A Large Neighborhood Search heuristic for Supply Chain Network Design

Majid Eskandarpour, Pierre Dejax, Olivier Péton

A Large Neighborhood Search heuristic for Supply Chain Network Design

Rapport Interne 14/3/AUTO
September 2014

Majid Eskandarpour Pierre Dejax Olivier Péton

Ecole des Mines de Nantes
IRCCyN UMR CNRS 6597
Institut de Recherche en Communications et Cybernétique de Nantes
Nantes, France

Abstract

Many exact or 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 proposed 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 involving multimodal transport. Location decisions for intermediate facilities (e.g. plants and distribution centers) are made using the LNS while transportation modes and product flow decisions are determined by a greedy heuristic. As a post-optimization step, we also use linear programming to determine the optimal product flows once the logistics network is fixed. Extensive experiments based on 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

1 Introduction

The goal of this paper is to describe and evaluate a Large Neighborhood Search approach to solve a facility location problem. The field of facility location has been very active since the description of the p-median problem by Hakimi [13] 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.

In addition to facility location decisions, multiple variants of SCND models include sizing decisions, allocation of products to facilities, supplier selection, choice of transportation modes, etc. The recent mathematical models generally include features such as multiple layers and types of facilities, multiple products and multiple periods. More advanced models integrate the bill of material for multi-component problems, multiple transportation modes, uncertainty, risk management, disruption, reverse logistics or sustainable development factors.

Most SCND models aim to determine the best network configuration regarding a single objective function representing an economic goal. The great majority of such problems are classified as NP-hard [12]. A large variety of exact or 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. Using heuristic or metaheuristic approaches is thus inevitable to tackle such hard problems.

Many heuristic methods have been used to solve SCND problems. Surprisingly enough, the Large Neighborhood Search (LNS) heuristic has almost never been used. The LNS was introduced by Shaw in a constraint programming framework [29]. The underlying principle is to destroy and repair the current solution iteratively in order to improve it progressively. Destroying the current solution consists of removing a subset of decision variables from the solution. Repairing the solution consists of restoring feasibility. This principle is similar to that of the *ruin and recreate* introduced by Schrimpf et al. [28]. The efficiency of the method relies on the choice and appropriate use of application-oriented *removal* and *repair* operators. Pisinger and Ropke present an extensive survey of the method [22]. This technique has proven its efficiency in several combinatorial optimization applications such as vehicle routing and scheduling problems.

To our knowledge, the use of the LNS for solving SCND problems is still very scarce. Copado-Méndez et al. model two case studies in chemical engineering and solve them with an LNS approach [4]. They randomly choose a set of variables to remove and invoke a solver to repair the solution by changing the value of the variables removed. 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. The LNS can be presented as a general framework in which a collection of potential operators can be used or not depending on the attributes of the model to be solved.

In this paper, we propose an LNS heuristic for solving an SCND model with four layers, multiple products and transportation modes. The model includes location decisions in the two intermediate levels. The main goal is to assess the efficiency of the LNS approach and to draw lessons for further research.

In particular, several challenges are emerging. Firstly, SCND models include both binary and continuous variables, which must be treated separately by the LNS algorithm. We adopt a hierarchical approach consisting, in each iteration, of identifying the locations with the LNS operators and then determining continuous variables by solving a linear programming sub-problem. Thus, the characteristics of the optimal solution are not known *a priori*. In vehicle routing or scheduling problems, the number of customers to visit or the number of tasks to schedule is known beforehand. The number of open facilities in a p-median problem is known by definition. This is not the case anymore in most SCND problems. The removal and repair operators must be able to lead to several network configurations. Lastly, our model includes two types of binary variables: location variables and the choice of transportation modes. In our heuristic, location decisions are fixed using the LNS while transportation modes are determined *a posteriori* by a greedy heuristic.

The remainder of the paper is organized as follows. In section 2, we present the position of our work in the SCND literature. Section 3 describes the problem and the model formulation. The proposed solution method is presented in section 4. Section 5 provides the computational results on generated test instances. The conclusion and suggested future research appear in section 6.

2 Position in the 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 reviews [6, 15, 16, 24, 27, 17] published in the last ten years.

Many metaheuristic and evolutionary methods have been recently developed. We can cite simulated annealing [e.g. 35, 30], tabu search [e.g. 7, 18], VNS [e.g. 9, 10], genetic algorithm [e.g. 1, 32], memetic algorithm [e.g. 20, 14], and scatter search [e.g. 8]. However, there is still a need to develop efficient

heuristic and metaheuristic methods to cover advanced models or large instances that cannot be solved by exact methods or MILP solvers.

One of the advanced features of the model proposed in this paper is the possibility of choosing among multiple transportation modes between vertices. As pointed out in [17], SCND models including the choice of transportation modes have not received much attention.

