International Journal for Traffic and Transport Engineering, 2022, 12(3): 340 - 351
DOI: http://dx.doi.org/10.7708/ijtte2022.12(3).04
UDC: 656.135:519.816]:004
629.326.2(44)

ADAPTING JSPRIT FOR THE ELECTRIC VEHICLE ROUTING PROBLEM WITH RECHARGING: IMPLEMENTATION AND BENCHMARK

Ayman Mahmoud[1], Tarek Chouaki[2], Sebastian Hörl[3], Jakob Puchinger[4]
[1,2,4] *Université Paris-Saclay, CentraleSupélec, Laboratoire Génie Industriel, 91190, Gif-sur-Yvette, France*
[2,3,4] *Institut de Recherche Technologique SystemX, Palaiseau 91120, France*

Received 16 July 2022; accepted 18 August 2022

Abstract: This article presents our adaptation of the Ruin-and-Recreate (R&R) algorithm to solve the electric vehicle routing problem with time windows and multiple trips. We implement this adaptation in JSprit, an open-source vehicle routing problem solver. We showcase the framework for a case study in Lyon, France. In the case study, we assess the efficiency impact of adding charging constraints to a simulation of a fleet of autonomous delivery robots. The framework is tested on benchmark instances and compared with results from literature.

Keywords: electric, vehicle, routing, JSprit, optimization.

1. Introduction

With the evolution of technology, policymakers are working towards making urban areas less polluted and carbon-free. One of the main factors affecting the quality of air and CO_2 emissions is freight flows. There is great interest in transforming the existing logistics fleet by integrating battery electrified vehicles (BEV). Especially, a significant potential is seen for last-mile logistics, where distances are short, and the flows of goods may be consolidated (Mucowska, 2021; Patella *et al.,* 2021)

Clearly, with electrification come operational challenges related to vehicle range and the necessity to recharge. Hence, a range of research endeavors is focused on solving problems associated with the specific characteristics of BEVs, such as the problem of locating recharging stations (Vosooghi *et al.,* 2020) or the electric vehicle routing problem (Erdoğan and Miller-Hooks, 2012; Schneider *et al.,* 2014).

For the latter class of problems, the present paper introduces distance constraints and recharging activities to JSprit, an open-source solver for common vehicle routing problems. The article is organized as follows; Section (2) presents the theoretical background and a literature review on existing use cases of JSprit, while Section (3) covers the implementational aspects of extending the framework. Our adaptations are exemplified in a case study for Lyon, France, in Section (4); benchmarking results are displayed in section (5), comparing the performance of the algorithm used in JSprit with results from (Schneider *et al.,* 2014). We conclude and discuss future work on the proposed approach in Section (6).

[1] Corresponding author: ayman.mahmoud@centralesupelec.fr

2. Background

Generally, Vehicle Routing Problems (VRP) aim to arrange a given set of transport jobs such as delivery pickups and drop-offs or depot visits into routes for a given (or variable) vehicle fleet. Algorithms solving VRPs decide which job to assign to which vehicle and how to temporally order the jobs along a vehicle's route. Furthermore, specific problem formulations can be derived by introducing constraints such as requiring that for a specific delivery, a pickup job comes before a drop-off job, that every vehicle can only cover a limited distance before returning to a depot, that certain jobs need to be performed within predefined time windows, and many more. The objective is often to minimize the total distance travelled, but more complex cost formulations exist considering the number of vehicles used or the time spent. (Vigo and Toth, 2014) provide an overview of VRP variants and their mathematical formulation.

2.1. Ruin & Recreate Heuristics

The Ruin & Recreate principle was first presented in (Schrimpf et al., 2000). The Ruin & Recreate heuristic is a large neighborhood search that combines elements of simulated annealing and threshold-accepting algorithms. Highlighting its performance for complex optimization problems with discontinuous search spaces and problems with many constraints, the researchers have tested their meta-heuristic on a wide range of VRP formulations.

