Adaptive large neighborhood search for multicommodity VRP

Wenjuan Gu, Diego Cattaruzza, Maxime Ogier, Frédéric Semet

Adaptive large neighborhood search for multicommodity VRP

Wenjuan Gu, Diego Cattaruzza, Maxime Ogier, Frédéric Semet
Univ. Lille, CNRS, Centrale Lille, Inria, UMR 9189 - CRIStAL
F-59000 Lille, France
Email: maxime.ogier@centralelille.fr

1 Introduction

In this work we study a vehicle routing problem where customers request multiple commodities. Different strategies to deliver a set of commodities to customers were presented in [1]. Among these strategies, a new one, the *commodity-constrained split-delivery mixed routing problem* (C-SDVRP) is presented and compared with classical ways to deliver multiple products (allowing to split a commodity or using vehicles dedicated to each commodity). In the C-SDVRP, the vehicles are flexible and can deliver any set of commodities, and a customer who requests multiple commodities can be delivered by different vehicles. When a commodity is delivered to a customer, the entire required amount is handed over. This, if the customer is visited more than once, the different vehicles will deliver different sets of commodities.

This problem arises for example in delivery of fresh fruits and vegetables to catering. In this case, products can easily be mixed into the same vehicle, and splitting the delivery of an individual commodity is not acceptable. This is also very relevant in the multi-depot case, where each each depot has a limited quantity of each commodity. A customer can then be delivered by vehicles coming from different depots.

The C-SDVRP has first been studied in [1]. The authors proposed a branch-and-cut algorithm able to solve 25 out of 64 small instances (15 customers) to optimality within 30 minutes, and a heuristic method. The heuristic consists in (1) making copies of each customer, one for each commodity required by the customer, and (2) using a heuristic for the capacitated VRP. In [2], the authors proposed an extended formulation for the C-SDVRP and developed a branch-price-and-cut algorithm. They solved to optimality instances with up to 40 customers and 3 commodities per customer within 2 hours.

This work proposes an adaptive large neighborhood search to solve the C-SDVRP, with the objective of obtaining very good solutions on small instances, and to be able to efficiently solve large instances.

2 Problem definition

The C-SDVRP can be defined based on a directed graph $G = (\mathcal{V}, \mathcal{A})$ in which $\mathcal{V} = \{0\} \bigcup \mathcal{V}_C$ is the set of vertices, and $\mathcal{A}$ is the set of arcs. More precisely, $\mathcal{V}_C = \{1, ..., N_C\}$ represents the set of customer vertices, and 0 is the depot. A cost c_{ij} is associated with each arc $(i, j) \in \mathcal{A}$ and represents the non-negative cost of travelling from i to j. Let $\mathcal{M}$ be the set of commodities that have to be delivered to the customers. Any customer $i \in \mathcal{V}_C$ may request any set of commodities. The depot contains a fleet of identical vehicles with capacity Q, able to deliver any subset of commodities. The objective is to minimize the total travelling cost.

The problem involves two related decisions: (1) finding a set of vehicle routes serving all customers; (2) selecting commodities delivered to each customer. The constraints are: (1) each route starts and ends at the depot; (2) the total quantity of commodities delivered by each vehicle does not exceed the vehicle capacity Q; (3) each commodity requested by each customer must be delivered by a single vehicle; (4) the demands of all customers need to be satisfied.

To solve the C-SDVRP, it is possible to duplicate the node associated with each customer by the number of commodities requested by the customer [1]. To each duplicated node, we then associate the demand of the customer for the corresponding commodity. For the sake of clarity, in the following we will call the duplicated nodes *customer commodity*.

3 Adaptive large neighborhood search

In order to solve the C-SDVRP for large instances, we propose a heuristic method based on the ALNS framework of [3]. Local search moves are also used in order to improve the solutions. A solution is represented as a set of routes. In order to take into account the specific features of C-SDVRP, a route can be represented: (1) as a sequence of *customers*, each customer having a set of commodities, or (2) as a sequence of *customer commodities*. In the first case, removing a customer from a route implies to remove the customer with the set of commodities delivered in this route, while in the second case it is possible to remove only one commodity.

3.1 General framework

ALNS relies on a set of removal and insertion heuristics which iteratively destroy and repair solutions. The probability to select a heuristic at a given iteration is influenced by its performance during past iterations. A sketch of the method is outlined in Algorithm 1.

An initial solution is constructed as follows: (1) give a random sequence of customers commodities to construct a giant tour, (2) apply a split procedure [4] to get a solution, (3) apply local search to improve this solution.

