




# VRPy: A Python package for solving a range of vehicle routing problems with a column generation approach

Romain Montagné<sup>1</sup>, David Torres Sanchez<sup>2</sup>, and Halvard Olsen Storbugt<sup>2</sup>

<sup>1</sup> EURODECISION <sup>2</sup> SINTEF Digital, Mathematics and Cybernetics

DOI: [10.21105/joss.02408](https://doi.org/10.21105/joss.02408)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

---

Editor: [Kakia Chatsiou](#) 

## Reviewers:

- [@bstabler](#)
- [@skadio](#)

Submitted: 10 June 2020

Published: 09 November 2020

## License

Authors of papers retain copyright and release the work under a Creative Commons Attribution 4.0 International License ([CC BY 4.0](#)).

## Introduction

The Vehicle Routing Problem (VRP) is amongst the most well known combinatorial optimization problems. The most classical version of the VRP, the Capacitated VRP (CVRP) (Laporte, 2007), can be described as follows. A fleet of vehicles with uniform capacity must serve customers with known demand for a single commodity. The vehicles start and end their routes at a common depot and each customer must be served by exactly one vehicle. The objective is to assign a sequence of customers to each vehicle of the fleet (a route), minimizing the total distance traveled, such that all customers are served and the total demand served by each vehicle does not exceed its capacity. Note that the VRP generalises the well-known traveling salesman problem (TSP) and is therefore computationally intractable.

Mathematicians have started tackling VRPs since 1959 (Dantzig & Ramser, 1959). Ever since, algorithms and computational power have not stopped improving. State of the art techniques include column generation approaches (Bramel & Simchi-Levi, 1997; Costa et al., 2019) on which `vrpy` relies; more details are given hereafter.

`vrpy` is of interest to the operational research community and others (e.g., logisticians, supply chain analysts) who wish to solve vehicle routing problems, and therefore has many obvious applications in industry.

## Features

`vrpy` is a Python package that offers an easy-to-use, unified API for many variants of vehicle routing problems including:

- the Capacitated VRP (CVRP) (Baldacci et al., 2010; Laporte, 2007),
- the CVRP with resource constraints (Laporte et al., 1985),
- the CVRP with time windows (Cordeau & Québec, 2000),
- the CVRP with simultaneous distribution and collection (Dell'Amico et al., 2006),
- the CVRP with pickups and deliveries (Desrosiers & Dumas, 1988),
- the CVRP with heterogeneous fleet (Choi & Tcha, 2007).

For each of these variants, it is possible to i/ set initial routes for the search (if one already has a solution at hand and wishes to improve it) ii/ lock routes (if part of the solution is imposed and must not be optimized) iii/ drop nodes (ignore a customer at the cost of a penalty).

`vrpy` is built upon the well known *NetworkX* library (Hagberg et al., 2008) and thus benefits from a user friendly API, as shown in the following quick start example:

```
from networkx import DiGraph
from vrpy import VehicleRoutingProblem

# Define the network
G = DiGraph()
G.add_edge("Source",1,cost=1,time=2)
G.add_edge("Source",2,cost=2,time=1)
G.add_edge(1,"Sink",cost=0,time=2)
G.add_edge(2,"Sink",cost=2,time=3)
G.add_edge(1,2,cost=1,time=1)
G.add_edge(2,1,cost=1,time=1)

# Define the customers demands
G.nodes[1]["demand"] = 5
G.nodes[2]["demand"] = 4

# Define the Vehicle Routing Problem
prob = VehicleRoutingProblem(G, load_capacity=10, duration=5)

# Solve and display solution value
prob.solve()
print(prob.best_value)
3
print(prob.best_routes)
{1: ["Source",2,1,"Sink"]}
```

## State of the field

Although the VRP is a classical optimization problem, to our knowledge there is only one dedicated package in the Python ecosystem that is able to solve such a range of VRP variants: the excellent OR-Tools (Google) routing library (Perron & Furnon, 2019), released for the first time in 2014. To be precise, the core algorithms are implemented in C++, but the library provides a wrapper in Python. Popular and efficient, it is a reference for `vrpy`, both in terms of features and performance. The current version of `vrpy` is able to handle the same variants as OR-Tools (mentioned in the previous section).

