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

Iman Dayarian
Teodor Gabriel Crainic
Michel Gendreau
Walter Rei

November 2013

CIRRELT-2013-67

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

Iman Dayarian[1,2,*], Teodor Gabriel Crainic[1,3], Michel Gendreau[1,4], Walter Rei[1,3]

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

[2] Department of Computer Science and Operations Research, Université de Montréal, Pavillon André-Aisenstadt, P.O. Box 6128, Station Centre-Ville, Montréal, Canada H3C 3J7

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

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

Abstract. We consider tactical planning for a particular class of multi-period vehicle routing problems (MPVRP). This problem involves optimizing product collection and distribution 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.

____

* Corresponding author: Iman.Dayarian@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2013

1. Introduction

The vehicle routing problem (VRP) is a difficult combinatorial optimization problem that is used 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 serve 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 the classical VRP, 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. (2013b), 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. (2012b) and Dayarian et al. (2013a); both studies assumed that the annual production is fixed. Dayarian et al. (2013b) 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 served by the same vehicle. The drivers to plan their daily operations also use this routing plan.

Dayarian et al. (2013b) proposed an exact methodology based on a branch-and-price approach and a multi-period model. 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. (2013b) provides optimal solutions for instances with up to twenty 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 a new metaheuristic based on the ALNS for our problem setting, which is described in more detail in Section 2.
- We design several new operators based on the special structure of the problem. We also adapt some existing operators in the literature.
- We propose a new adaptive layer for the ALNS in which destruction and construction heuristics are coupled to form the operators, rather than being treated independently.
- We propose a new diversity management system for the ALNS, which is based on extracting information from a list of diverse solutions and restarting the search from a diverse solution when it seems to be trapped in a local optimum.
- 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 perform a series of extensive numerical tests for a large set of randomly generated instances, to illustrate the performance of the algorithm in terms of computational time and solution quality.

The remainder of this paper is organized as follows. In Section 2, we describe the problem and the notation that we will use. Section 3 discusses the state-of-the-art of work in this field. In Section 4, we present the classical ALNS for the VRP, and in Section 5 we present our approach to the problem. In Section 6, we propose a series of bounds that allow us to evaluate the performance of the algorithm. The experimental results are reported in Section 7, and Section 8 provides concluding remarks.

2. Problem Statement and Notation

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. (2012b) and Dayarian et al. (2013b,a).

The problem can be formally described as follows: We wish to design a single tactical routing plan for a given horizon T. A plan consists of a set of routes, each performed by a single vehicle on every collection day of T. An unlimited fleet of identical vehicles is assumed to be available in 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 served 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 T 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 T. We associate with each period $\xi \in \Xi$ a weight W_ξ, representing the share of period ξ in horizon T. It is calculated by dividing the length of period ξ by the length of horizon T. 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. (2013b).

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 that is proportional to the length of the arc $dist_{ij}$. 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.

In each period, each producer $n_j \in \mathcal{N}$ produces a limited product quantity 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 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 periods and at most one failure in the other periods. In other words, for each route r, the following inequalities hold:

$$\sum_{j \in r} o_j^\xi \leq 2Q, \qquad s \in \mathcal{S} \tag{2}$$

and

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

The cost of the solution has three components: 1) the fixed vehicle costs; 2) the first-stage routing costs, which are the costs of the planned routes; and

3) the second-stage routing costs, which are the expected recourse costs in different periods of the horizon.

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. (2012b), 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 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. (2013a) 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 VRP that is the most similar to our problem is the multi-period or periodic MDVRP. In most studies of the multi-period vehicle routing problem (MPVRP), customers request a service that could be done 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 periodic vehicle routing problem (PVRP) in which the customers specify a service frequency and allowable combinations of visit days. A complete survey of the PVRP and its extensions can be found in Francis et al. (2008). The best-known algorithms for the PVRP are those of Cordeau et al. (1997), Hemmelmayr et al. (2009), Vidal et al. (2012), and Rahimi-Vahed et al. (2013). In our problem, all the producers need to be served every period on a daily basis. Moreover, the definition of the periods is based on the 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. (2013b), we are not aware of any previous study of the VRP with the multi-period configuration considered in this paper. Dayarian et al. (2013b) 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 and Toulouse, 2008; Crainic et al., 2009; Lahrichi et al., 2012a). Of the neighborhood search methods, the large neighborhood search (LNS) algorithms (Shaw, 1998) have proven to be successful for several classes of the VRP. The 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), Pisinger and Ropke (2010), Azi et al. (2010), 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. Classical ALNS for the VRP

The ALNS algorithm is an iterative process, in which at every iteration part of the current solution is destroyed using a destruction heuristic and then reconstructed using a construction heuristic in the hope of finding a better solution. The destruction heuristic usually disconnects a number $q \in [q_{min}, q_{max}]$ of the nodes from their current routes and places them into the *unassigned node pool*, Φ. Note that q_{min} and q_{max} are parameters whose values are to be tuned. The construction heuristic then inserts the nodes from Φ into the routes of the solution. The main components of the ALNS (Ropke and Pisinger, 2006; Pisinger and Ropke, 2007) are:

Adaptive search engine: At each iteration of the ALNS, one independently selects a destruction and a construction heuristic via a biased random mechanism, referred to as the roulette-wheel. It favors the heuristics that have been successful according to certain criteria in recent iterations. The adaptive layer of the ALNS procedure controls the functionality of the roulette-wheel. One associates with each heuristic a weight that is incremented during the search based on a scoring mechanism that measures the performance of the heuristic. The probability of selecting a given heuristic is proportional to the ratio of its weight to the sum of the weights of the other heuristics.

Adaptive weight adjustment: One divides the search into a number of fixed-length segments of consecutive iterations. In the first segment, all the heuristics have the same weight. At the end of each segment, the weights used to select the heuristics are updated.

Acceptance and stopping criteria: The new solution obtained via the destruction-construction procedure is usually accepted or rejected based on some criterion. This criterion is usually defined by the search paradigm applied at the master level, e.g., simulated annealing (SA) (see Kirkpatrick et al., 1983). 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

$$\exp(\frac{-\Delta f}{T}), \tag{4}$$

where $T > 0$ is the temperature parameter. The temperature is initialized to T^{init} and at the end of each iteration it is lowered by a cooling rate $c \in (0, 1)$: $T \leftarrow c.T$. The probability of accepting worse solutions reduces as T decreases. This allows the algorithm to progressively find better local optima. The stopping criterion is based either on a predetermined number of iterations or a predefined final temperature T^{fin}.

4.1. Selected destruction/construction heuristics

Several destruction and construction heuristics have been proposed, and some can be adapted to the VRP context. We focus on the destruction heuristics outlined below.

Random Removal: This heuristic (Ropke and Pisinger, 2006) randomly selects q nodes, removes them from their current position, and places them into Φ. The random nature of this heuristic diversifies the search.

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

Route Removal: This heuristic 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 consider the following construction heuristics:

Sequential Insertion: This heuristic inserts the nodes from Φ in order. Each node is placed in the position that incurs the minimal local cost.

Best-First Insertion: This heuristic 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.

