CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

An Adaptive Large Neighborhood Search Heuristic for a Multi-Period Vehicle Routing Problem

Iman Dayarian
Teodor Gabriel Crainic
Michel Gendreau
Walter Rei

October 2014

CIRRELT-2014-55

Bureaux de Montréal :
Université de Montréal
Pavillon André-Aisenstadt
C.P. 6128, succursale Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

An Adaptive Large Neighborhood Search Heuristic for a Multi-Period Vehicle Routing Problem†

Iman Dayarian[1,*], Teodor Gabriel Crainic[2,3], Michel Gendreau[1,2], Walter Rei[2,3]

[1] Department of Mathematics and Industrial Engineering, Polytechnique Montréal, P.O. Box 6079, Station Centre-ville, Montréal, Canada H3C 3A7

[2] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT)

[3] Department of Management and Technology, Université du Québec à Montréal, P.O. Box 8888, Station Centre-Ville, Montréal, Canada H3C 3P8

Abstract. We consider tactical planning for a particular class of multi-period vehicle routing problems (MPVRP). This problem involves optimizing product collection and redistribution from several production locations to a set of processing plants over a planning horizon. Each horizon consists of several days, and the collection-distribution are performed on a repeating daily basis. In this context, a single routing plan must be prepared for the whole horizon, taking into account the seasonal variations in the supply. We model the problem using a sequence of periods, each corresponding to a season, and intra-season variations are neglected. We propose an adaptive large-neighborhood search with several special operators and features. To evaluate the performance of the algorithm we performed an extensive series of numerical tests. The results show the excellent performance of the algorithm in terms of solution quality and computational efficiency.

Keywords: Multi-period vehicle routing problem, tactical planning, seasonal variation, adaptive large neighbourhood search.

Acknowledgements. Partial funding for this project was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial Research Chair, Collaborative Research and Development, and Discovery Grant programs. We also received support from the Fonds de recherche du Québec - Nature et technologies (FRQNT) through its Team Research Project program, and from the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through infrastructure grants and the support of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

†Revised version of the CIRRELT-2013-67.

---

* Corresponding author: Iman.Dayarian@gerad.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2014

1. Introduction

The vehicle routing problem (VRP) is a difficult combinatorial optimization problem that appears in many practical applications relating to the design and management of distribution systems. Studies of the classical VRP and its many variants and extensions, starting with the seminal work of Dantzig and Ramser (1959), represent a significant portion of the operations research literature (Toth and Vigo, 2002). The classical VRP, referred to as the capacitated vehicle routing problem (CVRP), concerns the determination of routes for a fleet of homogeneous vehicles, stationed at a central depot, that must service a set of customers with known demands (supplies). The goal is to design a collection of least-cost routes such that: 1) each route, performed by a single vehicle, begins at a depot, 2) each customer is visited once by exactly one vehicle, and 3) the quantity of goods delivered (collected) on each route does not exceed the vehicle capacity (Golden et al, 2008).

In classical settings, e.g., the CVRP, the routing plan is executed repeatedly over the planning horizon. The parameters of the problem, such as the quantities to be delivered (collected) at each customer location, are assumed fixed over the horizon and known a priori. However, in many real-life applications, this assumption may result in poor-quality routing plans. Our problem setting requires routing over relatively long horizons, in environments with significant seasonal fluctuations. This setting, milk collection and redistribution in the dairy industry of Quebec, initially introduced by Dayarian et al (2014a), has several problem-specific attributes and characteristics. The routing corresponds to the collection of milk from producers' farms followed by the distribution of the product to a set of processing plants. The routes must be designed in such a way that the plant demands are completely satisfied, while every producer is visited by exactly one vehicle and each vehicle delivers to just one plant per day. We assume that the daily quantity of milk produced satisfies the total plant demand.

The first studies of this problem were performed by Lahrichi et al (2013) and Dayarian et al (2014b); both studies assumed that the annual production is fixed. Dayarian et al (2014a) addressed a variant of the problem that accounted for seasonal variations in the supply. Because of contractual and service-consistency requirements, a single routing plan must be prepared for a given horizon. The contractual negotiations between the different stakeholders (producers, carriers, and plants) are based on a single routing plan. For service consistency, each producer should always be included in the same route and serviced by the same vehicle. The drivers also use this routing plan to schedule their daily operations.

Dayarian et al (2014a) proposed an exact methodology based a multi-period model and a branch-and-price approach. They divided the horizon into a series of periods, each a cluster of days with similar seasonal characteristics. The horizon can then be represented as a sequence of periods. The need to design a single plan for changing contexts recalls the *a priori* optimization framework for stochastic optimization problems. In stochastic programming, a two-stage model is often considered. The solution from the first stage is updated at the second stage as the values of the stochastic parameters are revealed.

The solution approach proposed by Dayarian et al (2014a) provides optimal solutions for instances with up to sixty producers. However, real-life problems may have several hundred producers. Therefore, we need solution approaches that can find good but not necessarily optimal solutions to larger problems. The main goal of this paper is to find such solutions using an effective adaptive large-neighborhood search (ALNS) framework (Pisinger and Ropke, 2007; Ropke and Pisinger, 2006). Our main contributions are as follows:

- We design an ALNS based metaheuristic for a complex vehicle routing problem. The proposed solution procedure includes a set of novel algorithmic features, including several new operators based on the special structure of the problem. These are detailed in Section 4.

- To evaluate the quality of the solution, we compute a series of lower and upper bounds on the value of the multi-period solution. We compare the solutions obtained through the ALNS with these bounds.

- We extensively analyze the performance of the method and its components in terms of computational time and solution quality, through a series of numerical tests on a large set of randomly generated instances.

The remainder of this paper is organized as follows. In Section 2, we describe the problem and the notation that we use. Section 3 discusses the state-of-the-art in this field. In Section 4, we present our ALNS-based approach to the problem. In Section 5, we propose a series of bounds that allow us to evaluate the performance of the algorithm. The experimental results are reported in Section 6, and Section 7 provides concluding remarks.

2. Problem Statement and Modeling

In this section, we introduce the problem; it is inspired by a dairy problem in Quebec. For a detailed description of the dairy transportation problem in Quebec (DTPQ), the reader is referred to Lahrichi et al (2013) and Dayarian et al (2014a,b).

The DTPQ can be briefly described as follows: In Quebec, the Fédération des producteurs de lait du Québec (FPLQ), a coalition of milk producers, is responsible for managing the collection and transportation of milk produced in the province. The FPLQ negotiates, on behalf of the producers, annual transportation contracts with the carriers (Lahrichi et al, 2013). Each contract with a carrier for a given horizon H is based on a single tactical routing plan. A plan consists of a set of routes, each performed by a single vehicle on every collection day of H. An unlimited fleet of identical vehicles is assumed to be available at multiple depots. On every collection day, each vehicle departs from a depot, collects a single product type from a subset of producers, delivers the collected product to a single plant, and then returns to its depot. This can be seen as an extension of the VRP with additional deliveries to multiple plants, and it is therefore NP-hard (Lenstra and Kan, 1981).

The producers' supply over the horizon may vary seasonally. The seasonal variations are often significant and may have a major impact on the routing. We assume that a year can be divided into several periods, each representing a seasonal production level. We take inter-period production variations into account; the potential intra-period fluctuations are neglected. Intra-period fluctuations can often be handled by leaving a spare capacity of 5%–10% on each vehicle when designing the routes. The producers' seasonal fluctuations are assumed to be perfectly positively correlated. This correlation arises because almost all the producers in a given geographical region are exposed to similar seasonal cycles. The plants must adjust their seasonal demands according to the supply so that the total supply always meets the total demand.

The proposed multi-period model has some similarities to an *a priori* optimization framework in the context of the vehicle routing problem with stochastic demand (VRPSD). In a two-stage formulation of a stochastic problem, the solution from the first stage is updated at the second stage as the exact values of the stochastic parameters are revealed. We seek a solution that minimizes the total expected cost of the original plan and the potential adjustments in the second stage. Similarly to algorithms for the VRPSD, in the context of our multi-period problem at the first stage we design a single plan for the planning horizon, taking into account possible supply changes between periods. At the second stage, the plan is adjusted based on the specificities of each period. In seasons with higher supply levels, at a given producer location a vehicle may have insufficient residual capacity to collect the supply. We refer to this as a *failure*. Following a failure, the vehicle usually travels to a plant to empty its tank and then proceeds to visit the remaining producers of the planned route. We refer to this extra travel as a *recourse* action.

Under our recourse policy, the vehicle always visits the producers in the order of the planned route; when a failure occurs, it travels to its assigned plant. Consequently, the total distance traveled corresponds to the fixed length of the planned route plus the length of the return trip to the plant.

The goal is to design a single least-cost collection-delivery plan for a given horizon, providing a certain level of service consistency and service quality, and taking into account the existence of several periods. We define a feasible plan to be one that is executable over the horizon with at most one failure per operation per route.

A single plan is necessary because 1) the contractual arrangements between the FPLQ and the carriers require a single plan that can be used for cost estimation for the whole horizon; and 2) there is a consistent driver-producer relationship when the producer is always serviced via the same route operated by the same vehicle. The second point leads to a familiar environment for the producer and the driver and avoids potential incompatibilities between the vehicles and the producer's facilities.

We control the desired service quality over a given horizon by setting a *service reliability threshold* (SRT), indicating the minimum percentage of days over the horizon H that the planned routes should be executable with no failures. The magnitude of the SRT has a major impact on the design of the plan. Clearly, if SRT = 100%, no failure occurs in any period of the horizon. However, this strategy is not cost-efficient, because it often requires many vehicles.

Let Ξ be the set of all periods in a given horizon H. We associate with each period $\xi \in \Xi$ a weight W_ξ, representing the share of period ξ in horizon H. It is calculated by dividing the length of period ξ by the length of horizon H. We also associate with each period ξ a production coefficient, P_ξ, which is defined to be the ratio of the production level in period ξ to a chosen *reference production level* P_{ref}. The choice of the reference production level is discussed in detail in Dayarian et al (2014a). Briefly, the reference period is obtained by merging the smallest subset of the periods with least production coefficients, forming a new period in such a way that the cumulative weight of the added periods to the subset covers the SRT. The newly obtained period, referred to as the *reference period*, substitutes the periods included in the subset. The production coefficient of the reference period corresponds to the largest coefficient among the added periods while its weight is equal to the cumulative weight of the substituted periods. For the sake of simplicity, all the production coefficients are divided by the reference period's coefficient P_{ref} (consequently, $P_{ref} = 1$). In order to provide plans respecting the defined SRT, one has to make sure that the designed routes do not face any failure in the reference period.

The model is defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$, where $\mathcal{V}$ and $\mathcal{A}$ are the node and arc sets, respectively. The node set contains the depots, producers, and plants; $\mathcal{V} = \mathcal{D} \cup \mathcal{N} \cup \mathcal{P}$. The arc set $\mathcal{A} \subset \mathcal{V} \times \mathcal{V}$ defines feasible movements between different locations in $\mathcal{V}$. For each pair of locations $n_i, n_j \in \mathcal{V}$, $n_i \neq n_j$, there exists an arc $(i, j) \in \mathcal{A}$. Each arc $(i, j) \in \mathcal{A}$ has an associated nonnegative travel cost c_{ij}, which is proportional to the length of the arc. An unlimited fleet of vehicles $\mathcal{K}$, with identical capacity Q, is available at each depot. However, employing vehicle $k \in \mathcal{K}$ incurs a fixed cost of c_k. Note that a naive upper bound on the number of vehicles can be obtained by assigning each producer to a vehicle.

In each period, each producer $n_j \in \mathcal{N}$ produces a limited quantity of product on a daily basis. The supply levels in period $\xi \in \Xi$ are given by a vector in which the jth parameter, denoted o_j^{ξ}, is the supply (offer) of producer j. Moreover, the supply of each producer n_j in the reference period is given by o_j^{ref}. Therefore, the supply of producer n_j in period ξ is

$$o_j^{\xi} = P_{\xi}.o_j^{ref} \qquad (j \in \mathcal{N}, \xi \in \Xi), \tag{1}$$

where P_{ξ} represents the production coefficient in period ξ. Each plant $p \in \mathcal{P}$ receives, on a daily basis, the collected product. The demand of each plant p in period ξ is given by D_p^{ξ}. The routes are designed to have no failures in the reference period and at most one failure in the other periods. In other words, for each route r, the following inequalities hold:

$$\sum_{j \in r} o_j^{\xi} \leq 2Q, \qquad (\xi \in \Xi) \tag{2}$$

and

$$\sum_{j \in r} o_j^{ref} \leq Q. \tag{3}$$

To obtain the "first-stage formulation" of the problem, we define binary variables x_{ijk}^{dp} equal to 1 if and only if vehicle k, departing form depot d and delivering to plant p, visits producer n_j immediately after visiting n_i. Therefore, the first-stage formulation takes the following form:

$$\min \quad mc_k + \sum_{p \in \mathcal{P}} \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} c_{ij} x_{ijk}^{dp} + \mathcal{F}(x) \tag{4}$$

subject to

$$m = \sum_{p \in \mathcal{P}} \sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{V}} x_{djk}^{dp}; \tag{5}$$

$$\sum_{i \in \mathcal{V}} x_{ihk}^{dp} - \sum_{i \in \mathcal{V}} x_{hik}^{dp} = 0 \quad (h \in \mathcal{V}, k \in \mathcal{K}, d \in \mathcal{D}, p \in \mathcal{P}); \tag{6}$$

$$\sum_{p \in \mathcal{P}} \sum_{d \in \mathcal{D}} \sum_{j \in \mathcal{V}} x_{djk}^{dp} \leq 1 \quad (k \in \mathcal{K}); \tag{7}$$

$$\sum_{i \in \mathcal{V}} o_i^{ref} \sum_{j \in \mathcal{V}} x_{ijk}^{dp} \leq Q \quad (k \in \mathcal{K}, d \in \mathcal{D}, p \in \mathcal{P}); \tag{8}$$

$$\sum_{d \in \mathcal{D}} \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{V}} o_i^{ref} \sum_{j \in \mathcal{V}} x_{ijk}^{dp} \geq D_p^{ref} \quad (p \in \mathcal{P}); \tag{9}$$

$$\sum_{i \in \mathcal{V}} \sum_{j \in \mathcal{V}} x_{ijk}^{dp} \leq |\mathcal{S}| - 1 \quad (k \in \mathcal{K}, d \in \mathcal{D}, p \in \mathcal{P}, \mathcal{S} \subseteq \mathcal{V}, |\mathcal{S}| \geq 2); \tag{10}$$

$$x_{ijk}^{dp} \in \{1, 0\} \quad (i, j \in \mathcal{V}, k \in \mathcal{K}, d \in \mathcal{D}, p \in \mathcal{P}). \tag{11}$$