Carlsson and Rönnqvist [3] 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. [11] 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, etc.) of the vehicles. The model is solved with a Lagrangean heuristic. The model in Wilhelm et al. [33] concerns the strategic design of an assembly system in the international business environment created by NAFTA. It therefore includes facility location and many associated decisions, such as the choice of technologies, capacities, suppliers and transportation modes. Cordeau et al. [5] develop a comprehensive multi-stage network design model: at the strategic level, they investigate facility location and capacity expansion decisions. At the tactical level, they also integrate the selection of transportation modes regarding fixed and variable costs and the capacity usage. 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 transportation mode determination are the following. Tiwari et al. [31] develop a 5-layer supply chain concerning strategic and tactical decisions. The solution space is explored through a hybrid Taguchi-Immune System metaheuristic approach, in which a *chromosome* is composed of location and transportation mode decisions. Wu et al. [34] study a spare parts logistics network encompassing three types of decision: facility location, item vendor selection and transportation mode. They compare two approaches consisting of determining all decisions simultaneously or determining the location decisions first. Sadjady and Davoudpour [26] propose a single-period two-echelon multi-commodity model regarding strategic and tactical decisions as well as the selection of transportation modes. The problem is solved with a Lagrangean relaxation heuristic. In the paper by Rahmaniani and Ghaderi [23], each arc between two facilities is associated with fixed and variable cost as well as a capacity. Olivares-Benitez et al. [19] propose a three-stage supply chain including plants, distribution centers and customers to minimize cost and temporal considerations. In order to reduce the shipping time from plants to customers, several transportation modes between two nodes are considered in terms of their own cost and travel time. Cardona-Valdés et al. [2] propose a two-echelon distribution network regarding economical and service level objectives. They incorporate demand uncertainty and transportation mode allocation decisions.

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 sequentially fix each type of decision variable. We adopt the latter approach, considering the location decisions in the first step and the transportation modes in the second step.

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 (DCs) and customers (retail stores or final customers), as depicted in Figure 1.

The location of suppliers and customers are known, whereas those of plants and DCs have to be determined from a list of candidate locations. At the first layer, suppliers provide the raw materials or components to the plants. These products are then converted to finished goods through value-added

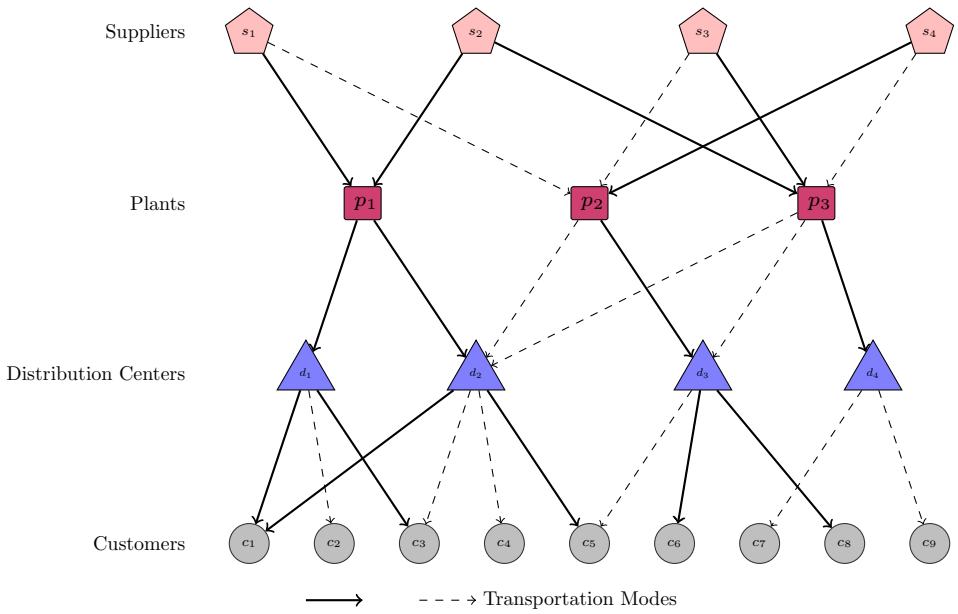

Figure 1: The supply chain considered

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. We do not consider single sourcing constraints, *i.e.* a facility can be delivered by several facilities from the preceding layer.

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 can be expressed as a single value limiting the output flow for each product at this facility. Without loss of generality, the mathematical model assumes that each product can be processed by every facility. If a facility cannot process one given product, this can be modeled by setting the corresponding capacity at 0.

The facility location decisions at plants and DCs are guided by two types of costs. Fixed costs for opening 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.

Several transportation modes are available, such as road, rail, inland navigation or air transport, to ship products between the nodes of the network. However we assume that a restricted list of suitable transportation modes has been identified *a priori* for each pair of nodes, with respect to criteria such as availability and safety, shipping costs, CO_2 emissions, shipment capacities, speed and frequency. At the strategic 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. All transportation modes have a known maximal load. Some modes also require a minimal quantity of goods to be shipped. We assume that only one transportation mode is selected between any pair of nodes and that all products are compatible enough to be loaded onto the same transportation mode.

With respect to the above-mentioned description, the main decisions are to select a subset of plants and DCs, to choose a transportation mode between suppliers, selected facilities and customers, and to determine the product flows in the logistics network. The objective function of this problem is to minimize the overall cost over one single period, including the fixed cost of opening facilities, processing costs and

transportation costs.

3.2 Data, sets, parameters and variables

We consider a set I of suppliers, a set J of plants, a set K of DCs, a set L of customers, a set P of products and a set M of potential transportation modes. The subsets of open plants and DCs are denoted as J^o and K^o. The SCND problem is defined on a directed graph $\psi = (V, A)$ with $V = I \cup J \cup K \cup L$ and the set A of arcs defines all possible links between facilities. This potentially includes all links between two successive layers represented in Figure 1. In practice, there may not be an arc between two vertices when the capacity is set at 0, or a transportation mode is unavailable or has an arbitrarily large value.