5. Proposed Solution Framework

Our algorithm is based on a general ALNS but incorporates a number of intensification and diversification strategies that improve its performance; an outline is presented in Algorithm 1.

Similarly to the classical ALNS, the search is divided into several segments, each a series of consecutive iterations. However, we dynamically adjust the length of each segment based on a criterion that will be described in Section 5.4. At each iteration, we explore the neighborhood of the current solution, generating potentially φ new solutions. We obtain the new solutions by applying a randomly selected destruction-construction operator to the current solution. At the end of each iteration, we apply an acceptance criterion to the best solution among the φ solutions found. If the solution satisfies the criterion, it replaces the current solution. Our acceptance criterion, which is based on a probabilistic threshold inspired by SA, is discussed in Section 5.4.

At the end of each segment, we apply a series of local search (LS) operators to the best solution found in the segment. If this gives an improvement, we update the current solution. To help the algorithm escape from local minima, we implement a diversity management mechanism; see Section 5.6.

We also propose the use of an enhanced central memory. It stores both high-quality solutions and a set of diverse solutions. We design several new destruction heuristics that use information extracted from the central memory; see Section 5.7. Moreover, we design new operators for our specific problem setting. The main components of our algorithm are described below.

5.1. 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 $Iter^{his}$ iterations is greater than δ_{max}, and we divide it by 2 if the number of such solutions is less than δ_{min}.

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 serving plants with an undersupply.

Our penalty function adds a penalty ρ to the local cost of removal or insertion in a given position, where

$$\rho = \eta D^{-}(s). \tag{5}$$

5.2. Destruction-construction operators

The new solutions are obtained by applying an operator *opr* $\in \Omega$, to the current solution, where Ω is the set of all operators. In the classical ALNS algorithm, the destruction and construction heuristics are selected independently, but we form *opr* by coupling a pair of heuristics. The main advantage is that we can weight the performance of each (destruction-construction) pair. We use two main types of heuristics:

1. Generic heuristics: A generic destruction heuristic can be coupled with any generic construction heuristic. The heuristics presented in Sections 4, 5.7, and 5.8 are all considered generic heuristics.
2. Specialized heuristics: For each specialized destruction heuristic, we develop a specialized construction heuristic. The resulting operator has a specific goal such as the generation of solutions with a certain level of diversity. We present our specialized heuristics in Section 5.9.

5.3. Central Memory

We consider a central memory, denoted Ψ, with a limited number of solutions. We extract from it different types of information for use in destruction heuristics. There is clearly a trade-off between search quality on the one hand and computational efficiency and memory requirements on the other. We use the extracted information to determine the relatedness between different nodes of the graph with respect to different criteria. We design a destruction heuristic based on each criterion (see Section 5.7). The central memory contains three lists of solutions:

- **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.

- **Diverse Solutions (Ψ_{DIV}):** A list of β_3 solutions, selected according to the quality-diversity criterion discussed in Section 5.6.1.

5.4. Acceptance criterion

Our acceptance criterion is inspired by SA, but we do not perform the cooling procedure at the end of every iteration. We perform the 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.

We divide the search into several segments. The length of each segment corresponds to the repetition schedule for a given temperature and therefore has a minimum length of δ iterations. If a new global best feasible solution is found in the current segment, the length of the segment is extended.

5.5. Adaptive Mechanism

At every iteration of the ALNS, we choose the operator to apply to the current solution via a roulette-wheel mechanism. Each operator *opr* is assigned a weight ω_{opr} according to its performance history. Given Ω, the operator set, the probability of selecting *opr* is $\omega_{opr}/\sum_{k\in\Omega}\omega_k$.

To evaluate the performance of the operators, we implement an adaptive weight adjustment procedure. After each block of γ segments, referred to as a mega-segment, we update the operator weights based on their long- and short-term performance history. The short-term history covers the last mega-segment, and the long-term history covers the entire search. Each operator is also assigned a score, which is reset to zero at the end of each mega-segment. Initially, all the weights are set to one and all the scores to zero. At each iteration, we update the scores. We do this by adding a bonus factor $\sigma_i, i \in \{1,\dots,4\}$, where $\sigma_i \leq \sigma_{i+1}, i \in \{1,2,3\}$, to the current score as follows:

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

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

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

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

In all other cases, the bonus factor is zero. Moreover, suppose that π_{opr} is the total score of *opr* obtained from ν_{opr} applications of *opr* in the last mega-segment. We control the influence of the short- and long-term history using a parameter $\alpha \in [0, 1]$, called the reaction factor, through the formula

$$\omega_{opr,\iota+1} = \omega_{opr,\iota}(1 - \alpha) + \alpha \frac{\pi_{opr}}{\nu_{opr}}. \tag{6}$$

Here $\omega_{opr,\iota}$ represents the weight of operator *opr* in mega-segment ι. A value of α close to zero increases the impact of the long-term history; a value close to one increases the impact of the short-term history.

5.6. Diversity Management

The diversity of the search is governed by the two mechanisms discussed below.

5.6.1. Diverse Solutions (Ψ_{DIV})

The decisions taken in some destruction heuristics use the information extracted from the central memory Ψ. Recall that this central memory is divided into lists of best feasible solutions, best infeasible solutions, and diverse solutions. Several strategies have been proposed for determining a set of diverse solutions (Vidal et al., 2012; Rahimi-Vahed et al., 2013).

We consider a new utility function that evaluates the solution based on $g(s) = C(s) - \upsilon DIV(s)$, where $C(s)$ is the quality and $DIV(s)$ is the diversity measure:

$$DIV(s) = \sum_{s' \in \Psi_{FS} \cup \Psi_{NFS}} \chi(s, s'). \tag{7}$$

Parameter υ is self-adjusting: if during the last $nbSegm^{DIV}$ segments no improved solution has been found, υ is doubled.

In Equation (7), $\chi(s, s')$ is the distance between solutions s and s':

$$\chi(s, s') = \sum_{i \in \mathcal{N}} (\mathbf{1}(succ^{s}_{n_i} \neq succ^{s'}_{n_i}) + \mathbf{2}(n^{s}_{d_i} \neq n^{s'}_{d_i}) + \mathbf{2}(n^{s}_{p_i} \neq n^{s'}_{p_i})). \tag{8}$$

Here $succ^{s}_{n_i}$, $n^{s}_{d_i}$, and $n^{s}_{p_i}$ are the successor, depot, and plant of node n_i in solution s.

We compare two solutions based on their node sequencing and their assignments to depots and plants. The weight of each sequencing difference is set to one, and the weight of each assignment difference is set to two. We first attempt to insert every new solution into Ψ_{FS} or Ψ_{NFS}. When a solution cannot be inserted or a solution currently in one of these lists is replaced, we must decide whether or not to add it to Ψ_{DIV}. If the number of solutions in Ψ_{DIV} is less than β_3, we add the new solution. Otherwise, we consider the current members of Ψ_{DIV} and the incoming solution, and we retain the β_3 best solutions as measured by $g(s)$ values.