In this formulation, the objective function computes the total cost of a solution, which has three components: 1) the fixed vehicle costs; 2) the first-stage routing cost, obtained by summing the costs of the planned routes; and 3) the second-stage routing cost $\mathcal{F}(x)$, which is defined as the average recourse costs computed over the different periods of the horizon (a full definition of $\mathcal{F}(x)$ is provided in equation (12) and model (13)-(21)). Constraint (5) counts the number of vehicles. The role of constraints (6) is to ensure that when a vehicle arrives at a producer it also leaves that producer. Constraints (7) specify that each vehicle is used at most once. Limits on vehicle capacity are imposed through constraints (8). Constraints (9) guarantee that the plant demands are satisfied. Finally, constraints (10) are standard subtour elimination constraints

To define the "second-stage" problem, let d_k and p_k indicate the original depot and the plant visited by vehicle k, respectively. For the sake of simplicity, in the second-stage formulation, the fixed first-stage variable $x_{ijk}^{d_k p_k}$ is reduced to x_{ijk}, as the information regarding d_k and p_k is encoded in index k. The parameter vector o^ξ represents the supplies in period ξ. We also define z_{ijk}^ξ as the flow on arc (i,j) for all $i,j \in \mathcal{V}$ traveled by vehicle k in period ξ. Finally, define the intermediate variable w_{ik}^ξ that takes the value 1 when a failure occurs as producer n_i is serviced by vehicle k in period ξ and 0 otherwise. Therefore, z^ξ and w^ξ represent the vectors of z_{ijk}^ξ and w_{ik}^ξ, respectively. The recourse problem is defined using the flow-based formulation (13)-(21). This second-stage formulation was first proposed by Dayarian et al (2014a) for the same problem.

$$\mathcal{F}(x) = \sum_{\xi \in \Xi} W_\xi F(x, o^\xi) \tag{12}$$

where

$$F(x, o^\xi) = \min \quad \sum_{k \in \mathcal{K}} \sum_{i \in \mathcal{N}} 2 c_{i p_k} w_{ik}^\xi \tag{13}$$

subject to

$$z_{ijk}^\xi \leq Q x_{ijk} \qquad (i,j \in \mathcal{V}, k \in \mathcal{K}), \tag{14}$$

$$w_{ik}^\xi \leq \sum_{j \in \mathcal{N}} x_{ijk} \qquad (i \in \mathcal{V}, k \in \mathcal{K}), \tag{15}$$

$$\sum_{j \in \mathcal{N}} z_{d_k jk}^\xi = 0 \qquad (k \in \mathcal{K}), \tag{16}$$

$$\sum_{j \in \mathcal{N} \cup \mathcal{P}} z_{ijk}^\xi = \sum_{j \in \mathcal{N} \cup \mathcal{D}} z_{jik}^\xi + o_i^\xi - Q w_{ik}^\xi \quad (i \in \mathcal{N}, k \in \mathcal{K}), \tag{17}$$

$$\epsilon z_{ijk}^\xi \geq x_{ijk} \qquad (i,j \in \mathcal{N} \cup \mathcal{P}, k \in \mathcal{K}), \tag{18}$$

$$\sum_{j \in \mathcal{N}} w_{jk}^\xi \leq 1 \qquad (k \in \mathcal{K}), \tag{19}$$

$$z^\xi \geq 0, \tag{20}$$

$$w_{ik}^\xi \in \{0,1\} \qquad (i \in \mathcal{N}, k \in \mathcal{K}). \tag{21}$$

Equation (12) defines $\mathcal{F}(x)$ as the average recourse cost over the considered planning horizon. In a given period, for a specific first-stage solution, the recourse cost is obtained by solving model (13)-(21). The objective function (13) returns the recourse cost given a first-stage solution x with respect to the production level in a given period ξ. As mentioned before, the recourse cost corresponds to the cost of a return trip to the plant from the failure point. Constraint (14) shows that the flows are nonzero only on the arcs of the planned routes and do not exceed the vehicle capacity. Constraint (15) specifies that a failure at producer n_i on route k can occur only if n_i is visited through route k. Constraints (16) assure that vehicles depart from the depots with empty tanks. Constraints (17) define when a failure occurs at a given producer n_i on a route. Constraint (18) guarantees that in any route only the initial arc leaving from the depot can have a zero flow. Parameter ϵ is a large constant such that both ϵQ and terms ϵo_j^ξ for all $j \in N$, and $\xi \in \Xi$, are integers. Such a constant is guaranteed to exist as long as all problem data are rational numbers. Based on constraints (17) and (18), if a vehicle is filled exactly by the load collected in a producer n_j, route failure will not occur at n_j, but rather at the next producer in the route. Constraints (21) guarantee that each vehicle faces at most one failure per period.

3. Literature Review

In this section, we review metaheuristic methods for VRPs with a similar structure to our problem.

Our problem setting has some special features:

1. The need to satisfy the plant demands; our problem can be seen as a many-to-one pickup and delivery problem (PDP).
2. The need to account for the production variations, while planning over a horizon.

Lahrichi et al (2013), investigating the same dairy application, considered a variant of the VRP with features similar to those of our problem. They used a generalized version of the Unified Tabu Search (Cordeau et al, 2001). They simultaneously considered the plant deliveries, different vehicle capacities, different numbers of vehicles at each depot, and multiple depots and periods. Dayarian et al (2014b) proposed a branch-and-price algorithm for a variant of the DTPQ in which a time window is associated with each producer, and the production levels over the horizon are assumed to be fixed.

The VRPs that are most similar to our problem are the multi-period (MPVRP) nd the periodic (PVRP) settings. In most studies of theMPVRP, customers request a service that could be performed over a multi-period horizon (see Tricoire, 2006; Angelelli et al, 2007; Wen et al, 2010; Athanasopoulos, 2011). The classical MPVRP is closely related to the PVRP, in which the customers specify a service frequency and allowable combinations of visit days. Surveys of these problems and extensions can be found in Francis et al (2008) and Vidal et al (2013). The best-known algorithms for the PVRP are those of Cordeau et al (1997), Hemmelmayr et al (2009), Rahimi-Vahed et al (2013) and, particularly, Vidal et al (2012) and Vidal et al (2014). In our problem, all the producers need to be serviced every period on a daily basis. Moreover, the definition of the periods is based on seasonal variations.

A single plan for a horizon of several periods has been investigated in the context of telecommunication network design (Kouassi et al, 2009; Gendreau et al, 2006). However, apart from the work of Dayarian et al (2014a), we are not aware of any previous study of the VRP with the multi-period configuration considered in this paper. Dayarian et al (2014a) used a branch-and-price approach to solve the problem that we investigate. However, their algorithm is able to solve instances with only up to twenty producers.

There are certain similarities between our problem and the consistent vehicle routing problem (ConVRP) introduced by Groër et al (2009). In the ConVRP, customers with known demands receive service either once or with a predefined frequency over a multiple-day horizon. Frequent customers must receive consistent service, which is defined as visits from the same driver at approximately the same time throughout the planning horizon (Tarantilis et al, 2012).

Complete surveys of metaheuristics for the VRP can be found in Gendreau et al (2008) and Vidal et al (2013). They include neighborhood searches (Gendreau et al, 1994; Cordeau et al, 2001; Rousseau et al, 2002; Bräysy, 2003), population-based methods such as evolutionary and genetic algorithms (Berger et al, 2003; Bräysy and Gendreau, 2005; Vidal et al, 2012), hybrid metaheuristics (Gehring and Homberger, 1999; Bent and Van Hentenryck, 2004; Homberger and Gehring, 2005) and parallel and cooperative metaheuristics (Crainic, 2008; Crainic et al, 2009; Lahrichi et al, 2012). Of the neighborhood search methods, the large neighborhood search (LNS) algorithms (Shaw, 1998) have proven to be successful for several classes of the VRP. ALNS (Ropke and Pisinger, 2006; Pisinger and Ropke, 2007), an extension of the LNS, is also related to the ruin-and-recreate approach of Schrimpf

(2000). Recently, ALNS has provided good solutions for a wide variety of vehicle routing problems; see for instance Ropke and Pisinger (2006), Gendreau et al (2010), Azi et al (2014), and Pepin et al (2009).

The MPVRP, as considered in this paper, has to date received limited attention. Based on the success of the ALNS, we propose an ALNS for our problem. This algorithm is outlined in the next section.

4. Proposed Solution Framework

The classical ALNS algorithm, as presented by (Ropke and Pisinger, 2006; Pisinger and Ropke, 2007), is an iterative process where, at each iteration, part of the current solution is destroyed and then reconstructed in the hope of finding a better solution. The destruction phase for the VRP consists in disconnecting a number $q \in [q_{min}, q_{max}]$ of nodes from their current routes and placing them into the *unassigned node pool* Φ. Note that q_{min} and q_{max} are parameters whose values are to be tuned. The construction phase then inserts the nodes from Φ into the routes of the solution. Destruction and construction are performed by appropriate heuristics, selected at each iteration from a given set of procedures via a biased random mechanism, referred to as roulette-wheel, favoring the heuristics that have been successful in recent iterations according to certain criteria (e.g., improvement in solution quality).

Our algorithm is based on the general ALNS concept, but incorporates a number of features that improve its performance; an outline of our procedure is presented in Algorithm 1. At each iteration, we explore the neighborhood of the current solution, generating potentially φ new solutions (lines 9-17). New solutions are obtained by applying an operator *opr* $\in \Omega$ to the current solution, where Ω is the set of all operators. Contrary to classical ALNS, the operators are built through coupling each combination of destruction and construction heuristics, described in Sections 4.3 and 4.4, respectively. (A similar idea of paring heuristics was used by Kovacs et al (2012) in the context of service technician scheduling.) The main advantage is that we can weight the performance of each (destruction-construction) pair. We select the operator to apply to the solution of the current iteration via a roulette-wheel mechanism (line 12).

At the end of each iteration, we apply an acceptance criterion to the best solution among the φ solutions found (lines 18-26). This criterion is usually defined by the search paradigm applied at the master level, e.g., simulated annealing (SA) (see Kirkpatrick et al, 1983). If the solution satisfies the criterion, it replaces the current solution. That is, the new solution s' replaces the current solution s if $f(s') < f(s)$, where $f(s)$ represents the value of solution s. In SA, with $\Delta f = f(s') - f(s)$, solution s' is accepted with probability

$$\exp(\frac{-\Delta f}{T}), \tag{22}$$

where $T > 0$ is the temperature parameter. The temperature is initialized to T^{init} and is lowered in the course of the search by a cooling rate $c \in (0,1)$: $T \leftarrow cT$ (line 41). The probability of accepting worse solutions reduces as T decreases. This allows the algorithm to progressively find better local optima. We perform the cooling procedure when no global best feasible solution has been found in the last δ iterations. This can be seen as a *dynamic repetition schedule* that dynamically defines the number of iterations executed at a given temperature. This procedure divides the search into several segments, each being a series of consecutive iterations. The length of each segment corresponds to the repetition schedule for a given temperature and therefore has a minimum length of δ iterations, where δ is a parameter to be tuned. If a new global best feasible solution is found in the current segment, the length of the segment is extended for another δ iterations (line 21).

To intensify the search, at the end of each segment, we apply a series of local search (LS) operators to the best solution found in the segment (lines 31-40). If this gives an improvement, we update the current solution.

We also propose the use of an enhanced central memory, which stores high-quality solutions. We design several new destruction heuristics that use information extracted from the central memory. Moreover, we design new operators for our specific problem setting. The main components of our algorithm are described next.

4.1. Search Space

It is well known in the metaheuristic literature that allowing the search into infeasible regions may lead to good solutions. We therefore permit infeasible solutions in which the plant demands are not completely satisfied. We evaluate the moves and solutions using a penalty function $f(s) = C(s) + \eta D^{-}(s)$, where $C(s)$ is the total operating cost of the solution (i.e., fixed, routing, and recourse costs) and $D^{-}(s)$ is the unsatisfied plant demand. The parameter η is initially set to 1. After each block of $Iter^{adj}$ iterations, we multiply η by 2 if the number of infeasible solutions in the last $Iter^{his}$

Algorithm 1 ALNS

```
s ← InitialSolution;
Initialize the weights π;
Set the temperature T;
iter ← 1;
segmentIter ← 1;
seg ← 1;
s_seg ← s;
repeat
    repeat
        s_iter ← s;
        q_iter ← Number of nodes to be removed;
        Opr_iter ← Select an operator;
        s' ← Opr_iter(s, q_iter);
        if f(s') < f(s_iter) then
            s_iter ← s';
        end if
    until iter/φ == 0
    if f(s_iter) < f(s*) and s_iter feasible then
        s* ← s_iter;
        s_seg ← s_iter;
        segmentIter ← 0;
    else
        if ACCEPT(s_iter, s) then
            s ← s_iter;
        end if
    end if
    if f(s_iter) < f(s_seg) then
        s_seg ← s_iter;
    end if
    Update the score of opr;
    if segmentIter == δ then
        s' ← LOCAL SEARCH(s_seg);
        if f(s') < f(s*) then
            s* ← s';
            segmentIter ← 0;
        else
            if f(s') < f(s) then
                s ← s';
            end if
            T ← c.T;
            s_seg ← s;
            seg ← seg + 1;
        end if
    end if
    if seg/γ == 0 then
        Update the weights;
    end if
    iter ← iter + 1;
    segmentIter ← segmentIter + 1;
until Stopping Criterion
return s*
```

iterations is greater than δ_{max}, and we divide it by 2 if the number of such solutions is less than δ_{min}. The two parameters δ_{min} and δ_{max} are to be tuned.

This penalty function is similar to that used in Taburoute (Gendreau et al, 1994) and the Unified Tabu Search (Cordeau et al, 2001). Our penalty strategy favors removal from routes serving plants with an oversupply and insertion into routes servicing plants being under-supplied. We add a penalty ρ to the local cost of removal or insertion in a given position, where

$$\rho = \eta D^{-}(s). \tag{23}$$

4.2. Adaptive Search Engine

We implement an adaptive weight adjustment procedure to represent the historic performance of the operators, and use these weights to bias their selection at each iteration. A *weight* ω_{opr} is thus assigned to each operator *opr*. Initially, all the weights are set to one. We update the operator weights after each block of γ segments, based on a combination of long and short-term performance history (lines 45-46). The probability of selecting *opr* is then defined as $\omega_{opr}/\sum_{k\in\Omega}\omega_k$.

The short-term performance of the operators is captured through a scoring mechanism. A *score* is assigned to each operator, the score being set to zero initially and after each γ segments. At each iteration, we then update the scores (line 30) by adding a bonus factor $\sigma_i, i \in \{1,\ldots,4\}$, where $\sigma_i \leq \sigma_{i+1}, i \in \{1,2,3\}$, to the current score as follows:

I. σ_4 if a new global best feasible solution has been found;

II. σ_3 if the new solution improves the current solution but not the global best feasible solution;

III. σ_2 if the new solution satisfies the acceptance criterion and is inserted into Ψ_{FS};

IV. σ_1 if the new solution satisfies the acceptance criterion but is not inserted into Ψ_{FS}.

The bonus factor is zero in all other cases.