We introduce the following notations:

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

Binary variable y_j is set at 1 if a facility $j \in J \cup K$ is open and 0 otherwise. In order to select the transportation modes throughout the network, we consider binary variables t_{ij}^m set at 1 if the transportation mode $m \in M$ is selected 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 $m \in M$.

3.3 Mathematical formulation

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

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

This objective function contains four terms, representing the sum of opening fixed costs, processing costs, and fixed and variable parts of transportation costs respectively.

Constraints (2) are the flow conservation constraints throughout the network.

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

Constraints (3) ensure the satisfaction of customer demands.

$$\sum_{k \in K} \sum_{m \in M} x_{kl}^{mp} \geq d_l^p \qquad \forall l \in L, p \in P \qquad (3)$$

Constraints (4)–(6) force the model to respect capacity constraint at suppliers, plants and DCs, respectively. In addition, (5) and (6) state that the products can be shipped only to open facilities.

$$\sum_{j\in J}\sum_{m\in M}\sum_{p\in P} x_{ij}^{mp} \leq cap_i \qquad \forall i \in I \tag{4}$$

$$\sum_{k\in K}\sum_{m\in M}\sum_{p\in P} x_{jk}^{mp} \leq cap_j\ y_j \qquad \forall j \in J \tag{5}$$

$$\sum_{l\in L}\sum_{m\in M}\sum_{p\in P} x_{kl}^{mp} \leq cap_k\ y_k \qquad \forall k \in K. \tag{6}$$

Constraints (7) ensure that one transportation mode at most is selected between two nodes. Constraints (8) – (9) guarantee that the volume limitation of each given mode is respected.

$$\sum_{m\in M} t_{ij}^{m} \leq 1 \qquad \forall (i,j) \in A \tag{7}$$

$$\sum_{p\in P} x_{ij}^{mp} \leq \bar{V}_{ij}^{m}\ t_{ij}^{m} \qquad \forall (i,j) \in A, m \in M \tag{8}$$

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

We also consider restrictions on the number of open facilities. Constraints (10)–(11) bound the number of open plants and DCs, respectively. These constraints can be discarded by setting minimal values at 0 and maximal values at $+\infty$.

$$J_{min} \leq \sum_{j\in J} y_j \leq J_{max} \tag{10}$$

$$K_{min} \leq \sum_{j\in K} y_j \leq K_{max} \tag{11}$$

Constraints (12) – (14) state non-negativity and binary restrictions on decision variables.

$$y_j \in \{0,1\} \qquad \forall j \in J \cup K \tag{12}$$

$$t_{ij}^{m} = \{0,1\} \qquad \forall (i,j) \in A, m \in M \tag{13}$$

$$x_{ij}^{mp} \geq 0 \qquad \forall (i,j) \in A, p \in P, m \in M \tag{14}$$

4 A Large Neighborhood Search heuristic

We propose an LNS heuristic able to deal with the three main types of decision variables: facility location, selection of transportation mode and calculation of optimal product flows. The location and transportation modes are modeled by binary variables while the product flows are modeled by continuous variables.

A key issue is to determine the number and location of plants and DCs. We call the *network configuration* the number of plants and DCs open in the current solution, represented by the pair $\left(\sum_{j\in J} y_j, \sum_{k\in K} y_k\right)$.

This network configuration has great influence on the whole solution. One of the main issues is that there are $(J_{max} - J_{min}) \times (K_{max} - K_{min})$ possible network configurations. Good heuristic methods

should explore all promising network configurations. The main challenges of the LNS algorithm are thus to handle both binary and continuous variables, to determine the strategy to visit and evaluate promising network configurations and lastly to select transportation modes.

The rest of this section is organized as follows. Section 4.1 gives an overview of the LNS approach. Sections 4.2, 4.3, and 4.4 describe the proposed removal and repair operators, and combinations of both. The management of network configurations is detailed in section 4.5. The proposed greedy heuristics to determine transportation modes and product flows are presented in section 4.6.

4.1 The proposed LNS heuristic

The LNS heuristic, depicted in Algorithm 1, is initialized with a simple greedy heuristic that iteratively opens facilities with the least fixed cost. Each network configuration is given an initial score.

Algorithm 1 LNS heuristic

Require: Initial solution $\mathcal{S}^0$ and score for each network configuration.

1: $BestSolution \leftarrow \mathcal{S}^0$
2: $CurrentSolution \leftarrow \mathcal{S}^0$
3: **while** the termination criterion is not satisfied **do**
4: randomly select one layer to modify between J and K
5: choose a network configuration using roulette wheel selection based on the scores
6: Selection of *Removal* and *Repair* operators to be applied
7: $\mathcal{S} \leftarrow Removal(CurrentSolution)$
8: $\mathcal{S} \leftarrow Repair(\mathcal{S})$
9: Apply a greedy heuristic to obtain transportation modes and product flows variables
10: **if** $\mathcal{S} < BestSolution$ **then**
11: $BestSolution \leftarrow \mathcal{S}$
12: $CurrentSolution \leftarrow \mathcal{S}$
13: **else**
14: **if** $AcceptanceCriterion(\mathcal{S})$ **then**
15: $CurrentSolution \leftarrow \mathcal{S}$
16: **end if**
17: **end if**
18: Update the score of each network configuration
19: **end while**
20: Post optimization: determine the optimal product flows with the simplex algorithm.
21: **return** $BestSolution$

A keypoint of the heuristic is that only one layer is modified in the current iteration. In line 4, this layer is randomly chosen between plants or DCs. A target network configuration for the next instance (line 5) is chosen *a priori*, before removal and repair operators are applied. This helps the operators decide how many facilities can be removed and rebuilt, which drastically simplifies the definition of removal and repair operators.