Performance-wise, `vrpy` ambitions to be competitive with OR-Tools eventually, at least in terms of solution quality. For the moment, benchmarks (available in the repository) for the CVRP on the set of Augerat instances (Augerat, 1995) show promising results: in the performance profile in Figure 1 below, one can see that nearly the same number of instances are solved within 10 seconds with the same relative error with respect to the best known solution (42% for `vrpy`, 44% for OR-Tools).

CVRP (Augerat instances) - Performance profile with a maximum run time of 10 sec

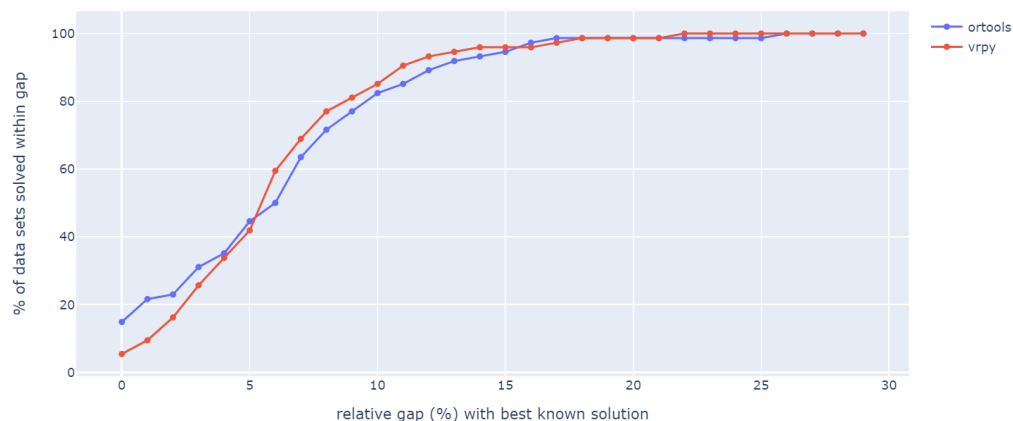


Figure 1: CVRP Performance profile

We do not claim to outperform OR-Tools, but aim to have results of the same order of magnitude as the package evolves, as there is still much room for improvement (see Section *Future Work* below). On the other hand, we are confident that the user friendly and intuitive API will help students, researchers and more generally the operational research community solve instances of vehicle routing problems of small to medium size, perhaps more easily than with the existing software.

py-ga-VRPTW is another library that is available but as mentioned by its authors, it is more of an experimental project and its performances are rather poor. In particular, we were not able to find feasible solutions for Solomon's instances (Solomon, 1987) and therefore cannot compare the two libraries. Also note that py-ga-VRPTW is designed to solve the VRPTW only, that is, the VRP with time windows.

## Mathematical background

vrpy solves vehicle routing problems with a column generation approach. The term *column generation* refers to the fact that iteratively, routes (or columns) are generated with a pricing problem, and fed to a master problem which selects the best routes among a pool such that each vertex is serviced exactly once. Results from the master problem are then used to search for new potential routes likely to improve the solution's cost, and so forth. This procedure is illustrated in Figure 2 below:

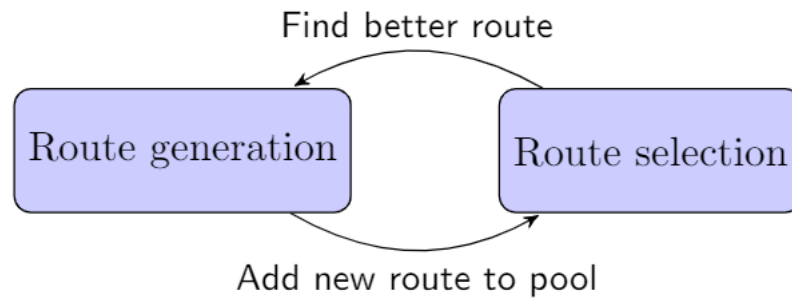


Figure 2: Column Generation

The master problem is a set partitioning linear formulation and is solved with the open source solver Clp from COIN-OR (Forrest et al., 2020), while the subproblem is a shortest elementary path problem with *resource constraints*. It is solved with the help of the cspy library (Torres Sanchez, 2020) which is specifically designed for such problems.