Let π_{opr} be the total score of *opr* obtained from ν_{opr} applications of *opr* in the last γ segments. We update the weight of each operator using a parameter

$\alpha \in [0, 1]$, called the *reaction factor*, through the formula

$$\omega_{opr,\iota+1} = \omega_{opr,\iota}(1 - \alpha) + \alpha \frac{\pi_{opr}}{\nu_{opr}}, \tag{24}$$

where $\omega_{opr,\iota}$ represents the weight of operator *opr* in ιth block of γ segments.

4.3. Destruction Heuristics

Several destruction heuristics have been proposed in the literature, and some can be adapted to our problem setting. We focus on the following destruction heuristics from the literature:

Worst Removal: Initially proposed by Rousseau et al (2002) and later used by Ropke and Pisinger (2006), it removes the q worst placed nodes and places them in Φ.

Route Removal: Removes a randomly selected route and places the corresponding nodes in Φ.

Cluster Removal: This heuristic (Pisinger and Ropke, 2007) removes a cluster of nodes from a route, based on their geographical region. It randomly selects a route from the current solution. It then applies the well-known Kruskal algorithm to find a minimum spanning tree for the nodes of this route, based on the arc length. When two forests have been generated, one of them is randomly chosen and its nodes are removed and placed in Φ.

Smart Removal: This heuristic (Rousseau et al, 2002) randomly selects a pivot node and removes portions of different routes around the pivot, based on a reference distance and a proximity measure.

We also define a series of memory-based destruction heuristics, which primarily differ in the way that the closeness of the removed nodes are weighted. The solution-cost-based related removal is adapted from existing heuristics proposed by Pisinger and Ropke (2007), while others are new.

Define Ψ, a *central memory* containing a limited number of solutions of two types:

- **Best Feasible Solutions (Ψ_{FS}):** A list of the β_1 best feasible solutions generated so far.

- **Best Infeasible Solutions (Ψ_{NFS}):** A list of the β_2 best infeasible solutions generated so far.

The size of the central memory follows from a trade-off between search quality on the one hand and computational efficiency and memory requirements on the other. We extract different types of information from the central memory, and use the extracted information to determine the closeness between different nodes of the graph with respect to different criteria. We design a destruction heuristic based on each criterion, obtaining the six heuristics below.

Solution-Cost-Based Related Removal

The solution-cost-based related removal heuristic, based on the historical node-pair removal (Pisinger and Ropke, 2007), associates with each arc $(u, v) \in A$ a weight $f^*(u, v)$. This weight indicates the value of the best-known solution that contains arc (u, v). Whenever a new solution is inserted into the central memory, we update the $f^*(u, v)$ value of all the arcs (u, v) in the solution.

Following a call to this heuristic, we perform a worst removal procedure in which the weight $f^*(u, v)$ replaces the cost of each arc $(u, v) \in A$. We repeat this process until q nodes have been removed and placed in Φ.

Route-Cost-Based Related Removal

The route-cost-based related removal heuristic, which is similar to the heuristic above, associates with each arc $(u, v) \in A$ a weight $r^*(u, v)$, indicating the value of the minimal-cost route found so far that contains arc (u, v). We perform a worst removal based on the $r^*(u, v)$ weights.

Paired-Related Removal

This heuristic investigates adjacent producer nodes. We give each arc (i, j) a weight $\varpi_{(i,j)}$, initially set to 0. The heuristic starts by adding a weight h_s to the weights of all the arcs used in the solutions of the central memory. When an arc (i, j) is used by solution s, we add the weight h_s to both (i, j) and (j, i). We compute h_s via $h_s = List.size() - pos_{inList}(s)$, where $List$ represents the list to which solution s belongs, $List.size()$ is the length of that list, and $pos_{inList}(s)$ is the position of solution s in that list. This procedure favors the solutions at the start of the lists. When a new solution is inserted into any of the lists, we update the weights h_s. We use the arc weights $\varpi_{(i,j)}$ to identify the q producer nodes that seem to be related to each other. An initial node n_i is randomly selected, removed, and placed in Φ. Then, while $|\Phi| < q$, we randomly select a node n_j from Φ and identify the node n_k in Φ that is the most closely related to node n_j (it has the highest $\varpi_{(j,k)}$). We then remove the node n_k and place it in Φ.

Route-Related Removal

This heuristic, similarly to the previous heuristic, adds a weight h_s to all pairs of nodes serviced by the same route in solution s. We assign weights as for the previous heuristic. We remove nodes from their current position following a similar procedure to that for the previous heuristic.

Depot-Producer-Related Removal

This heuristic attempts to identify the nodes that may be misassigned to a depot. A weight is assigned to each depot-node pair (n_d, n_i), for $d \in \mathcal{D}$ and $i \in \mathcal{N}$. The weight increases by h_s if, in solution s, producer i is assigned to a route departing from depot d. We calculate the value of h_s as for the paired-related removal heuristic. We select a node to remove via the following steps:

Step 1: We sort the producer-depot assignments in the current solution s according to the historical pair weights obtained as described above in $List_{i,d}(s)$.

Step 2: Starting from the producer-depot pair with the lowest weight, we remove nodes from their current position with probability

$$Pr_{n_i,n_{d_i}}(s) = \frac{rank(n_i)}{List_{i,d}(s).size()}, \tag{25}$$

where $rank(n_i)$ is the position of the pair (n_{d_i}, n_i) in $List_{i,d}(s)$. Moreover, $List_{i,d}(s).size()$ is the length of the node-depot list, which is the number of producer nodes. Accordingly, we remove the node with the lowest weight from its current position with probability 1.

Step 3: If the list is traversed to the end, but the number of removed nodes is less than q, we update the length of the list to $List_{i,d}(s).size() - |\Phi|$ and make the corresponding updates to the pair ranking. We then return to **Step 2.**

Plant-Producer-Related Removal

This heuristic follows the three steps above. It attempts to remove producer nodes based on the node-plant pair weights calculated from the historical information.

4.4. Construction Heuristics

After the destruction heuristic, the nodes that have been removed and placed in Φ are considered for reinsertion into routes. We consider the following construction heuristics from the literature:

Best-First Insertion: Inserts each node in the cheapest position. At each step it selects the node with the lowest insertion cost.

Regret Insertion: This heuristic (Ropke and Pisinger, 2006), orders the nodes in Φ by decreasing regret values. The regret value is the cost difference between the best insertion position and the second best. More generally, the k-regret heuristic defines the regret value with respect to the k best routes.

We also designed the following construction heuristic based on the characteristics of our problem.

Minimum-Loss Insertion This heuristic is based on the regret insertion heuristic but does not use ρ. It inserts nodes into the routes while attempting to maintain the feasibility of the solution at the minimal cost. The heuristic is based on the regret associated with the insertion of a node into a route servicing a plant with unsatisfied demand rather than in the best possible route. Clearly, the best candidate is a node for which the best possible position is in a route servicing a plant with unsatisfied demand. The best insertion candidate is determined using the following criterion:

$$n_i := \arg\min_{n_i \in \Phi} \Big(\min_{r \in \mathcal{R}_s^{D-}} (\Delta f_{r+n_i}(s)) - \min_{r \in \mathcal{R}_s} (\Delta f_{r+n_i}(s)) \Big), \qquad (26)$$

where $\mathcal{R}_s$ is the set of routes for solution s, and $\mathcal{R}_s^{D-}$ is the set of routes servicing plants with unsatisfied demand. If all the plant demands are met, the insertion order of the remaining nodes in Φ is defined as for the regret insertion operator.

4.5. Local Search

At the end of each segment, LS procedures are performed on the best solution found during the segment. Our LS procedures are inspired by the education phase of the genetic algorithm proposed by Vidal et al (2012). The procedures are restricted to the feasible region. We build each node's neighborhood using a threshold ϑ, which is computed as follows:

$$\vartheta = \frac{Z(s)}{nbArc(s)}, \tag{27}$$

where $Z(s)$ and $nbArc(s)$ are the sum of the arc costs and the number of arcs used in solution s. In our implementation, $Z(s)$ and $nbArc(s)$ are limited to the arcs between producer nodes; the recourse costs and the corresponding arcs are omitted. The value ϑ is the average length of the arcs between the producer nodes in solution s. The neighbour set of each node n_i contains all nodes n_j such that $c_{ij} \leq \vartheta$.

Suppose that n_u, assigned to route r_u, is a neighbor of n_v, assigned to route r_v. Moreover, suppose that n_x and n_y are immediate successors of n_u and n_v in r_u and r_v, respectively. For every node n_u and all of its neighbors n_v, we perform the LS operators in a random order. When a better solution is found, the new solution replaces the current solution. The LS stops when no operator generates an improved solution. The LS operators are:

Insertion 1: Remove n_u and reinsert it as the successor of n_v.

Insertion 2: Remove n_u and n_x; reinsert n_u after n_v and n_x after n_u.

Insertion 3: Remove n_u and n_x; reinsert n_x after n_v and n_u after n_x.

Swap 1: Swap the positions of n_u and n_v.

Swap 2: Swap the position of the pair (n_u, n_x) with n_v.

Swap 3: Swap the position of (n_u, n_x) with (n_v, n_y).

2-opt: If $r_u = r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_v) and (n_x, n_y).

2-opt* 1: If $r_u \neq r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_v) and (n_x, n_y).

2-opt* 2: If $r_u \neq r_v$, replace (n_u, n_x) and (n_v, n_y) with (n_u, n_y) and (n_x, n_v).

5. Bounds on the Multi-Period Solution

To evaluate the performance of our algorithm, we compute lower and upper bounds on the objective function value. This calculation is based on the set partitioning formulation of the problem (Dayarian et al, 2014a). Let the single-period problem that considers only the production levels in the reference period be Pb^{ref}, with optimal solution x^{ref}. Let Pb^{mp} be the multi-period problem, with optimal solution x^{*}.

Recall, the route cost, C, has three components: 1) fixed vehicle costs, 2) first-stage routing costs, and 3) second-stage routing costs (recourse costs). These components are denoted $c_f(x)$, $c(x)$, and $\mathcal{F}(x)$, respectively. That is, $C(x) = c_f(x)+c(x)+\mathcal{F}(x)$. For any feasible solution x to Pb^{mp}, the following inequality provides an upper bound on the value of the multi-period solution:

$$C(x^{*}) \leq C(x). \tag{28}$$

Moreover, because the fixed vehicle costs are significantly large compared to the total routing costs, the number of vehicles used in the multi-period solution is the minimum number of vehicles needed during the reference period, so the fixed vehicle costs are the same:

$$c_f(x^{*}) = c_f(x^{ref}). \tag{29}$$

Since x^{*} is also a feasible solution to P^{ref}, we have

$$c(x^{ref}) \leq c(x^{*}). \tag{30}$$

We combine (29) and (30) to obtain a lower bound on the value of the multi-period solution:

$$c_f(x^{ref}) + c(x^{ref}) \leq C(x^{*}). \tag{31}$$

We also consider a lower bound on the value of $\mathcal{F}(x^{*})$. Let $F(r,\xi)$ be the recourse cost in period $\xi \in \Xi$ for route $r \in \mathcal{R}_s$, where $\mathcal{R}_s$ is the set of routes in solution s. We have

$$\mathcal{F}(x) = \sum_{\xi \in \Xi} \sum_{r \in \mathcal{R}_s} W_{\xi} F(r,\xi). \tag{32}$$

Let the set of producer nodes visited by route r be $\mathcal{N}_r$, the plant to which r is assigned be p_r, and the set of all routes serving plant $p \in \mathcal{P}$ be $\mathcal{R}_s^p \subseteq \mathcal{R}_s$.

Then

$$F(r,\xi) \geq \quad 2\min_{i\in\mathcal{N}_r} c_{i,p_r}.t_r^{\xi} \tag{33}$$

$$\Rightarrow \mathcal{F}(x^*) \geq \quad 2\sum_{r\in\mathcal{R}_s} t_r^{\xi} \min_{i\in\mathcal{N}_r} c_{i,p_r} \tag{34}$$

$$= \quad 2\sum_{p\in\mathcal{P}}\sum_{r\in\mathcal{R}_s^p} t_r^{\xi} \min_{i\in\mathcal{N}_r} c_{i,p_r}, \tag{35}$$

where t_r^{ξ} is a binary parameter, which is equal to 1 if a failure occurs on route r in period ξ and 0, otherwise.

The minimum failure cost for a given instance can then be computed by first determining the minimum number of vehicles needed to service the plants and producers. We then assign the producers to vehicles (routes) while attempting to minimize the total failure cost. To do this, we assign failure points to the routes so that the total failure cost is minimized. The minimum number of vehicles, K^*, is obtained using equation (36).

$$K^* = \max\{\sum_{p\in\mathcal{P}}\lceil D_p/Q\rceil, \lceil\sum_{i\in\mathcal{N}} o_i/Q\rceil\}. \tag{36}$$

5.1. Minimum Failure Cost

Given the minimum number of vehicles, we can compute a lower bound on the total failure cost of Pb^{mp} based on inequality (35). We assign nodes to the restricted vehicle set $\mathcal{K}^*$, assuming that for a given route r, all the failures in different periods occur on the node that is closest to p_r. We assign the nodes by solving a bin-packing formulation that minimizes the failure cost, Table 1 displaying the notation:

$$Z = \min \quad \sum_{\xi \in \mathcal{S}} W_\xi \sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} 2c_{i,p} u^{\xi}_{ikp} \tag{37}$$

subject to

$$l_{kp} = \sum_{i \in \mathcal{N}} o_i x_{ikp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{38}$$

$$l_{kp} \leq Q y_{kp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{39}$$

$$\sum_{p \in \mathcal{P}} y_{kp} = 1 \quad (k \in \mathcal{K}^*); \tag{40}$$

$$\sum_{k \in \mathcal{K}^*} l_{kp} \geq D_p \quad (p \in \mathcal{P}); \tag{41}$$

$$\sum_{k \in \mathcal{K}^*} \sum_{p \in \mathcal{P}} x_{ikp} = 1 \quad (i \in \mathcal{N}); \tag{42}$$

$$x_{ikp} \leq y_{kp} \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{43}$$

$$P_\xi \sum_{p \in \mathcal{P}} l_{kp} \leq Q(1 + t^{\xi}_{k}) \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{44}$$

$$\sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} u^{\xi}_{ikp} = t^{\xi}_{k} \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{45}$$

$$u^{\xi}_{ikp} \leq x_{ikp} \quad (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{46}$$

$$y_{kp} \leq y_{k-1p} + y_{k-1p-1} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{47}$$

$$y_{11} = 1; \tag{48}$$

$$x_{ikp}, y_{kp}, t^{\xi}_{k}, u^{\xi}_{ikp} \in \{0,1\} \ (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*). \tag{49}$$

Constraints (38) and (39) ensure that the vehicle capacities are satisfied. Constraint (40) ensures that each vehicle is assigned to a single plant. Constraint (41) ensures that the plant demands are satisfied, and constraint (42) ensures that each producer is assigned to a single vehicle. Constraint (43) ensures that producers are assigned only to open routes. For each period ξ, constraints (44)–(46) determine the number and location of failures on each vehicle k. Constraints (47) and (48) break the possible symmetry due to the set of identical vehicles. The objective function, Z, provides a lower bound on the total failure cost. We assume that, for a given route, all the failures in different periods occur in the node that is closest to the assigned plant.