A pair of removal and repair operators is randomly selected from a pool of operators (line 6)

The transportation modes and product flows are determined only when all facilities have been settled (line 9). Determining the optimal flow is a linear program, which can be optimally solved in polynomial time. Nevertheless, this step has to be performed in each iteration and can represent a large computation effort, especially for large-size instances. Thus, we resort to a greedy fast heuristic consisting of assigning product flows to the closest facility via the cheapest transportation mode.

When the new solution improves the current solution, it is automatically accepted and saved (lines 11 and 12). Otherwise, an *acceptance criterion* similar to that of simulated annealing, is used to determine whether the new solution should replace the current solution (line 14). See [21] for the complete description of the acceptance criterion.

In line 17, the score of the current network configuration is updated based on the value of the objective function. At the last iteration, we slightly improve the solutions provided by the LNS method with a post-optimization step (line 20): instead of the greedy heuristic, we optimize the product flows with the simplex algorithm. The computation time is about 10 seconds for the largest instances.

4.2 Removal operators

The aim of the destruction operators is to remove some open facilities from one layer of the current solution. For the sake of simplicity, we describe the remove and repair operators and give mathematical notations only for the layer corresponding to plants. All formulas can be easily transposed to the case of DCs.

Let us assume that a number n_d of plants must be removed from the current solution. All removal (resp. repair) operators except the *random* one work as follows. For each candidate plant, we calculate a score representing the benefit of closing (resp. opening) this plant. This score can be based on distance, cost, demand or other logistic criteria. All candidate plants are ranked in order of scores. The n_d plants are selected with a biased roulette wheel giving much higher probability to the plants with the best scores (see [25] for more details).

Obviously, other decision variables related to the selected facilities have to be modified accordingly. If a facility is removed from the current solution, all associated variables (e.g. ingoing and outgoing flows) are set at 0.

1. **Random removal:** this operator randomly chooses n_d open facilities to be closed. Its aim is to diversify the search in the solution space.

2. **Total cost-based removal:** this operator closes facilities with the highest estimated cost. To this end, we sum two normalized indicators for each facility, representing fixed and variable costs.

   The 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 open plants $j' \in J^o$.

   The normalized variable cost of plant $j \in J$ is a similar 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:

   $$VC_j = \frac{\sum\limits_{p \in P} (a_j^p + \sum\limits_{i \in I} \sum\limits_{m \in M} v_{ij}^{mp} + \sum\limits_{k \in K^o} \sum\limits_{m \in M} v_{jk}^{mp})}{\max\limits_{j' \in J^o} (\sum\limits_{p \in P} (a_{j'}^p + \sum\limits_{i \in I} \sum\limits_{m \in M} v_{ij'}^{mp} + \sum\limits_{k \in K^o} \sum\limits_{m \in M} v_{j'k}^{mp}))}.$$

   Note that, in the above formula, only one m index in each sum has a non-zero value. The normalized cost indicator for plant j is $FC_j + VC_j$. Our numerical experiments show that this value balances fixed and variable costs. We recall that we describe the operators only for the plant layer. The reasoning is the same for the DC layer.

3. **Capacity utilization:** this operator removes facilities with the least capacity utilization. The capacity utilization ratio RC of facility $j \in J \cup K$ is computed as follows:

   $$RC_j = \frac{cap_j - \sum\limits_{i \in V} \sum\limits_{m \in M} \sum\limits_{p \in P} x_{ji}^{mp}}{cap_j},$$

which indicates the percentage of remaining capacity in facility j.

4. **Unit cost removal:** this operator closes facilities with the least performance in terms of fixed costs and utilization of capacity. The performance of one facility $j \in J \cup K$ is measured by the ratio $S_j = \dfrac{RC_j}{FC_j}$.

5. **Horizontal cluster removal:** removes a number of facilities of the same layer located in the same region. To this end, for each cluster to be removed, a first location is randomly selected and considered as a *seed*. Then, the nearest open facilities are closed. To avoid removing too many clusters from the same region, the choice of the next cluster to be removed is based on a distance criterion from previously removed clusters.

6. **Vertical cluster removal:** the goal is to close a number of plants and DCs related to each other, i.e. clusters of plants and DCs. A first possibility is to close one plant randomly as a seed and then the nearest open DCs. A second possibility is to close one DC randomly as a seed and then the nearest open plants. We choose randomly between these two possibilities.

4.3 Repair operators

The goal of repair operators is to restore the feasibility of a partial solution after a removal operator has been used. This consists of rebuilding a number of missing facilities. The number of facilities to be opened at each layer depends on the network configuration selected in line 5 of Algorithm 1. Thereafter, we denote by n_r the number of facilities to be opened.

1. **Random repair :** n_r closed facilities are randomly selected. This operator acts as a diversification tool.

