
    if $c(x') < c(x) + T$ **then**
      $x \leftarrow x'$;
    end if
  end for
  if $Nb_{req} \leqslant Nb_{user}$ **then**
    $n_{oper} \leftarrow AddNewRequest$
  end if
  $iter \leftarrow iter + 1$
  $Nb'_{req} \leftarrow \sum_{k \in K} \sum_{i \in P^u} v'_{i,k}$
  if $c(x') < c(x_b)$ **and** $Nb'_{req} = Nb_{req}$ **or** $Nb'_{req} \geqslant Nb_{req}$ **then**
    $x_b \leftarrow x'$
    $i_{imp} \leftarrow 0$
  end if
  if $i_{imp} > 0$ **then**
    $T \leftarrow T - T_{max}/T_{red}$
    if $T < 0$ **then**
      $r \leftarrow$ random number between 0 and 1
      $T \leftarrow r \times T_{max}$
      if $i_{imp} > n_{imp}$ **then**
        $x \leftarrow x_b$
        $i_{imp} \leftarrow 0$
      end if
    end if
  end if
end while
return x_b

A parallel insertion algorithm is proposed to obtain the initial routes for E-ADARP

Creating randomly vehicles $m \leq K$

Sorted customer pickup nodes by their earliest time window

The first m customers are assigned randomly to the generated vehicle

Insertion of requests with respect to cost minimization

Calculate distance between the last assigned element in each existing route with the first element in the list

Sort the vehicles in increasing order of distances calculated in the last step

Feasibility examination of the insertion for the selected node in first vehicle's route

Each request is inserted to its "best position" (increasing minimal cost)

References

- [1] C. Bongiovanni, M. Kaspi, N. Geroliminis, The electric autonomous dial-a-ride problem, Transportation Research Part B: Methodological 122 (2019) 436–45
- [2] T. Vidal, T. G. Crainic, M. Gendreau, C. Prins, A hybrid genetic algorithm with adaptive diversity managementfor a large class of vehicle routing problems with time-windows, Computers & operations research 40 (1) (2013)475–489.
- [3] J.-F. Cordeau, A branch-and-cut algorithm for the dial-a-ride problem, Operations Research 54 (3) (2006)573–586
- [4] S. Ropke, J.-F. Cordeau, G. Laporte, Models and branch-and-cut algorithms for pickup and delivery problemswith time windows, Networks: An International Journal 49 (4) (2007) 258–2

Meta-heuristic

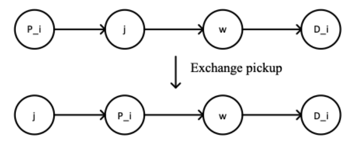
Bi-directional insertion algorithm

We design a bi-directional insertion algorithm to determine the position of recharging stations if we have got battery-infeasible solution. The pseudo-code is shown as below.

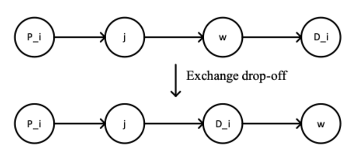
Algorithm 3 Bi-directional Insertion Algorithm Structure
Input: Battery infeasible solution x;
Output: Repaired feasible solution x_r;
$cap = 0$, $pos = \emptyset$;
for $i = 1 \rightarrow$ length(route) **do**
  $cap = cap + q_{route[i]}$;
  if $cap = 0$ **then**
    $pos = pos \cup i$;
  end if
end for
Calculate energy list that contains energy level at each node;
if energy levels in the route are positive **then**
  Backward Insertion Algorithm to repair the minimum battery level constraint;
else
  Forward insertion algorithm to repair the negative battery level;
  Adjust the values of elements in pos;
  Backward Insertion Algorithm to repair the minimum battery level constraint;
end if
return x_r

Local search operators: 3 intra-route operators

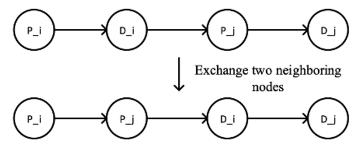
Exchange pickup operator
- Swapping the position of two consecutive nodes (i, j) which i is a pickup node and j is not its corresponding dropoff node
- In each iteration, one pickup node is selected randomly.



Exchange drop-off operator
- Swapping the position of two consecutive nodes (i, j) which j is a drop-off node and i is not its corresponding pickup node
- In each iteration, one dropoff node is selected randomly.

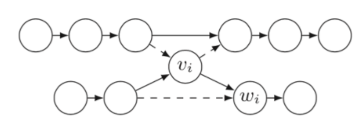


Exchange two neighboring nodes operator
- There is another situation where the successive node of pickup node P_i is its drop-off D_i, and the previous node of drop-off node D_j is its corresponding pickup P_j, but we can still exchange D_i and P_j to obtain a new solution.

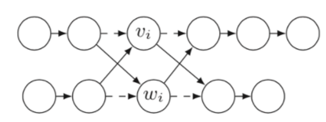


Local search operators: 3 inter-route operators

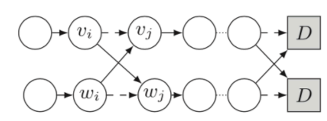
Relocate operator
- Removing a user request from its current route and re-insert the request in the best position in the route of another vehicle



Exchange operator
- Swapping two requests of two different routes.
- The pickup (drop-off) vertex of the first roue can only be inserted in the same position as the pickup (drop-off) vertex of the second route.