The Ruin & Recreate heuristic follows three steps: First, an initial, feasible solution is constructed. Such a feasible solution is an ensemble of routes (sequences of jobs) depending on the VRP formulation for each vehicle that fulfils a predefined set of constraints. After that, multiple iterations of executing the Ruin & Recreate steps are performed. The Ruin step selects a set of jobs to ruin based on various existing strategies (Christiaens and Berghe, 2020; Schrimpf *et al.,* 2000). These jobs are then removed from the existing job sequences. Following the Ruin step, the Recreate step finds a new configuration by re-inserting some or all of the removed jobs into the schedules, and a feasible solution is obtained. The Ruin & Recreate heuristic yields a feasible solution to the problem after every iteration, but the quality may vary. For that purpose, a cost function is defined, which can be based, for instance, on the total distance or time driven by the vehicles. The cost of a route is calculated given the sequence of jobs and, usually, based on cost matrices indicating the cost of moving between their respective locations. Finally, based on threshold acceptance, the algorithm decides whether to keep the previous solution or to continue with the new one, temporarily allowing a worsened objective to find even better solutions in future steps.

2.2. The JSprit Framework

JSprit is an open-source toolkit written in Java that implements the Ruin & Recreate principle in a modular and extensible way (JSprit, 2022). It solves a wide range of VRP variants, including Capacitated VRP (CVRP), Multiple Depot VRP, VRP with Time Windows (VRP-TW), VRP with backhauls, VRP with Pickup and Delivery (VRP-PD), VRP with heterogeneous fleet, the Traveling Salesman Problem (TSP), and the Dial-a-Ride Problem (DARP).

The open-source solver has been used in a range of research activities. For instance,

(Villanueva, 2020) address real-life waste collection problems as part of the ODL (Open Door Logistics) framework, and. (Karkula et al., 2019) presents a comparative study on open-source tools for solving capacitated vehicle routing problems with time windows.

JSprit can be integrated directly with the agent- and activity-based transport simulation framework MATSim (Martins-Turner *et al.,* 2019) with applications on various topics. (Martins-Turner et al., 2020) use JSprit to generate tour plans for a case study of Berlin's urban last-mile supply of grocery stores. For the same urban context, (Schlenther et al., 2020) use the tool in a study on multi-use autonomous taxis to transport passengers and goods. (Bean and Joubert, 2019) present a study on receiver agents with autonomous decision-making in MATSim. Their generated logistics flows are optimized using JSprit.

JSprit has not yet been used to solve electric VRP instances with recharging. However, (Ewert et al., 2021) use the tool to solve parcel delivery problems with range constraints. They introduce a vehicle type-specific distance constraint and demonstrate their model on real data in retail food distribution. Electric energy consumption is based on travel distance and vehicle type. Contrary to the present work, no recharging stations are considered.

2.3. Contribution

The contribution of the present paper is the introduction of (1) a flexibly adjustable energy consumption model into JSprit, (2) range constraints, and (3) explicit charging activities at distinct locations (charging stations) to the implementation of Ruin & Recreate in JSprit. Furthermore, experimental results on a custom case study and comparison with benchmarks from the literature are presented.

3. Implementation

The implementation of the Ruin & Recreate heuristic in JSprit has been adjusted to consider energy consumption and recharging at distinct charging stations with specifiable consumption and charging patterns. For that purpose, three major components have been added to the framework: (1) functionality to calculate the state of charge (SoC) of a vehicle at any point along a route, (2) a new insertion strategy that considers charging stations, and (3) a Ruin strategy that is aware of charging constraints. These components are described in the following.

3.1. Calculating the State of Charge

Generally, the cost of a route in JSprit is derived from following the movements and jobs along a route and summing up costs. Such costs can come from a predefined matrix of routed distances, but they can also be context-aware, e.g., taking into account the current time of day or a vehicle's load at any point along the route. The latter can be expressed as a state variable in JSprit, which can be updated along a route.

To consider the battery power consumption, we introduce a new state variable to each vehicle, representing the remaining energy left in each vehicle's battery. Traversing each vehicle's route, we decrease the energy level following the formulation of energy consumption presented in (Schneider *et al.,* 2014), incorporating speed, gradients, and load.