Algorithm 1 Adaptive large neighborhood search

generate an initial solution $s \in \{solutions\}$, $\rho \leftarrow 1$, $s_{best} \leftarrow s$
repeat
    Roulette wheel: select a removal heuristic h_{rem} and an insertion heuristic h_{ins}
    Destroy: $s_{rem} \leftarrow$ remove ρ customer commodities from s applying h_{rem}
    Repair: $s_{ins} \leftarrow$ insert removed customer commodities into s_{rem} applying h_{ins}
    Improve: $s' \leftarrow$ improve solution s_{ins} with local search
    if $f(s') < f(s_{best})$ **then** $s_{best} \leftarrow s'$ **end if**
    if accept(s', s) **then** $s \leftarrow s'$, $\rho \leftarrow 1$ **else** $\rho \leftarrow \rho + 1$, or ρ^{min} if $\rho = \rho^{max}$ **end if**
until stopping criterion is met
return s_{best}

At each iteration, a simulated annealing criterion is used to determine if the new solution s' is accepted. The number ρ of customer commodities to remove from the current solution s follows the scheme proposed in [5], with the aim of applying small moves when a new solution has just been accepted, while applying large move when no new solutions has been accepted in the most recent iterations.

3.2 Removal and insertion heuristics

Classical removal and insertion heuristics ([3]) have been implemented. Removal heuristics are: random removal, worst removal and Shaw removal with a relatedness measure based on distance. Insertion heuristics are: greedy insertion and regret insertion. Worst removal and Shaw removal work with customers, while random removal can be applied either with customers or customer commodities. Insertion heuristics work with customer commodities.

3.3 Local search moves

In order to improve a solution, we apply a set of local search moves. Seven different moves are implemented. In the first three moves routes are considered as a set of customers. These moves are: (1) insert customer, (2) swap customers, (3) 2-opt of customers. Then, we propose two other classical moves adapted for customer commodities: (4) insert customer commodity, and (5) swap customer commodities. We also propose two other moves: (6) erase route: from a given route, remove the customer commodities until no other route has capacity to accept another customer commodity; and (7) reassign commodities: for a given customer we propose a Mixed Integer Program to optimally assign all the commodities of this customer to the routes of the solution.

Given a feasible solution, all 7 moves are iteratively applied one after the other, until no one improves the solution. Then, routes of the current solution are concatenated and split algorithm is applied. If this gives a better solution, the procedure is repeated.

4 Preliminary results

The algorithm is implemented in C++ and run on a Intel (R) Core(TM) i7-6500U, 2.50GHz and 8GB of RAM. The instances are the ones proposed in [1], based on the R101 and C101 Solomon instances for the VRP. We consider 3 sets of instances with up to 3 commodities: small instances with 15 customers, mid-sized instances with 20 customers or 80 customers.

Global results are reported in Table 1. We report the average, minimum and maximum gap (gap%) between the value we obtained and the best known solution in literature [2], the number of optimal values we obtained and the average computational time in seconds.

Table 1: Global results.

$\lvert\mathcal{N}_C\rvert$	nb instances	av.gap (%)	min gap(%)	max gap(%)	nb optimal	av.time(s)
15	64	0.12	0.00	4.20	61	58
20	20	0.18	0.00	1.8	12	97
80	20	0.77	-0.88	6.69	-	1767

From Table 1, we can see that our algorithm can solve to optimality 61 out of 64 small instances in reasonable computing time. On mid-sized instances, we obtain very good quality results on a reasonable amount of time, and we provide 10 new best known values for mi-sized instances with 80 customers.

The prospects are to implement specific removal and insertion heuristics. Moreover, the proposed method can then be extended to other versions of the problems, like multi-depot version with a limited capacity of each commodity at the each depot.

References

[1] C. Archetti, A. M. Campbell, and M. G. Speranza, "Multicommodity vs. single-commodity routing", *Transportation Science* 50(2), 461-472, 2014.

[2] C. Archetti, N. Bianchessi, and M. G. Speranza, "A branch-price-and-cut algorithm for the commodity constrained split delivery vehicle routing problem", *Computers & Operations Research* 64, 1-10, 2015.

[3] S. Ropke, and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows", *Transportation Science*, 40(4), 455-472, 2006.

[4] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem", *Computers & Operations Research* 31(12),1985-2002, 2004.

[5] V. François, Y. Arda, Y. Crama, G. Laporte, "Large neighborhood search for multi-trip vehicle routing", *European Journal of Operational Research*, 255(2), 422-441, 2016.