This column generation procedure is very generic, as for each of the featuring VRP variants, the master problem is identical and partitions the customers into subsets (routes). It is the subproblem (or pricing problem) that differs from one variant to another. More specifically, each variant has its unique set of *resources* which must remain in a given interval. For example, for the CVRP, a resource representing the vehicle's load is carried along the path and must not exceed the vehicle capacity; for the CVRP with time windows, two extra resources must be considered: the first one for time, and the second one for time window feasibility. The reader may refer to (Costa et al., 2019) for more details on each of these variants and how they are dealt with within the framework of column generation.

Note that `vrpy` does not necessarily return an optimal solution. Indeed, once the pricing problems fails to find a route with negative marginal cost, the master problem is solved as a MIP. This *price-and-branch* strategy does not guarantee optimality. Note however that it can be shown (Bramel & Simchi-Levi, 1997) that asymptotically, the relative error goes to zero as the number of customers increases. To guarantee that an optimal solution is returned, the column generation procedure should be embedded in a branch-and-bound scheme (*branch-and-price*), which is beyond the scope of the current release, but part of the future work considered.

## Advanced Features

For more advanced users, there are different pricing strategies (approaches for solving sub-problems), namely sparsification strategies (Dell'Amico et al., 2006; Santini et al., 2018), as well as pre-pricing heuristics available that can lead to faster solutions. The heuristics implemented include a greedy randomized heuristic (for the CVRP and the CVRP with resource constraints) (Santini et al., 2018). Also, a diving heuristic (Sadykov et al., 2019) can be called to explore part of the branch-and-price tree, instead of solving the restricted master problem as a MIP.

Additionally, we have an experimental feature that uses Hyper-Heuristics for the dynamic selection of pricing strategies. The approach ranks the best pricing strategies as the algorithm is running and chooses according to selection functions based on (Ferreira et al., 2017; Sabar et al., 2015). The selection criteria has been modified to include a combination of runtime, objective improvement, and currently active columns in the restricted master problem. Adaptive

parameter settings found in (Drake et al., 2012) is used to balance exploration and exploitation under stagnation. The main advantage is that selection is done as the program runs, and is therefore more flexible compared to a predefined pricing strategy.

## Future Work

There are many ways `vrpy` could be improved. To boost the run times, specific heuristics for each variant could be implemented, e.g., Solomon's insertion algorithm (Solomon, 1987) for the VRPTW. Second, the pricing problem is solved with `cspy`, which is quite recent (2019) and is still being fine tuned. Also, currently, stabilization issues are dealt with a basic interior point based strategy which could be enhanced (Pessoa et al., 2018). Last but not least, there are many cutting strategies in the literature (Costa et al., 2019) that have not been implemented and which have proven to significantly reduce run times for such problems.

## Acknowledgements

We would like to thank reviewers Ben Stabler and Serdar Kadioglu for their helpful and constructive suggestions.

## References

- Augerat, P. (1995). *Approche polyédrale du problème de tournées de véhicules* [PhD thesis]. Institut National Polytechnique de Grenoble-INPG.
- Baldacci, R., Toth, P., & Vigo, D. (2010). Exact algorithms for routing problems under vehicle capacity constraints. *Annals of Operations Research*, 175(1), 213–245. <https://doi.org/10.1007/s10479-009-0650-0>
- Bramel, J., & Simchi-Levi, D. (1997). Solving the vrp using a column generation approach. In *The logic of logistics* (pp. 125–141). Springer. [https://doi.org/10.1007/0-387-22619-2\\_16](https://doi.org/10.1007/0-387-22619-2_16)
- Choi, E., & Tcha, D.-W. (2007). A column generation approach to the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34(7), 2080–2095. <https://doi.org/10.1016/j.cor.2005.08.002>
- Cordeau, J.-F., & Québec, G. d'études et de recherche en analyse des décisions (Montréal. (2000). *The vrp with time windows*. Groupe d'études et de recherche en analyse des décisions Montréal. <https://pdfs.semanticscholar.org/3aaa/a16ab53cf30c378fdb7c911fe0de39ee8997.pdf>
- Costa, L., Contardo, C., & Desaulniers, G. (2019). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, 53(4), 946–985. <https://doi.org/10.1287/trsc.2018.0878>
- Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, 6(1), 80–91. <https://doi.org/10.1287/mnsc.6.1.80>
- Dell'Amico, M., Righini, G., & Salani, M. (2006). A branch-and-price approach to the vehicle routing problem with simultaneous distribution and collection. *Transportation Science*, 40(2), 235–247. <https://doi.org/10.1287/trsc.1050.0118>
- Desrosiers, J., & Dumas, Y. (1988). The shortest path problem for the construction of vehicle routes with pick-up, delivery and time constraints. In *Advances in optimization and control* (pp. 144–157). Springer. [https://doi.org/10.1007/978-3-642-46629-8\\_10](https://doi.org/10.1007/978-3-642-46629-8_10)