Whenever a vehicle visits a charging station, battery power will be recharged. Spending

time at a depot that contains a charging station is different from visiting the charging station. The charging duration is calculated based on a predefined recharging rate *r* per vehicle type. Figure 1 shows a visualization of the SoC along a vehicle route with multiple stops at charging stations.

3.2. Inserting Charging Stations

JSprit already comes with strategies for inserting new jobs into existing vehicle routes. These insertion strategies are used to obtain an initial solution, where JSprit iteratively inserts all existing jobs into a set of initially empty vehicle routes. Later, during the Recreate phase, these strategies are used to re-insert jobs selected during the **Ruin** phase.

The insertion heuristics consider a set of constraints that need to be fulfilled. A range constraint has been added to work with electric vehicles, ensuring that no insertion is accepted that leads to a route with negative SoC at any point along the route. This constraint is checked during all job insertions to create the initial solution for the **Recreate** step.

The insertion strategies have been modified to facilitate the use of charging stations. Whenever a job is inserted, it is first tested whether the range constraint would be violated. If this is the case, the modified insertion strategy tries to insert a combination of a charging job and the planned job. The algorithm finds the closest charging stations based on the location of the preceding job. Then, it evaluates the global insertion cost after inserting the job and a charging activity using any of the closest stations at *any* point along the route. If any of the temporary configurations leads to a feasible solution that satisfies the acceptance threshold, the solution is accepted, and both the charging and the stop jobs are inserted. Otherwise, the proposed insertion is tagged infeasible, and the new job will need to be added at a different location proposed by the standard procedure.

To guide the insertion process, the set of relevant charging stations to choose from needs to be defined for every insertion. In each case they are sorted by the sum of distances to all the jobs in the route. By default, for instances with more than ten charging stations, we choose the 50% of the closest stations. This is done in order to avoid having to evaluate all the insertions for all charging stations and save computation time. However, we keep a minimum number of five evaluated charging stations in case sufficiently many are available.

3.3. Ruining Existing Solutions

By default, JSprit would use the existing Ruin strategies to remove activities along the route, including those for recharging. However, this may lead to infeasible solutions that cannot be repaired by the simple insertion process described above.

The existing Ruin strategies have been modified to counteract the emergence of infeasible solutions. Whenever a recharging activity is deleted from a route, it is first checked whether this would violate the vehicle's range constraint. If this is the case, the charging activity is left inside the route.

4. Case Study

We test the new functionality added to the JSprit framework on a case study for parcel deliveries in Lyon. The experiments are

inspired by existing studies that propose using delivery robots for sustainable last-mile logistics (Bakach *et al.,* 2021; Yu *et al.,* 2020).

A synthetic population for Lyon is created based on the open-source and open data methodology developed in (Hörl and Balac, 2021). Such a synthetic population is an individual-based artificial representation of the real population of the city, including sociodemographic attributes on the personal and household-level. Based on this sociodemographic and statistical information on the average number of parcel deliveries of household strata as reported by (Gardrat, 2019), a discrete number of parcels for an average day is generated for every household (Horl and Puchinger, 2022). For the case study, households with addresses in the Confluence peninsula South of the old town of Lyon, France, are chosen, leading to about *128* parcels to be delivered (Figure 1). Multiple delivery windows are defined for each parcel, based on the presence of household members at any time during the day.