The bound can be tightened if we acknowledge that on a given route not all periods have failures at the same node. Proposition 1 provides a condition

Table 1: Bin-packing notation for the minimum failure cost formulation

Notation	Description
x_{ikp}	1 if producer i is assigned to vehicle k and plant p;
y_{kp}	1 if vehicle k serves plant p;
o_i	supply of producer $i \in \mathcal{N}$;
D_p	demand of plant $p \in \mathcal{P}$;
$\mathcal{K}^*$	set of K^* identical vehicles;
t_k^ξ	1 if a failure in period ξ is assigned to vehicle k;
u_{ikp}^ξ	1 if a failure in period ξ is assigned to producer i on vehicle k, serving plant p;
l_{kp}	quantity delivered to plant p by vehicle k.

determining when two periods both encounter failure at the same node.

Proposition 1. *Two periods ξ_1 and ξ_2 may both encounter a failure at node n_j if the following inequality holds:*

$$\frac{Q}{P_2}(1 - \frac{P_2}{P_1}) \leq o_j. \tag{50}$$

Proof 1. *Assume that $P_1 \geq P_2$ and that in period ξ_1 the quantity collected prior to node n_j is Q. The quantity collected in period ξ_2 will then be $P_2\frac{Q}{P_1}$. Moreover, ξ_2 has a failure at node n_j if $P_2\frac{Q}{P_1} + P_2 o_j \geq Q$.* □

Including this condition in the model (37)–(49) may lead to an increase in the value of Z by assigning certain failure points to nodes that are farther from the plant. This occurs when two different periods cannot both encounter failure on the closest node to the plant.

6. Computational Experiments

We describe our computational experiments in the following sequence. In Section 6.1, we introduce the set of test problems. We calibrate the parameter values via extensive sensitivity analysis; the results of these tests are presented in Section 6.2. We also study the impact of different components of the algorithm based on a series of tests, which are presented in Section 6.3. Finally, the computational results for the test problems are presented in Section 6.4.

6.1. Test Instances

We consider instances with producer set sizes ranging from 40 to 200. The instances with 40, 50, and 60 producers were originally generated by Dayarian et al (2014a). We also created a set of larger instances with 100 and 200 producers to evaluate our heuristic on larger-scaled instances. Each instance was considered with 4 or 5 periods, to represent the multi-periodic aspect of the problem. For each case with 4 or 5 periods, 5 different scenarios $\{T1, \ldots, T5\}$ were explored, differing in terms of the distribution of the period weights and the SRT level. The details of the instances considered in this paper are presented in Table 2. The production levels and period weights are the same as in Dayarian et al (2014a) and are given in Table 3.

Table 2: Specifications of test instances

Number of producers	Number of depots	Number of plants
40	$2, 3$	$2, 3$
50	$2, 3-4, 6$	$2, 3-4, 6$
60	$2, 3-4, 6$	$2, 3-4, 6$
100	$2, 3, 6$	$2, 3, 6$
200	$3, 6$	$3, 6$

Table 3: Weight and production-level distribution of the periods (Dayarian et al, 2014a)

# periods	Type 1		Type 2		Type 3		Type 4		Type 5	
	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
4	1.30	20	1.30	25	1.20	35	1.10	30	1.10	40
	1.50	10	1.50	15	1.35	20	1.20	25	1.30	30
	1.70	10	1.70	10	1.50	15	1.40	15	1.70	10
	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$	P_ξ	$W_\xi\%$
	1.00	60	1.00	50	1.00	40	1.00	30	1.00	20
5	1.30	15	1.30	20	1.20	25	1.10	25	1.10	35
	1.50	15	1.50	15	1.35	20	1.20	20	1.20	25
	1.70	5	1.70	10	1.50	10	1.40	15	1.40	15
	1.90	5	1.90	5	1.65	5	1.70	10	1.70	5

We ran our ALNS algorithm for each of the test instances and investigated its performance in terms of solution quality and computational efficiency. The algorithm was coded in C++ and the tests were run on computers with a 2.67 GHz processor and 24 GB of RAM.

6.2. Parameter Settings and Sensitivity Analysis

Similarly to most metaheuristics, changing the values of the parameters may affect the performance (but not the correctness) of the algorithm.

We tune the parameters via a blackbox optimizer (Opal Audet et al, 2012). One drawback of this optimizer is that as the number of parameters increases, the accuracy of the algorithm decreases considerably. We therefore apply a two-phase procedure, where at each phase a subset of the parameters is tuned. In the first phase, the parameters that have a greater impact on the performance of the algorithm are adjusted using the blackbox optimizer. In the second phase, the less sensitive parameters are tuned via trial-and-error. As for the selection of the parameters to be included in each phase, it was made based on extensive preliminary tests.

We tune the parameters in the first subset by first determining a range for each parameter based on preliminary tests. We then find the best value for each parameter using the Opal algorithm (Audet et al, 2012). Opal takes an algorithm and a parameter vector as input, and it outputs parameter values based on a user-defined performance measure. Opal models the problem as a blackbox optimization, which is then solved by a state-of-the-art direct search solver.

To define a performance measure for Opal, we selected a restricted set of training instances. This set included instances ranging from 40 to 200 producer nodes, with 2 to 6 depots and plants. For a given vector of parameters, we ran each instance five times and recorded the average objective function value. The performance measure is defined to be the geometric mean of the average values of the training instances. Table 4 gives the values found for the first subset of parameters.

Table 4: Parameter values found using Opal

	Parameter	Range	Value
δ	Default segment length	[50, 150]	70
φ	Inner loop length	[3, 7]	6
γ	Number of segments to update operator weights	[1, 4]	2
α	Reaction factor in weight update	[0, 1]	0.25
c	Cooling rate for SA	[0.9980, 0.9998]	0.9987

We set the initial temperature to $T^{init} = \frac{0.05C(s_0)}{|\mathcal{N}|\ln(0.5)}$, where $C(s_0)$ is the value of the initial solution. By equation (22), setting the initial temperature to $\frac{0.05C(s_0)}{\ln(0.5)}$ allows us to accept solutions that are 5% different from the current solution with a probability of 50%. The choice of these values were inspired by the tuning performed by Pisinger and Ropke (2007). Preliminary tests showed that dividing this value by the number of producers improved the results; similar results were reported by Pisinger and Ropke (2007). We set the final temperature to $T^{fin} = T^{init}c^{25000}$, allowing a minimum of 25000 iterations. Table 5 gives the resulting values for the second subset.

Table 5: Parameter values found by trial and error

Parameter		Value
$[q_{min}, q_{max}]$	Bounds on number of nodes removed q	$[\min(5, 0.05\|\mathcal{N}\|), \min(20, 0.4\|\mathcal{N}\|)]$
$Iter^{adj}$	Number of iterations after which η is updated	20
$Iter^{his}$	History used to update η	100
δ_{min} and δ_{max}	Bounds on number of infeasible solutions used to update η	30 and 45
β_1, β_2	Lengths of lists in central memory	20, 20
$\sigma_1, \sigma_2, \sigma_3, \sigma_4$	Bonus factors for adaptive weight adjustment	1, 1, 1, 2

6.3. Evaluating the Contributions of the Algorithmic Components

We studied and now demonstrate the usefulness of various components of our algorithmic framework. We first examine the performance of the different operators, followed by an evaluation of the contribution of each destruction and construction heuristic. We then examine the gain of including the heuristics pairing feature, the local search operators, and the adaptive mechanism of the algorithm. Finally, we compare our algorithm with an adapted version of the basic ALNS proposed by Pisinger and Ropke (2007). These computations are based on a representative subset of 64 instances for different size combinations. The comparison is measured based on the following metrics:

best: The best value of the routing cost (solution's total cost excluding the vehicles fixed cost) found over five runs;

avg.: The mean value of the routing costs found over the five runs.

Furthermore, the variants obtained by excluding either a pairing of heuristics, or local search operators, or the adaptive layer, as well as the basic ALNS, are also compared on the basis of the CPU time:

T (s): The average computational time over five runs.

6.3.1. Evaluating the Performances of the Operators

Table 6: Final probabilities of choosing different destruction-construction pair

Destruction Heuristic	Construction Heuristic	min %	avg. %	max %
Worst Removal	Regret Insertion	1.03	6.16	17.41
	Best-First Insertion	0.01	1.76	6.56
	Minimum-Loss Insertion	0.00	2.17	4.75
Cluster Removal	Regret Insertion	1.16	6.27	11.14
	Best-First Insertion	0.11	3.19	9.52
	Minimum-Loss Insertion	0.03	3.19	6.68
Route Removal	Regret Insertion	0.00	2.32	10.51
	Best-First Insertion	0.00	0.50	4.03
	Minimum-Loss Insertion	0.00	1.81	6.03
Smart Removal	Regret Insertion	1.55	6.86	14.32
	Best-First Insertion	0.15	2.68	8.55
	Minimum-Loss Insertion	0.62	3.69	7.93
Paired-Related Removal	Regret Insertion	2.50	7.68	11.24
	Best-First Insertion	0.67	2.55	4.49
	Minimum-Loss Insertion	0.91	3.35	5.03
Solution-Cost-Based Related Removal	Regret Insertion	2.66	8.63	15.87
	Best-First Insertion	0.12	2.88	11.49
	Minimum-Loss Insertion	0.00	1.58	5.63
Route-Cost-Based Related Removal	Regret Insertion	0.46	6.66	17.20
	Best-First Insertion	0.00	1.95	7.69
	Minimum-Loss Insertion	0.00	1.07	7.65
Depot-Producer-Related Removal	Regret Insertion	0.47	7.26	23.31
	Best-First Insertion	0.01	2.47	11.98
	Minimum-Loss Insertion	0.11	2.60	9.62
Plant-Producer-Related Removal	Regret Insertion	0.53	7.07	13.16
	Best-First Insertion	0.02	1.85	6.08
	Minimum-Loss Insertion	0.02	1.81	8.18

Table 6 provides statistics on the probabilities of selecting different operators, computed at the end of the solution process of the instances in the representative instance subset. For each operator (pair of destruction-construction heuristics), three data are given:

min: The minimum probability of being selected at the end of the solution procedure among the 64 instances;

avg.: The average probability of being selected at the end of the solution procedure, considering the 64 instances;

max: The maximum probability of being selected at the end of the solution procedure among the 64 instances.

The minimum, average and maximum probabilities are distributed in [0.00, 2.66], [0.5, 8.63] and [4.03, 23.31] intervals, respectively. Moreover, the average of the values under the columns min, avg., and max are 0.49%, 3.70% and 9.85%, respectively. The results show that considering all the instances, at some point, each operator is useful. The significant variations between the min and max final probabilities in the case of some operators, such as the Depot-Producer-Related Removal with the Regret Insertion or the Route-Cost-Based Related Removal with the Regret Insertion show the importance of the adaptation layer. Even in the case of the operator formed of Route Removal with Best Insertion, which represents the smallest average final probability, in an instance its final probability was 4.03, which is larger than 1/27, its probability if no adaptation was considered. In fact, an operator may be strongly efficient in the case of an instance, while the same operator does not contribute significantly for another instance. The results also show that the adaptive layer of the algorithm allows the probability adjustment with respect to the characteristics of each instance.

Moreover, as we see in Table 6, the final probabilities of all operators that use the Regret Insertion outweigh the other operators. However, as we will show in Section 6.3.2, the exclusion of the operators that either use the Best-First Insertion or the Minimum-Loss Insertion leads to a degradation in the performance of the algorithm. Therefore, these operators are kept in the algorithm.

6.3.2. Evaluating the Contributions of the Heuristics

Table 7 provides statistics on the removal and insertion heuristics. We ran each instance five times while excluding one heuristic and keeping the others. Whenever a heuristic is excluded, the whole block of operators using that heuristic are disabled. For each instance, we recorded the average result over the five runs of the 64 instances of the representative set. The comparison is done based on the average percentage of solution degradation (columns Best sol. deg. over five runs and Avg. deg. over five runs) and also the maximum percentage of solution degradation (columns Max best sol.

deg. over five runs and Max avg. deg. over five runs). The maximum degradation shows the maximum loss corresponding to the exclusion of the block of operators using a specific heuristic in at least one of the instances of the representative set. Note that the values in the first two columns of Table 7 indicate the degradation in the geometric mean of the values obtained for all the instances in the considered subset. We use the geometric mean because the subset includes problems of different sizes with varying objective values. With the geometric mean the degradation's in smaller instances' objectives is not dominated by the larger ones.

Table 7: Evaluation of contribution (%) of each heuristic

Heuristic	Best sol. deg. over five runs	Avg. deg. over five runs	Max best sol. deg. over five runs	Max avg. deg. over five runs
Worst Removal	0.02	0.11	0.97	0.85
Cluster Removal	0.02	0.07	0.72	0.65
Route Removal	0.12	0.21	1.40	1.73
Smart Removal	0.03	0.11	0.74	0.90
Paired-Related Removal	0.08	0.13	1.37	1.23
Solution-Cost-Based Related Removal	0.10	0.16	1.53	1.35
Route-Cost-Based Related Removal	0.04	0.09	0.68	1.07
Depot-Producer-Related Removal	0.09	0.15	1.50	1.21
Plant-Producer-Related Removal	0.15	0.20	1.75	1.44
Regret Insertion	0.18	0.32	1.27	2.27
Best-First Insertion	0.03	0.08	0.92	0.86
Minimum-Loss Insertion	0.13	0.19	1.39	1.10

These results indicate the usefulness of all of our destruction and construction heuristics in the case of this problem setting. Overall, the plant-producer-related removal is the most efficient removal heuristic, followed by the route removal and Solution-Cost-Based Related Removal heuristics. Regret insertion is the most useful insertion heuristic, followed by the minimum-loss insertion heuristic.

