SOLVING A REAL-LIFE INVENTORY ROUTING PROBLEM USING A TWO-PHASE HEURISTIC

Pedro Munari
pedro.munari@gmail.com
Aldair Álvarez
aldairad147@gmail.com
Maria Gabriela Stevanato Furtado
gabisfurtado@gmail.com
Pedro Miranda
pluis1908@gmail.com
Amélia Stanzani
mel.stanzani@hotmail.com

We propose a heuristic algorithm to solve a real-life inventory routing problem (IRP) motivated by the bulk gas distribution operations of a multinational company that supplies gases for industrial applications. This problem takes into account multiple sources, time windows, driver-related constraints, customers orders, and many other practical constraints. The developed heuristic algorithm decouples the problem into two distinct phases at each iteration. In the first, it decides which resources to use, how much to deliver and priority values for each customer. Then, in the second phase, it determines which customers will be visited and the respective vehicle route. This heuristic was embedded in a multi-call approach that dynamically changes the values of the parameters until reaching the predefined stopping criterion. Computational experiments using real-life data showed that the heuristic is able to find solutions to most instances in short running times and hence can be used for efficient decision making in practice.

Palavras-chave: Inventory routing, Vendor managed inventory, Logistic ratio, Heuristic

1. Introduction

The inventory routing problem (IRP) emerges in logistic contexts in which a supplier has to determine which customers to visit and how much to deliver to each of them at each period of a time horizon, in addition to design the best routes for carrying out these visits. Hence, the IRP integrates two challenging problems, namely the multi-period inventory management problem and the vehicle routing problem. Given the complexity of the decisions involved, the research on solutions methods for the IRP has been very active in the past years (ANDERSSON et al., 2010; COELHO et al., 2014).

In this paper we propose a solution method for a practical variant of the IRP, motivated by an application in the bulk gas distribution industry under vendor-managed inventory (VMI) system. In VMI, the supplier manages the inventory of their customers, aiming at improving the efficiency of the overall system. In turn, the supplier must guarantee a minimum service level on the customers. Thus, the supplier must simultaneously decide when and how much to deliver to each customer and how to perform such deliveries. Particularly, this latter decision involves, among others, the choosing of the source of the product to deliver and the resources that must be used to perform the delivery (e.g. truck/trailer and driver).

An IRP variant was first studied by Bell et al. (1983), who solved an integrated inventory management and vehicle routing problem also for a gas distribution industry. Since then, several variants and solution methods have been studied in the literature. Dror et al. (1985) and Dror and Ball (1987) addressed long-term IRPs. Dror et al. (1985) solved the problem in a rolling horizon fashion, whereas Dror and Ball (1987) reduced the planning horizon to a single-period problem, defining costs reflecting long-term decisions, safety inventory levels and subsets of customers to be considered. Campbell and Savelsbergh (2004) develop a two-phase hybrid approach for a IRP minimizing transportation costs only. In their approach, the visit schedule and delivery sizes are determined solving an integer program while the delivery routes are constructed with heuristics algorithms. Savelsbergh and Song (2007) and Savelsbergh and Song (2008) proposed heuristic and hybrid algorithms to solve a variant of the IRP motivated by a real-life application which includes pickup and delivery routes spanning multiple time periods. Abdelmaguid et al. (2009) proposed heuristic algorithms for an IRP with backlogging. Le et al. (2013) used a column generation-based heuristic to solve an IRP with perishability constraints. Cordeau et al. (2014) solved the multi-product IRP with a three-phase decomposition-based heuristic algorithm. Benoist et al. (2011) solved an real-life IRP through

a rolling-horizon heuristic algorithm. The authors use a tailored short-term objective function designed to reflect long-term costs. Singh et al. (2015) developed an incremental heuristic approach for a IRP motivated by real-life distribution problem in a bulk gas industry.

