A large neighborhood search heuristic for supply chain network design

Majid Eskandarpour, Pierre Dejax, Olivier Péton

PII: S0305-0548(16)30275-1
DOI: 10.1016/j.cor.2016.11.012
Reference: CAOR 4125

To appear in: *Computers and Operations Research*

Received date: 28 August 2014
Revised date: 2 November 2016
Accepted date: 12 November 2016

Please cite this article as: Majid Eskandarpour, Pierre Dejax, Olivier Péton, A large neighborhood search heuristic for supply chain network design, *Computers and Operations Research* (2016), doi: 10.1016/j.cor.2016.11.012

A large neighborhood search heuristic for supply chain network design

Majid Eskandarpour [1,2], Pierre Dejax[1], Olivier Péton[1]

[1] *IMT Atlantique, 4 rue Alfred Kastler, F-44307 Nantes Cedex, France*
[2] *Department of Mathematics, Centre for Operational Research and Logistics, University of Portsmouth, Lion Gate Building, Lion Terrace, Portsmouth PO1 3HF,UK*

Abstract

Many exact and approximate solution techniques have been used to solve facility location problems and, more generally, supply chain network design problems. Yet, the Large Neighborhood Search technique (LNS) has almost never been suggested for solving such problems, although it has proven its efficiency and flexibility in solving other complex combinatorial optimization problems. In this paper, we propose an LNS framework for solving a four-layer single period multi-product supply chain network design problem. One important feature of the model is that it includes inter-modality: the itinerary followed by the cargo from origin to destination may take several transportation modes. Moreover, several modes may compete on some arcs. Location decisions for intermediate facilities (e.g. plants and distribution centers) are determined by the LNS while transportation modes and product flow decisions are determined by a greedy heuristic. As a post-optimization step, linear programming is used to optimize product flows once the structure of the logistics network is fixed. Extensive experiments, based on randomly generated instances of different sizes and characteristics, show the effectiveness of the method compared with a state-of-the-art solver.

Keywords: Supply chain network design, facility location, Large Neighborhood Search

*Corresponding author. Olivier Péton, IMT Atlantique, 4 rue Alfred Kastler, 44300 Nantes, France. olivier.peton@mines-nantes.fr, +33 251 858 313

A large neighborhood search heuristic for supply chain network design

Majid Eskandarpour [1,2], Pierre Dejax[1], Olivier Péton[1]

[1] *IMT Atlantique, 4 rue Alfred Kastler, F-44307 Nantes Cedex, France*
[2] *Department of Mathematics, Centre for Operational Research and Logistics, University of Portsmouth, Lion Gate Building, Lion Terrace, Portsmouth PO1 3HF,UK*

1. Introduction

The goal of this paper is to propose and evaluate a Large Neighborhood Search approach to solve a facility location and supply chain network design problem. The field of facility location has been very active since the description of the p-median problem by Hakimi [1] more than fifty years ago. In the field of supply chain management and logistics applications, seminal facility location models have been progressively incorporated into larger models, which now constitute the family of Supply Chain Network Design (SCND) problems. Most SCND models aim to design a supply chain optimally with regard to a single objective function representing an economic goal. The great majority of such problems are classified as NP-hard [2]. A large variety of exact and approximate solution techniques have been proposed for solving such problems. General solvers are often able to solve small- or medium-sized instances to optimality. However, rich models or large enough instances of classic models cannot be solved to optimality even by state-of-the-art solvers within acceptable time. Thus, there may be a need for customized algorithms and heuristics [3].

Many heuristic methods have been used to solve SCND problems. Surprisingly enough, the Large Neighborhood Search (LNS) heuristic has almost never been used in this context. It was introduced by Shaw in a constraint programming framework [4]. The underlying principle is to *destroy* and *repair* iteratively the current solution in order to improve it progressively. Destroying the current solution consists of de-selecting a subset of components (in our case: facilities) from the solution. Repairing the solution consists of restoring feasibility by selecting new facilities and modifying the product flows. This principle is similar to that of the *ruin and recreate* scheme introduced by Schrimpf et al. [5]. The efficiency of the method relies on the choice and appropriate use of application-oriented procedures called *removal* operators and *repair* operators. Pisinger and Ropke [6] present an extensive survey of the method.

The LNS technique has proven its efficiency in several fields of combinatorial optimization, such as vehicle routing and scheduling. To the best of our knowledge, the use of the LNS for solving SCND problems is still very scarce. Copado-Méndez et al. [7] model two case studies in chemical engineering and solve them with an LNS approach. The authors identify a benefit of the LNS: removal and repair operators, which are largely dependent on the models to be solved, provide high flexibility. It is a general framework in which potential operators can be used or not depending on the attributes of the model to be solved.

In this paper, we propose an LNS framework for solving an SCND model with four layers (suppliers, plants, distribution centers and customers), multiple products and transportation modes. The model includes

*Corresponding author. Olivier Péton, IMT Atlantique, 4 rue Alfred Kastler, 44300 Nantes, France. olivier.peton@mines-nantes.fr, +33 251 858 313

location decisions in the two intermediate levels. The main goal of our research is to assess the efficiency of the LNS approach for designing a fairly generic and realistic supply chain and to draw lessons for further research. In particular, several challenges are emerging. First, SCND models contain both binary and continuous variables, which must be treated separately. We adopt a hierarchical approach consisting, at each iteration, of identifying the facility locations with the LNS operators and then determining the values of the continuous variables by means of a greedy heuristic. Our model includes binary variables not only to model the selection of facilities but also to choose between competing transportation modes. In our heuristic, location decisions are fixed using the LNS while transportation modes are determined *a posteriori* by a greedy heuristic.

In the case of SCND problems, the role of removal operators is to de-select a subset of facility locations from the current solution. The role of repair operators is to select a subset of unselected facility locations. Contrary to vehicle routing or scheduling problems where the number of customers to visit or the number of tasks to schedule is known *a priori*, the number of selected candidates in the optimal solution of an SCND problem is not fixed. Thus, the LNS removal and repair operators must also manage the number of selected facility locations. For this purpose, we propose the notion of a *network structure*, which will be defined in Section 4.

The remainder of the paper is organized as follows. Section 2 reviews the SCND literature and describes its relation to our work. Section 3 describes the problem and the model formulation. The proposed solution method is presented in Section 4. Section 5 is dedicated to the description of the LNS operators. Section 6 provides the computational results for 60 randomly generated test instances. We tried to design a variety of realistic patterns of supply chains: all locations are generated in a territory divided into regions, and the regions can have their own special features (e.g., clusters of facilities or customers, low or high fixed costs). The conclusion and suggested future research appear in Section 7.

2. Related SCND literature

The large amount of work in the area of facility location and SCND problems has been classified and synthesized in a number of review papers. See, for example, the recent reviews [3, 8].

Many metaheuristic and evolutionary methods have been recently developed, including simulated annealing [e.g. 9, 10], tabu search [e.g. 11, 12], VNS [e.g. 13, 14], genetic algorithms [e.g. 15, 16], memetic algorithms [e.g. 17, 18], and scatter search [e.g. 19]. However, there is still a need to develop efficient and flexible heuristic and metaheuristic methods to solve hard problems or large instances that cannot be handled by exact methods or Mixed Integer Linear Programming (MILP) solvers.

Multiple variants of SCND models include the consideration of sizing decisions, allocation of products to facilities and supplier selection. The recent mathematical models generally include features such as multiple layers and types of facilities, multiple products and multiple time periods. Extended SCND models integrate the bill of material for complex end products, uncertainty, risk management, disruption, reverse logistics or sustainable development factors.

As mentioned above, one important feature of the model proposed in this paper is the possibility of choosing from multiple transportation modes between facilities, which had not received much attention in the SCND literature until recently [8]. Carlsson and Rönnqvist [20] describe a case study in the distribution of pulp from Sweden to several European countries. The international customers are supplied by three possible modes of transportation: vessel, train and lorry. The paper by Eskigun et al. [21] describes the outbound supply chain network of an automotive company. It is assumed that all vehicle types from the same plant are delivered to a destination using the same transportation mode to take advantage of economies of scale and to simplify the delivery process (e.g. loading, unloading, tracking) of the vehicles. The same assumption is

made in our model [8]. Cordeau et al. [22] develop a comprehensive multi-stage network design model: at the strategic level, they investigate facility location and capacity limits. At the tactical level, they also integrate the selection of transportation modes considering fixed and variable costs and the capacity limits. Their model is solved by two methods: a simplex-based branch-and-bound and a Benders decomposition approach. Other recent works proposing models closely related to ours and including the choice of transportation mode are the following.

Wu et al. [23] study a spare parts logistics network encompassing three types of decision: facility location, item vendor selection and choice of transportation mode. They study two approaches consisting of determining all decisions simultaneously or determining the location decisions first. Sadjady and Davoudpour [24] propose a single-period two-echelon multi-commodity model regarding strategic and tactical decisions as well as the choice of transportation modes. The problem is solved with a Lagrangean relaxation heuristic. Multi-modal SCND models can be a means to consider various costs and capacity limits on arcs [25], but also to manage the delivery time between plants and customers [26] or to handle economic and service level objectives [27]. Moreover, knowing that transport accounts for 22% of global CO_2 emissions, the choice of transportation modes is also a main driver for optimizing the environmental performance of a supply chain [28].

Taking transportation modes into account complicates the problem since it introduces many additional binary variables. Thus, most solution methods decompose the original problem into several sub-problems, or fix each type of decision variable sequentially. We adopt the latter approach: at each iteration, the location decisions are fixed first and then transportation modes are chosen.

3. Problem definition and modeling

3.1. Problem settings

We consider a multi-product supply chain network consisting of four layers: suppliers, production plants, distribution centers (denoted by DCs) and customers (retail stores or final customers), as depicted in Figure 1.

The locations of suppliers and customers are known, whereas those of plants and DCs have to be determined from a set of candidate locations. In the first layer, suppliers provide the raw materials or components to the plants. These products are then converted into finished goods through value-added operations performed in plants. As mentioned above, finished goods are shipped from plants to DCs and from DCs to customers. Customer demand is assumed deterministic and known a priori. We do not impose single sourcing constraints, *i.e.* an entity can be supplied by several entities from the preceding layer.

At the strategic level of planning that is considered here, the model treats facilities as black boxes: the detail of all internal activities such as storage, production operations and internal logistics, is not considered. Thus, the capacity of a facility is expressed as a single value limiting the total output flow throughout this facility.

The facility location decisions at plants and DCs are guided by two types of costs. Fixed costs of facilities are paid only if the corresponding facility is selected. Processing costs are variable costs assumed proportional to the level of activity (i.e. the outgoing flow) of the corresponding facility.