6.3.3. Evaluating the Performance of Destruction-Construction Heuristics Pairing

Table 8 synthesizes results on the contribution of particular algorithmic components. It provides, in particular, the average deterioration of the variant of the algorithm in which destruction and construction heuristics are considered individually rather than in pairs. This is equivalent to consider two separate pools of heuristics (destruction and construction), while the choice of heuristics from each pool is performed independently. In this variant, at the end of each iteration of the algorithm, the scores of the two heuristics used are incremented using the bonus factors, presented in Section 4.2.

The figures in Table 8 show that when no pairing is used to define the operators, the results observed deteriorate, on average, by 0.23%, 0.35% and 27.04% respectively for the best solution observed over five runs, the average solution quality obtained over five runs and the overall CPU time. Based on these results the use of the heuristics paring is motivated.

Table 8: Evaluation of contribution of algorithmic components

Algorithm	Best sol. degradation over five runs (%)	Ave. degradation over five runs (%)	CPU degradation (%)
No Heuristic Pairing	0.23	0.35	27.04
No Local Search	0.26	0.23	-2.04
No Adaptation	0.11	0.16	-5.73
Basic ALNS	2.54	3.82	75.51

6.3.4. Evaluating the Contribution of the Local Search

Table 8 also compares the results of our algorithm with the variant in which the local search operators at the end of each segment are disabled. The absence of the local search operators in the algorithm incurs a degradation of 0.26 % in the value of the best solution over five runs, a degradation of 0.23% in the average value of the five runs while causing a gain of 2.04 % in the CPU time. Considering the trade-off between CPU time and solution quality, it seems valuable to include the local search operators in the algorithm.

6.3.5. Evaluating the Contribution of the Adaptive Layer

We now turn to the impact of excluding the adaptive layer of the ALNS framework. This translates into the variant of the algorithm in which the roulette-wheel mechanism selects the operators equiprobably. We have already demonstrated the usefulness of each of our heuristics. When the adaptation is disabled, one may expect a significant deterioration in the performance if some less useful heuristics are kept. As shown in Table 8, the routing cost deterioration while disabling the adaptation mechanism is 0.11 % on the best solution over five runs and 0.16 % on the mean value of the five runs.

The relatively limited gains obtained using the adaptive layer tend to demonstrate the robustness of our algorithm (all operators defined are useful). Nonetheless, this algorithmic feature serves an important role. As illustrated in Section 6.3.1, given the number of operators that are used, the adaptive layer enables the algorithm to improve its choices iteratively by tailoring the selection probabilities according to the observed efficiency of the operators on the specific instance a. Accordingly, the use of the adaptive layer is motivated.

6.3.6. Evaluating the Performance of the basic ALNS

We finally compare the results obtained from our implementation of the basic ALNS introduced by Pisinger and Ropke (2007) with those obtained from our proposed algorithm. This translates in disabling several additional features proposed in this paper. These modifications are:

- Destruction-construction heuristics pairing is disabled. Each destruction or construction heuristic is treated separately;
- At each iteration, instead of φ neighbors of the current solution, only one neighbor is explored. In return the number of iterations before stopping the algorithm is set to 25000φ;
- The repetition schedule in the master level is disabled. This is equivalent to lowering the temperature in the SA mechanism at the end of each iteration.;
- Following the previous point, the weight adjustment of the heuristics is not performed dynamically: we adjust the weights after $\delta\gamma$ iterations;
- The local search operators are disabled;
- A noise to the insertion cost was added as described in Ropke and Pisinger (2006);
- A large penalty associated with infeasible solutions is added, as Pisinger and Ropke (2007) consider only feasible solutions;
- The list of employed destruction and construction heuristics in this variant is:
  - Random Removal;
  - Worst Removal;
  - Cluster Removal;
  - Route Removal;
  - Solution-Cost-Based Related Removal (Historical node-pair removal);

- Paired-Related Removal (Historical request-pair removal);
- Regret Insertion;
- Best Insertion.

Note that the historical request-pair removal proposed by Pisinger and Ropke (2007) is based on the memory of the top 100 solutions. Accordingly, we replace our central memory with a list of the top 100 solutions.

As reported in Table 8, our implementation of the ALNS based on the algorithm proposed by Pisinger and Ropke (2007) resulted in solutions with an average degradation in the value of the routing cost of 2.54% and 3.82% on the best and average over five runs, respectively. The larger CPU time (75.51% more) can be explained by the use of a larger number of iterations.

6.4. Computational Results

Detailed results obtained by applying our algorithm to the instances described in Section 6.1 are given in Tables 9 –16, where:

Bounds on opt. sol. are the lower and upper bounds obtained as described in Section 5;

BKS DCGR is the optimal solution from Dayarian et al (2014a), whenever it is available;

T (s) DCGR is the computational time of the branch-and-price algorithm of Dayarian et al (2014a);

ALNS best over 5 is the best solution found over 5 runs of the ALNS;

ALNS avg. over 5 is the average of the solutions found over the 5 runs;

% dev. total cost is the standard deviation of the total cost from the ALNS best over the 5 runs;

% dev. routing cost is the standard deviation of the routing cost from the ALNS best over the 5 runs;

T (s) ALNS avg. is the average computational time of the five runs;

% dev. ALNS best from DCGR is the deviation of the ALNS best from the BKS DCGR;

% dev. ALNS best from LB is the deviation of the ALNS best from the lower bound reported in column "**Bounds on opt. sol.**";

% dev. DCGR from LB is the deviation of the the BKS DCGR from the lower bound reported in column "**Bounds on opt. sol.**".

For the smaller instances (with 40, 50, and 60 producers), some optimal solutions are reported in Tables 9 – 13 in column BKS DCGR. We also generate lower and upper bounds as described in Section 5. The lower bound has two parts: 1) the value of the optimal solution for the VRP for the reference period, and 2) a lower bound on the total recourse cost, based on the bin-packing formulation described in Section 5. For the first part, we adapt the algorithm proposed by Dayarian et al (2014b) for the deterministic variant of the problem to solve the VRP corresponding to the reference period. This algorithm can solve some instances with up to 60 producers; we do not report bounds for larger problems. We solved the bin-packing formulation using Cplex 12.6. We compute the upper bound by evaluating the cost of the solution to the reference period, based on the objective function of the multi-period problem.

Table 9 gives the results for the instances with 40 producers. Results show that in the case of 20 out of the 29 instances with known optimal solutions, the best solution obtained by ALNS over 5 runs corresponds to the optimal solution. Moreover the average optimality gap of the best ALNS solutions over these 29 instances is 0.02%. The average deviation of the best ALNS solutions from the lower bound over the 34 instances for which the lower bound is available is 1.29%. We also calculated the deviation of the BKS DCGR from the lower bound, for the cases where both these values are available. The average deviation BKS DCGR from the lower bound over the 24 instances for which the BKS DCGR and the lower bound are available was 1.28 %. The similitude between the deviations from the lower bound in the case of the BKS DCGR and the ALNS Best shows the quality of the ANLS Best even when the BKS DCGR is not available for the basis of comparison. In terms of CPU time, in the case of the instances with 40 producers, the gain of using the ALNS compared to the exact solution method is significant (29 seconds vs. 6128 seconds on average).

Tables 10 and 11 report the results for the instances with 50 producers. We divided these instances into two groups with 2/3 or 4/6 depots and plants. Results show that, on average, an increase in the number of depots or plants does not necessarily affect the performance of the ALNS. A smaller number of available optimal solutions in the case of the BKS DCGR for the instances with a larger number of depot/plant shows the limits of the exact method. However, the comparison of the average optimality gap (% dev ALNS best from DCGR) in Tables 10 and 11, 0.05 % vs. 0.03 %, shows that the ALNS dealt well facing an increase in the number of depots/plants. Moreover, in Table 10, in the case of 17 out of 24 instances for which the optimal solutions are available, the ALNS best coincides with the optimal value. In terms of CPU, comparing the computation time of those 24 instances reached optimality using the algorithm of DCGR and the 40 instances solved by the ALNS, we observe a significant reduction (4509 vs. 42 seconds). As for the second part of instances with 50 producers, reported in Table 11, the ALNS best corresponds to the optimal solution BKS DCGR in the case of 7 instances out of 12 with known optimal solutions. The comparison of CPU based on only those 12 instances solved by the algorithm DCGR and all the 40 instances solved by the ALNS reveals a decrease from over 6300 seconds to 80 seconds. Similar to the case of the instances with 40 producers, comparable values representing the average deviation of ALNS best from the lower bound and the average deviation of DCGR from the lower bound , whenever the corresponding values are available. This further supports the claim that our ALNS is able to provide high-quality results (1.24 vs. 1.34 and 1.13 vs. 0.80).

Tables 12 and 13 show results for instances with 60 producers. The analyses of the results are more limited, as less information regarding the optimal solution values and the lower bounds is available for these instances. It can be observed that increasing the number of depots/plants made the problems harder on average. This is obvious from a larger average deviation of ALNS best from BKS DCGR in the case of instances with larger numbers of depots/plants. Moreover, an increase in the value of the average deviation of the routing cost from the best ALNS comparing to instances with 40 and 50 producers shows the higher difficulty of these instances. Similar to the results obtained for the instances with 40 and 50 producers, a significant reduction in CPU time is observed in the case of the instances with 60 producers (part 1: 8164 seconds for the exact algorithm vs. 55 second for the ALNS, part 2: 6297 seconds for the exact algorithm vs. 111 seconds for the ALNS).

Overall, an increase in the number of depots and/or plants (which potentially leads to a larger number of routes to be included in the solution), increases the average CPU time (e.g., in the case of instances with 40 producers: 22 seconds for 2D2P4S vs. 32 seconds for 3D3P4S, in the case of instances with 50 producers: 33 seconds for 2D2P4S vs. 100 for 6D6P4S, and in the case of instances with 60 producers: 41 second in the case of 2D2P4S vs. 145 seconds in the case of 6D6P4).

The results for the instances with 100 and 200 producers, reported in Tables 14 – 16, show that larger problems are more difficult. Increasing the number of plants has a greater impact than increasing the number of depots, on both the computational time and the deviation from the best solution obtained by restarting. It is however noticeable that the meta-heuristic we propose is able to generate good-quality solutions within low computational efforts even for these difficult instances.

7. Conclusions

We have investigated the design of tactical plans for a transportation problem inspired by real-world milk collection in Quebec. To take the seasonal variations into account, we modeled the problem as a multi-period VRP. We developed an ALNS algorithm incorporating several heuristics for this VRP.

We tested the algorithm on a large set of instances of different sizes. The results for the smaller instances were compared with the existing exact solutions in the literature. For the larger instances, where optimal solutions were not available, we computed lower and upper bounds on the value of the solution.

While the problem investigated in this paper is rather specific, we believe that many insights gained from the application of the proposed method to this problem could be extended to other complex vehicle routing problems.

Future research will include more attributes and constraints such as soft time windows on the collection, restrictions on the route length, and heterogeneous fleets of vehicles. We also plan to consider the situation where a vehicle may perform several deliveries to more than one plant per day. It would also be interesting to take into account the daily variations in the production levels. This transforms the problem into a VRP with stochastic demands.

8. Acknowledgements

Partial funding for this project was provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), through its Industrial

Research Chair, Collaborative Research and Development, and Discovery Grant programs. We also received support from the Fonds de recherche du Québec - Nature et technologies (FRQ-NT) through its Team Research Project program, and from the EMME/2-STAN Royalty Research Funds (Dr. Heinz Spiess). We also gratefully acknowledge the support of Fonds de recherche du Québec through infrastructure grants and of the support of Calcul Québec and Compute Canada through access to their high-performance computing infrastructure.

Angelelli E, Grazia S M, Savelsbergh MWP (2007) Competitive analysis for dynamic multiperiod uncapacitated routing problems. Networks 49(4):308–317

Athanasopoulos T (2011) The multi-period vehicle routing problem and its applications. PhD thesis, Department of Financial & Management Engineering, University of the Aegean, Chios, Greece

Audet C, Dang KC, Orban D (2012) Optimization of algorithms with OPAL. Tech. rep., GERAD

Azi N, Gendreau M, Potvin JY (2014) An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Computers & Operations Research 41:167–173

Bent R, Van Hentenryck P (2004) A two-stage hybrid local search for the vehicle routing problem with time windows. Transportation Science 38(4):515–530

Berger J, Barkaoui M, Bräysy O (2003) A route-directed hybrid genetic approach for the vehicle routing problem with time windows. INFOR 41:179–194

Bräysy O (2003) A reactive variable neighborhood search for the vehicle routing problem with time windows. INFORMS Journal on Computing 15:347–368

Bräysy O, Gendreau M (2005) Vehicle routing problem with time windows, part II: Metaheuristics. Transportation Science 39(1):119–139

Cordeau JF, Gendreau M, Laporte G (1997) A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30(2):105–119

Cordeau JF, Laporte G, Mercier A (2001) A unified tabu search heuristic for vehicle routing problems with time windows. Journal of The Operational Research Society 52:928–936

Crainic TG (2008) Parallel solution methods for vehicle routing problems. In: Golden B, Raghavan S, Wasil E (eds) The Vehicle Routing Problem: Latest Advances and New Challenges, Operations Research/Computer Science Interfaces, vol 43, Springer, New York, pp 171–198

Crainic TG, Crişan GC, Gendreau M, Lahrichi N, Rei W (2009) Multi-thread integrative cooperative optimization for rich combinatorial problems. In: IPDPS, pp 1–8

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Management Science 6(1):80–91

Dayarian I, Crainic TG, Gendreau M, Rei W (2014a) A branch-and-price approach for a multi-period vehicle routing problem. Computers & Operations Research DOI http://dx.doi.org/10.1016/j.cor.2014.06.004

Dayarian I, Crainic TG, Gendreau M, Rei W (2014b) A column generation approach for a multi-attribute vehicle routing problem. European Journal of Operational Research DOI http://dx.doi.org/10.1016/j.ejor.2014.09.015

Francis PM, Smilowitz KR, Tzur M (2008) The period vehicle routing problem and its extensions. In: Golden B, Raghavan S, Wasil E (eds) The Vehicle Routing Problem: Latest Advances and New Challenges, Operations Research/Computer Science Interfaces, vol 43, Springer, New York, pp 73–102

Gehring H, Homberger J (1999) A parallel hybrid evolutionary metaheuristic for the vehicle routing problem with time windows. In: Miettinen K, Makela M, Toivanen J (eds) Proceedings of EUROGEN99–Short Course on Evolutionary Algorithms in Engineering and Computer Science, pp 57–64

Gendreau M, Hertz A, Laporte G (1994) A tabu search heuristic for the vehicle routing problem. Management Science 40(10):1276–1290

Gendreau M, Potvin JY, Smires A, Soriano P (2006) Multi-period capacity expansion for a local access telecommunications network. European Journal of Operational Research 172:1051–1066