Basic variants of the IRP have also been addressed in recent years, focused on methods to obtain optimal solutions (COELHO et al., 2014). Archetti et al. (2007) and Solyalı and Süral (2011) proposed branch-and-cut (B&C) algorithms for a single-vehicle, single-product IRP. Coelho and Laporte (2013); Coelho and Laporte (2014) and Adulyasak et al. (2014a) also proposed B&C algorithms but for the multi-period, single-product IRP, while Desaulniers et al. (2016) developed a branch-price-and-cut (BPC) method for this problem. Also for the multi-vehicle, single-product IRP, metaheuristic algorithms were proposed by Shiguemoto and Armentano (2010); Archetti et al. (2012); Coelho et al. (2012); Adulyasak et al. (2014b); Santos et al. (2016); and Alvarez et al. (2018). In particular, Archetti et al. (2016) and Alvarez et al. (2018) address the IRP using as objective function the so called logistic ratio. This objective is the ratio of the total routing cost to the total quantity distributed, i.e., the distribution cost per unit delivered.

In this paper we propose a construction heuristic based on the product consumption at customers through the time horizon. It ranks customers according to product shortage and solves a series of resource constrained shortest path problems to determine shifts that satisfy all the constraints required by the company. The remainder of this paper is organized as follows. In Section 2, we describe the IRP addressed in this paper. A detailed description of the method is given in Section 3. In Section 4, we present the results of computational experiments with the proposed heuristic using instances created from real-life data. We then close this paper in Section 5 with conclusions and suggestions for future developments.

2. Problem description

The IRP variant addressed in this paper is concerned with the repeated distribution of a single product to a set of customers $\mathcal{C} = \{1,2,\dots,\mathrm{n}\}$, over a discrete and finite planning horizon $\mathcal{T} = \{1,2,\dots,\mathrm{T}\}$. This variant is motivated by the distribution process of a multinational company that supplies industrial gases for different applications. The company works with three types of gas distribution: pipelines, bulk and cylinders. In this paper, we focus on the bulk gas distribution, which poses a challenge when combined with VMI. The company monitors (telemetry and forecast) and must manage the customers inventories. In this case, several decisions need to be

made simultaneously: when to deliver to each customer; how much to deliver in each visit; how to combine such deliveries into feasible routes; and which resources to use.

To accomplish this, we must determine a set of driver shifts to satisfy customer demand requirements, maximum duration and additional technical constraints. We are given a set of transportation and human resources (trucks and drivers), which are located at one single base (depot). Drivers have different availabilities according to their time windows. There is a heterogeneous fleet of vehicles and each vehicle can perform multiple routes in the planning horizon. Multiple production sites (sources) are available in the problem, which means that the vehicles can load the product at different points. Each source has a list of vehicles allowed to load. A fixed service time (loading/unloading) is incurred at each source and customer (safety reasons).

Each shift must start at the base, visit a subset of customers and/or sources, and then return to the base. All these visits must satisfy vehicle capacity, customer tank capacity and safety level, customer time windows, maximum driving duration, trailer usage and minimum operation quantity. Customers are divided into two classes: VMI customers and call-in customers. For each VMI customer the company monitors its tank level and guarantees that this level never becomes lower than a given safety level. Call-in customers place orders through the planning horizon and they are mandatory. A customer can be further classified as layover, which means that the duration of a shift containing this customer can be extended by including a resting time for the driver. The cost of the shifts is proportional to its duration (driver costs) and includes the driving time, the idle time and the loading/unloading time. A travel cost proportional to the traveled distance is also incurred (fuel consumption costs). When a rest appears in a shift, there is an additional layover cost.

The objective is to minimize the logistic ratio, which is given by the total transportation cost of the shifts divided by the total quantity delivered over the planning horizon. Holding costs are not included in this objective. The logistic ratio corresponds to the average cost per unit of delivered product and captures the long-term impact of a short-term planning given that it focuses on the efficiency of the process (BENOIST et al., 2011; SINGH et al., 2015; ARCHETTI et al., 2016; ALVAREZ et al., 2018).

3. A two-phase heuristic for the addressed IRP variant

The purpose of the heuristic is to determine a set of shifts that minimizes the logistic ratio, while satisfies customer demand requirements and technical constraints. This is an iterative procedure that, at each iteration, finds a single shift by choosing: an available driver; the starting and ending time of the shift, which must be inside of one the time windows of the driver; the trailer used by the driver; the sequence of customers and sources that will be visited; and how much will be delivered (collected) to each customer (at each source).