Companies conducting SCND studies usually do a preliminary filtering of possible facility locations. When too few facilities are selected, transportation costs are prohibitive. On the other hand, selecting too many facilities increases the supply chain complexity and the fixed costs, requiring possibly prohibitive capital investments. According to Bode and Wagner [29], there is a general consensus that supply chains have become increasingly complex over the last decades and that this is not a desirable feature. Thus, one important assumption in our study is that decision makers or industrial experts are able to give a good *a*

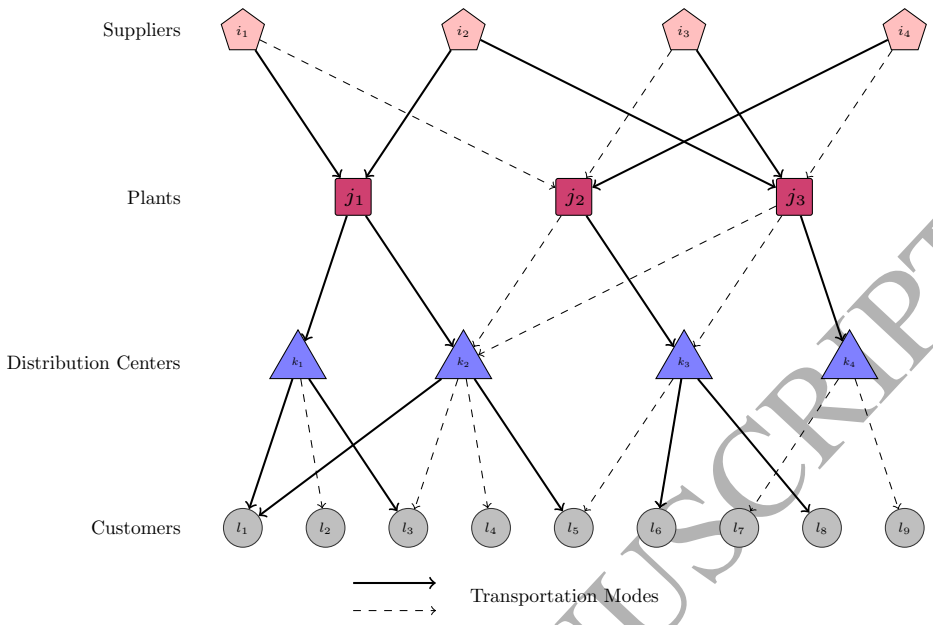

Figure 1: The supply chain considered

priori approximation of the minimum and maximum number of plants and DCs to be selected. These values are determined within reasonably narrow ranges, later denoted by $\{J_{min}, \ldots, J_{max}\}$ and $\{K_{min}, \ldots, K_{max}\}$, respectively.

In our study, several transportation modes can be used, such as road, rail, inland navigation or air transport, to ship products between the nodes of the network. It is assumed that a restricted set of suitable transportation modes has been identified *a priori* for each pair of nodes, with respect to multiple criteria such as availability and safety, shipping costs, GHG emissions, shipment capacities, speed and frequency. At the strategic decision level, the cost of most transportation modes is assumed linear with respect to the quantity carried. However, some transportation modes incur a fixed charge. For example, a company with an internal fleet of trucks will pay a fixed cost (amortization, maintenance, insurance, etc.) even if the vehicles are not used. In addition, some transportation modes also require a minimal quantity of goods to be shipped. It is assumed that only one transportation mode is chosen between any pair of nodes and that all products are compatible enough to be loaded onto the same transportation mode. At a strategic decision level, it is considered that enough vehicles will be available on each arc of the network. Thus, there is no fleet restriction and no upper capacity limit on the arcs.

The main decisions to be made in our problem are: (i) to select facility locations (plants and DCs), (ii) to choose transportation modes between suppliers and plants, plants and DCs, and DCs and customers, (iii) to determine the product flows throughout the logistics network, in order to satisfy all customer demands and meet given logistics constraints. The objective function of this problem is to minimize the overall costs over one single period of planning (typically, but not necessarily, one year). Fixed facility costs and variables costs related to processing and transportation operations are considered. By processing costs, we mean procurement costs from suppliers, transformation costs within plants and material handling or storage costs at DCs.

3.2. Data sets, parameters and variables

We consider a set I of suppliers, a set J of candidate plants, a set K of candidate DCs, a set L of customers, a set P of products and a set M of candidate transportation modes. The SCND problem is defined on a directed graph $\psi = (V, A)$. The set V of nodes includes sets I, J, K and L. The set A of arcs defines all possible links between nodes. This potentially includes all links between two successive layers represented in Figure 1.

We introduce the following notations:

- d_l^p: demand of customer $l \in L$ for product $p \in P$;
- cap_i: capacity at node $i \in I \cup J \cup K$;
- c_j: fixed cost of selecting facility $j \in J \cup K$;
- v_{ij}^{mp}: variable transportation cost of one unit of product $p \in P$ on arc $(i, j) \in A$ by mode $m \in M$;
- a_i^p: processing cost of one unit of product $p \in P$ at $i \in I \cup J \cup K$;
- g_{ij}^m: fixed cost of transportation mode $m \in M$ along arc $(i, j) \in A$;
- V_{ij}^m: minimum threshold volume for using transportation mode $m \in M$ along arc $(i, j) \in A$.

We also define lower and upper bounds on the number of selected facility locations: the number of selected plants must lie in the set $\{J_{min}, \ldots, J_{max}\}$ and the number of selected DCs must lie in the set $\{K_{min}, \ldots, K_{max}\}$. J_{min} and K_{min} can be calculated as a straightforward lower bound: considering the plants in non-increasing order of their capacities, the sum of customer demands must be larger than the capacity in the $J_{min} - 1$ first plants and smaller than the capacity in the J_{min} first plants (the same principle can be applied to DCs). The four parameters J_{min}, J_{max}, K_{min} and K_{max} can also be set by decision-makers depending on preliminary logistics studies, the history of the company and the sector of activity.

The binary variable y_j is equal to 1 if facility $j \in J \cup K$ is selected and 0 otherwise. In order to choose the transportation modes throughout the network, the value of binary variable t_{ij}^m is set at 1 if transportation mode $m \in M$ is chosen for arc $(i, j) \in A$ and 0 otherwise. Continuous variables x_{ij}^{mp} represent the flow of product $p \in P$ on arc $(i, j) \in A$ using transportation mode $m \in M$.

Without loss of generality, it is assumed that each product can be processed by every facility. In practice, if a facility cannot process one given product, this can be handled by ignoring the corresponding variable x_{ij}^{mp} or imposing its value as 0 in the model. Similarly, an arc $(i, j) \in A$ (with associated variables t_{ij}^m and x_{ij}^{mp}) is defined only when it is conceivable.

3.3. Mathematical formulation

In order to represent and solve the problem described above, we propose the following MILP model minimizing the economic objective (1):

$$min\ z = \sum_{j \in J \cup K} c_j y_j + \sum_{(i,j) \in A} \sum_{m \in M} \sum_{p \in P} (a_i^p + v_{ij}^{mp}) x_{ij}^{mp} + \sum_{(i,j) \in A} \sum_{m \in M} g_{ij}^m t_{ij}^m \quad (1)$$

The first term in the objective function represents the sum of the facility fixed costs. The second term represents the processing costs and variable transportation costs. The last term represents the fixed transportation costs.

Constraints (2) and (3) are the flow conservation constraints at plants and DCs per product, respectively.

$$\sum_{i\in I}\sum_{m\in M} x_{ij}^{mp} = \sum_{k\in K}\sum_{m\in M} x_{jk}^{mp} \qquad \forall j \in J, p \in P \tag{2}$$

$$\sum_{J\in J}\sum_{m\in M} x_{jk}^{mp} = \sum_{l\in L}\sum_{m\in M} x_{kl}^{mp} \qquad \forall k \in K, p \in P \tag{3}$$

Constraints (4) ensure the satisfaction of customer demands. Note that an equivalent formulation with a greater than or equal to sign would lead to the same optimal value of the objective function.

$$\sum_{k\in K}\sum_{m\in M} x_{kl}^{mp} = d_l^p \qquad \forall l \in L, p \in P \tag{4}$$

Constraints (5)–(7) model capacity constraints at suppliers, plants and DCs, respectively. In addition, (6) and (7) state that the products can be shipped only to selected facilities.

$$\sum_{j\in J}\sum_{m\in M}\sum_{p\in P} x_{ij}^{mp} \leq cap_i \qquad \forall i \in I \tag{5}$$

$$\sum_{k\in K}\sum_{m\in M}\sum_{p\in P} x_{jk}^{mp} \leq cap_j\ y_j \qquad \forall j \in J \tag{6}$$

$$\sum_{l\in L}\sum_{m\in M}\sum_{p\in P} x_{kl}^{mp} \leq cap_k\ y_k \qquad \forall k \in K. \tag{7}$$

Constraints (8) ensure that at most one transportation mode is chosen between two nodes. Constraints (9) guarantee that the minimal capacity of each transportation mode is satisfied. Constraints (10) state that if transportation mode m is not selected on the arc $(i, j) \in A$, then there is no flow on this arc. The parameter M represents a big number. For instance, it can be set to the value of the total demand. This constraint can also be used to limit the flow on each arc by setting smaller values of M.

$$\sum_{m\in M} t_{ij}^m \leq 1 \qquad \forall (i,j) \in A \tag{8}$$

$$\sum_{p\in P} x_{ij}^{mp} \geq V_{ij}^m\ t_{ij}^m \qquad \forall (i,j) \in A, m \in M \tag{9}$$

$$\sum_{p\in P} x_{ij}^{mp} \leq M\ t_{ij}^m \qquad \forall (i,j) \in A, m \in M \tag{10}$$

Constraints (11) and (12) bound the number of selected plants and DCs, respectively.

$$J_{min} \leq \sum_{j\in J} y_j \leq J_{max} \tag{11}$$

$$K_{min} \leq \sum_{j\in K} y_j \leq K_{max} \tag{12}$$

Finally, constraints (13) – (15) state non-negativity and binary restrictions on decision variables.

$$y_j \in \{0,1\} \qquad \forall j \in J \cup K \tag{13}$$
$$t_{ij}^m \in \{0,1\} \qquad \forall (i,j) \in A, m \in M \tag{14}$$
$$x_{ij}^{mp} \geq 0 \qquad \forall (i,j) \in A, p \in P, m \in M \tag{15}$$

4. A Large Neighborhood Search heuristic

The LNS framework which we propose to solve the SCND model is described in the following. It deals with the three main types of decision variables: facility location (binary variables), choice of transportation modes (binary variables) and calculation of optimal product flows (continuous variables). A key issue is to determine the number and location of plants and DCs. In our implementation, each iteration of the LNS framework imposes a fixed number of selected facilities. For this purpose, we define the *network structure* as follows.

Definition The pair $(\sum_{j \in J} y_j, \sum_{k \in K} y_k)$, representing the number of plants and DCs selected in a solution of model (1)-(15), is called the *network structure*.

Note that the network structure is only a pair of integer values that gives a rough description of the general shape of the supply chain but does not state which particular facilities are selected. Since the number of selected facility locations is bounded by constraints (11) and (12), there are $(J_{max}-J_{min}+1)\times(K_{max}-K_{min}+1)$ possible network structures. This value is very small compared to the $\frac{J_{max}!}{J_{min}!(J_{max}-J_{min})!} \times \frac{K_{max}!}{K_{min}!(K_{max}-K_{min})!}$ potential combinations of variables y_j $(j \in J \cup K)$. One key principle of our heuristic is to intensify the search around the most promising network structures. This requires being able to assess each network structure, in addition to individual solutions. Similarly to the adaptive LNS scheme [30], a score is calculated for each network structure. This score is updated every 100 iterations of the LNS heuristic.