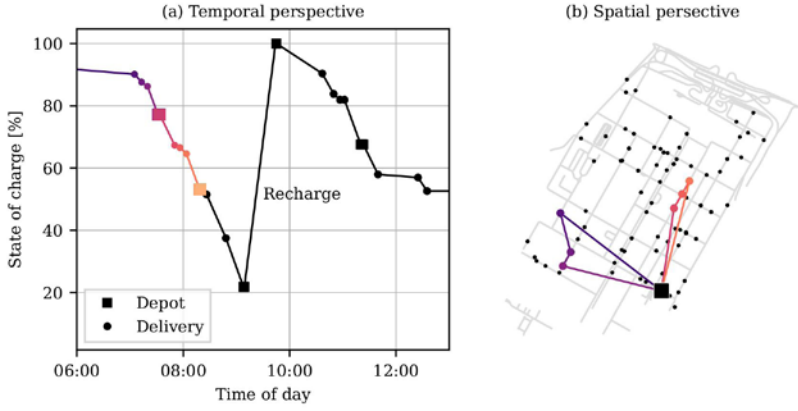

Fig. 1.
Temporal and Spatial Perspective
Example for, a temporal and spatial analysis of the SoC and the routes generated by the algorithm for the time between 06:00 at 13:00

In the test case, the parcels are delivered by automated electric delivery robots from an Urban Consolidation Center (UCC) located at the Eastern shore of the study area. Each vehicle can carry a maximum of four parcels at a time and perform multiple tours during the day. The objective is to minimize the total distance driven by the entire fleet for one day. Distances are calculated based on the shortest path through the road network, and travel times are derived using a speed of *3 km/h*. For each vehicle of the variable size fleet, a battery capacity of *8000mAh LiPo 18.5V* is assumed with consumption of *24.67 Wh/km* based on data obtained from the *Starship* delivery robot (Starship, 2022). The initial SoC is set to *100%*.

To summarize, we present an electric Multi-Trip Vehicle Routing Problem with Time Windows (e-MTVRPTW), demonstrating the interaction of existing functionality in

JSprit with our electric vehicle functionality including recharging at the depot.

The pickup and delivery problem is optimized using JSprit, with and without charging constraints. In each case, *1,000* iterations are performed. Furthermore, to test the sensitivity of the solutions, the experiments are repeated with *1.3, 1.6* and *two* times the baseline demand.

As an example of the detailed information derived from the JSprit runs, Figure 1a shows the SoC at and in between depots, delivery, and recharging activities for one vehicle route. Figure 1 shows the spatial trace of a vehicle. Aggregated metrics can be derived for each scenario using this information.

Table 1 shows the optimization results regarding problem size, the number of vehicles needed, the distance driven, and the number of charging activities. The last column indicates whether the problem has been solved with or without charging and range constraints. The results show that for the case without electric consumption constraints, four vehicles are needed for three demand levels; only for the highest demand scenario, five vehicles are necessary to deliver all parcels within the constraints given by the delivery time windows. In the eVRP case, the number of vehicles needed doubles in the solutions found by JSprit and even increases to *13* vehicles for the highest demand scenario. The results show that about every second vehicle must perform a charging activity in these scenarios. The results stay very similar in terms of distance, as the charging station is located directly at the depot.

Table 1

Optimization Results with Sensitivity Against Increased Demand

Scaling factor	1.0	1.3	1.6	2.0	eVRP
Parcels	128	165	198	256	
Vehicles	4	4	4	5	
	7	8	9	13	✓
Distance	55.19	63.26	71.74	95.23	
	59.29	62.21	71.89	90.49	✓
Charging activities	4	4	5	4	✓

5. Benchmarking Results

JSprit uses the Ruin & Recreate algorithm that has shown promising results in solving VRP variants (Schrimpf *et al.*, 2000). To assess the performance of our extension for electric vehicles, we can consider existing research common eVRP problems (Erdoğan and Miller-Hooks, 2012; Goeke, 2019; Montoya *et al.,* 2015; Schneider *et al.*, 2014). The research of (Erdoğan and Miller-Hooks, 2012) introduced the Green Vehicle Routing Problem (G-VRP), in which they consider a vehicle routing problem involving alternative fuel vehicles (AFVs) that have limited travel range and require recharge during routing. The results showed that the limitation of the driving range severely increased the total travelled distance, which takes into account the detours to the charging stations.