The first step in the heuristic consists in computing the inventory/tank levels I_{it} at each customer $i \in \mathcal{C}$ and time period $t \in \mathcal{T}$, given the respective initial inventory level I_{i0} and the product consumption d_i^t (demand). This is done by subtracting the product consumption of each time period from the inventory level on the previous period ($I_{i,t-1}$). For a given time period $t \in \mathcal{T}$, the inventory level of customer $i \in \mathcal{C}$ is given by:

$$I_{it} = \max\{0, I_{i0} - \sum_{h=1}^{t} d_i^h\}. \quad (1)$$

These levels are used to determine which customers must be visited and when they must be visited to avoid their tank level to become lower than the safety level. After their computation, we create a list containing the time windows of all drivers, sorted in the ascending order of their respective opening time instant. Then, we start the iterative process from the first time window in the list.

Each iteration starts with the driver that corresponds to the current time window. Then, we select the earliest available trailer that can be assigned to the driver. With the driver and trailer chosen, we select a subset of customers for potential visits, given by those that can be serviced using the current trailer and whose time windows have overlap with the current driver's time window. For each customer $i \in \mathcal{C}$ selected in this subset, we first define its delivery amount q_i as follows. For VMI customers, this is computed as

$$q_i = \min\{C_i - I_{ia}, Q\}, \quad (2)$$

where C_i is the tank capacity of the customer, I_{ia} is the inventory level of the customer at the opening time of its time window and Q is the trailer capacity. If q_i is smaller than the minimum delivery quantity imposed by the customer, then q_i is set to 0. For call-in customers, the demand corresponds to the actual demand of the order. We also define a time window for each customer, based on the value defined for q_i. It opens at the first time instant in which we can deliver the

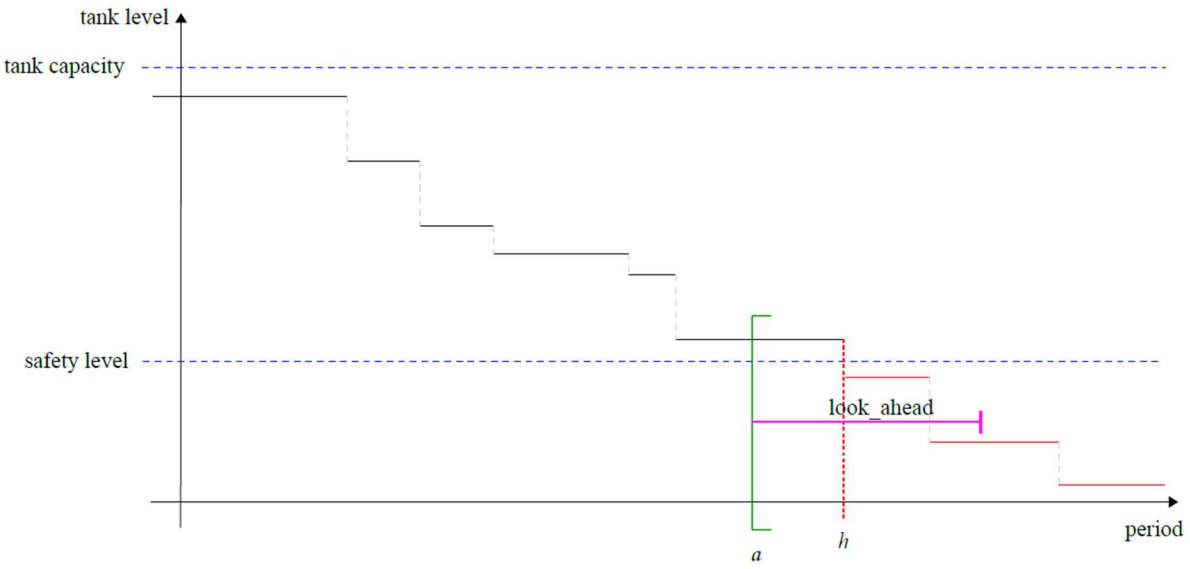
product amount q_i without violating the customer capacity; and closes in the latest time instant in which the safety level is still not violated without the delivery of q_i.