The main challenges of the LNS algorithm are (i) to handle both binary and continuous variables, (ii) to determine the strategy to visit and evaluate promising network structures and (iii) to choose transportation modes. The remainder of this section is organized as follows. The first subsection depicts the proposed LNS framework. Then, subsequent subsections detail the critical points of the heuristic: how to determine an initial solution and to initialize the score of each network structure (4.2), how the scores are used to choose the network structure at the next iteration (4.3), how to determine the number of facilities to de-select or select (4.3), and how to determine the transportation modes and product flows (4.5).

4.1. The proposed LNS framework

The proposed LNS framework is depicted in Algorithm 1.

The initialization of the score of all network structures (line 1) and the determination of the initial solution (line 2) are detailed in Section 4.2. The termination criterion (line 5) is based on a maximal number of iterations (*IterMax*) but other classic criteria would be possible (e.g. number of iterations without improvement, computing time limit). A key point of Algorithm 1 is that each iteration modifies the value of facility location variables in only one layer in the network (plants or DCs). At each iteration, the active layer is randomly selected (line 6). The network structure for the next iteration (line 7) is chosen before removal and repair operators are applied. It is called the *target network structure* since it helps define how many facilities should be de-selected by the removal operators and selected by the repair operators (line 8). More details are given in Sections 4.3 and 4.4. The removal and repair operators are also randomly selected, from a set of LNS operators (line 9) described in Section 5.

Algorithm 1 LNS framework for our supply chain network design problem

for each network structure, **do** calculate an initial score
Determine the initial solution $\mathcal{S}^0$
Best Solution: $\mathcal{S}^* \leftarrow \mathcal{S}^0$
Current solution $\mathcal{S} \leftarrow \mathcal{S}^0$
for $iter = 1$ to $IterMax$ **do**
  Randomly choose the active layer between J or K
  Choose a target network structure using a biased roulette wheel principle based on the scores
  Apply Algorithm 3 to set the number of facilities to be removed (f^-) and repaired (f^+)
  Randomly choose a *Removal* and a *Repair* operator
  if $f^- > 0$ **then** $\mathcal{S}' \leftarrow Removal(\mathcal{S})$, **else** $\mathcal{S}' \leftarrow \mathcal{S}$
  if $f^+ > 0$ **then** $\mathcal{S}'' \leftarrow Repair(\mathcal{S}')$, **else** $\mathcal{S}'' \leftarrow \mathcal{S}'$
  Apply Algorithm 4 to choose transportation modes and to set the value of product flow variables
  Denote $z_{\mathcal{S}''}$ and $z_{\mathcal{S}^*}$ the objective values of solutions $\mathcal{S}''$ and $\mathcal{S}^*$, respectively
  if $z_{\mathcal{S}''} < z_{\mathcal{S}^*}$ **then**
    $\mathcal{S}^* \leftarrow \mathcal{S}''$
    $\mathcal{S} \leftarrow \mathcal{S}''$
  else
    if $AcceptanceCriterion(\mathcal{S}'')$ **then** $\mathcal{S} \leftarrow \mathcal{S}''$
  end if
  if $iter \mod 100 = 0$ **then** update the score of network structures
end for
$\mathcal{S}^* = PostOptimization(\mathcal{S}^*)$
return $\mathcal{S}^*$

The transportation modes and product flows are determined only when the values of all facility location variables have been set (line 12). The problem of determining the optimal product flows is a linear program, which could be optimally solved in polynomial time. Nevertheless, this problem has to be solved at each iteration. Since this can represent a non negligible computation effort, especially for large-sized instances, we resort instead to a fast greedy heuristic consisting of assigning product flows to the nearest facility, via the cheapest transportation mode (see Section 4.5). Note that this heuristic, as well as many operators, often use the general notion of the *nearest facility*. Unless otherwise specified, the Euclidean distance is used to calculate the nearest facility (but real distances could be used for a specific application).

When the new solution improves the current solution, it is automatically accepted and saved (lines 14 and 16). Otherwise, an *acceptance criterion* similar to that of simulated annealing, is used to determine whether the new solution should replace the current solution (line 18). A description of the acceptance criterion is given by Pisinger and Ropke [30]. Following Pisinger and Ropke [30], the initial temperature is set such that a solution that is 10% worse than the initial solution is accepted with a probability of 0.5. The temperature is decreased by 0.05% in every iteration.

In line 20, the score of the current network structure is updated based on the value of the objective function. This step is performed every 100 iterations

Finally, after the last iteration, the best solution $\mathcal{S}^*$ provided by the LNS method is slightly improved with a post-optimization step (line 22), which consists of optimally determining the values of the continuous flow variables x_{ij}^{mp}, given the facility location choices corresponding to $\mathcal{S}^*$. Since this post-optimization has to be performed only once, resorting to an LP solver is not too time-consuming. For this purpose, the primal simplex algorithm is used.

4.2. Initializing the score of all network structures and the initial solution

The score of each network structure is initialized as follows. For each network structure, a simple greedy heuristic is first applied. This heuristic selects facilities with a *least fixed cost* rule, and assigns customer demands to DCs as long as they can be satisfied by the installed capacity. Then, this solution is improved by running 100 iterations of a simplified version of Algorithm 1 with only two removal operators (*capacity utilization* and *unit cost*, see Section 5.1) and one repair operator (cost-based repair, see Section 5.2).

Let us denote by N the set of all possible network structures, z_n the best value of the objective function obtained with network structure $n \in N$, and $z_n^* = \min_{n \in N} z_n$ the overall best objective value found. The initial score of the network structure $n \in N$ is set as:

$$score(n) = \frac{z_n - z_n^*}{\sum_{n \in N} (z_n - z_n^*)}, \quad n = 1, \ldots, |N|. \tag{16}$$

The best solution, with objective value z_n^*, encountered during the initialization step is used as the initial solution of the LNS framework. Note that this initialization step can represent up to 10% of the total computing time, but helps to evaluate the solution space considerably.

4.3. Choosing a target network structure

The exploration of the solution space by Algorithm 1 must achieve a trade-off between having a good coverage of all network structures and dedicating enough computational effort to the most promising ones. The target network structure is chosen by the biased roulette wheel selection principle depicted in Algorithm 2. This procedure considers a set of scores (here the scores of all network structures) ranked in non-increasing order, and selects one score, giving a higher probability to the first ones. The parameter $\alpha \geq 1$ controls the degree of randomness: a low value of α corresponds to higher randomness.

Algorithm 2 Biased roulette wheel selection principle

Require: A list $\mathcal{L}$ of scores, sorted in non-increasing order.
Require: $\alpha \geq 1$: a randomness parameter
1: Generate a random number ρ according to a continuous uniform distribution in $[0, 1)$.
2: Choose the score at position $\lceil \rho^{\alpha} |\mathcal{L}| \rceil$ in $\mathcal{L}$.

The scores are recalculated every 100 iterations as in formula (16). Only the last 100 iterations are taken into account when recomputing them. If some network structure has not been considered during the last 100 iterations, its score is not modified. This approach raises several issues. First, in many adaptive LNS implementations, it is more common to use an exponential smoothing formula to decrease the importance of earlier results gradually (see formula (17) in [30]). We implemented both the full recalculation of all scores and exponential smoothing. It turned out that full recalculation yields slightly better results. Second, in the adaptive LNS, the score reflects not only the objective function value of a solution, but also the ability to yield overall best solutions, to improve the current solution and even to find previously unvisited solutions. In our approach, the score is based on the objective function value only. One advantage is that there is no need to tune additional parameters. A counterpart is that our approach does not reward diversification. Hopefully, this is offset by three LNS operators (random removal, random repair and history swap described in Section 5) which aim to diversify the search in the solution space.

4.4. Determining the number of facilities to de-select/select

The main parameters of the LNS operators are the number of facilities to de-select / select. The number of selected facilities in the active layer at the current iteration is called f. The number of facilities in the same layer of the target network structure is called f'. Going from f to f' facilities requires de-selecting a number f^- of facilities and selecting a number f^+ of facilities. Parameters f^- and f^+ must be chosen such that $f' = f - f^- + f^+$.

Algorithm 3 Determination of f^- and f^+

Require: Number f of selected facilities in the active layer at the current iteration.
Number f' of selected facilities in the active layer in the target network structure.
1: **if** $f = f'$ **then**
2: $f^- = f^+ = \lceil 0.2 \times f \rceil$.
3: **end if**
4: **if** $f > f'$ **then**
5: Select randomly between approach 1 and approach 2:
6: Approach 1: set $f^- = f - f'$ and $f^+ = 0$
7: Approach 2: set $f^- = \lceil 0.2 \times f \rceil + f - f'$ and $f^+ = \lceil 0.2 \times f \rceil$.
8: **end if**
9: **if** $f < f'$ **then**
10: Select randomly between approach 1 and approach 2:
11: Approach 1: set $f^- = 0$ and $f^+ = f' - f$
12: Approach 2: set $f^- = \lceil 0.2 \times f \rceil$ and $f^+ = \lceil 0.2 \times f \rceil + f' - f$
13: **return** values of f^- and f^+
14: **end if**

Algorithm 3 details the determination of f^- and f^+. We randomly choose between two approaches.

The first approach is to choose f^- and f^+ with minimal adjustments. If $f > f'$ then $f - f'$ facilities are de-selected and the value of f^+ is set at 0. Conversely, if $f < f'$, all facilities are kept and the value of f^+ is set at $f' - f$. The second approach assumes that at least 20% of the f current facilities should be modified. If $f = f'$, both the value of f^- and f^+ are set at $\lceil 0.2 \times f \rceil$. If $f \neq f'$, the value of one parameter is set at $\lceil 0.2 \times f \rceil$ while the value of the other is set at $\lceil 0.2 \times f \rceil + |f - f'|$.

4.5. Determining transportation modes and product flows

Algorithm 4 is the greedy heuristic called at line 12 of Algorithm 1. It determines the transportation modes and the product flows between nodes. First, the transportation modes and the product flows between all DCs and customers are determined. Then, a similar approach is used to determine the transportation modes and product flows between plants and DCs and between suppliers and plants.