5.6.2. Diversity Segment

If after $nbSegm^{DIV}$ segments, no new best global solution is found, we devote a complete segment to the generation of diverse solutions. In this *diversity segment*, at each iteration, we randomly select one operator from the *specialized operators* (Section 5.9). Similarly to the regular segments, at each iteration, we generate φ solutions. We take the most diverse one according to $DIV(s)$ that is within a radius of r of the best-known solution and apply an acceptance criterion. In this criterion, the new solution s' replaces the current solution s if $g(s') < g(s)$.

5.7. Generic Destruction Heuristics

We now describe the generic destruction heuristics. Some are new, while others are adapted from existing heuristics proposed by Pisinger and Ropke (2007), which primarily differ in the way that the relatedness are weighted. We use 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 in 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.

Algorithm 1 ALNS

```
 1: $s \leftarrow$ InitialSolution;
 2: Initialize the weights $\pi$;
 3: Set the temperature $T$;
 4: $iter \leftarrow 0$;
 5: $segmentIter \leftarrow 0$;
 6: $seg \leftarrow 0$;
 7: $s_{seg} \leftarrow s$;
 8: repeat
 9:     repeat
10:         $s_{iter} \leftarrow s$;
11:         $q_{iter} \leftarrow$ Number of nodes to be removed;
12:         $Opr_{iter} \leftarrow$ Select an operator;
13:         $s' \leftarrow Opr_{iter}(s, q_{iter})$;
14:         if $f(s') < f(s_{iter})$ then
15:             $s_{iter} \leftarrow s'$;
16:         end if
17:     until $iter/\varphi == 0$
18:     if $f(s_{iter}) < f(s^*)$ and $s_{iter}$ feasible then
19:         $s^* \leftarrow s_{iter}$;
20:         $s_{seg} \leftarrow s_{iter}$;
21:         $segmentIter \leftarrow 0$;
22:     else
23:         if ACCEPT($s_{iter}$, $s$) then
24:             $s \leftarrow s_{iter}$;
25:         end if
26:     end if
27:     if $f(s_{iter}) < f(s_{seg})$ then
28:         $s_{seg} \leftarrow s_{iter}$;
29:     end if
30:     Update the score of $opr$;
31:     if $segmentIter == \delta$ then
32:         $s' \leftarrow$ LOCAL SEARCH($s_{seg}$);
33:         if $f(s') < f(s^*)$ then
34:             $s^* \leftarrow s'$;
35:             $segmentIter \leftarrow 0$;
36:         end if
37:     else
38:         if $f(s) < f(s)$ then
39:             $s \leftarrow s'$;
40:         end if
41:         $T \leftarrow c.T$;
42:         $s_{seg} \leftarrow s$;
43:         $seg \leftarrow seg + 1$;
44:     end if
45:     if $seg/\gamma == 0$ then
46:         Update the weights;
47:     end if
48:     $iter \leftarrow iter + 1$;
49:     $segmentIter \leftarrow segmentIter + 1$;
50: until Stopping Criterion
51: return $s^*$
```

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 served 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 mis-assigned to a depot. Each depot-node pair (n_d, n_i), for $d \in \mathcal{D}$ and $i \in \mathcal{N}$, is assigned a weight. 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

$$Pr_{n_i,n_{d_i}}(s) = \frac{rank(n_i)}{List_{i,d}(s).size()} \tag{9}$$

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.

5.8. Generic Construction Heuristics

After the destruction heuristic, the nodes that have been removed and placed in Φ are considered for reinsertion into routes.

Sequential Insertion with Plant Satisfaction

This heuristic is identical to sequential insertion except that insertions are not penalized with the parameter ρ. To overcome possible infeasibilities, we use the following two-phase insertion heuristic. In the first phase we insert nodes only for routes serving plants with an unsatisfied demand. When all the plant demands are met we move to the second phase. The remaining nodes may now be inserted in any route, as in normal sequential insertion. This procedure does not necessarily guarantee the feasibility of the new solution, because it depends on the infeasibility level of the original solution and the nodes in Φ.

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. This heuristic is based on the regret associated with the insertion of a node in a route serving 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 serving a plant with unsatisfied demand. The best insertion candidate is determined using the following criterion:

$$n_i := arg \min_{n_i \in \Phi} (\min_{r \in R_s^{DP}} (\Delta f_{r+n_i}(s)) - \min_{r \in R_s} (\Delta f_{r+n_i}(s))), \tag{10}$$

where R_s is the set of routes for solution s, and R_s^{DP} is the set of routes serving 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.

5.9. Specialized Operators

We also design specialized operators for our problem setting. Each consists of a pair of destruction and construction heuristics that work together; they are not coupled with any other heuristics. The destruction and construction heuristics cooperate to achieve diverse solutions (recall that these operators are only used in the diversity segments). We use the three operators below.

5.9.1. Depot-Exchange Operator

This operator changes the depot of a subset of the nodes to enhance the diversification.

Depot-Exchange Removal

This heuristic selects the nodes to be removed via the following steps:

Step 1: Randomly select a pair of depots, n_{d_1} and n_{d_2}.

Step 2: Sort the nodes n_i, $i \in \mathcal{N}$ in $List_{reg}(n_{d_1}, n_{d_2})$ according to the regret of assigning them to n_{d_1} or n_{d_2} using the following formula:

$$regret(n_i)_{d_1,d_2} = |dist_{d_1,n_i} - distd_2, n_i|. \tag{11}$$

Step 3: Starting from the node with the lowest regret value, remove nodes from their current positions with probability

$$Pr_{n_i} = \frac{rank(n_i)}{List_{reg}(n_{d_1}, n_{d_2}).size()}, \tag{12}$$

where $rank(n_i)$ is the position of node n_i in $List_{reg}(n_{d_1}, n_{d_2})$, so the position of the node with the smallest regret value is $List_{reg}(n_{d_1}, n_{d_2}).size()$.

Step 4: If the $List_{reg}(n_{d_1}, n_{d_2})$ is completely traversed but still $|\Phi| < q$, replace $List_{reg}(n_{d_1}, n_{d_2}).size()$ by $List_{reg}(n_{d_1}, n_{d_2}).size() - |\Phi|$, update $rank(n_i)$, and go to **Step 3**.

Depot-Exchange Insertion

Following a call to the above heuristic, we reinsert nodes from Φ into routes using this heuristic. It is based on the regret insertion heuristic, while nodes are preassigned to the depots. This preassignment uses the depots n_{d_1} and n_{d_2} selected for the removal heuristic. Each node $n_i \in \Phi$ is assigned to n_{d_1} with probability

$$Pr_{n_i,d_1} = 1 - \frac{dist_{d_1,n_i}}{dist_{d_1,n_i} + dist_{d_2,n_i}}, \tag{13}$$

and to n_{d_2} with probability $1 - Pr_{n_i,d_1}$.

5.9.2. Plant-Exchange Operator

Similarly to the depot-exchange operator, this operator changes a subset of the producer-plant assignments.