Since it may be impossible to visit all selected customers due to capacity and timing requirements, we need to choose which ones to visit in the shift. To do so, we associate a priority value p_i to each selected customer $i \in \mathcal{C}$, trying to assess the urgency to serve the customer. This urgency is quantified by checking whether the customer safety level will be violated in the next `look_ahead` time periods, starting from the opening of the current driver's time window. Hence, `look_ahead` is a parameter that suggests how long we should look ahead in the time horizon, to have a good assessment of the right time instant to visit the customer. If the safety level of customer $i \in \mathcal{C}$ is not violated in this time interval, then we set $p_i = 0$. Otherwise, the priority is set as

$$p_i = T - h, \tag{3}$$

where T is the last period of the horizon and h is the period in which the safety level becomes violated (for VMI customers) or the latest time that the order can be delivered (for call-in customers). An illustration of how the parameter look_ahead works is presented in Figure 1 for the tank level (inventory) of a given customer. Notice that at the beginning of the current time window (a) the tank level is higher than the safety level. However, by looking to the next `look_ahead` periods, we observe that the safety level becomes violated at time period h.

Figure 1 - Illustration of the `look_ahead` parameter



Source: Created by the authors

Additionally, to include a long-term assessment in the priority values, we verify the tank level also at the end of the planning horizon. If it violates the safety level, then the priority is updated as follows

$$p_i = p_i + \texttt{longterm\_multiplier} * (\mathrm{T} - a). \quad (4)$$

Hence, the parameter `longterm_multiplier` imposes an extra penalty to avoid the inventory level of the customers to get too low on the long-term, aiming to prevent the accumulation of large demands in latter periods, which can complicate to serve all customers in the next iterations of the heuristic. At this point, we have assigned priorities to all customers that can be visited by the driver and trailer chosen at the current iteration. Then, we can start the next phase of the iteration, in which we build and solve a resource constrained shortest path problem with recharge (RCSPPR), based on the data defined in the previous steps. This RCSPPR is defined by $\mathrm{n} + \mathrm{ns} + 1$ nodes, given by the n customers, the ns sources and the base. There is one arc (i, j) linking each pair of nodes i and j, except when j cannot be visited by the current trailer. There are three resources in the problem: vehicle capacity, customer time windows and route duration. The shortest path must start and return to the depot (base) and respect the limits impose to all these resources. Recall that if the route includes at least one layover customer, then the maximum driving duration can be extended including a resting time for the driver. The path can visit the source nodes as many times as possible and the vehicle becomes fully loaded every time this visit happens.

To solve the RCSPPR we use a standard insertion heuristic in which a seed node is chosen to start a new path. The node corresponding to a customer $\mathrm{i} \in \mathcal{C}$ with the smallest p_i (priority values) is chosen as the seed. Then, new nodes are iteratively added to the path until no more nodes can be added without violating the resources. Again, nodes corresponding to customers with the smallest priorities are the first to be included. Every time a node $\mathrm{i} \in \mathcal{C}$ is chosen to enter the path, we test the feasibility and compute the potential increasing in the total travel cost after inserting this node considering each two adjacent nodes in the path. We analyze this based on three different types of insertion:

- Insert node i only;
- Insert a visit to the closest source node and then to node i;
- Insert a visit to node i and then to the closest source node;

– Insert a visit to the closest source node, then to node i and then to the closest source again.

The node is then inserted in the position that is feasible and results in the minimum increase of the travel cost, according to one of these insertion types.

By solving the RCSPPR the shift is determined and then included in the partial solution. Then, we update all data regarding inventory levels of the visited customers, the load level of the used trailer and the next time instant in which the driver will be available. This finishes one iteration of the heuristic. A new driver time windows is then selected from the sorted list defined at the beginning of the method and a new iteration starts. The heuristic finishes when this list becomes empty. If the inventory levels of all customers are higher than the safety level at each time period, then we found a feasible solution of the problem. Since the method is a construction heuristic that takes a series of decisions sequentially and without any backtrack, it may finish without obtaining a feasible solution.