The work of (Erdoğan and Miller-Hooks, 2012) invited many researchers to work on the same instances and propose different heuristics with better performance and exact approaches. Most notably, (Schneider *et al.,* 2014) proposed a hybrid Variable Neighborhood Search/Tabu Search (VNS/TS) algorithm, which we use as a benchmark for our JSprit extension. Their approach includes four main steps. Step (1) begins the "Preprocessing Phase," in which the algorithm removes infeasible jobs. During the "Shaking Phase" (2), the VNS component finds a neighbouring solution to the problem. In the "Perturbation Phase" (3), the VNS component randomly cross-exchanges customer sequences between different routes. The final "Intensification Phase" (4) uses the TS to improve randomly generated solutions. The authors use the Simulated Annealing heuristic (SA) as their acceptance criterion, where deteriorating solutions are accepted with a certain probability.

The benchmark consists of four sets with ten instances each featuring 20 customers ("Small Instances"). The sets have different characteristics regarding the distribution of the clients (C: clustered, U: random) and the number of available fueling (charging) stations (S). In addition to the four sets, there are instances with 111 to 500 customers ("Large Instances").

The objective is to find the shortest distance with a limited number of vehicles while assuring all feasible customers are visited once, the tour length cannot exceed 11 hours, and the charging duration is fixed at 30 minutes. This section analyzes the first promising benchmark results.

Tables 2.1, 2.2, and 2.3 present the results of JSprit solving all small instances. We used JSprit to solve the small instances by setting the number of iterations to 2,000. The best solution found and the average running time of 10 executions are shown and compared with the results of the VNS/TS heuristic. In the tables, *#m* refers to the number of tours, *#c* refers to the number of served customers, and *%f* compares the objective values found by JSprit with VNS/TS. Only solutions which have a matching number of customers are valid to be compared (see Discussion).

For the small instances, JSprit found 30 matching solutions out of 40 instances, two better solutions than VNS/TS, and 8 solutions with worse objective values. For the instances (*S2_4i4s, S2_4i6s, S2_4i10s*) JSprit results are not matching the results found in Schneider, Stenger & Goeke 2014, but the (%f) is smaller than 0,01.

The large instances have been solved using JSprit by setting the number of iterations to 1,000 for the instances with 111 customers (Table 3), and to 500 for those between 200 and 500 customers (Table 4). For the instances with 111 customers, JSprit consistently requires on more tour than VNS/TS with the same number of unserved customers, while for the largest instances (Table 4), unserved customers are frequently offset against additional tours.

Table 2.1

Results of Ruin & Recreate Algorithm on the 20 Customers' G-Vrp Instances (U)

	VNS/TS			**R&R (JSprit)**			
	Best (mi)	# m	# c	Best (mi)	# m	# c	%f
20c3sU1	1797,49	6	20	1797,49	6	20	0,00
20c3sU2	1574,77	6	20	1574,77	6	20	0,00
20c3sU3	1704,48	7	20	1704,48	7	20	0,00
20c3sU4	1482,00	6	20	1482,00	6	20	0,00
20c3sU5	1689,37	5	20	1689,37	5	20	0,00
20c3sU6	1618,65	6	20	1618,65	6	20	0,00
20c3sU7	1713,66	6	20	1713,66	6	20	0,00
20c3sU8	1706,50	6	20	1706,50	6	20	0,00
20c3sU9	1708,81	6	20	1708,81	6	20	0,00
20c3sU10	1181,31	5	20	1181,31	5	20	0,00

Table 2.2

Results of Ruin & Recreate Algorithm on the 20 Customers' G-Vrp Instances (C)

	VNS/TS			**R&R (JSprit)**			
	Best (mi)	# m	# c	Best (mi)	# m	# c	%f
20c3sC1	1173,57	4	20	1173,57	4	20	0,00
20c3sC2	1539,97	5	19	1539,97	5	19	0,00
20c3sC3	880,20	3	12	880,20	3	12	0,00
20c3sC4	1059,35	4	18	1108,46	4	18	4,43
20c3sC5	2156,01	7	19	2179,42	7	19	1,07
20c3sC6	2758,17	8	17	2758,17	8	17	0,00
20c3sC7	1393,99	4	6	1393,99	4	6	0,00
20c3sC8	3139,72	9	18	3139,72	9	18	0,00
20c3sC9	1799,94	6	19	1808,59	6	19	0,48
20c3sC10	2583,42	8	15	2583,42	8	15	0,00