Plant-Exchange Removal

We randomly select two plants, n_{p_1} and n_{p_2}. We then sort the nodes based on the regret value $regret(n_i)_{p_1,p_2} = |dist_{p_1,n_i} - dist_{p_2,n_i}|$. Starting from the node with the lowest regret value, we remove nodes from their current positions with probability

$$Pr_{n_i} = \frac{rank(n_i)}{List_{reg}(n_{p_1}, n_{p_2}).size()}. \tag{14}$$

Plant-Exchange Insertion

This is a regret insertion heuristic that restricts the insertion of each node $n_i \in \Phi$ to a preassigned producer-plant pair. The preassignments use the plants n_{p_1} and n_{p_2} selected for the removal heuristic. The probability function is

$$Pr_{n_i,p_j} = 1 - \frac{dist_{p_j,n_i}}{dist_{p_1,n_i} + dist_{p_2,n_i}}, \qquad j \in \{1,2\}. \tag{15}$$

5.9.3. Tabu-Based Operator

This operator pairs the most random removal and one of the most successful insertion heuristics, i.e., the random removal heuristic and the regret insertion heuristic. To diversify the search, it attempts to avoid generating the same solutions by prohibiting the reassignment of removed nodes to their previous routes. The removal heuristic records a list of the routes to which the removed nodes were previously assigned. The regret insertion heuristic is as described in Section 4 except that it avoids reinserting a node into its previous route. The removal heuristic has a high level of diversity, and the insertion heuristic is designed to insert removed nodes into the best routes (provided they are different from the previous routes).

5.10. 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 a genetic algorithm proposed by Vidal et al. (2012). The procedures are restricted to the feasible region. We build each node's neighborhood using a granularity threshold ϑ, initially proposed by Toth and Vigo (2003), which is computed as follows:

$$\vartheta = \lambda \frac{Z(s)}{nbArc(s)}, \tag{16}$$

where $Z(s)$ and $nbArc(s)$ are the sum of the arc costs and the number of arcs used in solution s, and λ is a suitable sparsification factor. 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 $\frac{Z(s)}{nbArc(s)}$ 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 $dist_{ij} \leq \vartheta$.

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 as follows:

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).

6. 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., 2013b). Let the single-period problem that considers only the production levels in the reference period be P^{ref}, with optimal solution x^{ref}. Let P^{mp} be the multi-period problem, with optimal solution x^*.

The route cost 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. For any feasible solution x to P^{mp}, the following inequality provides an upper bound on the value of the multi-period solution:

$$c_f(x^*) + c(x^*) + \mathcal{F}(x^*) \leq c_f(x) + c(x) + \mathcal{F}(x). \tag{17}$$

Moreover, since 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:

$$c_f(x^*) = c_f(x^{ref}). \tag{18}$$

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

$$c(x^{ref}) \leq c(x^*). \tag{19}$$

We combine (18) and (19) to obtain a lower bound on the value of the multi-period solution:

$$c_f(x^{ref}) + c(x^{ref}) \leq c_f(x^*) + c(x^*) + \mathcal{F}(x^*). \tag{20}$$

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

$$\mathcal{F}(x) = \sum_{\xi \in \Xi} \sum_{r \in \mathcal{R}_s} F(r,\xi). \tag{21}$$

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

$$F(r,\xi) \geq \quad 2 \min_{i \in \mathcal{N}_r} dist_{i,p_r}.t_r^{\xi} \tag{22}$$

$$\Rightarrow \mathcal{F}(x^*) \geq \quad 2 \sum_{r \in \mathcal{R}_s} t_r^{\xi} \min_{i \in \mathcal{N}_r} dist_{i,p_r} \tag{23}$$

$$= \quad 2 \sum_{p \in \mathcal{P}} \sum_{r \in \mathcal{R}_s^p} t_r^{\xi} \min_{i \in \mathcal{N}_r} dist_{i,p_r}, \tag{24}$$

where $F(r,\xi)$ is the recourse cost on route r in period ξ, and 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 minimal failure cost for a given instance can then be computed by first determining the minimum number of vehicles needed to serve 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. We perform this two-step procedure by solving the bin-packing models discussed below.

6.1. Minimum Number of Vehicles

We first present the model that allows us to determine the minimum number of vehicles to cover the plant demands. Table 1 gives the notation, and the constraints are as follows:

1. Each producer is assigned to one vehicle and each vehicle to one plant;
2. The vehicle capacities are respected;
3. The plant demands are satisfied.

Table 1: Bin-packing notation for minimum number of vehicles

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 $p \in \mathcal{P}$.

The formulation is

$$\min \quad \sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}} y_{kp} \tag{25}$$

subject to

$$\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}} x_{ikp} = 1 \quad (i \in \mathcal{N}); \tag{26}$$

$$\sum_{i\in\mathcal{N}} \sum_{k\in\mathcal{K}} o_i x_{ikp} \geq D_p \quad (p \in \mathcal{P}); \tag{27}$$

$$\sum_{i\in\mathcal{N}} o_i x_{ikp} \leq Q \quad (p \in \mathcal{P}, k \in \mathcal{K}); \tag{28}$$

$$x_{ikp} \leq y_{kp} \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}); \tag{29}$$

$$x_{ikp}, y_{kp} \in 0, 1 \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}), \tag{30}$$

where the objective function minimizes the number of vehicles. Constraint (26) ensures that all the producers are assigned to exactly one route. Constraint (27) ensures that the plant demands are satisfied, and Constraint (28) ensures that the vehicle capacities are respected.

6.2. Minimum Failure Cost

Given the minimum number of vehicles, we can compute a lower bound on the total failure cost of P^{mp} based on inequality (24). Let the minimum number of vehicles be K^*. 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 an extension of the first bin-packing formulation that minimizes the failure cost.

Table 2: Bin-packing notation for minimum failure cost

Notation	**Description**
$\mathcal{K}^*$	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.

Table 2 gives the notation, and the formulation is

$$Z = \min \quad \sum_{\xi \in \mathcal{S}} Pr(\xi) \sum_{p \in \mathcal{P}} \sum_{i \in \mathcal{N}} 2.d_{i,p} u_{ikp}^\xi \tag{31}$$

subject to

$$l_{kp} = \sum_{i \in \mathcal{N}} o_i x_{ikp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{32}$$

$$l_{kp} \leq Q y_{kp} \sum_{i \in \mathcal{N}} o_i x_{ikp} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{33}$$

$$\sum_{p \in \mathcal{P}} y_{kp} = 1 \quad (k \in \mathcal{K}^*); \tag{34}$$

$$\sum_{k \in \mathcal{K}^*} l_{kp} \geq D_p \quad (p \in \mathcal{P}); \tag{35}$$

$$\sum_{k \in \mathcal{K}^*} \sum_{p \in \mathcal{P}} x_{ikp} = 1 \quad (i \in \mathcal{N}); \tag{36}$$

$$x_{ikp} \leq y_{kp} \quad (i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{37}$$

$$Pt(\xi) \sum_{p \in \mathcal{P}} l_{kp} \leq Q(1 + t_k^\xi) \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{38}$$

$$\sum_{p\in\mathcal{P}}\sum_{i\in\mathcal{N}} u^{\xi}_{ikp} = t^{\xi}_{k} \quad (\xi \in \mathcal{S}, k \in \mathcal{K}^*); \tag{39}$$

$$u^{\xi}_{ikp} \leq x_{ikp} \quad (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{40}$$

$$y_{kp} \leq y_{k-1p} + y_{k-1p-1} \quad (p \in \mathcal{P}, k \in \mathcal{K}^*); \tag{41}$$

$$y_{11} = 1; \tag{42}$$

$$x_{ikp}, y_{kp}, t^{\xi}_{k}, u^{\xi}_{ikp} \in \{0,1\} \quad (\xi \in \mathcal{S}, i \in \mathcal{N}, p \in \mathcal{P}, k \in \mathcal{K}^*). \tag{43}$$

Constraints (32) and (33) ensure that the vehicle capacities are satisfied. Constraint (34) ensures that each vehicle is assigned to a single plant. Constraint (35) ensures that the plant demands are satisfied, and Constraint (36) ensures that each producer is assigned to a single vehicle. Constraint (37) ensures that producers are assigned only to open routes. For each period ξ, Constraints (38)–(40) determine the number and location of failures on each vehicle k. Constraints (41) and (42) break the possible symmetry due to the set of identical vehicles. The objective function, Z, represents 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 not all periods have failures at the same node. Proposition 1 provides a condition determining when two periods both encounter failure at the same node.