Gendreau M, Potvin JY, Bräysy O, Hasle G, Løkketangen A (2008) Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography. In: Golden B, Raghavan S, Wasil E (eds) The Vehicle Routing Problem - Latest Advances and New Challenges, Springer Verlag, Heidelberg

Gendreau M, Potvin JY, Bräysy O, Hasle G, Løkketangen A (2010) Large neighborhood search. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, vol 146, International Series in Operations Research & Management Science, Springer, Boston, pp 399–419

Golden B, Raghavan S, Wasil EA (2008) The Vehicle Routing Problem: Latest Advances and New Challenges. Operations Research/Computer Science Interfaces Series, 43, Springer

Groër C, Golden B, Wasil E (2009) The consistent vehicle routing problem. Manufacturing & Service Operations Management 11(4):630–643

Hemmelmayr VC, Doerner KF, Hartl RF (2009) A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research 195(3):791–802

Homberger J, Gehring H (2005) A two-phase hybrid metaheuristic for the vehicle routing problem with time windows. European Journal of Operational Research 162(1):220–238

Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220:671–680

Kouassi R, Gendreau M, Potvin JY, Soriano P (2009) Heuristics for multi-period capacity expansion in local telecommunications networks. Journal of Heuristics 15(4):381–402

Kovacs AA, Parragh SN, Doerner KF, Hartl RF (2012) Adaptive large neighborhood search for service technician routing and scheduling problems. Journal of Scheduling 15(5):579–600

Lahrichi N, Crainic TG, Gendreau M, Rei W, Crişan GC, Vidal T (2012) An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Tech. rep., CIRRELT, Montreal

Lahrichi N, Crainic T, Gendreau M, Rei W, Rousseau LM (2013) Strategic analysis of the dairy transportation problem. Journal of Operational Research Society URL http://dx.doi.org/10.1057/jors.2013.147

Lenstra JK, Kan AHGR (1981) Complexity of vehicle routing and scheduling problems. Networks 11:221–227

Pepin AS, Desaulniers G, Hertz A, Huisman D (2009) A comparison of five heuristics for the multiple depot vehicle scheduling problem. Journal of Scheduling 12(1):17–30

Pisinger D, Ropke S (2007) A general heuristic for vehicle routing problems. Computers & Operations Research 34:2403–2435

Rahimi-Vahed A, Crainic TG, Gendreau M, Rei W (2013) A path relinking algorithm for a multi-depot periodic vehicle routing problem. Journal of Heuristics 19(3):497–524

Ropke S, Pisinger D (2006) An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows. Transportation Science 40:455–472

Rousseau LM, Gendreau M, Pesant G (2002) Using constraint-based operators to solve the vehicle routing problem with time windows. Journal of Heuristics 8:43–58

Schrimpf G (2000) Record breaking optimization results using the ruin and recreate principle. Journal of Computational Physics 159(2):139–171

Shaw P (1998) Using constraint programming and local search methods to solve vehicle routing problems. In: Maher M, Puget JF (eds) Principles and Practice of Constraint Programming – CP98, Lecture Notes in Computer Science, vol 1520, Springer Berlin Heidelberg, pp 417–431

Tarantilis C, Stavropoulou F, Repoussis P (2012) A template-based tabu search algorithm for the consistent vehicle routing problem. Expert Systems with Applications 39(4):4233–4239

Toth P, Vigo DE (eds) (2002) The Vehicle Routing Problem, vol 9. Society for Industrial and Applied Mathematics

Tricoire F (2006) Optimization des tournées de véhicules et de personnels de maintenance: Application à la distribution et au traitement des eaux. PhD thesis, IRCCyN - Institut de Recherche en Communications et en Cybernétique de Nantes, Nantes, France

Vidal T, Crainic TG, Gendreau M, Lahrichi N, Rei W (2012) A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research 60(3):611–624

Vidal T, Crainic TG, Gendreau M, Prins C (2013) Heuristics for multi-attribute vehicle routing problems: A survey and synthesis. European Journal of Operational Research 231(1):1–21

Vidal T, Crainic TG, Gendreau M, Prins C (2014) A Unified Solution Framework for Multi-Attribute Vehicle Routing Problems. European Journal of Operational Research 234(3):658–673

Wen M, Cordeau JF, Laporte G, Larsen J (2010) The dynamic multi-period vehicle routing problem. Computers & Operations Research 37(9):1615–1623

Table 9: Results for instances with 40 producers

	Instance	Bounds on opt. sol.	BKS DCGR	T (s) DCGR	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s) ALNS avg.	% dev ALNS best from DCGR	% dev ALNS best from LB	% dev DCGR from LB
	pr-40-2D2P4S-T1	[11864.8, 12128.9]			12048.20	12048.6	0.01	0.03	24		1.55	
2 depots	pr-40-2D2P4S-T2	[11557.2, 11840.2]	11823.9	13800	**11823.9**	11823.9	0.00	0.00	21	0.00	2.31	2.31
2 plants	pr-40-2D2P4S-T3	[9738.9, 9991.16]	9863.76	3870	**9863.76**	9865.91	0.03	0.14	29	0.00	1.28	1.28
4 periods	pr-40-2D2P4S-T4		11466.5	1711	**11466.5**	11468.7	0.04	0.20	16	0.00		
	pr-40-2D2P4S-T5	[11555.4, 11793.8]	11691.7	1219	**11698.7**	11698.7	0.00	0.00	22	0.06	1.24	1.18
	pr-40-2D2P5S-T1	[11473.7, 11658]			11628.50	11628.8	0.01	0.03	26		1.35	
2 depots	pr-40-2D2P5S-T2		11489	1866	**11489**	11489	0.00	0.00	18	0.00		
2 plants	pr-40-2D2P5S-T3	[9725.6, 9937.98]	9855.34	6236	**9855.34**	9855.34	0.00	0.00	29	0.00	1.33	1.33
5 periods	pr-40-2D2P5S-T4	[11680.8, 11946.6]	11882.7	2641	**11882.7**	11882.7	0.00	0.00	25	0.00	1.73	1.73
	pr-40-2D2P5S-T5	[11992.7, 12333.9]	12108	3998	12114.00	12115.6	0.02	0.06	31	0.05	1.01	0.96
	pr-40-2D3P4S-T1	[11550.9, 11679.6]	11648.8	9529	**11648.8**	11648.8	0.00	0.00	26	0.00	0.85	0.85
2 depots	pr-40-2D3P4S-T2	[11727.5, 11921.5]	11874.4	11328	**11874.4**	11876.3	0.02	0.10	27	0.00	1.25	1.25
3 plants	pr-40-2D3P4S-T3	[11932.3, 12204.6]	12127.2	3268	**12127.2**	12127.2	0.00	0.00	26	0.00	1.63	1.63
4 periods	pr-40-2D3P4S-T4	[13414.2, 13903.3]			13661.20	13665.8	0.04	0.18	28		1.84	
	pr-40-2D3P4S-T5	[11885.8, 12220.5]	12051.8	14214	**12051.8**	12051.8	0.00	0.00	29	0.00	1.40	1.40
	pr-40-2D3P5S-T1	[11830.3, 11961.3]	11935.4	17992	**11935.4**	11935.4	0.00	0.00	29	0.00	0.89	0.89
2 depots	pr-40-2D3P5S-T2	[13482.6, 13796.4]			13697.80	13698.1	0.01	0.02	29		1.60	
3 plants	pr-40-2D3P5S-T3	[11899.8, 12120.7]	12075.1	7726	**12075.1**	12080	0.05	0.20	27	0.00	1.47	1.47
5 periods	pr-40-2D3P5S-T4		11833.3	1675	**11833.3**	11836	0.03	0.11	33	0.00		
	pr-40-2D3P5S-T5	[11509.4, 11634.9]	11593.3	1920	11599.90	11604.1	0.04	0.18	32	0.06	0.79	0.73
	pr-40-3D2P4S-T1	[11139.5, 11277.3]	11245.7	575	**11245.7**	11245.7	0.00	0.00	26	0.00	0.95	0.95
3 depots	pr-40-3D2P4S-T2				10095.50	10095.5	0.00	0.00	25			
2 plants	pr-40-3D2P4S-T3	[9789.62, 10040.5]			9899.19	9903.43	0.06	0.24	31		1.12	
4 periods	pr-40-3D2P4S-T4	[12237.3, 12361.8]	12308.5	4903	**12308.5**	12308.5	0.00	0.00	22	0.00	0.58	0.58
	pr-40-3D2P4S-T5	[9616.2, 9904.28]	9802.38	11247	9802.64	9802.64	0.00	0.00	27	0.00	1.94	1.94
	pr-40-3D2P5S-T1	[9605.56, 9787.56]	9750.78	2081	**9750.78**	9750.78	0.00	0.00	25	0.00	1.51	1.51
3 depots	pr-40-3D2P5S-T2	[12127, 12366.1]	12353.2	10346	12353.40	12353.4	0.00	0.00	22	0.00	1.87	1.87
2 plants	pr-40-3D2P5S-T3	[9765.74, 9966.62]			9879.28	9880.55	0.02	0.08	35		1.16	
5 periods	pr-40-3D2P5S-T4		10184.4	2229	**10184.4**	10184.4	0.00	0.00	26	0.00		
	pr-40-3D2P5S-T5	[11133.9, 11367]			11264.80	11264.8	0.00	0.00	32		1.18	
	pr-40-3D3P4S-T1		11713.9	5572	11715.40	11717.5	0.02	0.10	31	0.01		
3 depots	pr-40-3D3P4S-T2	[11031.8, 11195.3]	11139.9	893	11164.70	11164.7	0.00	0.00	32	0.22	1.20	0.98
3 plants	pr-40-3D3P4S-T3	[11640.3, 11881.5]	11830	5014	11832.10	11837	0.05	0.22	34	0.02	1.65	1.63
4 periods	pr-40-3D3P4S-T4	[11426.4, 11643.5]	11547.3	14750	**11547.3**	11547.3	0.00	0.00	29	0.00	1.06	1.06
	pr-40-3D3P4S-T5	[11207.4, 11428.9]			11307.60	11307.6	0.00	0.00	33		0.89	
	pr-40-3D3P5S-T1	[11243, 11380]			11344.00	11344	0.00	0.00	34		0.90	
3 depots	pr-40-3D3P5S-T2	[11512.2, 11684.8]			11639.90	11639.9	0.00	0.00	28		1.11	
3 plants	pr-40-3D3P5S-T3	[11611.1, 11827.8]	11798.8	7902	**11798.8**	11800.8	0.03	0.12	36	0.00	1.62	1.62
5 periods	pr-40-3D3P5S-T4	[11500.8, 11695.6]	11618.6	6668	**11618.6**	11620.4	0.02	0.09	30	0.00	1.02	1.02
	pr-40-3D3P5S-T5	[11210.8, 11400.9]	11269.8	2536	11282.20	11282.2	0.00	0.00	35	0.11	0.64	0.53
	Avg.			6128 (29)			0.01 (40)	0.05 (40)	28 (40)	0.02 (29)	1.29 (34)	1.28 (24)

Table 10: Results for instances with 50 (1) producers

	Instance	Bounds on opt. sol.	BKS DCGR	T (s) DCGR	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s) ALNS avg.	% dev ALNS best from DCGR	% dev ALNS best from LB	% dev DCGR from LB
	pr-50-2D2P4S-T1	[11744.2, 11885.8]	11877.4	2405.58	11877.4	11877.4	0.00	0.00	25	0.00		1.13
2 depots	pr-50-2D2P4S-T2				13686.3	13692.1	0.05	0.20	33			
2 plants	pr-50-2D2P4S-T3	[11844.2, 12090.1]	12035.5	2951.33	12037.3	12037.5	0.00	0.02	30	0.01		1.62
4 periods	pr-50-2D2P4S-T4	[11514.7, 11782.1]			11669.4	11669.4	0.00	0.00	34		1.34	
	pr-50-2D2P4S-T5				13868.5	13869.1	0.01	0.03	42			
	pr-50-2D2P5S-T1		13657.2	1008.49	**13657.2**	13661.1	0.04	0.16	43	0.00		
2 depots	pr-50-2D2P5S-T2	[11471.2, 11698.6]	11634.2	2153.47	**11634.2**	11635.1	0.02	0.07	38	0.00		1.42
2 plants	pr-50-2D2P5S-T3	[11794.8, 12010.3]	11968.3	1596.54	**11968.3**	11968.3	0.00	0.00	32	0.00		1.47
5 periods	pr-50-2D2P5S-T4		13744	9383.57	**13744**	13744	0.00	0.00	36	0.00		
	pr-50-2D2P5S-T5	[12049.6, 12185.9]			12185.9	12185.9	0.00	0.00	34		1.13	
	pr-50-2D3P4S-T1	[13221.3, 13449.7]			13399.9	13400.1	0.00	0.02	39		1.35	
2 depots	pr-50-2D3P4S-T2	[15806, 16235.5]	16145.2	1528.7	**16145.2**	16146.7	0.01	0.06	41	0.00		2.15
3 plants	pr-50-2D3P4S-T3				15709	15713.6	0.03	0.14	41			
4 periods	pr-50-2D3P4S-T4				12332.4	12334.5	0.03	0.10	30			
	pr-50-2D3P4S-T5				15676.4	15687.7	0.08	0.34	39			
	pr-50-2D3P5S-T1		15463.5	4878.09	15464.8	15481.8	0.13	0.56	38	0.01		
2 depots	pr-50-2D3P5S-T2				12201.8	12201.8	0.00	0.00	32			
3 plants	pr-50-2D3P5S-T3		15688.4	3060.48	15701.3	15703.2	0.01	0.06	48	0.08		
5 periods	pr-50-2D3P5S-T4	[15918.2, 16359]	16195.7	7951.72	16207	16215.4	0.07	0.27	52	0.07		1.74
	pr-50-2D3P5S-T5	[13299.1, 13511.3]			13482.4	13482.5	0.00	0.00	51		1.38	
	pr-50-3D2P4S-T1	[13495.1, 13731.3]			13669.2	13669.2	0.00	0.00	40		1.29	
3 depots	pr-50-3D2P4S-T2				14190.2	14190.4	0.00	0.01	37			
2 plants	pr-50-3D2P4S-T3				13963.2	13963.2	0.00	0.00	39			
4 periods	pr-50-3D2P4S-T4	[13184.5, 13457.6]			13305.7	13305.7	0.00	0.00	41		0.92	
	pr-50-3D2P4S-T5	[13247.1, 13570.4]	13423.9	6900.11	**13423.9**	13429.8	0.06	0.29	47	0.00		1.33
	pr-50-3D2P5S-T1	[13177.4, 13371.5]	13359.2	5660.5	**13359.2**	13359.8	0.01	0.03	45	0.00		1.38
3 depots	pr-50-3D2P5S-T2		14570	4916.23	14570	14571.3	0.01	0.04	45	0.00		
2 plants	pr-50-3D2P5S-T3				13862	13862.3	0.00	0.02	40			
5 periods	pr-50-3D2P5S-T4		14314	8315.24	14314	14317.5	0.03	0.13	38	0.00		
	pr-50-3D2P5S-T5	[13608.8, 13887.6]	13756.1	691.23	**13756.1**	13772.9	0.15	0.63	49	0.00		1.08
	pr-50-3D3P4S-T1	[11647.2, 11824.8]	11778.8	10131	**11778.8**	11785.1	0.08	0.36	42	0.00		1.13
3 depots	pr-50-3D3P4S-T2		15029.4	6451.89	**15029.4**	15030.6	0.01	0.04	44	0.00		
3 plants	pr-50-3D3P4S-T3		15402.1	378.3	**15402.1**	15407.9	0.05	0.24	53	0.00		
4 periods	pr-50-3D3P4S-T4	[15018.4, 15444]	15136.5	2375.72	**15136.5**	15139.8	0.03	0.13	51	0.00		0.79
	pr-50-3D3P4S-T5	[14896.1, 15167.4]	15052.3	2957.44	**15052.3**	15053.6	0.01	0.07	49	0.00		1.05
	pr-50-3D3P5S-T1	[14909.1, 15103.7]	15072.3	3972.12	**15072.3**	15074.7	0.03	0.13	50	0.00		1.09
3 depots	pr-50-3D3P5S-T2	[15132.4, 15533.8]	15340.7	13820.4	**15340.7**	15340.7	0.00	0.00	54	0.00		1.38
3 plants	pr-50-3D3P5S-T3		15415.3	1540.71	**15415.3**	15421	0.05	0.24	58	0.00		
5 periods	pr-50-3D3P5S-T4		15022.5	3184.63	**15022.5**	15022.5	0.00	0.00	48	0.00		
	pr-50-3D3P5S-T5	[11758.8, 11991.5]			11905	11908.3	0.04	0.15	55		1.24	
	Avg.			4509 (24)			0.03 (40)	0.11 (40)	42 (40)	0.05 (24)	1.24 (7)	1.34 (14)