2-opt operator
- Selecting two random routes and removes an arc from each of them. The removed arc is connected with the remaining part of the other route in the route pair



Reduce neighborhood size

In the local search process, the neighborhood is searched by using different operators. As we iterate the meta-heuristic thousands of times to find high-quality solutions, the size of the neighborhood has a direct impact on computational efficiency. Three technic are used to recuce the neighborhood size: time window tightening, arc elimination, customer correlation measure [2].

Results and Future Work

Meta-heuristic performance

We use three set of instances to conduct experiments: adapted Cordeau instances[3], adapted Uber instances, and we newly introduce the adapted Ropke instances [4] to conduct large-scale experiments.

Parameter adjusting and design decision

Table 1
Sensitivity analysis for the number of iterations

Nb_{iter}	1000	2000	3000	4000	5000	6000
Average gap (%)	1.31%	0.93%	0.65%	0.49%	0.40%	0.38%
Average CPU (s)	46.47	100.83	136.98	179.28	203.43	247.69

Table 2
Sensitivity analysis for t_{max}

t_{max}	0.6	0.9	1.2	1.5	1.8	2.1	2.4
Average gap (%)	1.97 %	1.45%	1.2%	0.51%	1.16%	2.07%	2.11 %
Average CPU (s)	227.51	228.52	234.11	231.04	238.58	258.94	269.33

Table 3
Contribution of operators

Operator	Average gap (%)	Average CPU (s)
base	0.37%	203.43
2-opt	4.14%	217.03
exchange	0.75%	214.45
relocate	1.45%	155.38
exchange pickup	0.64%	190.59
exchange drop-off	0.68%	207.95
exchange neighboring	0.62%	201.70

Algorithm performance

We have compared our results with the best-known solutions of the exact model. the average solution gap between our algorithm results and the reported exact results is only 0.58%. Surprisingly, three new best solutions have been found

Table 6
Results comparison on Cordeau instances $\gamma = 0.7$

Instance	DA + LS algorithm		Three-index model[9][a]		Two-index model[9][a]		Gap%	
$\gamma = 0.7$	BC(min)	AT(s)	Obj(min)	CPU(s)	Obj(min)	CPU(s)	3index	2index
a2-16	**240.66**	172.46	240.66*	29.4	240.66*	5.4	0	0
a2-20	NA	427.00	NA	7200	NA	7200	NA	NA
a2-24	364.19	337.16	358.21*	3539.4	358.21*	961.2	1.67 %	1.67 %
a3-18	**240.58**	111.07	240.58*	642.6	240.58*	48	0	0
a3-24	281.82	237.78	277.72*	2957.4	277.72*	152.4	1.48%	1.48%
a3-30	NA	232.04	NA	7200	NA	7200	NA	NA
a3-36	NA	321.11	NA	7200	494.04	7200	NA	NA
a4-16	**223.13**	51.17	223.13*	2179.2	223.13*	67.2	0	0
a4-24	318.29	122.56	321.03	7200	318.21*	1834.8	-0.85%	0.03%
a4-32	**427.92***	206.37	NA	7200	430.07	7200	NA	-0.50%
a4-40	NA	310.56	NA	7200	NA	7200	NA	NA
a4-48	NA	371.45	NA	7200	NA	7200	NA	NA
a5-40	**434.49***	236.87	NA	7200	447.63	7200	NA	-2.94%
a5-50	**625.95***	328.42	NA	7200	NA	7200	NA	NA
Avg		224.11		5296.29		4333.5	NA[b]	NA[b]

[a] Results on a 3.6 GHz Intel(R) Core(TM) with 16 Gb of RAM.
[b] NA indicates the gap cannot be calculated due to the unequal number of solved instances.
* Numbers in bold with star indicate new solution found by proposed algorithm.

Table 4
Results comparison on Cordeau and Uber instances $\gamma = 0.1$

Cordeau instances	DA + LS algorithm	Three-index model[9]	Two-index model[9]
AT(s)	203.43	4505.23	1210.54
Gap	-	NA[a]	0.37%
Uber instances	**DA + LS algorithm**	**Three-index model[9]**	**Two-index model[9]**
AT(s)	281.71	5422.16	1280.83
Gap	-	NA[a]	0.69%

[a] NA indicates the gap cannot be calculated due to the unequal number of solved instances.

Table 5
Results comparison on Cordeau and Uber instances $\gamma = 0.4$

Cordeau instances	DA + LS algorithm	Three-index model[9]	Two-index model[9]
AT(s)	265.58	4547.79	1233.47
Gap	-	NA[a]	0.64%
Uber instances	**DA + LS algorithm**	**Three-index model[9]**	**Two-index model[9]**
AT(s)	324.84	5451.17	2169.25
Gap	-	NA[a]	0.57%

[a] NA indicates the gap cannot be calculated due to the unequal number of solved instances.

Conclusion:

- An adapted deterministic annealing meta-heuristic to tackle the Electric Autonomous Dial-A-Ride Problem (E-ADARP).
- Efficient operators to enhance the local search and a bi-directional insertion to insert recharging stations and determine recharging duration.
- Experiments on existing E-ADARP instances and new, larger E-ADARP instance.
- Comparing to best-known solutions, several new best solutions are found, the average gap of proposed algorithm is 0.58%.
- The effect of allowing multiple visits on the recharging stations has been investigated

Future work

The E-ADARP might be improved to take into account more real-life characteristics, such as time-dependent travel times. The objective functions may consider users' convenience with less waiting times which may conflicting with the global financial optimization. Operators of our DA meta-heuristic may be extended to consider additional constraints

Journée de l'ED Interfaces
11 February, ENSTA