2. **Total cost repair:** this operator 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. Following [19], the candidate locations $j \in J$ are ranked in ascending order of the values

$$S_j = \frac{c_j}{cap_j} + \frac{\sum\limits_{i \in I \cup K^o} (\mu_{ij} \sum\limits_{p \in P} a_j^p) + \sum\limits_{i \in I} (\mu_{ij} \sum\limits_{p \in P} \max\limits_{m \in M} v_{ij}^{mp}) + \sum\limits_{k \in K^o} (\mu_{jk} \sum\limits_{p \in P} \max\limits_{m \in M} v_{jk}^{mp})}{\sum\limits_{i \in I \cup K^o} \mu_{ij}}$$

   where $\mu_{ij} = \min(cap_i, cap_j)$ is the maximal admissible product flow between the candidate location and the adjacent layers. In the above formula, the first term represents the unit fixed cost. The second term takes into account the processing and variable transportation costs of all types of product between each candidate facility and open facilities in adjacent layers.

3. **Best substitution:** the idea of this operator is to substitute the facilities that have just been closed by the best set of substituting facilities. It can be applied only if $n_d = n_r$.

   The set $J \backslash J^o$ of closed plants can be partitioned into two subsets: the set J^c of plants that have just been closed by the removal operator at the current iteration and the set $J \backslash (J^o \cup J^c)$ of plants which were already closed at the beginning of the current iteration. The best set of substituting facilities can be determined by solving an assignment problem, in which each element of J^c is assigned by an alternative plant in $J \backslash (J^o \cup J^c)$. The criterion to be minimized is the sum of a normalized distance between plants and a normalized fixed cost.

4. **Unit cost ratio:** this operator favors facilities with high available capacity and lower fixed costs. For each closed facility, it is based on the capacity/cost ratio $\frac{cap_j}{c_j}$.

5. **Horizontal cluster:** this operator is symmetric to the *horizontal cluster* removal operator.

6. **Vertical cluster:** this operator is symmetric to the *vertical cluster* removal operator.

7. **Cluster Customers-DC:** the idea of this operator is to open DCs near clusters of customers. First, one cluster of customers is selected. Then a closed DC in the neighborhood of the cluster of customers is randomly selected and opened. The procedure repeats n_r times. This operator relies on the definition of clusters. More details will be given in section 5.2.1.

8. **Top-down flow assignment:** in a partially destroyed solution, part of the demand may not be satisfied. This operator repairs the destroyed solution by adding material flow corresponding to the unsatisfied demand.

   We first add product flow from I to J and then from J to K, thus we call this operator *top-down flow assignment*. For each supplier, there is some unsatisfied demand for outgoing products. The corresponding quantities are sorted in descending order and placed in a priority list. We assign each element of this list to the nearest plant and update its capacity. If the nearest plant is among the non-operating plants, we open it.

   This reparation procedure continues until the target network configuration has not been reached. If the solution is still not feasible, the objective function is penalized by a quantity proportional to the amount of unsatisfied demand. If a feasible solution is found before the final network configuration is reached, we save the value of the objective function and the best solution found during the process is conserved. Hence, this operator may generate a solution with fewer open facilities than in the target network configuration.

9. **Bottom-up flow assignment:** this operator is symmetric to the *top-down assignment* described above. It assigns unsatisfied flows from customers to suppliers.

4.4 Removal and repair

These operators combine one *removal* and one *rebuild* operator sequentially.

1. **Swap operator:** if $n_d = n_r$, then we can use a swap operator, which sequentially removes and adds one facility. We first randomly choose one open facility and remove it from the current solution. Then, all closed facilities are ranked in increasing distance to the one that was just removed. One of them is selected according to the biased roulette wheel principle.

2. **History swap operator:** the goal of this operator is to diversify the search by strongly favoring facilities that were not frequently opened in previous iterations. A historical record of open and closed facilities in all iterations is collected. The history swap operator is one in which the facility to be closed (resp. opened) is selected with a biased roulette wheel based on the maximal (resp. minimal) use in past iterations.

4.5 Network configuration

In the network considered, the location decision variables are regrouped into two layers. The number of active plants in the solution varies from J_{min} to J_{max} and the number of active DCs varies from K_{min} to K_{max}. Thus the number of possible network configurations is $(J_{max} - J_{min}) \times (K_{max} - K_{min})$.

The LNS algorithm must have a good coverage of all network configurations, but their systematic exploration would be time consuming. We adopt an adaptive approach that gives a score to each network configuration. In each iteration, the next network configuration is randomly selected according to this score. This enables more computational effort to be dedicated to the most promising network configurations. The following subsections detail how we give a score to each network configuration, choose one at each iteration, and update its score during the LNS heuristic.

4.5.1 Initializing the score of all network configurations

To give an initial score to each network configuration, the problem is solved with each configuration for a predetermined number of iterations. Let us denote as N the set of all possible configurations, z_n the value of the objective function obtained with configuration $n \in N$, and $z_{n^*} = \min_{n \in N} z_n$ the overall best solution. The score w_n of configuration $n \in N$ is calculated as follows:

$$w_n = \frac{z_n - z_n^*}{\sum_{n \in N} (z_n - z_n^*)}, n = 1, .., N \tag{15}$$

4.5.2 Choosing a network configuration

At each iteration, the *LNS* heuristic explores the solution space of location variables y_j according to the current network configuration. Having only one active configuration enables the search space, and thus the computation time to be reduced.