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

$$\frac{Q}{Pt_2}(1 - \frac{Pt_2}{Pt_1}) \leq o_j. \tag{44}$$

Proof. Assume that $Pt_1 \geq Pt_2$ 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 $Pt_2.\frac{Q}{Pt_1}$. Moreover, ξ_2 has a failure at node n_j if $Pt_2.\frac{Q}{Pt_1} + Pt_2.o_j \geq Q$. □ □

Including this condition in the model (31)–(43) 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.

7. Computational Experiments

We now describe our computational experiments. In Section 7.1, 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 7.2. We also study the impact of different components of the algorithm based on a series of tests, which are presented in Section 7.3. Finally, the computational results for the test problems are presented in Section 7.4.

7.1. Test Problems

We consider the test problems proposed by Dayarian et al. (2013b) as well as extensions of them. The extensions increase the size of the instances. Dayarian et al. (2013b) generated instances with 15 or 20 producers, 2 or 3 depots, and 2 or 3 plants. Each instance was solved 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 $\{T1, \dots, T5\}$ 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 3.

Table 3: Specifications of test problems

Number of producers	Number of depots	Number of plants
15	$2, 3$	$2, 3$
20	$2, 3$	$2, 3$
40	$2, 3$	$2, 3$
100	$2, 3, 6$	$2, 3, 6$
200	$3, 6$	$3, 6$

The production levels and period weights are the same as in Dayarian et al. (2013b); Table 4 gives the production levels and weights.

We ran our ALNS algorithm for each of the above test problems 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.

7.2. 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 to be described later. 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, based on extensive preliminary tests. It divides the

Table 4: Weight and production-level distribution of the periods

# periods	Type 1		Type 2		Type 3		Type 4		Type 5	
	P_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$
	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_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$	P_s	$W_s\%$
	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

parameters into two subsets. The first subset contains the less sensitive parameters, and the second subset includes the parameters with a greater impact on the performance of the algorithm. We tune the parameters in the first subset by trial-and-error; Table 5 gives the resulting values.

Table 5: Parameter values found by trial and error

	Parameter	Value
$[q_{min}, q_{max}]$	Bounds on number of nodes removed q	$[\min(5, 0.05\lvert\mathcal{N}\rvert), \min(20, 0.4\lvert\mathcal{N}\rvert)]$
$Iter^{adj}$	Number of iterations after which η is updated	20
$Iter^{his}$	History used to update η	100
δmin and δmax	Bounds on number of infeasible solutions used to update η	30 and 45
β_1, β_2, β_3	Lengths of lists in central memory	20, 20, 10
λ	Sparsification factor in granularity threshold	1
$\sigma_1, \sigma_2, \sigma_3, \sigma_4$	Bonus factors for adaptive weight adjustment	1, 1, 1, 2

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 Eq. (4), 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%. Our 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.

We tune the parameters in the second subset by first determining a range for each parameter based on extensive 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; see Audet et al. (2012).

To define a performance measure for Opal, we selected a restricted set of training instances. This set included instances ranging from 20 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 6 gives the values found for the second subset of parameters.

Table 6: 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	Impact of long-term/short-term memory in weight update	[0, 1]	0.25
c	Cooling rate for SA	[0.9980, 0.9998]	0.9987
r	Acceptance radius gap in diversity segments	[0.01, 0.07]	0.05
$nbSegm^{DIV}$	Call diversity segment after observing no improvement in this number of segments	[25, 100]	45

7.3. 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. For each instance, we recorded the average result over the five runs. The values in Table 7 indicate the degradation in the geometric mean of the values obtained for all the instances in the training set. We use the geometric mean because the training set includes problems of different sizes with varying objective values. With the geometric mean the smaller instances are not dominated by the larger ones.

The plant-producer-related removal is the most efficient removal heuristic, followed by the route removal and smart removal heuristics. Minimum-loss insertion is the most useful insertion heuristic, followed by the regret insertion heuristic. We do not include the specialized operators in this evaluation because their main goal is to create diversity in the search. However, we have studied the impact of excluding the diversity segment. Our tests on the training set show that the solutions found without the diversity segment are on average 0.01% better. However, in some cases, particularly for smaller instances, the diversity segment helps us to escape from local optima. We have also evaluated the LS operators for the training set. The solutions found without these operators are on average 0.16% worse.

Table 7: Evaluation of contribution of each heuristic

Heuristic	Solution degradation without this heuristic (%)
Random Removal	-0.03
Worst Removal	0.00
Route Removal	0.05
Cluster Removal	0.02
Smart Removal	0.05
Solution-Cost-Based Related Removal	0.02
Route-Cost-Based Related Removal	0.04
Paired-Related Removal	-0.03
Route-Related Removal	0.02
Depot-Producer-Related Removal	0.04
Plant-Producer-Related Removal	0.07
Sequential Insertion	-0.01
Sequential Insertion with Plant Satisfaction	0.03
Best-First Insertion	0.02
Regret Insertion	0.04
Minimum-Loss Insertion	0.07

7.4. Computational Results

Table 8 presents the results of applying our algorithm to the instances described in Section 7.1. In this table,

ALNS best is the average of the best solutions found;

ALNS avg. is the mean value of the average of the solutions found over the five runs;

% dev. is the average of standard deviation over the five runs;

T (s) is the average computational time.

Detailed results for each instance are given in Tables 9–15. The standard deviations reported in Table 8 are based on the routing costs; the fixed vehicle costs have been removed. Tables 9–15 report the deviations based on both the total cost and the routing costs.

Table 8: Results for instances of different sizes

Instance size	ALNS best	ALNS avg.	% dev.	T (s)
15	5074.80	5074.80	0.00	5
20	5935.33	5935.68	0.05	7
40	11551.42	11552.10	0.04	26
100	31951.60	31976.71	0.48	129
200	53601.86	53654.35	0.51	235

For the smaller instances, optimal solutions reported by Dayarian et al. (2013b). In Tables 9 and 10, these solutions are given in column BKS DCGR. For the larger instances, we generate lower and upper bounds as described in Section 6. 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, obtained by solving the bin-packing formulations in Section 6. We used Cplex 12.2 to solve these problems. We compute the upper bound by evaluating the cost of the VRP for the reference period, based on the objective function of the multi-period problem. We adapt the algorithm proposed by Dayarian et al. (2013a) for a similar problem to solve the VRP for the reference period. This algorithm can solve problems with up to 50 producers; we do not report bounds for larger problems.