Table 2.3

Results of Ruin & Recreate Algorithm on the 20 Customers' G-Vrp Instances (S)

	VNS/TS			**R&R (JSprit)**			
	Best (mi)	# m	# c	Best (mi)	# m	# c	%f
S1_2i6s	1578,12	6	20	1578,12	6	20	0,00
S1_4i6s	1397,27	5	20	1397,27	5	20	0,00
S1_6i6s	1560,49	5	20	1571,29	5	20	0,69
S1_8i6s	1692,32	6	20	1692,32	6	20	0,00
S1_10i6s	1173,48	4	20	1173,48	4	20	0,00
S2_2i6s	1633,1	6	20	1645,8	6	20	0,78
S2_4i6s	1532,96	5	19	1505,06	6	19	-1,82
S2_6i6s	2431,33	7	20	2431,33	7	20	0,00
S2_8i6s	2158,35	7	16	2175,65	7	16	0,80
S2_10i6s	1958,46	6	17	1792,45	6	17	-9,26
S1_4i2s	1582,2	6	20	1582,2	6	20	0,00
S1_4i4s	1460,09	5	20	1460,09	5	20	0,00
S1_4i6s	1397,27	5	20	1397,27	5	20	0,00
S1_4i8s	1397,27	6	20	1397,27	6	20	0,00
S1_4i10s	1396,02	5	20	1396,02	5	20	0,00
S2_4i2s	1059,35	4	18	1106,29	4	18	4,24
S2_4i4s	1446,08	5	19	1446,18	5	19	0,00*
S2_4i6s	1434,14	5	20	1434,23	5	20	0,00*
S2_4i8s	1434,14	5	20	1484,61	6	20	3,52
S2_4i10s	1434,13	5	20	1434,23	5	20	0,00*

Table 3

Results of Ruin & Recreate Algorithm on the 111 Customers' G-Vrp Instances

	VNS/TS			R&R (JSprit)			
	Best (mi)	# m	# c	Best (mi)	# m	# c	%f
111c_21s	4797,15	17	109	4910,76	17	109	2,36
111c_22s	4802,16	17	109	5167,42	18	109	7,60
111c_24s	4786,96	17	109	4965,51	18	109	3,73
111c_26s	4778,62	17	109	4985,56	18	109	4,33
111c_28s	4799,15	17	109	5101,43	18	109	6,30

Table 4

Results of Ruin & Recreate Algorithm on the (200 – 500) Customers' G-Vrp Instances

	VNS/TS			R&R (JSprit)			
	Best (mi)	# m	# c	Best (mi)	# m	# c	%f
200c_21s	8963,46	35	192	9506,93	32	191	-
250c_21s	10800,18	39	237	12007,94	41	236	-
300c_21s	12594,77	46	283	14469,29	50	282	-
350c_21s	14323,02	51	329	16156,95	55	328	-
400c_21s	16850,21	61	378	19330,38	65	376	-
450c_21s	18521,23	68	424	22717,95	78	423	-
500c_21s	21170,9	76	471	25446,35	85	469	-

As the tables show, JSprit presents promising solutions for the small instances. Additionally, the extension shows a better performance than the VNS/TS approach in terms of execution time (for the defined number of iterations). For instance, for the "U" instances, our extension was able to match the solutions found with VNS/TS with an average execution time of 9 seconds versus 39 seconds. For the larger instances, we expect that increasing the number of iterations would lead to solutions closer to the best solutions known in the literature, resulting in longer execution times.

6. Discussion

The benchmark study was subject to a sensitivity analysis, and we tried different variations of ruin strategies (Random, Worst, String, Radial, Cluster). We identified that the best ruin strategy is to randomly select between the different approaches at each iteration of the algorithm. There is no preprocessing phase in our implementation, the algorithm tries to assign all customers in an instance. Instead of having a hard constraint to visit all customers, our algorithm adds a penalty when a customer is not visited, in consequence, we cannot ensure that all clients are served. This is the case of our results in Table 4. The fact that not all customers are visited in all instances is largely related to our choice of the penalty value.