We recall that, in each iteration, only one layer among plants and DCs is modified. Without loss of generality, let us assume that the selected layer corresponds to the plants (a similar reasoning is used if it corresponds to the DCs). The number of open plants must be within the interval $[J_{min}, J_{max}]$. Assume that the number of open facilities in the current solution is $\bar{j}$. In order to avoid too large variations from one iteration to another, only slight variations in network configuration are authorized.

Then, the number of open facilities at the next iteration is chosen from the next three possibilities: $\max(J_{min}, \bar{j} - 0.2 \times J_{max})$, $\bar{j}$ and $\min(J_{max}, \bar{j} + 0.2 \times J_{max})$.

Following the idea of the adaptive large neighborhood search [21], the probability of choosing each possibility depends on the network configuration score, which is calculated from the network configuration's performance in past iterations. The score is updated after each segment of 100 iterations, with an exponential smoothing formula (see [21]). Hence, in the final iterations the LNS heuristic tends to focus on network configurations with the highest scores.

4.6 Determining transportation modes and product flows

In line 9 of Algorithm 1, we apply a greedy heuristic to select the product flows and the transportation modes between facilities. We first determine the transportation modes and the product flows between all DCs and customers, which is described in Algorithm 2. Then, we adopt the same principle to determine transportation modes and product flows between plants and DCs and between suppliers and DCs.

Algorithm 2 is based on a priority order defined by the largest demands. All demands d_l^p are ranked in descending order (line 2) and we keep assigning products in descending order of this priority order (line 3) with a greedy criterion based on the transportation cost (line 5). Sometimes it is impossible to assign the whole customer demand d_l^p due to the capacity limitation cap_k (line 6). Then, the cheapest transportation mode is selected for arc $(k, l) \in A$ (line 9). Other available modes for this arc are discarded by increasing their cost to an arbitrary large value (line 10).

Algorithm 2 Assignment of transportation modes and product flows

Require: d_l^p : demand of customer $l \in L$ for product $p \in P$, cap_k : capacity of DC $k \in K$, v_{kl}^{mp} : variable transportation cost for product p on arc (k,l) with mode $m \in M$.

Initialization of transportation modes: $t_{ij}^m = 0, \forall (i,j) \in A, m \in M$.
Build a list $ListD$ of demands with all demands d_l^p in decreasing order.
for all demands $d_l^p \in ListD$ **do**
  while $d_l^p > 0$ **do**
    select the DC k^* and the transportation mode m^* with minimum cost $v_{k^*l}^{m^*p}$
    calculate the value of a maximal feasible shipment $x_{k^*l}^{m^*p} = \min(cap_k, d_l^p)$
    update remaining capacity at DC: $cap_k \leftarrow cap_k - x_{k^*l}^{m^*p}$
    update customer demand: $d_l^p \leftarrow d_l^p - x_{k^*l}^{m^*p}$
    update transportation mode list: $t_{kl}^{m*} = 1$
    update $ListD$ and values $v_{k^*l}^{mp}$ for $m \neq m^*$
  end while
end for
return values of x_{kl}^{mp} and t_{ij}^m

5 Computational experiments

In this section, we detail the computational experiments we performed in order to validate the proposed LNS heuristic. For this purpose, we generated a set of instances of different sizes and characteristics. Since these experiments rely on generated instances, we first explain the main principles of the data generation. Then, we explain how the parameters of the LNS were set. Lastly, we detail the numerical results by analyzing the efficiency of each removal and repair operator and comparing the results of the LNS with those of a state-of-the-art MILP solver. All algorithms were coded in C++ and performed on a computer with four Intel 3.0 GHz CPUs and 8 GB of RAM.

5.1 Test instances

We generated a total of 60 instances, with 15 distinct sizes and 4 types of supply chain network configurations. The size of these instances is 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 upper limits $|J_{max}|$ and $|K_{max}|$ on the number of plants and DCs that can be opened. Similarly to [5], we set the number of potential suppliers and plants to $|I| = |J| = 0.1 \times |L|$. The number of potential DCs was set to $|K| = 0.2 \times |L|$. The values J_{max} and K_{max} were set to $0.5 \times |J|$ and $0.5 \times |K|$, respectively. Table 1 displays the value of all parameters for each size of instance. The goal of generating small test instances is to compare LNS solutions with known optimal solutions obtained with an MILP solver. The aim of large instances is to study how the LNS behaves when the solver is unable to solve the instances to optimality.

5.2 Data generation

5.2.1 Generation of four types of logistics network

All locations were generated on a 200×200 grid. Since the physical layout of suppliers, facilities and customers may influence the network configuration, we generated four types of pattern corresponding to different realistic situations:

Table 1: Characteristics of test instances

Problem size	$\|I\|$	$\|J\|$	$\|K\|$	$\|L\|$	$\|J_{max}\|$	$\|K_{max}\|$
s1	6	6	12	60	3	6
s2	7	7	14	70	4	7
s3	8	8	16	80	4	8
s4	9	9	18	90	5	9
s5	10	10	20	100	5	10
s6	12	12	24	120	6	12
s7	14	14	28	140	7	14
s8	16	16	32	160	8	16
s9	18	18	36	180	9	18
s10	20	20	40	200	10	20
s11	22	22	44	220	11	22
s12	24	24	48	240	12	24
s13	26	26	52	260	13	26
s14	28	28	56	280	14	28
s15	30	30	60	300	15	30