Table 9 gives the results for the instances with 15 producers. For the instances with 2 or 3 depots and plants and 4 or 5 periods, our algorithm was able to find the optimal solutions with a standard deviation of zero. The computational time is about 1/80th of that required by the branch-and-price algorithm of Dayarian et al. (2013b).

Table 10 gives the results for the instances with 20 producers for which the optimal solutions are available. These instances have 2 or 3 depots and 3 plants. Table 11 gives the results for the instances with 20 producers for which the optimal solutions are unknown. For 18 of the instances in Table 10, every run of the algorithm found the optimal solution. For 19 of the instances in Table 11, our solution lies between the computed bounds. We also calculate the value of $\frac{LB}{ALNS_{best}}$. For Table 10, $ALNS_{best}$ is the optimal value for each instance; the average value of this ratio is 0.991. For Table 11, the average value is 0.989. This comparative factor between instances in Tables 10 and 11 indicates the quality of the solutions obtained for instances with 20 producers, for which the optimal solutions are available.

For the instances with 40 producers, all the solutions found lie between the computed bounds. The computational time is less than 2% of the time needed to solve the single-period VRP using the branch-and-price algorithm. The results for the instances with 100 and 200 producers 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.

8. 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.

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.

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.

References

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

T. Athanasopoulos. *The Multi-Period Vehicle Routing Problem and Its Applications*. PhD thesis, Department of Financial & Management Engineering, University of the Aegean, Chios, Greece, 2011.

C. Audet, K.-C. Dang, and D. Orban. Optimization of algorithms with OPAL. Technical report, GERAD, 2012.

N. Azi, M. Gendreau, and J.-Y. Potvin. An adaptive large neighborhood search for a vehicle routing problem with multiple trips. Technical Report CIRRELT-2010-08, CIRRELT, 2010.

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

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

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

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

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

J.-F. Cordeau, G. Laporte, and A. Mercier. A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52:928–936, 2001.

T. G. Crainic and M. Toulouse. Explicit and emergent cooperation schemes for search algorithms. In V. Maniezzo, R. Battiti, and J.-P. Watson, editors, *Learning and Intelligent Optimization*, volume 5313 of *Lecture Notes in Computer Science*, pages 95–109. Springer Berlin Heidelberg, 2008.

T. G. Crainic, G. C. Crişan, M. Gendreau, N. Lahrichi, and W. Rei. Multi-thread integrative cooperative optimization for rich combinatorial problems. In *IPDPS*, pages 1–8, 2009.

G. B. Dantzig and J. H. Ramser. The truck dispatching problem. *Management Science*, 6(1):80–91, 1959.

I. Dayarian, T.G. Crainic, M. Gendreau, and W. Rei. A column generation approach for a multi-attribute vehicle routing problem. Technical Report CIRRELT-2013-57, Montreal, CIRRELT, 2013a.

I. Dayarian, T.G. Crainic, M. Gendreau, and W. Rei. A branch-and-price approach for a multi-period vehicle routing problem. Technical Report CIRRELT-2013-60, Montreal, CIRRELT, 2013b.

P. M. Francis, K. R. Smilowitz, and M. Tzur. The period vehicle routing problem and its extensions. In B. Golden, S. Raghavan, and E. Wasil, editors, *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43 of *Operations Research/Computer Science Interfaces*, pages 73–102. Springer US, 2008.

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

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

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

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

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

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

V. C. Hemmelmayr, K. F. Doerner, and R. F. Hartl. A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3):791–802, 2009.

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

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

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

N. Lahrichi, T. G. Crainic, M. Gendreau, W. Rei, G. C. Crişan, and T. Vidal. An integrative cooperative search framework for multi-decision-attribute combinatorial optimization. Technical report, CIRRELT, Montreal, 2012a.

N. Lahrichi, T.G. Crainic, M. Gendreau, W. Rei, and L-M. Rousseau. Strategic analysis of the dairy transportation problem. Technical Report CIRRELT-2012-80, Montreal, Montreal, Forthcoming in *Journal of Operational Research Society*, CIRRELT, 2012b.

J. K. Lenstra and A. H. G. Rinnooy Kan. Complexity of vehicle routing and scheduling problems. *Networks*, 11:221–227, 1981.

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

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

D. Pisinger and S. Ropke. Large Neighborhood Search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, chapter 13, pages 399–419. Springer US, Boston, MA, 2010.

A. Rahimi-Vahed, Teodor Gabriel Crainic, Michel Gendreau, and Walter Rei. A path relinking algorithm for a multi-depot periodic vehicle routing problem. *Journal of Heuristics*, 19(3):497–524, 2013.

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

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

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

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg, 1998.

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

P. Toth and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. *INFORMS Journal on Computing*, 15:333–346, 2003.

P. Toth and D. Editors Vigo, editors. *The Vehicle Routing Problem*, volume 9. Society for Industrial and Applied Mathematics, 2002.