Table 11: Results for instances with 50 (2) producers

	Instance	Bounds on opt. sol.	BKS DCGR	T (s) DCGR	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s) ALNS avg.	% dev ALNS best from DCGR	% dev ALNS best from LB	% dev DCGR from LB
	pr-50-4D4P4S-T1				15234	15250.7	0.12	0.59	63			
4 depots	pr-50-4D4P4S-T2		15422.3	2112	15429.6	15430	0.00	0.02	64	0.05		
4 plants	pr-50-4D4P4S-T3		15511.3	13351	**15511.3**	15513.3	0.01	0.06	67	0.00		
4 periods	pr-50-4D4P4S-T4	[15320.7, 15605]			15429.1	15432.2	0.02	0.10	55		0.71	
	pr-50-4D4P4S-T5	[14728.2, 14880.9]	14811.3	667	**14811.3**	14811.3	0.00	0.00	70	0.00	0.56	0.56
	pr-50-4D4P5S-T1	[14755.5, 14935.2]	14870.7	1196	**14870.7**	14870.7	0.00	0.00	74	0.00	0.78	0.78
4 depots	pr-50-4D4P5S-T2	[15440.5, 15642.6]			15634.2	15634.2	0.00	0.00	61		1.25	
4 plants	pr-50-4D4P5S-T3		15503.5	6885	**15503.5**	15503.5	0.00	0.00	69	0.00		
5 periods	pr-50-4D4P5S-T4				15454	15455.3	0.01	0.05	69			
	pr-50-4D4P5S-T5		15203.7	16713	15224.7	15245.2	0.16	0.75	69	0.14		
	pr-50-4D6P4S-T1				15228.5	15235.5	0.05	0.25	86			
4 depots	pr-50-4D6P4S-T2				11970.5	11970.5	0.00	0.00	72			
6 plants	pr-50-4D6P4S-T3				11622.9	11626.02	0.02	0.10	75			
4 periods	pr-50-4D6P4S-T4				15885.4	15920.7	0.25	1.02	80			
	pr-50-4D6P4S-T5				15681.7	15682.6	0.01	0.04	65			
	pr-50-4D6P5S-T1				15438.3	15438.3	0.00	0.00	67			
4 depots	pr-50-4D6P5S-T2				15992.7	15996.8	0.04	0.15	77			
6 plants	pr-50-4D6P5S-T3				11595.3	11597.7	0.02	0.11	74			
5 periods	pr-50-4D6P5S-T4	[11816.7, 11978.9]			11978.9	11978.9	0.00	0.00	73		1.37	
	pr-50-4D6P5S-T5				15073.9	15085.5	0.09	0.44	97			
	pr-50-6D4P4S-T1	[15117, 15320.5]	15264.5	2051	**15264.5**	15264.5	0.00	0.00	69	0.00	0.98	0.98
6 depots	pr-50-6D4P4S-T2				15247	15254.8	0.06	0.30	87			
4 plants	pr-50-6D4P4S-T3	[18744.2, 19117.6]	18872.1	7721	18877.1	18881.1	0.02	0.12	86	0.03	0.71	0.68
4 periods	pr-50-6D4P4S-T4	[14770.7, 15049.7]			14988.9	14989.2	0.00	0.02	79		1.48	
	pr-50-6D4P4S-T5		15388.7	13250	15389.7	15392.9	0.03	0.12	72	0.01		
	pr-50-6D4P5S-T1		15328.8	2725	**15328.8**	15328.8	0.00	0.00	71	0.00		
6 depots	pr-50-6D4P5S-T2	[14821.1, 15082.5]			15054.5	15054.5	0.00	0.00	77		1.57	
4 plants	pr-50-6D4P5S-T3	[18739.8, 19047.3]			18893	18896.9	0.02	0.11	90		0.82	
5 periods	pr-50-6D4P5S-T4		15185.6	8753	15204.6	15210.6	0.05	0.22	94	0.13		
	pr-50-6D4P5S-T5	[15070.9, 15371.9]	15218.8	1990	**15218.8**	15218.8	0.00	0.00	85	0.00	0.98	0.98
	pr-50-6D6P4S-T1	[15130.6, 15351.8]			15311.4	15312.1	0.01	0.03	104		1.19	
6 depots	pr-50-6D6P4S-T2	[15245.3, 15511.4]			15425.2	15425.4	0.00	0.01	89		1.18	
6 plants	pr-50-6D6P4S-T3	[15477.7, 15900.2]			15791.9	15794.7	0.02	0.10	100		2.03	
4 periods	pr-50-6D6P4S-T4				15030.9	15033.2	0.02	0.10	100			
	pr-50-6D6P4S-T5				15085.8	15097.9	0.10	0.50	107			
	pr-50-6D6P5S-T1				15060.9	15074	0.13	0.63	107			
6 depots	pr-50-6D6P5S-T2				15191.9	15196.7	0.04	0.17	92			
6 plants	pr-50-6D6P5S-T3	[15454.6, 15803.3]			15724.1	15728.4	0.03	0.14	116		1.74	
5 periods	pr-50-6D6P5S-T4	[15366.7, 15558.1]			15479.6	15480.9	0.01	0.05	102		0.73	
	pr-50-6D6P5S-T5	[15177.2, 15381.8]			15339.2	15341.2	0.02	0.07	113		1.07	
	Avg.			6303 (13)			0.03 (40)	0.16 (40)	82 (40)	0.03 (12)	1.13 (17)	0.80 (5)

Table 12: Results for instances with 60 (1) producers

	Instance	Bounds on opt. sol.	BKS DCGR	T (s) DCGR	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s) ALNS avg.	% dev ALNS best from DCGR	% dev ALNS best from LB	% dev DCGR from LB
	pr-60-2D2P4S-T1	[13861.3, 14030.4]	14015.6	6956	**14015.6**	14016.7	0.01	0.05	34	0.00	1.11	1.11
2 depots	pr-60-2D2P4S-T2				17488.6	17495.9	0.05	0.24	45			
2 plants	pr-60-2D2P4S-T3				15999.3	16005.4	0.05	0.19	34			
4 periods	pr-60-2D2P4S-T4	[16930, 17197.9]			17076.2	17078.8	0.02	0.10	43		0.86	
	pr-60-2D2P4S-T5				15920.8	15926.8	0.05	0.19	51			
	pr-60-2D2P5S-T1	[15373.4, 15630.1]			15612	15613.8	0.02	0.07	55		1.55	
2 depots	pr-60-2D2P5S-T2	[16798.4, 17042.9]			17025.2	17025.2	0.00	0.00	39		1.35	
2 plants	pr-60-2D2P5S-T3				15886.1	15893.4	0.06	0.23	39			
5 periods	pr-60-2D2P5S-T4				17619.8	17622.4	0.02	0.09	51			
	pr-60-2D2P5S-T5	[14342.6, 14489.1]			14444	14444.9	0.01	0.04	42		0.71	
	pr-60-2D3P4S-T1	[16956.1, 17174.9]			17157.2	17157.2	0.00	0.00	48		1.19	
2 depots	pr-60-2D3P4S-T2				17863.6	17877.8	0.10	0.40	58			
3 plants	pr-60-2D3P4S-T3	[17429.3, 17847.8]	17622.5	4023	17630	17644.1	0.09	0.38	56	0.04	1.15	1.11
4 periods	pr-60-2D3P4S-T4				17316.1	17348.8	0.21	0.97	57			
	pr-60-2D3P4S-T5				15434.6	15434.6	0.00	0.00	60			
	pr-60-2D3P5S-T1				15334.9	15335.7	0.01	0.03	56			
2 depots	pr-60-2D3P5S-T2				17521.5	17533.6	0.09	0.39	57			
3 plants	pr-60-2D3P5S-T3	[17383.7, 17750.7]			17593.2	17599.8	0.05	0.20	59		1.21	
5 periods	pr-60-2D3P5S-T4				17873.9	17879.2	0.03	0.14	69			
	pr-60-2D3P5S-T5	[17150.6, 17481.9]			17325.1	17325.1	0.00	0.00	51		1.02	
	pr-60-3D2P4S-T1		17342.8	1903	17345.2	17355.8	0.09	0.42	52	0.01		
3 depots	pr-60-3D2P4S-T2	[15747.9, 16111]			16013.7	16014.6	0.01	0.03	52		1.69	
2 plants	pr-60-3D2P4S-T3	[15425.9, 15771.9]			15629.3	15633.8	0.03	0.14	53		1.32	
4 periods	pr-60-3D2P4S-T4				14284.4	14294.1	0.08	0.30	56			
	pr-60-3D2P4S-T5	[15287.5, 15606.5]			15430.6	15430.6	0.00	0.00	49		0.94	
	pr-60-3D2P5S-T1	[15078, 15272.9]			15253.9	15254.6	0.01	0.03	49		1.17	
3 depots	pr-60-3D2P5S-T2				14386.8	14396	0.11	0.41	55			
2 plants	pr-60-3D2P5S-T3	[15369.9, 15657.3]			15555.3	15556.2	0.01	0.04	58		1.21	
5 periods	pr-60-3D2P5S-T4	[15908.7, 16165.2]	16075.8	13039	16082.9	16082.9	0.00	0.00	61	0.04	1.09	1.05
	pr-60-3D2P5S-T5				17568.8	17596.8	0.18	0.80	66			
	pr-60-3D3P4S-T1				15546.9	15549.7	0.03	0.12	60			
3 depots	pr-60-3D3P4S-T2				17046.2	17046.2	0.00	0.00	53			
3 plants	pr-60-3D3P4S-T3		15499.1	10616	15508.1	15509.9	0.01	0.06	70	0.06		
4 periods	pr-60-3D3P4S-T4				16905.7	16908.3	0.03	0.14	70			
	pr-60-3D3P4S-T5				17217.9	17217.9	0.00	0.00	64			
	pr-60-3D3P5S-T1				17098.3	17099.7	0.02	0.08	59			
3 depots	pr-60-3D3P5S-T2		17071.9	12448	**17071.9**	17073.5	0.01	0.07	68	0.00		
3 plants	pr-60-3D3P5S-T3				15506.3	15512.5	0.05	0.21	73			
5 periods	pr-60-3D3P5S-T4				17113.2	17113.5	0.00	0.02	60			
	pr-60-3D3P5S-T5				15655.9	15660	0.06	0.25	70			
	Avg.			8164 (6)			0.04 (40)	0.17 (40)	55 (40)	0.03 (6)	1.17 (15)	1.09 (3)