- **Pattern** 1**:** the coordinates of all vertices were randomly generated with a uniform distribution in the interval $[0, 200]$ along each axis.
- **Pattern** 2**:** we assumed that around 60% of all facilities are located in a few clusters representing large cities or areas with a large density. The remaining 40% are scattered randomly throughout the grid. To define the coordinates of each cluster, the 200×200 grid was divided into 25 sub-grids. We generated one cluster at most in each sub-grid and the facilities of each cluster were randomly generated within the corresponding sub-grid.
- **Pattern** 3**:** we modeled industrial regions with clusters of suppliers and clusters of plants. The coordinates of a *seed* cluster of suppliers were first selected randomly with a uniform distribution in the 200×200 grid. Then, the coordinates of a cluster of plants were chosen randomly in the same sub-grid. The remaining vertices were randomly generated.
- **Pattern** 4**:** we modeled areas with a high density of customers. 60% of all DCs and customers were regrouped into 4 to 5 clusters. A *seed* location was first selected randomly with a uniform distribution in the 200×200 grid. Then, a cluster of customers and a cluster of DCs were generated in the same sub-grid as the seed. The procedure was repeated until 60% of the customers and DCs had been generated. All remaining vertices were randomly generated.

5.2.2 Generation of transportation modes

We assumed three transportation modes in the network. Table 2 shows the main characteristics of these modes. Column 2 states whether the corresponding transportation mode is subject to fixed costs. Column 3 displays the relative variable cost of each mode compared with the others. Columns 4 and 5 detail the minimal load limitations and other restrictions on each transportation mode.

For example, mode 1 can be an internal fleet of trucks. Mode 2 can concern an outsourced fleet of trucks for the delivery of goods to the customers. Mode 3 can correspond to a vessel or a train, with a minimal amount of load on each shipment and that only for long-distance trips.

The fixed cost of mode 1 was assumed to be 10000. The variable cost of mode 2 was defined as 20% more expensive than the variable cost of mode 1. We also assumed that the fixed cost of mode 3 was zero and its variable was 80% of that of mode 1.

Table 2: Characteristics of transportation modes

Transportation mode	Fixed cost	Variable cost	Min limitation	Restrictions
mode 1	✓	Intermediate	×	×
mode 2	×	Highest	×	DCs to customers
mode 3	×	Lowest	✓	Long distance only (suppliers to plants, plants to DCs)

However, mode 1 can be used without any restriction whereas mode 2 is limited to the distribution to customers and mode 3 has a minimum product flow restriction and can be used only for a subset of long-haul trips. Admissible trips with mode 3 were generated by dividing the 200×200 grid into 25 sub-grids. Only a subset of sub-grids allows mode 3. Moreover, the two ends of the trip must be as far apart as distinct sub-grids that both allow mode 3.

5.2.3 Fixed cost of facility location

The fixed cost of opening facilities varies with the price of the real estate market in each region. Investment costs are significantly larger in sub-grids with a higher density of customers. We divided the 200×200 grid into 20×20 sub-grids of equal size and generated distinct fixed costs in each sub-grid, with a relative difference of about 60% between the cheapest and the most expensive sub-grids.

We assumed economies of scale when building large facilities. Thus the fixed costs are also roughly proportional to the square root of the facilities capacity.

5.2.4 Variable costs

The variable transportation cost between two nodes depends on the arc length, the transportation modes and local factors. A variable transportation cost tc_{ij}^{1p} on arc $(i,j) \in A$ with mode 1 for product p was determined randomly in the interval $[0.8, 1.2] \times t_{ij} \times \tau$, where t_{ij} is the distance between nodes i and j, and τ is a parameter 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, we assumed slightly increasing transportation costs layer by layer. Thus, τ was randomly chosen in the interval $[1, 1.3]$ for the transportation between a supplier and a plant, in the interval $[1.2, 1.4]$ between plants and DCs and in the interval $[1.3, 1.5]$ for the distribution to customers. As explained before, the variable transportation costs for modes are proportional to that of mode 1.

The unit processing cost for a product $p \in P$ was generated as the sum of the purchase cost (in the interval $[130, 150]$), production cost (in the interval $[130, 150]$) and warehousing and logistics costs (in the interval $[100, 120]$). Then, for every node $n \in I, J, K$ and every product $p \in P$, we generated the corresponding costs, which were noised by multiplying them by a factor randomly chosen in the interval $[0.9, 1.2]$.

Since the variable costs (fixed transportation cost, variable transportation cost, and processing cost) influence the network configuration, we considered two levels of variable costs. Following [5] and [26], half of the instances were generated in such a way that variable costs represented $40\% - 50\%$ of the total network cost. In the other instances this cost represented $20\% - 30\%$ of the total cost. In the latter case, the fixed cost values were multiplied by 2.

5.2.5 Capacity of facilities

Assume that u is the sum of all customer demands. Then, the capacity of each DC was chosen randomly with a uniform distribution in the interval $[\frac{u}{|K_{max}|}, 1.1 \times \frac{u}{|K_{max}|}] \times 1.5$. The same formula was also

applied to plants, with 2 instead of 1.5. Accordingly, the capacity of suppliers was chosen randomly with a uniform distribution in the interval $[\frac{u}{|I|}, 1.1 \times \frac{u}{|I|}] \times 3$.