Algorithm 4 Selection of transportation modes and calculation of product flows between DCs and customers

Require: d_l^p : demand of customer $l \in L$ for product $p \in P$,
cap_k : capacity of selected DCs $k \in K^0 (y_k = 1)$,
v_{kl}^{mp} : variable transportation cost for product p on arc (k, l) with mode $m \in M$.
1: Initialization of decision variables associated with the selection of transportation modes: $t_{kl}^m = 0, \forall k \in K, l \in L, m \in M$.
2: Build a list $ListD$ of demands in non-increasing order of the values d_l^p.
3: **for** all demands $d_l^p \in ListD$ **do**
4: **while** $d_l^p > 0$ **do**
5: select the DC k^* and the available transportation mode m^* minimizing the cost of carrying $\min(cap_{k^*}, d_l^p)$ units along arc (k^*, l)
6: **if** $min(cap_{k^*}, d_l^p) \geq V_{ij}^{m^*}$ **then**
7: set $t_{k^*l}^{m^*} = 1$
8: set $x_{k^*l}^{m^*p} = \min(cap_{k^*}, d_l^p)$
9: **else**
10: $x_{k^*l}^{m^*p} = 0$
11: **end if**
12: update remaining capacity at DC k^*: $cap_{k^*} \leftarrow cap_{k^*} - x_{k^*l}^{m^*p}$
13: update customer demand: $d_l^p \leftarrow d_l^p - x_{k^*l}^{m^*p}$
14: update $ListD$.
15: **end while**
16: **end for**
17: **return** values of x_{kl}^{mp} and t_{ij}^m

Algorithm 4 is based on a priority order defined by the largest demands. All demands d_l^p are ranked in non-increasing order (line 2) and we keep assigning products in non-increasing order of this priority order (line 3) with a greedy criterion based on the transportation cost (line 5). On line 5, the quantity $\min(cap_{k^*}, d_l^p)$ represents the value of a maximal flow on arc (k^*, l). The total transportation cost is calculated by the formula $\min(cap_{k^*}, d_l^p) \times v_{k^*l}^{m^*p} + g_{k^*l}^m$, where $g_{k^*l}^m$ and $v_{k^*l}^{m^*p}$ represent the fixed and variable costs associated with transportation mode m on the arc (k^*, l), respectively. If there is no fixed cost (see Table 2) then $g_{k^*l}^m = 0$. On line 6, the minimal flow imposed by constraints (9) is checked. Then, transportation mode m^* is selected (line 7) and the value of the flow on arc (i, j) is set to the capacity cap_{k^*} or to the demand d_l^p, whatever is smaller (line 8).

Note that, in order to be concise, we present a simplified version of Algorithm 4. In practice, the value of $v^{mp}_{k^*l}$ is set at $+\infty$ once there is no residual capacity at facility k^*. Moreover, the value of x^{mp}_{kl} is set at 0 for all unused DCs k once the demand of customer l is fulfilled.

5. Description of the LNS operators

This section describes a set of 6 removal operators (Section 5.1), 9 repair operators (Section 5.2) and 2 combined *removal+repair* operators (Section 5.3). All combinations between removal and repair operators are admitted. Most removal and repair operators use a biased roulette wheel selection principle similar to that of Algorithms 2 and 3 in [31]. All candidate facility locations are first evaluated based on problem-specific metrics (e.g, total or unit cost, capacity utilization). Then, they are ranked in non-increasing order according to this metric. Finally, the biased roulette wheel principle (Algorithm 2) is called f^- times (removal operator) or f^+ times (repair operator).

It is assumed that all facilities are located in a territory partitioned into *regions* (this is without loss of generality since the whole territory may constitute a single region). In addition, depending on the experimental data, some regions may comprise clusters of suppliers, plants, DCs or customers, i.e. a high density of these facilities.

Note that most operators are general purpose operators that work for all types of instances. Some operators (removal operators 5 and 6, and repair operators 5, 6 and 7) are specifically designed to handle the presence of clusters and can be selected only in this case. Combined operators (Section 5.3) apply only if $f^+ = f^-$. Hence, on line 9 of Algorithm 1, the operators are chosen randomly among the applicable ones.

5.1. Removal operators

The aim of the removal operators is to de-select f^- facilities from one layer of the current solution. This layer may concern either plants or DCs.

When a facility is selected, some decision variables related to this facility must be modified accordingly. For example, if a plant is de-selected, the values of all ingoing and outgoing flows are immediately set at 0. All other flows have to be recomputed after the repair operator has been called (line 12 of Algorithm 1).

1. **Random removal:**
   This operator randomly de-selects f^- facilities from the current solution. Its aim is to diversify the search in the solution space.
2. **Cost-based removal:**
   This operator de-selects facilities with the highest estimated cost. Without loss of generality, let us describe the operator applied to the plant layer only. The reasoning is exactly the same for the DC layer. Let us denote J^o and K^o the subsets of selected plants and DCs. A normalized fixed cost FC_j of facility $j \in J$ is defined as the ratio $FC_j = \dfrac{c_j}{\max\limits_{j' \in J^o} c_{j'}}$ between its fixed cost c_j and the maximal one among selected plants $j' \in J^o$.
   Following the idea of Olivares-Benitez et al. [26], a normalized variable cost of plant $j \in J$ is calculated as a ratio in which each term includes the processing costs as well as the ingoing and outgoing transportation costs related to j. It is expressed by the following formula:

$$VC_j = \frac{\sum\limits_{p\in P}(a^p_j + \sum\limits_{i\in I'}\sum\limits_{m\in M} v^{mp}_{ij} + \sum\limits_{k\in K'}\sum\limits_{m\in M} v^{mp}_{jk})}{\max\limits_{j'\in J'}(\sum\limits_{p\in P}(a^p_{j'} + \sum\limits_{i\in I'}\sum\limits_{m\in M} v^{mp}_{ij'} + \sum\limits_{k\in K'}\sum\limits_{m\in M} v^{mp}_{j'k}))},$$

where $I' \subset I$, $J' \subset J^o$ and $K' \subset K^o$ denote the set of nodes actually connected to j in the current solution. Since only one transportation mode is allowed between any pair of nodes, only one term in each sum of the type $\sum_{m \in M} v_{\cdot\cdot}^{mp}$ has non-zero value.

The sum $FC_j + VC_j$ of the two normalized costs gives a global cost indicator for plant j in the interval $]0, 2]$. Our numerical experiments show that this indicator balances the impact of fixed and variable costs. All plants are ranked by non-increasing value of $FC_j + VC_j$. Then, f^- plants are de-selected using the biased roulette wheel principle.

3. **Capacity utilization:**
This operator tends to de-select facilities with the least capacity utilization. The ratio RC of unused capacity at facility $j \in J^o$ or $j \in K^o$ indicates the proportion of remaining capacity at facility j. It is computed as follows:

$$RC_j = \frac{cap_j - \sum_{i \in \gamma(j)} \sum_{m \in M} \sum_{p \in P} x_{ji}^{mp}}{cap_j},$$

where $\gamma(j) \subset V$ is the set of direct successors of node j. Open facilities with the highest ratio will have a higher probability of being de-selected by the biased roulette wheel principle.

4. **Unit cost removal:**
This operator de-selects facilities with the least performance in terms of fixed costs and utilization of capacity. The performance of one facility $j \in J^o$ or $j \in K^o$ is measured by the ratio $\frac{RC_j}{FC_j}$.

5. **Cluster removal:**
This operator applies to supply chains in which some regions regroup clusters of plants or DCs. It is advisable to de-select all facilities of a cluster simultaneously, as the repair operator would otherwise be prone to re-selecting the same individual elements back into the solution (see e.g. [30], [32]).
The size of a cluster is determined by the total number of facilities in the cluster. The number of clusters to be de-selected is estimated as $n_c = \lceil f^- / c_{max} \rceil$, where f^- is the number of facilities that must be de-selected from the current active layer and c_{max} is the size of the largest cluster. Algorithm 5 describes the cluster removal operator. On line 4, ties are randomly broken. Note than if f^- is smaller than the total number of facilities in the n_c clusters, the operator does not de-select all facilities in the n_c clusters (line 5). On the contrary, it may rarely happen that the value of f^- exceeds the total number of facilities in the n_c clusters. In this case, the instruction at line 5 de-selects all facilities in the n_c clusters, and the cluster removal operator must be completed by randomly de-selecting other facilities in adjacent regions.

Algorithm 5 Cluster removal operator

Require: f^-: number of facilities to de-select, c_{max}: maximum number of facilities in a cluster.

1: Calculate $n_c = \lceil f^- / c_{max} \rceil$
2: Select n_c clusters randomly, and select one facility (called *seed*) randomly in each of these clusters.
3: In each of the n_c clusters, rank all facilities (including the seed) by increasing Euclidean distance to the seed.
4: Merge the n_c ranking lists into a single list, by taking first all facilities of rank 1 (seeds) then all facilities of rank 2, etc.
5: Remove f^- facilities of the merged list with the biased roulette wheel principle.

6. **Vertical cluster removal:**

This operator applies to supply chains with clusters of plants or DCs. The goal is to de-select pairs of plants and DCs related to each other. It is called vertical because it de-selects pairs of facilities that do not belong to the same layer.
At each iteration, we randomly choose between two approaches. The first approach is to de-select one plant randomly and then the nearest DC. A symmetric approach is to de-select one DC randomly and then the nearest plant. The process is repeated until at least 20% of the facilities have been de-selected from the current solution (the parameter f^- and the biased roulette wheel are not used by this operator).

5.2. Repair operators

The outcome of a removal operator is a partially destroyed solution. The goal of repair operators is to restore feasibility and reach the target network structure, by adding f^+ facilities to the partial solution. It may happen that the repaired solution is still not feasible. In this case, each unit of unsatisfied demand is penalized in the objective function by a large value set at $\max_{j \in J \cup K} c_j$. Hence, the current iteration of Algorithm 1 will not register an unfeasible solution as the new best solution (line 15) or an acceptable one (line 18).