The heuristic runs quickly for most instances considered in the computational experiments (see Section 4). It has a few parameters to be set by the user before running, namely `look_ahead` and `longterm_multiplier`. These parameters have practical meanings and are typically considered in the decision-making process of logistic activities. Hence, setting good values for them should not be a challenge to a decision maker and may even allow them to analyze different scenarios. In addition, the data collected on different time periods may be rather heterogeneous regarding the number of customers, time periods, available vehicles, drivers' schedules and other features.

Another way of considering these parameters is recurring to a multi-call strategy, in which the heuristic is called several times, but using different parameter settings. For each parameter that changes the performance of the heuristic significantly, we define a range such that the heuristic is called for each discrete value selected from this range. As a result, we have a multi-call construction heuristic for addressed IRP variant. A pseudo-code for this heuristic is shown in Algorithm 1.

Algorithm 1 – The two-phase heuristic in the multi-call strategy

```
   input  : Instance, parameters;
   output: Solution S*;
1  begin
2      S* ← ∅;
3      while there are parameter values not tested do
4          Compute the inventory levels of customers using (1);
5          Create a list containing the time windows of all drivers, sorted in ascending order of their
             opening time;
6          S ← ∅;
7          while exist an available time window in the list do
8              Get the driver corresponding to the current time window;
9              Get the next available trailer;
10             for all customer i do
11                 Set the delivery amount as in (2);
12                 Compute the priority values as in (3) and (4) ;
13             end
14             Apply the insertion heuristic to solve the RCSPPR;
15             Check if a new feasible shift for the driver was created;
16             Save the shift in solution S;
17             Update the inventory levels for each served customer;
18             Set the next time instant that the driver and trailer will be available;
19         end
20         if a better solution has been found then
21             Update the best solution S*;
22         end
23         Update the heuristic parameters;
24     end
25 end
```

Source: Created by the authors

4. Results

We have applied the proposed multi-call heuristic to the problem instances available at the website of the ROADEF/EURO 2016 Challenge (ROADEF, 2016). These instances were created using real-life data from Air Liquide, a multinational company that serves approximately one million customers worldwide, in various industries. Hence, they are very challenging and include a large number of time periods, customers and other elements, as presented in Table 1. To be concise, we consider in this paper only instances in set B. The heuristic was implemented in language C/C++ and the computer used to run the experiments is a Linux PC with processor Intel Core i7-3540M and 8 GB of RAM. The maximum running time was set as 3600 seconds.

Table 2 presents the computational results of the proposed heuristic using the instances described in Table 1. The first column gives the name of the instance; the remaining six columns show the logistic ratio of the best solution found within the time limit specified in the header of

the column; and the last column gives the best-known solution (BKS) as provided in (ROADEF, 2016), obtained within a time limit of 1800 seconds. Notice that the proposed heuristic obtains feasible solutions for most instances, which is already a challenging task. In addition, it achieves reasonably good solution values even for short running times.

Table 1 – Main characteristics of the real-life instances from ROADEF (2016)

Instance	*Periods*	*Customers*	*Sources*	*Drivers*	*Trailers*	*Layover*	*Call-in*	*Orders*
V2.12	240	324	1	13	15	12	12	24
V2.13	240	53	1	5	5	14	14	0
V2.14	840	53	1	5	5	14	14	0
V2.15	240	134	1	4	3	16	16	4
V2.16	240	184	1	7	4	5	5	1
V2.17	840	134	1	4	3	16	16	4
V2.18	840	134	1	4	3	16	16	4
V2.19	840	53	1	5	5	14	14	0
V2.20	840	184	1	7	4	5	5	1
V2.21	840	184	1	7	4	5	5	1
V2.22	504	324	1	13	15	12	12	24
V2.23	504	324	1	13	15	12	12	24
V2.24	240	32	2	5	6	0	0	30
V2.25	840	32	2	5	6	0	0	30
V2.26	840	32	2	5	6	0	0	30