Several design decisions have been taken when implementing charging activities into JSprit, mainly regarding the Ruin strategy. As is standard in the literature on Ruin & Recreate, the algorithm maintains feasible solutions after each step. Accordingly, the existing JSprit modules are implemented such that this condition holds implicitly after applying a Ruin strategy. No additional checks for constraint violations are performed after. However, the energy consumption constraint may be violated when removing a recharging activity from a vehicle's route.

We have considered two options to solve the issue: As one option, a Recreate strategy may have been introduced that can repair

any ruined solution. While it would be the more flexible option in the conceptual structure of the Ruin & Recreate approach, it would require inserting charging activities independently of the jobs, which, in turn, would lead to considerable changes in the core implementation of JSprit, where insertions are job-based.

Hence, our chosen approach is to introduce customized versions of the existing Ruin strategies to avoid creating ruined solutions with constraint violations. This approach has the advantage of being less invasive to the existing code base and following the implicit policy of existing Ruin strategies. However, the approach prevents the removal of recharging activities that are relevant for the ruined route but not necessarily for a follow-up solution. This adds more complexity to the algorithm and may also restrict the search space or slow down convergence. The topic provides potential for future research.

For now, our implementation has been tested with up to 500 customers. Beyond that limit, the run time of the extensive search for potential charging activities along a route becomes prohibitive. To remedy the problem, more efficient strategies for inserting charging activities may be introduced in the future (Kullman et al., 2021). Furthermore, we can build on multi-threading functionality of JSprit to gain efficiency. While it is available for standard problem formulations, it has not yet been adapted to be compatible with our extensions.

7. Conclusion

In this paper, we present an extension for the open-source VRP solver JSprit that considers electric vehicles including energy consumption, charging constraints and charging infrastructure. While the performance of the JSprit implementation is inferior to existing benchmarks from literature on the Green Vehicle Routing Problem, we can show that it frequently reproduces best-known solutions for small-scale problems. To our knowledge, it is the first time that the problem has been solved using the Ruin & Recreate approach which is the basis of JSprit.

The main advantage of our approach is that it is based on an open-source and extensible framework that already covers a wide range of VRP variants. For the future, we propose methodological improvements in terms of varying search heuristics, and speed-ups through parallelization.

Acknowledgements

The code related to the experiments presented in this paper can be obtained from the authors, and the detailed results with the sensitivity analysis can be shared upon request.

This paper presents work developed at IRT SystemX in the scope of the project LEAD, which has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement no. 861598. **This work has also received funding from the Région Île-de-France within the frame of the Future Cities Lab**. The content of this paper does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in this paper lies entirely with the authors.

References

Bakach, I.; Campbell, A.M.; Ehmke, J.F. 2021. A two-tier urban delivery network with robot-based deliveries, *Networks* 78(4): 461-483.

Bean, W.L.; Joubert, J.W. 2019. Modelling receiver logistics behaviour. In *Procedia Computer Science, The 10th International Conference on Ambient Systems, Networks and Technologies (ANT 2019) / The 2nd International Conference on Emerging Data and Industry 4.0 (EDI40 2019) / Affiliated Workshops* 151: 763–768.

Christiaens, J.; Berghe, G.V. 2020. Slack Induction by String Removals for Vehicle Routing Problems, *Transportation Science* 54(2): 417–433.

Erdoğan, S.; Miller-Hooks, E. 2012. A Green Vehicle Routing Problem, *Transportation Research Part E: Logistics and Transportation Review, Select Papers from the 19th International Symposium on Transportation and Traffic Theory* 48(1): 100–114.

Ewert, R.; Martins-Turner, K.; Thaller, C.; Nagel, K. 2021. Using a route-based and vehicle type specific range constraint for improving vehicle routing problems with electric vehicles, *Transportation Research Procedia* 52: 517-524.