F. Tricoire. *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, 2006.

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

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

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

Table 9: Results for instances with 15 producers

	Instance	BKS DCGR	T (s)	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-15-2D2P4S-T1	4353.62	19	**4353.62**	**4353.62**	0	0	3
2 depots	pr-15-2D2P4S-T2	4395.48	26	**4052.82**	**4052.82**	0	0	4
2 plants	pr-15-2D2P4S-T3	4478.78	24	**5930.40**	**5930.40**	0	0	6
4 periods	pr-15-2D2P4S-T4	4434.82	13	**4395.48**	**4395.48**	0	0	4
	pr-15-2D2P4S-T5	4472.04	7	**4090.51**	**4090.51**	0	0	4
	pr-15-2D2P5S-T1	4358.84	22	**4478.78**	**4478.78**	0	0	3
2 depots	pr-15-2D2P5S-T2	4403.64	30	**4148.54**	**4148.54**	0	0	4
2 plants	pr-15-2D2P5S-T3	4439.31	16	**5959.89**	**5959.89**	0	0	6
5 periods	pr-15-2D2P5S-T4	4449.81	9	**4434.82**	**4434.82**	0	0	3
	pr-15-2D2P5S-T5	4476.56	7	**4115.57**	**4115.57**	0	0	4
	pr-15-2D3P4S-T1	5855.70	1422	**5894.16**	**5894.16**	0	0	5
2 depots	pr-15-2D3P4S-T2	5860.58	911	**4472.04**	**4472.04**	0	0	3
3 plants	pr-15-2D3P4S-T3	5831.72	964	**4158.00**	**4158.00**	0	0	4
4 periods	pr-15-2D3P4S-T4	5821.90	998	**5837.27**	**5837.27**	0	0	5
	pr-15-2D3P4S-T5	5843.12	947	**4358.84**	**4358.84**	0	0	4
	pr-15-2D3P5S-T1	5871.89	1103	**5898.82**	**5898.82**	0	0	5
2 depots	pr-15-2D3P5S-T2	5886.37	928	**4055.18**	**4055.18**	0	0	4
3 plants	pr-15-2D3P5S-T3	5843.29	972	**4403.64**	**4403.64**	0	0	4
5 periods	pr-15-2D3P5S-T4	5843.12	980	**4098.60**	**4098.60**	0	0	4
	pr-15-2D3P5S-T5	5832.51	874	**5945.35**	**5945.35**	0	0	6
	pr-15-3D2P4S-T1	4052.82	58	**4439.31**	**4439.31**	0	0	3
3 depots	pr-15-3D2P4S-T2	4090.51	23	**4118.45**	**4118.45**	0	0	4
2 plants	pr-15-3D2P4S-T3	4148.54	34	**5985.97**	**5985.97**	0	0	7
4 periods	pr-15-3D2P4S-T4	4115.57	18	**4132.68**	**4132.68**	0	0	4
	pr-15-3D2P4S-T5	4158.00	34	**4449.81**	**4449.81**	0	0	3
	pr-15-3D2P5S-T1	4055.18	30	**5900.75**	**5900.75**	0	0	5
3 depots	pr-15-3D2P5S-T2	4098.60	27	**4476.56**	**4476.56**	0	0	3
2 plants	pr-15-3D2P5S-T3	4118.45	28	**5896.57**	**5896.57**	0	0	6
5 periods	pr-15-3D2P5S-T4	4132.68	24	**4152.31**	**4152.31**	0	0	4
	pr-15-3D2P5S-T5	4152.31	17	**5866.92**	**5866.92**	0	0	6
	pr-15-3D3P4S-T1	5930.40	759	**5843.12**	**5843.12**	0	0	6
3 depots	pr-15-3D3P4S-T2	5959.89	838	**5871.89**	**5871.89**	0	0	6
3 plants	pr-15-3D3P4S-T3	5894.16	600	**5843.12**	**5843.12**	0	0	6
4 periods	pr-15-3D3P4S-T4	5837.27	447	**5855.70**	**5855.70**	0	0	6
	pr-15-3D3P4S-T5	5898.82	764	**5886.37**	**5886.37**	0	0	6
	pr-15-3D3P5S-T1	5945.35	717	**5821.90**	**5821.90**	0	0	6
3 depots	pr-15-3D3P5S-T2	5985.97	835	**5860.58**	**5860.58**	0	0	6
3 plants	pr-15-3D3P5S-T3	5900.75	615	**5843.29**	**5843.29**	0	0	6
5 periods	pr-15-3D3P5S-T4	5896.57	561	**5831.72**	**5831.72**	0	0	6
	pr-15-3D3P5S-T5	5866.92	473	**5832.51**	**5832.51**	0	0	6
	Avg.	5074.80	429	**5074.80**	**5074.80**	0	0	5

Table 10: Results for instances with 20 producers with available optimal solutions

	Instance	LB	BKS DCGR	LB/BKS	T (s)	ALNS best over 5	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-20-2D3P4S-T1	5810.84	5873.92	0.989	3934	**5873.92**	**5873.92**	0.00	0.00	6
2 depots	pr-20-2D3P4S-T2	5827.34	5907.21	0.986	7338	**5907.21**	**5907.21**	0.00	0.00	6
3 plants	pr-20-2D3P4S-T3	5851.1	5890.07	0.993	4431	**5890.07**	**5890.07**	0.00	0.00	7
4 periods	pr-20-2D3P4S-T4	5790.35	5807.1	0.997	4240	**5807.1**	**5807.1**	0.00	0.00	7
	pr-20-2D3P4S-T5	5789.32	5861.86	0.988	3617	**5861.86**	**5861.86**	0.00	0.00	6
	pr-20-2D3P5S-T1	5827.04	5883.98	0.990	3371	**5883.98**	**5883.98**	0.00	0.00	7
2 depots	pr-20-2D3P5S-T2	5848.16	5919.85	0.988	3422	**5919.85**	**5919.85**	0.00	0.00	7
3 plants	pr-20-2D3P5S-T3	5854.21	5888.36	0.994	4814	**5888.36**	**5888.36**	0.00	0.00	7
5 periods	pr-20-2D3P5S-T4	5826.85	5856.77	0.995	2239	**5856.77**	**5856.77**	0.00	0.00	7
	pr-20-2D3P5S-T5	5808.6	5833.37	0.996	2166	**5833.37**	**5833.37**	0.00	0.00	7
	pr-20-3D3P4S-T1	5951.3	6013.02	0.990	13090	**6013.02**	**6013.02**	0.00	0.00	8
3 depots	pr-20-3D3P4S-T2	5974.3	6043.07	0.989	9930	**6043.07**	**6043.07**	0.00	0.00	8
3 plants	pr-20-3D3P4S-T3	5950.7	6026.6	0.987	10404	**6026.6**	**6026.6**	0.00	0.00	8
4 periods	pr-20-3D3P4S-T4	5898.75	5948.7	0.992	6706	**5948.7**	**5948.7**	0.00	0.00	7
	pr-20-3D3P4S-T5	5917.5	5980.32	0.989	7072	**5980.32**	5981.44	0.03	0.12	8
	pr-20-3D3P5S-T1	5979.4	6032.42	0.991	13749	**6032.42**	**6032.42**	0.00	0.00	8
3 depots	pr-20-3D3P5S-T2	6002.4	6067.63	0.989	14925	**6067.63**	**6067.63**	0.00	0.00	8
3 plants	pr-20-3D3P5S-T3	5962.9	6037.43	0.988	9448	**6037.43**	6038.69	0.03	0.11	8
5 periods	pr-20-3D3P5S-T4	5950.56	6016.16	0.989	7633	**6016.16**	**6016.16**	0.00	0.00	8
	pr-20-3D3P5S-T5	5924.85	5982.43	0.990	5735	**5982.43**	**5982.43**	0.00	0.00	8
	Avg.	5887.324	5943.514	0.991	6913	5943.5135	5943.63	0.00	0.01	7

Table 11: Results for instances with 20 producers without available optimal solutions