Table 13: Results for instances with 60 (2) producers

	Instance	Bounds on opt. sol.	BKS DCGR	T (s) DCGR	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s) ALNS avg.	% dev ALNS best from DCGR	% dev ALNS best from LB	% dev DCGR from LB
	pr-60-4D4P4S-T1		18968.5	4877	18985.2	18988.3	0.02	0.10	78	0.09		
4 depots	pr-60-4D4P4S-T2	[18834.5, 19092.3]			19014.1	19014.1	0.00	0.00	65		0.95	
4 plants	pr-60-4D4P4S-T3		19389.3	8976	**19389.3**	19389.3	0.00	0.00	67	0.00		
4 periods	pr-60-4D4P4S-T4				15596.8	15601.7	0.04	0.16	67			
	pr-60-4D4P4S-T5				18591.4	18601.1	0.07	0.34	75			
	pr-60-4D4P5S-T1				18605.2	18610.8	0.03	0.18	81			
4 depots	pr-60-4D4P5S-T2				15782.6	15783.4	0.01	0.02	73			
4 plants	pr-60-4D4P5S-T3		19367.5	14702	**19367.5**	19367.5	0.00	0.00	78	0.00		
5 periods	pr-60-4D4P5S-T4	[18781.7, 19134.6]	18963.3	2780	18969.4	18977.3	0.05	0.24	83	0.03	1.00	0.97
	pr-60-4D4P5S-T5		18874.6	673	18895.2	18907.4	0.07	0.36	100	0.11		
	pr-60-4D4P5S-T5				22473.5	22484.3	0.06	0.28	118			
4 depots	pr-60-4D6P4S-T1				22368.2	22390.1	0.11	0.56	139			
6 plants	pr-60-4D6P4S-T2				22314.5	22323.9	0.05	0.27	135			
4 periods	pr-60-4D6P4S-T3				22570.1	22586	0.08	0.41	122			
	pr-60-4D6P4S-T4				15573.4	15584.4	0.09	0.41	109			
	pr-60-4D6P4S-T5				15496.7	15501	0.03	0.14	107			
4 depots	pr-60-4D6P5S-T1				22885.8	22897.7	0.07	0.33	132			
6 plants	pr-60-4D6P5S-T2				22363.6	22375.2	0.06	0.31	148			
5 periods	pr-60-4D6P5S-T3				22313.4	22322.2	0.05	0.26	129			
	pr-60-4D6P5S-T4		22065.6	9770	22366	22383.5	0.09	0.45	113	1.36		
	pr-60-4D6P5S-T5		18935	368	**18935**	18935.5	0.00	0.02	99	0.00		
6 depots	pr-60-6D4P4S-T2				19413.6	19420.9	0.05	0.22	86			
4 plants	pr-60-6D4P4S-T3				19023.9	19025.6	0.01	0.06	98			
4 periods	pr-60-6D4P4S-T4		18673.1	11446	18682.6	18688	0.04	0.18	81	0.05		
	pr-60-6D4P4S-T5				18911.9	18951	0.24	1.15	94			
	pr-60-6D4P5S-T1				18962.8	18966.6	0.03	0.12	91			
6 depots	pr-60-6D4P5S-T2				18906.3	18909.4	0.02	0.09	87			
4 plants	pr-60-6D4P5S-T3				19022.8	19028	0.03	0.15	105			
5 periods	pr-60-6D4P5S-T4				19427.3	19438.2	0.06	0.28	93			
	pr-60-6D4P5S-T5		18785.4	631	**18785.4**	18786	0.00	0.02	116	0.00		
	pr-60-6D6P4S-T1		22386.6	15214	22394.4	22412.3	0.10	0.49	155	0.03		
6 depots	pr-60-6D6P4S-T2				22451.1	22458	0.04	0.21	149			
6 plants	pr-60-6D6P4S-T3		22538.9	12429	22541	22559	0.09	0.45	147	0.01		
4 periods	pr-60-6D6P4S-T4				21723.1	21737	0.08	0.46	144			
	pr-60-6D6P4S-T5				15313.1	15332.3	0.14	0.66	128			
	pr-60-6D6P5S-T1				15227.5	15266.7	0.29	1.38	124			
6 depots	pr-60-6D6P5S-T2				21995.5	22030	0.19	1.06	161			
6 plants	pr-60-6D6P5S-T3		22582.8	3524	22593.7	22603.5	0.05	0.26	149	0.05		
5 periods	pr-60-6D6P5S-T4		22342.9	1690	22371.5	22376.3	0.02	0.13	151	0.13		
	pr-60-6D6P5S-T5		22299	1081	22324.4	22339.7	0.08	0.44	154	0.11		
	Avg.			6297 (14)			0.06	0.32	111	0.14 (14)	0.97 (2)	0.97 (1)

Table 14: Results for instances with 100 producers (1)

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-100-2D2P4S-T1	29519.2	29532.1	0.05	0.21	72
2 depots	pr-100-2D2P4S-T2	29831.6	29838.2	0.03	0.14	71
2 plants	pr-100-2D2P4S-T3	30193.8	30204.8	0.05	0.18	73
4 periods	pr-100-2D2P4S-T4	29968.2	29988.2	0.09	0.35	74
	pr-100-2D2P4S-T5	30251.3	30268	0.07	0.28	77
	pr-100-2D2P5S-T1	29580.4	29591.2	0.04	0.18	79
2 depots	pr-100-2D2P5S-T2	29892.1	29910.4	0.07	0.29	78
2 plants	pr-100-2D2P5S-T3	29965	29988.7	0.09	0.37	77
5 periods	pr-100-2D2P5S-T4	30100.9	30119	0.07	0.29	83
	pr-100-2D2P5S-T5	30228.9	30248.8	0.08	0.33	85
	pr-100-2D3P4S-T1	26407.4	26416.5	0.04	0.22	57
2 depots	pr-100-2D3P4S-T2	26585.5	26609.1	0.10	0.48	56
3 plants	pr-100-2D3P4S-T3	26830.6	26857.3	0.12	0.56	60
4 periods	pr-100-2D3P4S-T4	26666.9	26710.9	0.19	0.92	60
	pr-100-2D3P4S-T5	26925.1	26939.5	0.07	0.32	57
	pr-100-2D3P5S-T1	26415.1	26430.8	0.07	0.36	61
2 depots	pr-100-2D3P5S-T2	26626.5	26648.8	0.09	0.45	63
3 plants	pr-100-2D3P5S-T3	26671.8	26691.4	0.09	0.43	61
5 periods	pr-100-2D3P5S-T4	26786	26832.1	0.22	1.00	63
	pr-100-2D3P5S-T5	26859.8	26920	0.27	1.26	66
	pr-100-2D6P4S-T1	26940.4	26964.9	0.10	0.47	98
2 depots	pr-100-2D6P4S-T2	27148.6	27179.1	0.14	0.60	99
6 plants	pr-100-2D6P4S-T3	27418.9	27462.9	0.19	0.81	113
4 periods	pr-100-2D6P4S-T4	27164.5	27178.7	0.08	0.35	119
	pr-100-2D6P4S-T5	27413.6	27464.6	0.25	1.05	114
	pr-100-2D6P5S-T1	26946.5	26980.7	0.15	0.67	114
2 depots	pr-100-2D6P5S-T2	27171.6	27218.3	0.20	0.87	116
6 plants	pr-100-2D6P5S-T3	27225.8	27248.9	0.11	0.47	121
5 periods	pr-100-2D6P5S-T4	27338.8	27347.6	0.04	0.18	130
	pr-100-2D6P5S-T5	27430.4	27451.9	0.10	0.44	131
	pr-100-3D2P4S-T1	23774.1	23791.6	0.11	0.43	89
3 depots	pr-100-3D2P4S-T2	24038.8	24049.3	0.05	0.20	86
2 plants	pr-100-3D2P4S-T3	24269.8	24296.2	0.14	0.52	92
4 periods	pr-100-3D2P4S-T4	24070.5	24084.5	0.08	0.33	83
	pr-100-3D2P4S-T5	24289.4	24300.6	0.06	0.24	85
	pr-100-3D2P5S-T1	23808.4	23811	0.02	0.07	86
3 depots	pr-100-3D2P5S-T2	24062.1	24073.7	0.06	0.22	97
2 plants	pr-100-3D2P5S-T3	24110.6	24127.9	0.10	0.38	97
5 periods	pr-100-3D2P5S-T4	24204.8	24233.4	0.14	0.53	106
	pr-100-3D2P5S-T5	24311	24315.2	0.03	0.11	107
	pr-100-6D2P4S-T1	26283.4	26289.4	0.03	0.15	95
6 depots	pr-100-6D2P4S-T2	26482.9	26487.5	0.02	0.10	94
2 plants	pr-100-6D2P4S-T3	26721.2	26724.1	0.01	0.06	110
4 periods	pr-100-6D2P4S-T4	26557	26572	0.07	0.34	89
	pr-100-6D2P4S-T5	26790.2	26812.8	0.10	0.45	116
	pr-100-6D2P5S-T1	26319.1	26324.4	0.03	0.12	100
6 depots	pr-100-6D2P5S-T2	26533.7	26541.4	0.03	0.17	121
2 plants	pr-100-6D2P5S-T3	26592	26598.1	0.03	0.13	116
5 periods	pr-100-6D2P5S-T4	26710.7	26729.7	0.08	0.39	122
	pr-100-6D2P5S-T5	26793.7	26801.2	0.04	0.16	113
	Avg.	33630.72	33655.19	0.11	0.49	113

Table 15: Results for instances with 100 producers (2)

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-100-3D3P4S-T1	27704.8	27740	0.14	0.59	101
3 depots	pr-100-3D3P4S-T2	27904.7	27927.7	0.11	0.45	102
3 plants	pr-100-3D3P4S-T3	28143.9	28163.3	0.09	0.35	101
4 periods	pr-100-3D3P4S-T4	27803.8	27852.6	0.20	0.82	107
	pr-100-3D3P4S-T5	28037.4	28081.3	0.18	0.70	110
	pr-100-3D3P5S-T1	27768.7	27779.1	0.05	0.19	107
3 depots	pr-100-3D3P5S-T2	27990.1	28015.9	0.10	0.42	107
3 plants	pr-100-3D3P5S-T3	28006.1	28023.4	0.07	0.28	111
5 periods	pr-100-3D3P5S-T4	28038	28067.5	0.14	0.54	119
	pr-100-3D3P5S-T5	28067.9	28076.4	0.04	0.15	121
	pr-100-3D6P4S-T1	33482.8	33489.6	0.03	0.14	134
3 depots	pr-100-3D6P4S-T2	33605.1	33652.9	0.16	0.81	136
6 plants	pr-100-3D6P4S-T3	33501.3	33534.9	0.11	0.59	148
4 periods	pr-100-3D6P4S-T4	33185.2	33195.4	0.04	0.22	150
	pr-100-3D6P4S-T5	33413.7	33435.1	0.08	0.42	157
	pr-100-3D6P5S-T1	33531.2	33560.7	0.10	0.52	139
3 depots	pr-100-3D6P5S-T2	33751.2	33760.5	0.04	0.19	141
6 plants	pr-100-3D6P5S-T3	33512.3	33540.2	0.09	0.48	154
5 periods	pr-100-3D6P5S-T4	33500.2	33519.6	0.07	0.35	163
	pr-100-3D6P5S-T5	33345.4	33362.3	0.06	0.30	162
	pr-100-6D3P4S-T1	26829.5	26838.3	0.04	0.18	109
6 depots	pr-100-6D3P4S-T2	27056.5	27069	0.05	0.24	112
3 plants	pr-100-6D3P4S-T3	27256.4	27289.8	0.14	0.60	115
4 periods	pr-100-6D3P4S-T4	26980.3	26994.1	0.06	0.27	122
	pr-100-6D3P4S-T5	27225.3	27239.1	0.07	0.30	125
	pr-100-6D3P5S-T1	26852.4	26858.2	0.03	0.13	114
6 depots	pr-100-6D3P5S-T2	27089.7	27110.9	0.09	0.40	117
3 plants	pr-100-6D3P5S-T3	27140.4	27152.4	0.05	0.23	120
5 periods	pr-100-6D3P5S-T4	27147.4	27177.6	0.13	0.57	129
	pr-100-6D3P5S-T5	27176.4	27190.5	0.06	0.28	132
	pr-100-6D6P4S-T1	30673	30705.2	0.15	0.68	210
6 depots	pr-100-6D6P4S-T2	30878.8	30919.7	0.16	0.70	208
6 plants	pr-100-6D6P4S-T3	31076	31109.8	0.15	0.66	218
4 periods	pr-100-6D6P4S-T4	30795.9	30824.8	0.12	0.53	226
	pr-100-6D6P4S-T5	31072.9	31099.1	0.10	0.46	231
	pr-100-6D6P5S-T1	30729	30754.4	0.10	0.47	215
6 depots	pr-100-6D6P5S-T2	30929.2	30974.6	0.19	0.83	216
6 plants	pr-100-6D6P5S-T3	30995.4	31027.9	0.13	0.58	223
5 periods	pr-100-6D6P5S-T4	30964.4	31026.3	0.24	1.07	233
	pr-100-6D6P5S-T5	30943.6	31002.7	0.27	1.19	240
	Avg.	29852.7	29878.6	0.11	0.47	150

Table 16: Results for instances with 200 producers

	Instance	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-200-3D3P4S-T1	53888	53915.7	0.06	0.26	167
3 depots	pr-200-3D3P4S-T2	54490.3	54514.3	0.06	0.22	165
3 plants	pr-200-3D3P4S-T3	55283.6	55326.7	0.10	0.36	172
4 periods	pr-200-3D3P4S-T4	54907	54985.3	0.17	0.63	186
	pr-200-3D3P4S-T5	55642	55682.7	0.09	0.33	192
	pr-200-3D3P5S-T1	53968.6	53995.9	0.06	0.24	174
3 depots	pr-200-3D3P5S-T2	54525.8	54564.7	0.08	0.33	176
3 plants	pr-200-3D3P5S-T3	54817.5	54860.6	0.09	0.35	182
5 periods	pr-200-3D3P5S-T4	55176.1	55210.7	0.08	0.30	184
	pr-200-3D3P5S-T5	55519.3	55551.3	0.08	0.29	195
	pr-200-3D6P4S-T1	49392.1	49440.8	0.11	0.54	207
3 depots	pr-200-3D6P4S-T2	49871.4	49977.6	0.26	1.20	202
6 plants	pr-200-3D6P4S-T3	50415.7	50505.7	0.21	0.95	194
4 periods	pr-200-3D6P4S-T4	50163.1	50193.6	0.08	0.36	202
	pr-200-3D6P4S-T5	50753.3	50767.6	0.04	0.17	204
	pr-200-3D6P5S-T1	49435.4	49509.5	0.17	0.82	217
3 depots	pr-200-3D6P5S-T2	49913.3	50014.5	0.23	1.06	215
6 plants	pr-200-3D6P5S-T3	50124.4	50178.7	0.12	0.56	213
5 periods	pr-200-3D6P5S-T4	50382.9	50439.2	0.13	0.56	212
	pr-200-3D6P5S-T5	50668.5	50700.9	0.08	0.35	218
	pr-200-6D3P4S-T1	50071.9	50118	0.10	0.47	189
6 depots	pr-200-6D3P4S-T2	50576.1	50606.9	0.08	0.33	194
3 plants	pr-200-6D3P4S-T3	51270.7	51318.6	0.11	0.46	200
4 periods	pr-200-6D3P4S-T4	50913.8	50987.5	0.16	0.69	216
	pr-200-6D3P4S-T5	51526.7	51605.9	0.18	0.75	211
	pr-200-6D3P5S-T1	50113	50152.8	0.10	0.46	188
6 depots	pr-200-6D3P5S-T2	50644.7	50688.8	0.11	0.46	198
3 plants	pr-200-6D3P5S-T3	50942.2	50967.5	0.07	0.29	202
5 periods	pr-200-6D3P5S-T4	51235.6	51311.9	0.18	0.74	217
	pr-200-6D3P5S-T5	51426.1	51537	0.26	1.06	224
	pr-200-6D6P4S-T1	57968.5	58008.4	0.08	0.37	318
6 depots	pr-200-6D6P4S-T2	58522.4	58565.8	0.09	0.38	329
6 plants	pr-200-6D6P4S-T3	58977.3	59060.3	0.16	0.69	351
4 periods	pr-200-6D6P4S-T4	58463.3	58536	0.15	0.65	359
	pr-200-6D6P4S-T5	59006.3	59039.8	0.07	0.29	369
	pr-200-6D6P5S-T1	58006.9	58058.9	0.11	0.49	331
6 depots	pr-200-6D6P5S-T2	58597.2	58654.8	0.12	0.50	330
6 plants	pr-200-6D6P5S-T3	58701.8	58742.1	0.09	0.36	344
5 periods	pr-200-6D6P5S-T4	58880.1	58907.4	0.05	0.23	363
	pr-200-6D6P5S-T5	58891.3	58969.3	0.17	0.72	379
	Avg.	53601.9	53654.3	0.12	0.51	235