1. **Random repair:**
   This operator randomly selects f^+ unselected facilities. It acts as a diversification mechanism.
2. **Cost-based repair:**
   This operator is directly inspired from Olivares-Benitez et al. [26]. It follows the same approach as the greedy repair heuristic (or best insertion) in vehicle routing problems. The principle is to select iteratively the facility whose insertion minimizes the cost of the future solution. If the active layer is that of plants, the candidate facilities $j \in J \backslash J^o$ are ranked in non-decreasing order of the values

$$cost_j = \frac{c_j}{cap_j} + \frac{\sum_{i \in I \cup K^o} (\mu_{ij} \sum_{p \in P} a_j^p) + \sum_{i \in I} (\mu_{ij} \sum_{p \in P} \max_{m \in M} v_{ij}^{mp}) + \sum_{k \in K^o} (\mu_{jk} \sum_{p \in P} \max_{m \in M} v_{jk}^{mp})}{\sum_{i \in I \cup K^o} \mu_{ij}},$$

   where $\mu_{ij} = \min(cap_i, cap_j)$ is the value of a maximal admissible product flow between a candidate facility j and a facility i in an adjacent layer. If the active layer is that of DCs, the candidate facilities $k \in K \backslash K^o$ are ranked in non-decreasing order of the values

$$cost_k = \frac{c_k}{cap_k} + \frac{\sum_{j \in J^o \cup L} (\mu_{jk} \sum_{p \in P} a_k^p) + \sum_{j \in J^o} (\mu_{jk} \sum_{p \in P} \max_{m \in M} v_{jk}^{mp}) + \sum_{l \in L} (\mu_{kl} \sum_{p \in P} \max_{m \in M} v_{kl}^{mp})}{\sum_{j \in J^o \cup L} \mu_{jk}},$$

   where $\mu_{jk} = \min(cap_j, cap_k)$ and $\mu_{kl} = \min(cap_k, \sum_{p \in P} d_l^p)$ represent the value of a maximal admissible product flows between a candidate facility k and a plant $j \in J^o$ or a customer $l \in L$. In the above formulae, the first terms represent a fixed cost ratio. The second terms take into account the processing and variable transportation costs of all types of products between each candidate facility and selected facilities in adjacent layers.
3. **Best substitution:**
   This operator applies only when $f^- = f^+$. It does not use the biased roulette wheel; its principle is to model and solve a pairing subproblem optimally.
   Without loss of generality, let us assume that the active layer is that of plants (exactly the same reasoning can be made for DCs). The set of plants that have just been de-selected by the removal

operator is called J^c. Thus, the set $J \backslash (J^o \cup J^c)$ represents all plants that are unselected after a removal operator has been used, i.e. before executing line 11 of Algorithm 1.
The goal of this operator is to replace the elements of J^c by an optimal subset of elements in $J \backslash (J^o \cup J^c)$. This is modeled by a pairing problem. The cost of replacing plant $i \in J^c$ by plant $j \in J \backslash (J^o \cup J^c)$ includes the fixed cost of j and the Euclidean distance $\delta_{i,j}$ between i and j. Since these two components are completely different both in nature and in value, the normalized values $\frac{c_j}{\max\limits_{j \in J} c_j}$ and $\frac{\delta_{ij}}{\max\limits_{i \in I, j \in J} \delta_{ij}}$ are considered. The criterion to be minimized is the sum of the normalized pairing costs, which take values in the range $]0, 2]$. The problem is solved optimally with an ILP solver within a very short computation time.

4. **Unit cost ratio:**
This operator favors facilities with a higher capacity and lower fixed costs. For each candidate facility $j \in J \backslash J^o$ or $j \in K \backslash K^o$, the ratio $\frac{c_j}{cap_j}$ between the capacity and the fixed cost is calculated. Since smaller values of this ratio are more desirable, the facilities are ranked in non-decreasing order of the ratio. Then, f^+ candidates are selected with the biased roulette wheel principle.
5. **Cluster repair:**
This operator is symmetric to the *cluster removal* operator.
6. **Vertical cluster repair:**
This operator is symmetric to the *vertical cluster* removal operator.
7. **Cluster Customers-DC:**
The goal of this operator is to select DCs near clusters of customers. It can be called only if the active layer is that of DCs and if the data set has clusters of customers. This is particularly the case for data sets corresponding to Pattern 4 (see Section 6.2.1 below).
First, one region with a cluster of customers is randomly chosen. Then, an unselected DC is sought in the same region and selected. If there is no DC in the same region, it is sought in adjacent regions. The procedure repeats until f^+ DCs have been selected.
8. **Top-down assignment:**
In a partially destroyed solution, part of the demand is no longer satisfied. The top-down flow assignment operator repairs a solution by greedily augmenting the material flow in the network, from the upstream nodes (suppliers) to the downstream nodes (customers). The key idea is that facilities are selected when the flow cannot be augmented anymore.
To illustrate this idea, let us assume that the active layer is that of plants (the reasoning is the same with DCs). While de-selecting a plant $j \in J^c$, the values of all the flow variables on arcs adjacent to j are also set at zero. For a given supplier $i \in I$, the value $\varphi_i = \sum\limits_{j \in J^c} \sum\limits_{p \in P} x_{ij}^{m^* p}$ corresponds to the quantity removed from the outgoing flow at i, regardless of the type of product. We call it *unassigned flow*.
Algorithm 6 describes the top-down flow assignment operator. For each supplier, the operator calculates the total amount of removed flow (line 2). These removed flows are ranked in non-increasing order of their values and placed in a priority list (line 3). Then, the operator tries to insert elements of the priority list into the existing network, giving priority to the shortest arc (i, j), $i \in I, j \in J$ (line 7). When the material flow can no longer be augmented, the operator has to select a new facility. The greedy choice is based on the first element of the priority list: the nearest plant to the current supplier is selected.
Note that all feasible solutions are evaluated, even if they are found before the target network structure is reached. The best solution found during the process is kept, even if it has fewer selected facilities than the target network structure.

Algorithm 6 Top-down flow assignment repair operator in the plant layer

Require: A partially destroyed solution. f^+: number of facilities to repair.

$nr = 0$ (number of facilities repaired)
for each supplier $i \in I$, calculate the unassigned flow $\varphi_i = \sum_{j \in J^c} \sum_{p \in P} x_{ij}^{m^* p}$.
Build list Φ of unassigned flows, with pair (i, φ_i) ranked in non-increasing order of φ_i.
while $nr \leq f^+$ **do**
  continue = TRUE
  while *continue* = TRUE **do**
    Select the first element (i, φ_i) in Φ and assign as much flow φ' as possible to the arc originating at i and terminating at the nearest non-saturated selected plant.
    if $\varphi' = 0$ **then**
      continue = FALSE
    else
      Remove the pair (i, φ_i) from Φ
      If $\varphi' < \varphi_i$, **then** update the list Φ with the pair $(i, \varphi_i - \varphi')$.
    end if
  end while
  Select the unselected plant nearest to supplier i.
  $nr \leftarrow nr + 1$
end while

9. **Bottom-up assignment:** This operator is symmetric to the *top-down assignment* described above, but starts at the downstream nodes (customers) and finishes at the upstream nodes (suppliers).

5.3. Combined removal and repair operators

1. **Swap operator:**
   A swap is a one-to-one exchange of status between a selected facility and an unselected one. In the swap operator, swaps are performed sequentially f^+ times. Of course, this operator can be called only if $f^- = f^+$.
   First, the operator randomly de-selects one facility (called *seed* facility). Then, all un-selected facilities are ranked by increasing Euclidean distances from the seed facility. One of them is selected according to the biased roulette wheel principle. This process is repeated f^+ (or equivalently f^-) times.
2. **History swap operator:**
   The goal of this operator is to diversify the search by strongly favoring facilities that were not frequently selected in previous iterations. We keep a historical record of selected and de-selected facilities at all iterations. The history swap operator builds a priority list in which the candidate facilities are ranked by non-increasing frequency of presence in the past solutions. First, f^- facilities are de-selected with the biased roulette wheel selection. Then, the priority list is reversed and $f^+ = f^-$ facilities are selected with the biased roulette wheel principle.

6. Computational experiments

This section details the computational experiments performed in order to validate the proposed LNS framework. For this purpose, a set of instances of different sizes and features was generated (see Section

6.1). This was necessary since there were no existing benchmark instances corresponding to our problem. Since these experiments rely on randomly generated instances, we explain the main principles of the data generation in Section 6.2. The discrete uniform distribution was used to generate values randomly in finite discrete sets. The continuous uniform distribution was used to generate values randomly in intervals (e.g. costs, coordinates of facilities).

In Section 6.3, we analyze the efficiency of each removal and repair operator in terms of their contribution to the value of the objective function. In Section 6.4, we present the numerical results obtained with the LNS heuristic and compare them with those of a state-of-the-art MILP solver. Even though strategic problems such as SCND problems are not addressed everyday by companies, being able to solve them efficiently and in a short amount of time is also essential. Indeed, strategic projects generally require decision support systems able to rapidly prototype and evaluate multiple scenarios, and to perform sensitivity or robustness analyses. This calls for the development of heuristic methods with a good trade-off between solution quality and computational effort [3, 33].

All algorithms were coded in C++. The solver used was IBM ILOG CPLEX 12.5. The calculations were performed on a computer with four Intel 3.0 GHz CPUs and 8 GB of RAM.

6.1. Test instances

We generated 60 instances grouped into 15 problem sets[1]. Each problem set contains 4 instances of the same size, determined by the number $|I|$ of suppliers, the number $|J|$ of candidate plants, the number $|K|$ of candidate DCs, the number $|L|$ of customers, and the upper limits J_{max} and K_{max} on the number of plants and DCs. Similarly to Cordeau et al. [22], the number of suppliers and candidate plants is set at $|I| = |J| = 0.1 \times |L|$.

The number of candidate DCs was set at $|K| = 0.2 \times |L|$. The values J_{max} and K_{max} were set at $\lceil 0.5 \times |J| \rceil$ and $0.5 \times |K|$, respectively. For each instance, the values J_{min} and K_{min} were set in such a way to ensure feasibility. To do so, all plants were ranked by non-increasing capacity. Then, the first plants were greedily selected until the cumulated capacity exceeded the total demand. The same procedure was applied to the DCs.

Table 1 displays the values of all the parameters for each problem set. Columns 2–5 detail the number of suppliers, plants, DCs and customers, respectively. In addition, the number $|P|$ of products was set at 5. Columns 6 and 7 represent the maximal number of plants and DCs in the network structures. Since the minimal values J_{min} and K_{min} are defined according to the total customer demand, they differ slightly from one instance to another in the same problem set. The value of J_{min} ranges from 1 to 7. The value of K_{min} ranges from 7 to 20.

Columns 8 to 10 report the average number of binary variables, continuous variables and constraints over the four instances of each problem set. Column 11 indicates the maximal number of network structures in each problem set.

The goal of generating small test instances is to compare LNS solutions with known optimal solutions obtained with an MILP solver. The aim of generating large instances is to study how the LNS behaves when the solver is unable to solve the instances to optimality within a pre-specified time limit.

6.2. Data generation

6.2.1. Generation of four supply chain patterns

The physical layout of nodes in the supply chain is likely to impact on the optimal network structure. For each of the 15 problem sets, we generated four types of supply chain patterns corresponding to various

[1] These instances are available from the authors upon request.

Table 1: Characteristics of test instances

Problem set	$\|I\|$	$\|J\|$	$\|K\|$	$\|L\|$	J_{max}	K_{max}	Variables *binary*	Variables *continuous*	Constraints	*Network structures*
s1	6	6	12	60	3	6	1 620	8 009	3 026	6
s2	7	7	14	70	4	7	2 203	10 909	4 039	9
s3	8	8	16	80	4	8	2 872	14 239	5 194	12
s4	9	9	18	90	5	9	3 651	18 117	6 515	12
s5	10	10	20	100	5	10	4 522	22 457	7 984	12
s6	12	12	24	120	6	12	6 531	32 475	11 357	15
s7	14	14	28	140	7	14	8 902	44 300	15 316	32
s8	16	16	32	160	8	16	11 562	57 570	19 788	35
s9	18	18	36	180	9	18	14 638	72 920	24 900	55
s10	20	20	40	200	10	20	17 886	89 127	30 408	78
s11	22	22	44	220	11	22	21 837	108 855	36 843	84
s12	24	24	48	240	12	24	26 064	129 957	43 778	90
s13	26	26	52	260	13	26	30 567	152 444	51 213	98
s14	28	28	56	280	14	28	35 336	176 258	59 138	135
s15	30	30	60	300	15	30	40 588	202 488	67 770	162

realistic situations.