- Drake, J. H., Özcan, E., & Burke, E. K. (2012). An improved choice function heuristic selection for cross domain heuristic search. *International Conference on Parallel Problem Solving from Nature*, 307–316. [https://doi.org/10.1007/978-3-642-32964-7\\_31](https://doi.org/10.1007/978-3-642-32964-7_31)
- Ferreira, A. S., Gonçalves, R. A., & Pozo, A. (2017). A multi-armed bandit selection strategy for hyper-heuristics. *2017 IEEE Congress on Evolutionary Computation (Cec)*, 525–532. <https://doi.org/10.1109/CEC.2017.7969356>
- Forrest, J. J., Vigerske, S., Ralphs, T., Hafer, L., jpfasano, Santos, H. G., Saltzman, M., h-i-gassmann, Kristjansson, B., & King, A. (2020). *Coin-or/clp: Version 1.17.6* (releases/1.17.6) [Computer software]. Zenodo. <https://doi.org/10.5281/zenodo.3748677>
- Hagberg, A., Swart, P., & S Chult, D. (2008). *Exploring network structure, dynamics, and function using networkx*. Los Alamos National Lab.(LANL), Los Alamos, NM (United States). [https://conference.scipy.org/proceedings/scipy2008/paper\\_2/full\\_text.pdf](https://conference.scipy.org/proceedings/scipy2008/paper_2/full_text.pdf)
- Laporte, G. (2007). What you should know about the vehicle routing problem. *Naval Research Logistics (NRL)*, 54(8), 811–819. <https://doi.org/10.1002/nav.20261>
- Laporte, G., Nobert, Y., & Desrochers, M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5), 1050–1073. <https://doi.org/10.1287/opre.33.5.1050>
- Perron, L., & Furnon, V. (2019). *OR-Tools* (Version 7.2). Google. <https://developers.google.com/optimization/>
- Pessoa, A., Sadykov, R., Uchoa, E., & Vanderbeck, F. (2018). Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2), 339–360. <https://doi.org/10.1287/ijoc.2017.0784>
- Sabar, N. R., Zhang, X. J., & Song, A. (2015). A math-hyper-heuristic approach for large-scale vehicle routing problems with time windows. *2015 IEEE Congress on Evolutionary Computation (Cec)*, 830–837. <https://doi.org/10.1109/CEC.2015.7256977>
- Sadykov, R., Vanderbeck, F., Pessoa, A., Tahiri, I., & Uchoa, E. (2019). Primal heuristics for branch and price: The assets of diving methods. *INFORMS Journal on Computing*, 31(2), 251–267. <https://doi.org/10.1287/ijoc.2018.0822>
- Santini, A., Plum, C. E., & Ropke, S. (2018). A branch-and-price approach to the feeder network design problem. *European Journal of Operational Research*, 264(2), 607–622. <https://doi.org/10.1016/j.ejor.2017.06.063>
- Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2), 254–265. <https://doi.org/10.1287/opre.35.2.254>
- Torres Sanchez, D. (2020). Cspy: A python package with a collection of algorithms for the (resource) constrained shortest path problem. *Journal of Open Source Software*, 5(49), 1655. <https://doi.org/10.21105/joss.01655>