OR-Tools' Vehicle Routing Solver: a Generic Constraint-Programming Solver with Heuristic Search for Routing Problems

Thibaut Cuvelier, Frederic Didier, Vincent Furnon, Steven Gay, Sarah Mohajeri, Laurent Perron
Google Paris, France
{tcuvelier,fdid,vfurnon,stevengay,mohajeri,lperron}@google.com

Keywords: *vehicle-routing problems, constraint programming, heuristics, metaheuristics.*

1 Introduction

OR-Tools [1] is the general-purpose optimisation toolbox open-sourced by Google in 2015[1], being in development since 2008. This toolkit provides a uniform interface to several solvers, both first- and third-party. In particular, it offers a high-level interface for vehicle-routing problems (VRPs). OR-Tools contains several solvers, in particular two CP solvers, CP* (since the first open-source release) and CP-SAT (gold-medal winner at several MiniZinc competitions [2], developed since 2009), , but also two linear solvers: the simplex-based Glop (since 2014[2]), and PDLP [3], a first-order large-scale linear solver. OR-Tools is being actively developed, with approximately quarterly releases. Outside Google, the solver suite is easily accessible via Google Cloud, either for solving VRPs[3] or mixed-integer linear programs, although the latter API is not yet in general access[4].

The routing component has historically played a strong role in the development of the overall solver; its major focus is on solving large-scale industrial vehicle-routing problems with complex constraints: vehicle capacities with various starting/ending depots, client time windows considering road traffic and driver breaks, pick-up-and-delivery precedence rules, incompatible shipments within the same vehicle, solution similarity to a previous call to the solver, etc. To this end, a high-level modelling API is proposed to the users in Python, C++, Java, and C#, using only routing concepts, even though the user has access to the underlying constraint-programming model.

From an algorithmic point of view, the routing solver is organised in three parts: (i) first-solution heuristics generate good potential vehicle tours; (ii) local search improves the first solutions, with metaheuristics to guide the search; (iii) a CP engine proves the optimality of the best solution or improves upon it. The main difference with many academic solvers is the focus on generality in the solver, including its heuristics.

[1]https://groups.google.com/g/or-tools-discuss/c/OMDgOZE7z 4/m/hkwZaKTRla4J

[2]https://ai.googleblog.com/2014/09/sudoku-linear-optimization-and-ten-cent.html

[3]https://cloud.google.com/optimization/docs/overview

[4]https://aihub.cloud.google.com/u/0/p/products%2F03a54ca4-f9ba-489b-bbb3-b6ca8c22c5cf

2 Modelling vehicle-routing problems

VRP-like models can be directly entered using the routing modelling API, a high-level interface dedicated to this family of problems. The solver ingests the distance-weighted directed graph through a callback function (`RoutingModel:: RegisterTransitCallback`). Vehicles are represented by a set of optimisation variables (`RoutingModel::Start` and `RoutingModel::Next`). The solver represents quantities that accumulate along a path, like distance, time, or vehicle load, with `RoutingDimension` and `CumulVar`. This mechanism is used to model time windows.

Some specific problems have a dedicated interface, like pickup and delivery, where the same vehicle must first pickup a parcel on its route before delivering it (`RoutingModel::AddPickupAndDelivery`).

Arc-routing problems are not supported out of the box, but users can perform the transformation to usual vehicle-routing problems by representing arcs to visit by nodes [4]. Edge routing corresponds to arc routing, the only difference being that traversing each edge once (in either direction) is sufficient: the distinction can be modelled using a disjunction (`RoutingModel::AddDisjunction`).

3 Solving vehicle-routing problems

OR-Tools provides several kinds of first-solution heuristics, like savings or a greedy, cheapest-arc algorithm. The next step of the solver is to call iterative improvement techniques to refine the first solutions, using heuristic operators. Many such algorithms have been implemented for VRP-like problems[5]. Their goal is to alter the first solutions slightly to improve them by exploring the neighbourhood they define. This process generates many candidate solutions (typically, at least hundreds of thousands per second): not all of them are feasible or improve the solution; a filtering system ensures that only feasible solutions are further considered.

To continue improving the solution when basic local-search heuristics cannot make any more progress, a metaheuristic is used. OR-Tools implements simulated annealing, tabu search, and guided local search (GLS). The end of the search with OR-Tools is performed with a CP solver: either CP* or CP-SAT.

References

[1] OR-Tools. URL: https://github.com/google/or-tools/

[2] P. J. Stuckey, T. Feydy, A. Schutt, G. Tack, J. Fischer. *The MiniZinc Challenge.* AI Magazine, volume 35, issue 2, 2014.

[3] David Applegate, Mateo Díaz, Oliver Hinder, Haihao Lu, Miles Lubin, Brendan O'Donoghue, Warren Schudy. *Practical large-scale linear programming using primal-dual hybrid gradient.* Advances in Neural Information Processing Systems, 2021.

[4] H. Longo, M. Poggi de Aragão, E. Uchoa. *Solving capacitated arc routing problems using a transformation to the CVRP.* Computers & Operations Research, volume 33, issue 6, 2006.

[5] 5https://github.com/google/or-tools/blob/v9.2/ortools/constraint solver/routing parameters.proto#L115-L370