All data were generated on a 200×200 grid. Each axis is divided into 5 segments of equal lengths, leading to 25 sub-grids of size 40×40, called *regions*, as introduced in Section 5. A *cluster* is a dense set of nodes located in the same region. In our data generation, clusters of plants or distribution centers were limited respectively to a maximum of four and eight facilities within a given region. Clusters of suppliers or customers do not have this limitation. A given region could comprise several clusters provided they respect the above limitation rule in total.

- **Pattern** 1**:** the coordinates of all nodes (suppliers, plants, DCs, customers) were generated randomly over the whole 200×200 grid.
- **Pattern** 2**:** we assumed that around 60% of all nodes are located in a few clusters. The remaining 40% of nodes are scattered randomly throughout the grid. First, 4 or 5 distinct regions were randomly chosen (the choice between 4 and 5 is also random). Then, 60% of the nodes were generated randomly in these regions, with a limit of 4 plants and 8 DCs. The remaining 40% of nodes were generated randomly.
- **Pattern** 3**:** this pattern models industrial regions with clusters of suppliers and clusters of plants. First, 4 or 5 distinct regions were randomly chosen (the choice between 4 and 5 is also random). Then, 60% of all suppliers and plants were randomly generated in the chosen regions. All remaining nodes were generated randomly.
- **Pattern** 4**:** this pattern models populated regions with a high density of customers. Since DCs are often located near customers, candidate DCs are also positioned in the same regions. First, 4 or 5 distinct regions were randomly chosen (the choice between 4 and 5 is also random). Then, 60% of all DCs and customer locations were randomly generated in the chosen regions. All remaining nodes were generated randomly.

6.2.2. Customer demands and capacities

Yeh [34] generated customer demands randomly over the interval $[100, 300]$. We use the same generation rule for every customer and every product $p \in P$. We defined the sum $D = \sum_{l \in L} \sum_{p \in P} d_l^p$ of all customer demands. Then, the capacity of all facilities was chosen randomly between 1.1 and ζ_{max} times the ratio $D/|K_{max}|$. Parameter ζ_{max} is set at 1.5 in the case of DCs, 2 in the case of plants, and 3 in the case of suppliers.

6.2.3. Generation of transportation modes

In order to test our model on a rich case corresponding to realistic situations, three transportation modes were assumed in the network. These modes may have different characteristics and some of the modes cannot be used over the whole network.

Table 2 shows the main characteristics of these modes. Column 2 states whether the corresponding transportation mode is subject to fixed costs (which are incurred regardless of the distance traveled). Column 3 displays the relative value of the variable cost for each transportation mode. Column 4 indicates if the transportation mode is subject to minimal load limitations. Column 5 details in which part(s) of the network each transportation mode is available.

Table 2: Characteristics of transportation modes

Transportation mode	Fixed cost	Variable cost	Minimum load	Availability of transportation mode
mode 1	✓	Intermediate	no	All arcs
mode 2	no	Highest	no	DCs to customers
mode 3	no	Lowest	✓	Suppliers to plants, plants to DCs (long-haul trips)

For example, although this is not a limitation, mode 1 could be an internal fleet of trucks which can be used on each arc in the network. The fixed cost of mode 1 is assumed to be 10000. This value was set such that the fixed cost of mode 1 represents around 5% of the total transportation costs. Mode 2 could represent an outsourced fleet of trucks for the delivery of goods to customers. Its variable cost is 20% higher than that of mode 1. Mode 3 could correspond to inland navigation or rail transportation. It is used for long-haul trips only (the two ends of the trip must be in distinct 40×40 regions). Its variable cost is 20% lower than that of mode 1. In return, not all locations are accessible by mode 3 and it is subject to a minimal load constraint value for each shipment. More precisely, the load of a trip between locations i and j with mode 3 is bounded below by $V_{ij}^3 = \lceil \beta \times \min(cap_i, cap_j) \rceil$, where β is a parameter generated in the interval $[0.4, 0.7]$.

Thus, mode 1 competes with mode 3 in the two upstream layers (suppliers to plants and plants to DCs) and with mode 2 in the last layer (from DCs to customers).

6.2.4. Fixed cost of facilities

Assuming economies of scale, the fixed cost of a facility is estimated to be roughly proportional to the square root of its capacity. It is defined as $c_j = \Phi\sqrt{cap_j}$, where the parameter Φ represents the price of the real estate market at each location. To generate Φ, the whole 200×200 grid was divided into small areas of size 5×5 and each area was labeled with a price category (very expensive, expensive, intermediate or cheap) depending on the number of candidate facilities and customers within a short distance. Very expensive areas correspond to 5% of the most "dense" areas. Then, expensive areas represent 50% of facilities located around the very expensive areas. Finally, the remaining areas have been randomly divided into two categories: intermediate

(40%) and cheap (5%). Cheap areas model, for example, local incentives for attracting economic activities in remote locations. The value of parameter Φ for cheap, intermediate, expensive and very expensive areas was generated randomly in the interval [5000,20000], [20000,35000], [35000,50000] and [50000,60000], respectively. This typology of costs was inspired by press releases concerning logistics real estate. Our goal was to perform numerical experiments on datasets inspired by realistic fixed costs rather than on purely randomized ones.

6.2.5. Variable costs

We detail how the processing and transportation costs were generated. Then, it is explained how these costs were scaled with fixed costs.

- **Processing costs**

  As defined earlier, the processing costs (parameters a_i^p) are the sum of all costs incurred at each node of the supply chain network: procurement of raw material and components from suppliers, manufacturing operations, storage and handling operations. The procurement cost of products provided by suppliers was randomly generated in the interval $[130, 150]$, the manufacturing cost at plants in the interval $[130, 150]$ and the warehousing and logistics costs at DCs in the interval $[100, 120]$. Then, following [22], these costs were multiplied by a random noising factor which is randomly chosen in the interval $[0.9, 1.2]$.

- **Transportation costs**

  The variable transportation cost between two nodes depends on the Euclidean distance between the nodes, the transportation modes and local factors. As indicated in Table 2, mode 1 is available for each arc $(i, j) \in A$ and each product $p \in P$. A variable transportation cost on each arc and each product $p \in P$ was generated as follows. The Euclidean distance between the endpoints of the arc was multiplied by a unitary cost generated randomly in the interval $[0.8, 1.2]$ and by a parameter τ representing the cost in each layer of the supply chain. Due to the added value of products along the supply chain and the transportation of smaller lot sizes in the downstream part of the supply chain, slightly increasing transportation costs are assumed from one layer to the next. Thus, τ was randomly chosen in the interval $[1, 1.3]$ for transportation from a supplier to a plant, in the interval $[1.2, 1.4]$ from plants to DCs and in the interval $[1.3, 1.5]$ for distribution to customers. As explained in Section 6.2.3, variable costs of transportation mode 2 are 20% higher that those of mode 1. Variable costs of mode 3 are 20% lower.

The variable transportation cost and variable processing cost influence the design of the network. In order to scale fixed and variable costs, the same approach used by Cordeau et al. [22] and Sadjady and Davoudpour [24] was followed. In each of the 15 problem sets, successive adjustments of variable costs were made so that they would represent between 40% and 50% of the total costs in two out of the four instances, and between 20% and 30% of the total costs in the two remaining instances. Table 3 shows how instances with so-called small (s) and large (L) variable costs are spread among the problem sets and patterns.

6.3. Evaluation of the LNS operators

In order to evaluate the relevance of the proposed LNS operators, we selected a subset of 15 *representative instances* out of the 60 instances. The choice of these instances obeys the following rules: (i) the 15 representative instances belong to 15 distinct problem tests, (ii) the instances taken from problem sets $s1, s2, s3$ and $s4$ have distinct patterns (same constraints for $s5, s6, s7$ and $s8$, etc.), (iii) 8 instances with large variable cost and 7 instances with small variable cost are selected.

Table 3: Small (s) and Large (L) variable costs in test instances

Problem set	Pattern 1	Pattern 2	Pattern 3	Pattern 4
s1	L	L	s	s
s2	L	s	L	s
s3	L	s	s	L
s4	s	L	L	s
s5	s	L	s	L
s6	s	s	L	L
s7	L	s	s	L
s8	s	L	s	L
s9	s	s	L	L
s10	L	s	L	s
s11	s	L	L	s
s12	L	L	s	s
s13	s	L	s	L
s14	L	s	L	s
s15	s	L	L	s

Table 4 displays the results of the operators' evaluation. Two protocols were followed. The goal of the first protocol was to evaluate the contribution of a given operator against all the others together. First, the LNS was run with all the operators and obtained an objective value z_1. Then, successively for each operator to be evaluated, the LNS was run with all the operators, except the operator considered. We obtained an objective value denoted z_2.

The individual contribution of each operator is measured by the ratio:

$$\frac{z_2 - z_1}{z_1} \times 100. \tag{17}$$

Since the objective function must be minimized, a positive ratio indicates that the operator contributes to the efficiency of the LNS.

The second protocol evaluates the contribution of a single (removal or repair) operator at a time, against the corresponding random operator only. More precisely, in order to evaluate the contribution of a given removal operator, the LNS was run with both the random removal operator and the considered operator and obtained an objective value z_1. Then, the LNS was run with the random removal operator only and obtained an objective value z_2. In both cases, all repair operators were also used. The individual contribution of the considered operator is again measured by the ratio (17). Similarly, to evaluate the contribution of a given repair operator, z_1 represents the objective value obtained with both the random repair operator and the considered repair operator. z_2 represents the objective value obtained with the random operator only. In both cases, all removal operators were also used.

Five runs on each of the 15 representative instances were performed. Since the results were quite stable, Table 4 only reports the average results.

The second column exhibits the contribution of operators evaluated through protocol 1. It shows only positive indicator values, with similar orders of magnitude. The third column relates to protocol 2. It shows that almost all operators, except *history swap*, yield positive indicator values. Note however the fairly large dispersion of the indicator values for the latter protocol. The slightly negative value for the *history swap* operator is understandable since this operator corresponds to a pure diversification factor, which is likely to be effective only when combined with other operators.