Gardrat, M. 2019. Survey methodology: the decoupling of household purchases and recovery of goods [In French: Méthodologie d'enquête: le découplage de l'achat et de la récupération des marchandises par les ménages] [Rapport de recherche] LAET (Lyon, France); Métropole de Lyon. 114p.

Goeke, D. 2019. Granular tabu search for the pickup and delivery problem with time windows and electric vehicles, *European Journal of Operational Research* 278(3): 821–836.

Hörl, S.; Balac, M. 2021. Synthetic population and travel demand for Paris and Île-de-France based on open and publicly available data, *Transportation Research Part C: Emerging Technologies* 130: 103291.

Horl, S.; Puchinger, J. 2022. From synthetic population to parcel demand: A modeling pipeline and case study for last-mile deliveries in Lyon. *Transportation Reaseach Arena (TRA),* Lisbon, Portugal.

JSprit. 2022. A Java based, open-source toolkit for solving rich traveling salesman (TSP) and vehicle routing problems (VRP). Available from Internet: <https://jsprit.github.io/>.

Karkula, M.; et al. 2019. Comparison of capabilities of recent open-source tools for solving capacitated vehicle routing problems with time windows, *Carpathian Logistics Conference,* Zakopane, Poland, 72-77.

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.

Martins-Turner, K., *et al.* 2019. Agent-based Modelling and Simulation of Tour Planning in Urban Freight Traffic. *Transportation Research Procedia, Urban Mobility – Shaping the Future Together mobil.TUM 2018 – International Scientific Conference on Mobility and Transport Conference Proceedings* 41: 328–332.

Martins-Turner, K., *et al.* 2020. Electrification of Urban Freight Transport - a Case Study of the Food Retailing Industry. *Procedia Computer Science, The 11th International Conference on Ambient Systems, Networks and Technologies (ANT) / The 3rd International Conference on Emerging Data and Industry 4.0 (EDI40) / Affiliated Workshops* 170: 757–763.

Montoya, A. *et al.* 2015. *The electric vehicle routing problem with partial charging and nonlinear charging function (Research Report).* LARIS. 11p.

Mucowska, M. 2021. Trends of Environmentally Sustainable Solutions of Urban Last-Mile Deliveries on the E-Commerce Market—A Literature Review, *Sustainability* 13(11): 5894.

Patella, S.M.; Grazieschi, G.; Gatta, V.; Marcucci, E.; Carrese, S. 2020. The adoption of green vehicles in last mile logistics: A systematic review, *Sustainability* 13(1): 6.

Schlenther, T.; Martins-Turner, K.; Bischoff, J.F.; Nagel, K. 2020. Potential of private autonomous vehicles for parcel delivery, *Transportation Research Record* 2674(11): 520-531.

Schneider, M.; Stenger, A.; Goeke, D. 2014. The electric vehicle-routing problem with time windows and recharging stations, *Transportation Science* 48(4): 500-520.

Schrimpf, G.; Schneider, J.; Stamm-Wilbrandt, H.; Dueck, G. 2000. Record breaking optimization results using the ruin and recreate principle, *Journal of Computational Physics* 159(2): 139-171.

Starship. 2022. Available from Internet: <https://www.starship.xyz/>.

Vigo D.; Toth, P. 2014. *Vehicle Routing: Problems, Methods, and Application, Second Edition*. Volume 18 of MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics. 481p.

Villanueva, R.S. 2020. A pragmatic approach to improve the efficiency of the waste management system in Stockholm through the use of Big Data, Heuristics and open source VRP solvers: A real life waste collection problem; Stockholm's waste collection system and inherent vehicle routing problem, VRP. *Degree project in electrical engineering, second cycle, KTH.*

Vosooghi, R.; Puchinger, J.; Bischoff, J.; Jankovic, M.; Vouillon, A. 2020. Shared autonomous electric vehicle service performance: Assessing the impact of charging infrastructure, *Transportation Research Part D: Transport and Environment* 81: 102283.

Yu, S.; Puchinger, J.; Sun, S. 2020. Two-echelon urban deliveries using autonomous vehicles, *Transportation Research Part E: Logistics and Transportation Review* 141: 102018.