	Instance	LB	UB	ALNS best over 5	LB/ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-20-2D2P4S-T1	6162.22	6301	6237.18	0.988	6237.18	0.00	0.00	7
2 depots	pr-20-2D2P4S-T2	6186.47	6366.29	6266.92	0.987	6266.92	0.00	0.00	7
2 plants	pr-20-2D2P4S-T3	6158.79	6499.43	6226.73	0.989	6226.73	0.00	0.00	7
4 periods	pr-20-2D2P4S-T4	6121.98	6481.62	6178.21	0.991	6178.21	0.00	0.00	6
	pr-20-2D2P4S-T5	6185.52	6539.06	6246.02	0.990	6246.02	0.00	0.00	7
	pr-20-2D2P5S-T1	6182.71	6302.52	6259.28	0.988	6259.28	0.00	0.00	7
2 depots	pr-20-2D2P5S-T2	6220.79	6367.63	6301.07	0.987	6301.07	0.00	0.00	8
2 plants	pr-20-2D2P5S-T3	6160.79	6431.58	6218.3	0.991	6218.3	0.00	0.00	7
5 periods	pr-20-2D2P5S-T4	6172.34	6484.31	6258.85	0.986	6258.85	0.00	0.00	7
	pr-20-2D2P5S-T5	6158.74	6543.94	6230.57	0.988	6230.57	0.00	0.00	7
	pr-20-3D2P4S-T1	5552.08	5602.42	5588.46	0.993	5588.46	0.00	0.00	6
3 depots	pr-20-3D2P4S-T2	5578.15	5640.67	5604.2	0.995	5604.2	0.00	0.00	6
2 plants	pr-20-3D2P4S-T3	5542.82	5639.71	5627.18	0.985	5627.86	0.03	0.13	6
4 periods	pr-20-3D2P4S-T4	5520.98	5597.4	5597.4	0.986	5600.46	0.12	0.62	6
	pr-20-3D2P4S-T5	5550.18	5623.64	5623.64	0.987	5623.64	0.00	0.00	7
	pr-20-3D2P5S-T1	5553.98	5614.93	5597.81	0.992	5597.81	0.00	0.00	7
3 depots	pr-20-3D2P5S-T2	5565.85	5639.79	5616.21	0.991	5616.21	0.00	0.00	6
2 plants	pr-20-3D2P5S-T3	5548.67	5626.75	5620.96	0.987	5620.96	0.00	0.00	7
5 periods	pr-20-3D2P5S-T4	5543.6	5622.43	5622.43	0.986	5624.03	0.04	0.18	7
	pr-20-3D2P5S-T5	5532.29	5621.31	5621.31	0.984	5627.81	0.15	0.75	7
	Avg.			5927.14	0.989	5927.73	0.02	0.08	7

Table 12: Results for instances with 40 producers

	Instance	Bounds on opt. sol.	T (s)	ALNS best	ALNS avg. over 5	% dev total cost	% dev routing cost	T (s)
	pr-40-2D2P4S-T1	[12229.2, 12405.5]	5441	12389.5	12389.5	0.00	0.00	19
2 depots	pr-40-2D2P4S-T2	[12336, 12558.3]	5337	12535.1	12535.1	0.00	0.00	20
2 plants	pr-40-2D2P4S-T3	[12556.2, 12780.2]	5342	12752.5	12754.4	0.02	0.08	19
4 periods	pr-40-2D2P4S-T4	[12569.8, 12700.3]	5386	12679.6	12679.6	0.00	0.00	19
	pr-40-2D2P4S-T5	[12700.4, 12881.6]	5416	12856.6	12860.2	0.04	0.15	21
	pr-40-2D2P5S-T1	[12241.8, 12415.4]	5344	12398.7	12398.7	0.00	0.00	22
2 depots	pr-40-2D2P5S-T2	[12359.1, 12574.4]	5374	12555	12555	0.00	0.00	21
2 plants	pr-40-2D2P5S-T3	[12463.5, 12652]	5358	12632.8	12632.8	0.00	0.00	22
5 periods	pr-40-2D2P5S-T4	[12609.8, 12742.5]	5344	12730.4	12730.4	0.00	0.00	22
	pr-40-2D2P5S-T5	[12693.3, 12844.1]	5352	12825.6	12825.6	0.00	0.00	22
	pr-40-2D3P4S-T1	[11826.5, 12019.5]	1540	11956.1	11959.6	0.04	0.15	27
2 depots	pr-40-2D3P4S-T2	[11893.3, 12135.6]	1536	12039.1	12040	0.02	0.06	26
3 plants	pr-40-2D3P4S-T3	[11933.6, 12188.6]	1540	12093.4	12093.4	0.00	0.00	26
4 periods	pr-40-2D3P4S-T4	[11838.9, 12033.8]	1532	11965.4	11965.4	0.00	0.00	26
	pr-40-2D3P4S-T5	[11902.4, 12191.2]	1540	12072.3	12073.6	0.02	0.07	28
	pr-40-2D3P5S-T1	[11846.3, 12040.4]	1525	11984.5	11986.5	0.02	0.08	29
2 depots	pr-40-2D3P5S-T2	[11916.4, 12157.7]	1535	12074.6	12075.9	0.02	0.07	28
3 plants	pr-40-2D3P5S-T3	[11899.8, 12120.9]	1725	12045.8	12045.8	0.00	0.00	28
5 periods	pr-40-2D3P5S-T4	[11911.8, 12110.6]	1515	12068.7	12068.7	0.00	0.00	29
	pr-40-2D3P5S-T5	[11895.3, 12128.5]	1534	12046.8	12046.8	0.00	0.00	28
	pr-40-3D2P4S-T1	[9681.72, 9862.4]	640	9794.28	9794.28	0.00	0.00	27
3 depots	pr-40-3D2P4S-T2	[9725.53, 9955.19]	640	9860.69	9860.69	0.00	0.00	27
2 plants	pr-40-3D2P4S-T3	[9770.77, 10051]	634	9916.4	9917.29	0.02	0.08	26
4 periods	pr-40-3D2P4S-T4	[9655.76, 9937.56]	650	9763.42	9768.03	0.07	0.29	26
	pr-40-3D2P4S-T5	[9726.14, 10051.9]	648	9824.06	9824.06	0.00	0.00	25
	pr-40-3D2P5S-T1	[9688.24, 9873.74]	641	9812.15	9812.64	0.01	0.05	29
3 depots	pr-40-3D2P5S-T2	[9748.89, 9974.3]	648	9888.96	9888.96	0.00	0.00	30
2 plants	pr-40-3D2P5S-T3	[9749.21, 9990.35]	635	9892.71	9893.51	0.01	0.04	29
5 periods	pr-40-3D2P5S-T4	[9728.32, 9990.4]	638	9863.89	9863.89	0.00	0.00	30
	pr-40-3D2P5S-T5	[9704.04, 10017.8]	644	9829.02	9830.13	0.02	0.11	30
	pr-40-3D3P4S-T1	[11525.3, 11697.1]	229	11642.8	11642.8	0.00	0.00	24
3 depots	pr-40-3D3P4S-T2	[11569.3, 11788.6]	233	11709.7	11709.7	0.00	0.00	24
3 plants	pr-40-3D3P4S-T3	[11618, 11833.3]	228	11718.2	11718.2	0.00	0.00	25
4 periods	pr-40-3D3P4S-T4	[11525.5, 11675.7]	233	11624.6	11629.1	0.06	0.27	26
	pr-40-3D3P4S-T5	[11570.5, 11807.1]	228	11727.6	11727.8	0.00	0.02	27
	pr-40-3D3P5S-T1	[11530.4, 11701.8]	233	11654	11654	0.00	0.00	25
3 depots	pr-40-3D3P5S-T2	[11592.2, 11802.1]	221	11736	11736	0.00	0.00	28
3 plants	pr-40-3D3P5S-T3	[11595, 11779.8]	229	11688.2	11688.2	0.00	0.00	27
5 periods	pr-40-3D3P5S-T4	[11593.8, 11742.4]	252	11718.6	11718.6	0.00	0.00	28
	pr-40-3D3P5S-T5	[11576.9, 11743.8]	231	11688.9	11688.9	0.00	0.00	30
	Avg.	[11412.5, 11623.9]	1949	11551.417	11552.1	0.01	0.04	26

Table 13: 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 14: 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 15: 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.	53601.855	53654.3425	0.12	0.51	235