Table 4: Average contribution of each removal and repair operator

Operator	*Contribution with protocol 1*	*Contribution with protocol 2*
1. Random removal	0.43	–
2. Cost-based removal	0.22	0.81
3. Capacity utilization	0.25	0.98
4. Unit cost removal	0.31	0.95
5. Cluster removal	0.25	0.57
6. Vertical cluster removal	0.28	0.45
1. Random repair	0.17	–
2. Cost-based repair	0.22	0.85
3. Best substitution	0.14	0.34
4. Unit cost ratio	0.36	1.04
5. Cluster repair	0.20	0.24
6. Vertical cluster repair	0.25	0.14
7. Cluster Customers-DC	0.42	0.05
8. Top-down assignment	0.37	0.22
9. Bottom-up assignment	0.24	0.09
1. Swap	0.24	0.67
2. History swap	0.47	−0.06

Table 5: Operator utility

Operator	% of fruitful iterations	best	improving	accepted
		(% of the results in column 2)		
1. Random removal	3.5	1.1	52.8	46.1
2. Cost-based removal	3.9	1.1	56.5	42.4
3. Capacity utilization	4.0	1.3	57.7	40.9
4. Unit cost removal	4.1	2.0	58.0	40.0
5. Cluster removal	3.1	0.9	52.4	46.7
6. Vertical cluster removal	3.1	0.8	52.1	47.1
1. Random repair	2.9	1.0	53.4	45.6
2. Cost-based repair	4.6	1.2	59.1	39.7
3. Best substitution	11.5	1.3	55.0	43.7
4. Unit cost ratio	4.6	2.0	60.5	37.5
5. Cluster repair	1.5	0.3	39.3	60.4
6. Vertical cluster repair	2.9	0.5	53.1	46.4
7. Cluster customers-DC	1.4	0.5	39.7	59.8
8. Top-down assignment	6.3	2.5	59.6	37.9
9. Bottom-up assignment	4.5	0.4	51.5	48.1
1. Swap	7.9	2.3	48.7	49.0
2. History swap	0.6	1.1	93.1	5.7
Average	*4.1*	*1.2*	*55.4*	*43.4*

Table 5 analyzes the utility of each operator over five runs on the representative subset of 15 instances. For each operator, column 2 presents the average percentage of fruitful iterations, i.e. the iterations that result in a new best solution, an improvement of the current solution or a deterioration of the current solution, which is accepted by the acceptance criterion. Columns 3–5 show how this percentage is split among the three categories.

These results show that no operator outperforms another and each of them brings a specific contribution. Some operators seem to have a negligible effect, but ignoring them may worsen the quality of the solution. For example, *vertical cluster removal* and *random repair* do not look very useful for yielding new best known solutions, but they may help escape from local optima. The main key performance factor is probably the simultaneous use of several operators, which enables the search procedure to be intensified or diversified. Identifying which interactions between operators favor good results is still an open question.

6.4. Computational results

6.4.1. Comparison of LNS vs CPLEX results

The results of the proposed LNS heuristic were compared against the optimal solutions or lower bounds provided by CPLEX 12.5 with a maximal CPU time of 6 hours. The heuristic was run 10 times on each of the 60 instances. Preliminary experiments concluded that setting $IterMax = 25000$ in Algorithm 1 ensured a good trade-off between the CPU time and the quality of the solution.

The computational results are presented in Tables 6–8. Columns 2 and 3 present the CPU time (in seconds) for CPLEX and the LNS heuristic, respectively. The running times in column 3 correspond to the average value of the 10 runs for each instance.

An empty entry in column 2 means that the limit of 6 hours was reached without finding an optimal solution. Columns 4, 5, and 6 present the minimal, average, and maximal relative gap (in %) between the results found by the LNS and CPLEX, over 10 runs of the LNS heuristic.

The relative gap is calculated as the ratio

$$R\_Gap = \frac{z_{LNS} - z_{CPLEX}}{z_{CPLEX}} \times 100,$$

where z_{LNS} and z_{CPLEX} are the best feasible solutions obtained by the LNS heuristic and CPLEX, respectively.

Column 7 presents the CPLEX MIP gap (in %). This gap is calculated as the ratio

$$MIP\_Gap = \frac{z_{CPLEX} - LB_{CPLEX}}{LB_{CPLEX}} \times 100,$$

between the best upper bound and the best lower bound obtained by CPLEX.

As shown in Tables 6, 7 and 7, the relative gap *R_Gap* obtained with the proposed heuristic remains of the same order of magnitude (around 2% on average), independently of the size of the instances. In contrast, the *MIP_Gap* of CPLEX, which is 0% for small instances, increases with the size of instances up to similar values than the *R_Gap*. However, solutions are obtained with LNS in an amount of time considerably smaller than with CPLEX, and the difference increases with the size of the instances.

In 21 out of the 60 instances, there is no guarantee that the best feasible solution identified by CPLEX after 6 hours of computation is optimal. For these 21 instances, the average value of columns 4–6 (min., avg., and max. relative gap) is 1.09%, 1.73%, and 2.48%, respectively, while the average MIP gap is 1.10%. Note that this MIP gap is satisfying given that the data used in real-life applications often contain a margin of error larger than 1% [22]. If these 21 instances are ignored, the average value of columns 4–6 is 1.44%,

Table 6: Comparison between CPLEX and the LNS (sets s1 to s5)

Set	CPU (seconds)		*R_Gap* (%)			*MIP_Gap*
	CPLEX	LNS	Min	Avg.	Max	(%)
s1	16	41	1.08	1.40	1.81	0
	60	65	1.85	2.32	3.41	0
	13	65	1.68	1.84	2.51	0
	220	65	2.11	2.59	2.98	0
s2	97	66	0.99	0.99	0.99	0
	58	106	1.28	1.96	1.99	0
	58	106	1.51	1.67	1.74	0
	74	106	1.17	1.70	2.07	0
s3	400	94	1.55	2.01	2.47	0
	125	148	1.02	2.73	3.50	0
	831	148	1.80	2.46	2.73	0
	411	145	1.16	2.12	3.68	0
s4	89	120	1.75	1.89	2.25	0
	570	190	2.46	2.94	4.52	0
	293	190	1.24	1.71	3.14	0
	583	192	1.38	1.96	2.72	0
s5	223	146	1.75	2.55	3.37	0
	574	232	1.42	1.80	2.19	0
	536	233	0.79	1.16	1.83	0
	1 184	233	1.32	2.32	4.14	0
Average			1.47	2.01	2.70	

Table 7: Comparison between CPLEX and the LNS (sets s6 to s10)

Set	CPU (seconds)		*R_Gap* (%)			*MIP_Gap*
	CPLEX	LNS	Min	Avg.	Max	(%)
s6	136	182	0.91	1.95	3.24	0
	129	283	0.91	0.91	0.91	0
	2 321	281	1.22	2.14	3.58	0
	834	279	0.80	1.84	3.00	0
s7	1 214	229	1.69	2.55	4.42	0
	1 218	344	1.51	2.60	4.53	0
	1 124	345	0.99	1.47	2.18	0
	1 119	338	0.94	1.29	2.24	0
s8	19 260	291	1.62	2.36	2.83	0
		412	2.19	2.80	3.68	0.99
	6 471	421	1.06	2.42	3.65	0
	9 682	411	2.42	3.92	4.64	0
s9	4 014	369	1.35	2.05	2.70	0
	4 200	498	1.52	1.72	1.85	0
		501	1.24	1.36	1.74	0.04
	4 947	497	1.45	2.20	2.84	0
s10	17 684	544	1.88	2.86	4.08	0
		630	1.69	2.13	2.66	1.08
		642	2.44	2.87	3.50	0.19
	4 279	648	1.28	1.95	2.79	0
Average			1.46	2.17	3.05	

Table 8: Comparison between CPLEX and the LNS (sets s11 to s15)

Set	CPU (seconds)		*R_Gap* (%)			*MIP_Gap*
	CPLEX	LNS	Min	Avg.	Max	(%)
s11		669	1.92	2.36	2.91	0.20
		707	1.51	1.99	2.57	0.15
		731	1.81	2.83	3.55	0.77
	16 735	723	1.52	2.54	3.19	0
s12		756	1.03	1.72	2.10	0.88
		815	-0.18	0.08	0.54	1.89
		861	0.17	1.19	1.90	1.33
		833	0.84	1.46	3.05	1.49
s13	14 407	953	1.46	2.09	3.02	0
		1 001	-0.03	0.95	1.80	2.13
		1 035	0.97	1.26	1.51	2.78
		1 006	0.33	0.68	1.34	2.18
s14	10 671	1114	2.28	2.37	3.10	0
		1 242	2.03	2.71	3.60	0.41
		1 237	1.87	2.66	3.93	0.24
		1 239	0.97	1.22	1.27	0.15
s15		1 364	-0.80	0.25	1.38	2.31
		1 477	1.69	2.55	4.09	0.59
		1 510	1.33	2.19	3.07	1.16
		1 470	-0.10	1.06	1.95	2.08
Average			1.03	1.71	2.49	

2.09%, and 2.89%, respectively. The negative percentages in Table 8 mean that the heuristic identified better solutions than CPLEX.

6.4.2. Analysis of relative gaps

Figure 2 illustrates how the relative gap is reduced during the search process in the LNS algorithm for problem set s15, pattern 1. The initial solution has a 8.23% gap from the lower bound LB_{CPLEX} provided by CPLEX. The relative gap is 5.92%. After 4200 iterations, the relative gap becomes negative. After 20200 iterations, the relative gap becomes −0.8%.

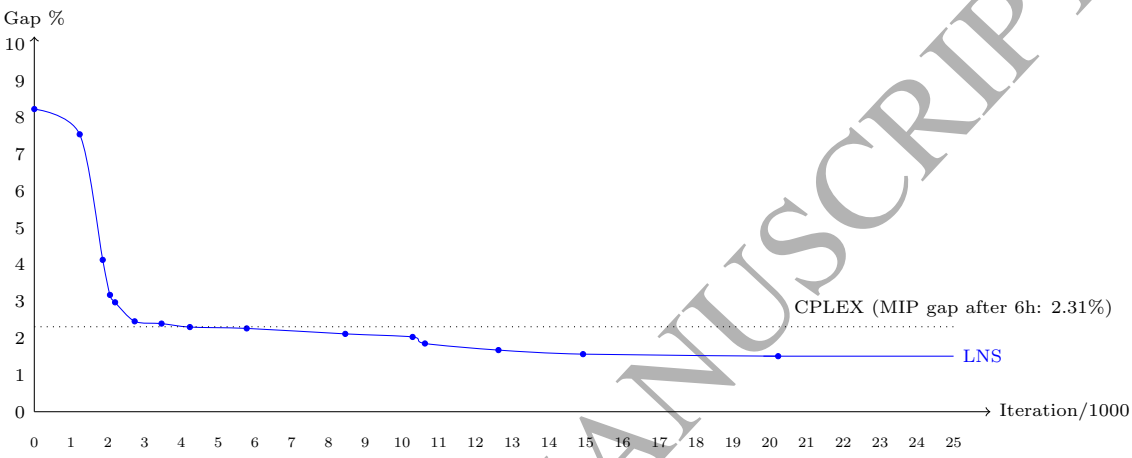

Figure 2: Example of LNS iterations (Test set s15, pattern 1)

The relative gap ranges from −0.8% to 4.64% over all test instances, with an average value of 2.01%. The maximum running time of the LNS is 1 510 seconds with an average of 526 seconds, which shows the ability of the heuristic to find good results within an acceptable time. The best, average, and the worst results (columns 4–6) are quite different. This suggests that reinforced intensification procedures would probably improve the overall results and reduce the relative difference between runs at the expense of a longer (but affordable) computation time.

Besides, the difference between the results of the four patterns is non-significant, both in terms of the optimality gap and the computation time. Moreover, there is no significant difference in the results of instances with small and large variable costs. This indicates that the proposed LNS framework is flexible enough to fit with various types of supply chain.