Source: Created by the authors

Table 2 – Results obtained by the proposed heuristic within 3600 seconds

Instance	$\leq$ 5 sec	$\leq$ 30 sec	$\leq$ 60 sec	$\leq$ 180 sec	$\leq$ 1800 sec	$\leq$ 3600 sec	BKS
V2.12	–	–	–	–	–	–	0.010024
V2.13	0.051006	0.049933	0.049933	0.04904	0.047358	0.047171	0.028875
V2.14	–	0.067901	0.066857	0.060496	0.058771	0.058189	0.034971
V2.15	0.046357	0.040264	0.040264	0.040264	0.039946	0.039946	0.024993
V2.16	0.022856	0.019329	0.019329	0.019329	0.019235	0.019146	0.011783
V2.17	–	–	–	–	–	–	0.03213
V2.18	–	–	–	–	–	–	0.031882
V2.19	–	0.062705	0.062705	0.059695	0.058671	0.058671	0.034022
V2.20	–	–	0.024866	0.024583	0.023175	0.023175	0.017486
V2.21	–	–	0.024562	0.024521	0.023708	0.023701	0.016806
V2.22	–	–	–	–	–	–	0.012667
V2.23	–	–	–	–	–	–	0.012603
V2.24	0.021129	0.021129	0.021047	0.020942	0.020942	0.020942	0.011219
V2.25	0.021097	0.02069	0.02069	0.020513	0.020399	0.020399	0.011451
V2.26	0.021377	0.021367	0.021367	0.021367	0.020614	0.020614	0.011281

Source: Created by the authors

5. Conclusions and future developments

We proposed a construction heuristic for a practical variant of the inventory routing problem, which considers real-life requirements motivated by the bulk gas distribution activities of a multinational company. Each iteration of this heuristic has two main phases. In the first, it defines priority values for customers, based on the inventory level of a given time interval. Then, in the second phase, a feasible route is defined for the customers with the best priorities. The resulting route is included in the solution as a single shift and the process is repeated. The heuristic is based on two main parameters: `look_ahead`, which tells how many periods ahead we should look to check if a customer safety level is violated; and `longterm_multiplier`, which defines an extra penalty to avoid the inventory level of the customers to get too low on the long-term. These parameters have practical meanings and may be intuitively handled by the decision makers. We have also proposed a multi-call strategy, in which different values are automatically tested for these parameters.

The results of computational experiments using instances created from real-life data indicate that the heuristic was able to find feasible solutions to most instances and in short running times. Hence, we believe that it may be of great value in practice, to help decision makers to quickly determine a feasible policy that integrates inventory and routing decisions. This heuristic can also be used to find initial solutions for more elaborated methods, such as metaheuristics and matheuristics. Therefore, an interesting future development would be to design local search heuristics to be used jointly with the proposed heuristic in a metaheuristic framework. Also, different strategies may be tried to define the priorities values of customers as well as to solve the resource constrained shortest path problem.

6. Acknowledgements

This work was supported by the São Paulo Research Foundation (FAPESP) and the National Council for Scientific and Technological Development (CNPq).

REFERENCES

ABDELMAGUID, T. F.; DESSOUKY, M. M.; ORDÓÑEZ, F. Heuristic approaches for the inventory-routing problem with backlogging. **Computers and Industrial Engineering,** 56(4):1519–1534, 2009.

ADULYASAK, Y.; CORDEAU, J.-F.; JANS, R. Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. **INFORMS Journal on Computing**, 26(1):103–120, 2014a.

ADULYASAK, Y.; CORDEAU, J.-F.; JANS, R. Optimization-based adaptive large neighborhood search for the production routing problem. **Transportation Science**, 48(1):20–45, 2014b.

ALVAREZ, A.; MUNARI, P.; MORABITO, R. Iterated local search and simulated annealing algorithms for the inventory routing problem. **International Transactions in Operational Research**, Online first, 2018.

ANDERSSON, H.; HOFF, A.; CHRISTIANSEN, M.; HASLE, G.; LØKKETANGEN, A. Industrial aspects and literature survey: Combined inventory management and routing. **Computers & Operations Research**, 37(9):1515–1536, 2010.

ARCHETTI, C.; BERTAZZI, L.; HERTZ, A.; GRAZIA SPERANZA, M. A hybrid heuristic for an inventory routing problem. **INFORMS Journal on Computing**, 24(1):101–116, 2012.