6.4.3. Impact of transportation modes

The goal of this section is to assess the impact of multi-modality. To do so, the LNS heuristic was run (i) with the full model described in Section 3, allowing all transportation modes as defined in Table 2, (ii) with a simplified model that allows only transportation mode 1. 10 runs of the LNS algorithm were performed on each of the 15 representative test instances described above.

Table 9 reports the average difference between the full model and the simplified model. Columns 2 and 3 represent the percentages of plants and DCs that differ in each scenario. Column 4 indicates the total percentage cost decrease when multiple modes of transportation are allowed. Columns 5–7 show how this cost decrease splits among three categories: the fixed cost of facilities, the processing cost and the transportation

Table 9: Influence of inter-modal transportation

Test instance	*% of plants modified*	*% of DCs modified*	*% of* cost decrease	*including: fixed cost*	*operating cost*	*transportation cost*
$i1$	0	0	4.76	0	1.10	3.66
$i2$	22	7	5.31	1.18	1.96	2.17
$i3$	0	12	4.74	2.16	1.22	1.36
$i4$	22	17	3.59	2.18	0.95	0.46
$i5$	30	30	6.47	3.92	1.47	1.08
$i6$	42	29	6.92	3.05	1.76	2.11
$i7$	7	25	4.81	2.40	1.18	1.23
$i8$	18	21	4.88	1.43	1.49	1.96
$i9$	27	19	5.17	2.25	1.16	1.76
$i10$	20	22	7.49	3.39	1.05	3.05
$i11$	36	30	6.55	2.34	1.37	2.84
$i12$	38	37	7.23	4.84	0.75	1.64
$i13$	42	30	6.92	3.31	1.45	2.16
$i14$	32	25	5.75	3.44	0.76	1.55
$i15$	30	27	7.84	3.84	1.62	2.38
Average	24.40	22.07	5.90	2.65	1.29	1.96

cost. These results show a cost reduction of 5.90% on average, which is due to the transport itself (1.96%), to the fixed cost of the selected facilities (2.65%) and to the processing costs (1.29%). The results in columns 2 and 3 and the decrease in fixed costs clearly show that using multiple transportation modes influences not only the variable transportation and processing costs but also the design of the network.

7. Conclusion

In this paper, we proposed and tested a new approach for solving a supply chain network design problem, based on the Large Neighborhood Search framework. For that purpose, a 4-layer multi-product supply chain network design model was considered. This model includes production plant and distribution center location as well as the choice of transportation modes. The LNS framework had indeed never been used before to address generic supply chain design models.

In the proposed LNS heuristic, removal and repair operators determine the locations of two layers of facilities: plants and distribution centers. At each iteration, a greedy heuristic is called upon to select the appropriate transportation modes and to determine the product flows through the network. Finally, a post-optimization step using linear programming is used for determining the optimal product flows.

The performance of the proposed method was tested on a variety of randomly generated instances with various sizes and layouts, which we specifically generated. We provide extensive comparisons with optimal solutions or bounds obtained with CPLEX. The numerical results show the stability of our LNS framework and its efficiency, in terms of both the quality of the solution and the computation time, especially for large problems. These results confirm the fact that standard solvers may be used for solving small- or medium-sized SCND instances. But obtaining efficient solutions to realistic, large-sized problems in reasonable computing time requires the development of specific solution techniques.

Our solution technique relies on the notion of a *network structure*, which helps the LNS algorithm focus on a small range of good values of the key decision variables. In real-life applications, a reasonable range can often be obtained from preliminary studies or may naturally arise from budget limitations or organizational

constraints. If this range becomes too wide, the problems become much harder to solve. One possible way to deal with larger network structure intervals would be to use an adaptive approach. We could start to explore a reasonable interval, and shift the bounds of the range dynamically.

This research shows that the principles of the LNS can be used successfully for supply chain network design problems. Thus, further research could aim to adapt our heuristic framework to more complex models, for example with multiple periods, complex bills of materials or multiple objectives. Since sustainable supply chain network design has become a major trend in recent years [28], we plan to consider a second objective related to the environmental impact of the supply chain. The first step will be to evaluate CO_2 emissions arising from transportation activities and facilities. This requires modeling the CO_2 emissions and embedding the LNS heuristic into a bi-objective framework to determine an approximation of the Pareto front.

Acknowledgements: This research was supported partially by the Région Pays de la Loire through the research project OLASI, which has been labeled by the competitiveness clusters EMC2 and Nov@log. This support is gratefully acknowledged. We thank the three anonymous reviewers for their valuable comments and suggestions which considerably helped improve the paper.

References

References

[1] S. L. Hakimi, Optimum locations of switching centers and the absolute centers and medians of a graph, Operations Research 12 (3) (1964) 450–459.

[2] A. Gupta, J. Könemann, Approximation algorithms for network design: A survey, Surveys in Operations Research and Management Science 16 (1) (2011) 3 – 20.

[3] S. A. Alumur, B. Y. Kara, M. T. Melo, Location Science, Springer International Publishing, 2015, Ch. 16 Location and Logistics, pp. 419–441.

[4] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, in: M. Maher, J.-F. Puget (Eds.), Principles and Practice of Constraint Programming – CP98, Vol. 1520 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 417–431.

[5] G. Schrimpf, J. Schneider, H. Stamm-Wilbrandt, G. Dueck, Record breaking optimization results using the ruin and recreate principle, Journal of Computational Physics 159 (2000) 139–171.

[6] D. Pisinger, S. Ropke, Large neighborhood search, in: M. Gendreau, J.-Y. Potvin (Eds.), Handbook of Metaheuristics, Vol. 146 of International Series in Operations Research & Management Science, Springer, 2010, pp. 399–419.

[7] P. J. Copado-Méndez, C. Blum, G. Guillén-Gosálbez, L. Jiménez, Large neighbourhood search applied to the efficient solution of spatially explicit strategic supply chain management problems, Computers & Chemical Engineering 49 (11) (2013) 114–126.

[8] M. T. Melo, S. Nickel, F. Saldanha-da Gama, Facility location and supply chain management - A review, European Journal of Operational Research 196 (2) (2009) 401–412.

[9] M. Yaghini, M. Momeni, M. Sarmadi, A simplex-based simulated annealing algorithm for node-arc capacitated multicommodity network design, Applied Soft Computing 12 (9) (2012) 2997 – 3003.

[10] P. Subramanian, N. Ramkumar, T. Narendran, K. Ganesh, PRISM: PRIority based SiMulated annealing for a closed loop supply chain network design problem, Applied Soft Computing 13 (2) (2013) 1121 – 1135.

[11] L. Der-Horng, D. Meng, A heuristic approach to logistics network design for end-of-lease computer products recovery, Transportation Research Part E 44 (3) (2008) 455 – 474.

[12] M. Melo, S. Nickel, F. Saldanha-da Gama, A tabu search heuristic for redesigning a multi-echelon supply chain network over a planning horizon, International Journal of Production Economics 136 (1) (2012) 218 – 230.

[13] M. Eskandarpour, E. Nikbakhsh, S. Zegordi, Variable neighborhood search for the bi-objective post-sales network design problem: A fitness landscape analysis approach, Computers & Operations Research 52, Part B (2014) 300 – 314.

[14] M. Eskandarpour, S. Zegordi, E. Nikbakhsh, A parallel variable neighborhood search for the multi-objective sustainable post-sales network design problem, International Journal of Production Economics 145 (1) (2013) 117 – 131.

[15] F. Altiparmak, M. Gen, L. Lin, I. Karaoglan, A steady-state genetic algorithm for multi-product supply chain network design, Computers & Industrial Engineering 56 (2) (2009) 521 – 537.

[16] H. Wang, H. Hsu, A closed-loop logistic model with a spanning-tree based genetic algorithm, Computers & Operations Research 37 (2) (2010) 376–389.

[17] M. Pishvaee, R. Zanjirani Farahani, W. Dullaert, A memetic algorithm for bi-objective integrated forward/reverse logistics network design, Computers & Operations Research 37 (6) (2010) 1100–1112.

[18] R. Jamshidi, S. Fatemi Ghomi, B. Karimi, Multi-objective green supply chain optimization with a new hybrid memetic algorithm using the Taguchi method, Scientia Iranica 19 (6) (2012) 1876 – 1886.

[19] F. Du, G. W. Evans, A bi-objective reverse logistics network analysis for post-sale service, Computers & Operations Research 35 (8) (2008) 2617 – 2634.

[20] D. Carlsson, M. Rönnqvist, Supply chain management in forestry—case studies at Södra Cell AB, European Journal of Operational Research 163 (3) (2005) 589 – 616.

[21] E. Eskigun, R. Uzsoy, P. V. Preckel, G. Beaujon, S. Krishnan, J. D. Tew, Outbound supply chain network design with mode selection, lead times and capacitated vehicle distribution centers, European Journal of Operational Research 165 (1) (2005) 182 – 206.

[22] J.-F. Cordeau, F. Pasin, M. Solomon, An integrated model for logistics network design, Annals of Operations Research 144 (2006) 59–82.

[23] M. Wu, Y. Hsu, L. Huang, An integrated approach to the design and operation for spare parts logistic systems, Expert Systems with Applications 38 (4) (2011) 2990–2997.

[24] H. Sadjady, H. Davoudpour, Two-echelon, multi-commodity supply chain network design with mode selection, lead-times and inventory costs, Computers & Operations Research 39 (7) (2012) 1345 – 1354.

[25] R. Rahmaniani, A. Ghaderi, A combined facility location and network design problem with multi-type of capacitated links, Applied Mathematical Modelling 37 (9) (2013) 6400–6414.

[26] E. Olivares-Benitez, R. Ríos-Mercado, J. González-Velarde, A metaheuristic algorithm to solve the selection of transportation channels in supply chain design, International Journal of Production Economics 145 (1) (2013) 161–172.

[27] Y. Cardona-Valdés, A. Álvarez, J. Pacheco, Metaheuristic procedure for a bi-objective supply chain design problem with uncertainty, Transportation Research Part B: Methodological 60 (2014) 66 – 84.

[28] M. Eskandarpour, P. Dejax, J. Miemczyk, O. Péton, Sustainable supply chain network design: An optimization-oriented review, Omega 54 (2015) 11–32.

[29] C. Bode, S. M. Wagner, Structural drivers of upstream supply chain complexity and the frequency of supply chain disruptions, Journal of Operations Management 36 (2015) 215–228.

[30] D. Pisinger, S. Ropke, A general heuristic for vehicle routing problems, Computers & Operations Research 34 (8) (2007) 2403 – 2435.

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

[32] G. M. Ribeiro, G. Laporte, An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem, Computers & Operations Research 39 (3) (2012) 728 – 735.

[33] M. Melo, S. Nickel, S. da Gama F., Network design decisions in supply chain planning, Tech. rep., Technical Report 140, ITWM - Fraunhofer Institute for Industrial Mathematics, Kaiserslautern, Germany (2008).

[34] W. Yeh, An efficient memetic algorithm for the multi-stage supply chain network problem, International Journal of Advanced Manufacturing Technology 29 (7-8) (2006) 803–813.