ARCHETTI, C.; BERTAZZI, L.; LAPORTE, G.; SPERANZA, M. G. A branch-and-cut algorithm for a vendor-managed inventory-routing problem. **Transportation Science**, 41(3):382–391, 2007.

ARCHETTI, C.; DESAULNIERS, G.; SPERANZA, M. G. Minimizing the logistic ratio in the inventory routing problem. **EURO Journal on Transportation and Logistics**, pages 1–18, 2016.

BELL, W. J.; DALBERTO, L. M.; FISHER, M. L.; GREENFIELD, A. J.; JAIKUMAR, R.; KEDIA, P.; MACK, R. G.; PRUTZMAN, P. J. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. **Interfaces**, 13(6):4–23, 1983.

BENOIST, T.; GARDI, F.; JEANJEAN, A.; ESTELLON, B. Randomized local search for real-life inventory routing. **Transportation Science**, 45(3):381–398, 2011.

CAMPBELL, A. M.; SAVELSBERGH, M. W. P. A decomposition approach for the inventory-routing problem. **Transportation Science**, 38(4):488–502, 2004.

COELHO, L. C.; CORDEAU, J.-F.; LAPORTE, G. Consistency in multi-vehicle inventory-routing. **Transportation Research Part C: Emerging Technologies**, 24:270–287, 2012.

COELHO, L. C.; CORDEAU, J.-F.; LAPORTE, G. Thirty years of inventory routing. **Transportation Science**, 48(1):1–19, 2014.

COELHO, L. C.; LAPORTE, G. A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. **International Journal of Production Research**, 51(23):7156–7169, 2013.

COELHO, L. C.; LAPORTE, G. Improved solutions for inventory-routing problems through valid inequalities and input ordering. **International Journal of Production Economics**, 155:391–397, 2014.

CORDEAU, J.-F.; LAGANÀ, D.; MUSMANNO, R.; VOCATURO, F. A decomposition-based heuristic for the multiple-product inventory-routing problem. **Computers & Operations Research**, 55(2):153–166, 2014.

DESAULNIERS, G.; RAKKE, J. G.; COELHO, L. C. A branch-price-and-cut algorithm for the inventory-routing problem. **Transportation Science**, 50(3):1060–1076, 2016.

DROR, M.; BALL, M. Inventory/routing: Reduction from an annual to a short-period problem. **Naval Research Logistics**, 34:891–905, 1987.

DROR, M.; BALL, M.; GOLDEN, B. A computational comparison of algorithms for the inventory routing problem. **Annals of Operations Research**, 4(1):1–23, 1985.

LE, T.; DIABAT, A.; RICHARD, J.-P.; YIH, Y. A column generation-based heuristic algorithm for an inventory routing problem with perishable goods. **Optimization Letters**, 7(7):1481–1502, 2013.

ROADEF. **ROADEF/EURO challenge 2016: Inventory Routing Problem**, 2016. Available at: <http://challenge.roadef.org/2016/en>. Last access: 07 May 2018.

SANTOS, E.; OCHI, L.; SIMONETTI, L.; GONZÁLEZ, P. A hybrid heuristic based on iterated local search for multivehicle inventory routing problem. **Electronic Notes in Discrete Mathematics**, 52:197–204, 2016.

SAVELSBERGH, M.; SONG, J.-H. Inventory routing with continuous moves. **Computers & Operations Research**, 34(6):1744–1763, 2007.

SAVELSBERGH, M.; SONG, J.-H. An optimization algorithm for the inventory routing problem with continuous moves. **Computers & Operations Research**, 35(7):2266–2282, 2008.

SHIGUEMOTO, A. L.; ARMENTANO, V. A. A Tabu Search procedure for coordinating production, inventory and distribution routing problems. **International Transactions in Operational Research**, 17(2):179–195, 2010.

SINGH, T.; ARBOGAST, J. E.; NEAGU, N. An incremental approach using local-search heuristic for inventory routing problem in industrial gases. **Computers and Chemical Engineering**, 80:199–210, 2015.

SOLYALI, O.; SÜRAL, H. A branch-and-cut algorithm using a strong formulation and an a priori tour-based heuristic for an inventory-routing problem. **Transportation Science**, 45(3):335–345, 2011.