5.2.6 Demand

As proposed in [36], the customer demands were randomly generated according to a uniform distribution in the interval $[100, 300]$.

5.3 Parameter setting

Preliminary experiments concluded that running the LNS algorithm during 10000 iterations for test instances 1 to 10 and 15000 iterations for test instances 11 to 15 ensured a good trade-off between the computational time and the quality of the solution. The initial score of each network configuration was calculated by running each configuration at least 75 times. The parameters of the simulated annealing acceptance criterion such as cooling rate and initial temperature were tuned following [21].

5.4 Evaluation of the LNS operators

In order to evaluate the pertinence of the LNS operators, we selected a representative subset of 15 instances.

The first protocol compares the results obtained with and without each operator. First, we ran the LNS with all operators and obtained an objective value z_1. Then, we excluded each of the operators one by one while keeping the others. We obtained an objective value z_2. The individual contribution of each operator is measured by the quantity

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

The second protocol compares the results obtained with the random operator only and with the random operator + the assessed operator. More precisely, if we assess a removal operator, z_1 represents the value of the objective obtained with the random destruction + the assessed operator followed by all repair operators. z_2 represents the objective obtained with the random destruction operator only, followed by all repair operators. If we assess a repair operator, z_2 represents the objective obtained with all removal operators followed by random repair + the repair assessed operator. Value z_1 represents the objective obtained with all removal operators followed by the random repair operator only. The individual contribution of each operator is measured by the quantity

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

Table 3 shows the contribution of each operator measured by both protocols. We performed five runs on the 15 representative instances. Since the results were quite stable, we only report the average results.

The second column shows only positive numbers, which means that all operators contribute to the efficiency of the LNS. The third column shows that almost all operators, except *history swap*, give positive numbers, which means that they contribute to the efficiency of the LNS. The negative value for *history swap* is quite normal since this operator is a pure diversification factor, which is likely to work only when combined with other operators.

Table 4 analyzes the utility of each operator over five runs on a representative subset of 15 instances. For each operator, column 2 presents the average percentage of fruitful iterations, i.e. the iterations which 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 into the three categories.

Table 3: Average contribution of each removal and repair operator

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

Table 4: Operator utility

Operator	% of useful iterations	best	improving	accepted
		(% of the results in column 2)		
Random removal	3.5	1.1	52.8	46.1
Total cost-based removal	3.9	1.1	56.5	42.4
Capacity utilization	4.0	1.3	57.7	40.9
Unit cost removal	4.1	2.0	58.0	40.0
Horizontal cluster removal	3.1	0.9	52.4	46.7
Vertical cluster removal	3.1	0.8	52.1	47.1
Random repair	2.9	1.0	53.4	45.6
Total cost-based	4.6	1.2	59.1	39.7
Best substitution	11.5	1.3	55.0	43.7
Unit cost ratio	4.6	2.0	60.5	37.5
Horizontal cluster	1.5	0.3	39.3	60.4
Vertical cluster	2.9	0.5	53.1	46.4
Cluster customers-DC	1.4	0.5	39.7	59.8
Top-down assignment	6.3	2.5	59.6	37.9
Bottom-up assignment	4.5	0.4	51.5	48.1
Swap	7.9	2.3	48.7	49.0
History swap	0.6	1.1	93.1	5.7
Average	*4.1*	*1.2*	*55*	*43*

These results show that no operator outperforms another. Some operators seem to have a negligible effect, but removing 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 in escaping 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.

5.5 Computational 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 computational time of 3 hours. The heuristic was run 10 times on each of the 60 instances. The computational results are presented in Tables 5–7. Columns 3 and 4 present the computational time (in seconds) for Cplex and the LNS heuristic, respectively. Columns 5 and 6 present the minimal and average gap (in %) between the results found by the LNS and Cplex. Columns 7 and 8 in Tables 6–7 present the minimal and average gap (in %) between the LNS and the lower bounds found by Cplex. These columns are filled only when Cplex cannot find any optimal solution after 3 hours of computation (in this case, column 3 contains an $*$).

Table 5: Comparison between Cplex and the LNS (sets s1 to s5)

Set	CPU (seconds)		UB GAP (%)	
	Cplex	LNS	Min	Avg.
s1	37	16	0.86	1.52
	109	18	0.82	1.65
	82	18	0.59	0.78
	38	18	1.01	1.06
s2	651	26	1.51	2.25
	411	29	1.18	1.86
	502	29	1.75	2.45
	480	29	1.22	1.49
s3	326	36	2.38	2.74
	503	40	0.99	2.00
	662	40	0.68	0.99
	224	40	0.55	1.54
s4	1150	47	0.76	1.07
	1072	63	1.21	1.38
	668	52	1.45	1.79
	146	51	0.88	1.32
s5	686	57	0.59	1.09
	788	63	0.66	1.41
	597	62	0.57	0.71
	682	63	0.78	1.00
Average			1.02	1.50

In 23 out of the 60 instances, Cplex finds no optimal solution after 3 hours of computation. For these 23 instances, the average gap is 0.55% and 1.20% from the upper bound and 2.75% and 3.41% from the lower bound. If we ignore these 23 instances, the average value of the optimality gaps in columns 4 and

Table 6: Comparison between Cplex and the LNS (sets s6 to s10)

Set	CPU (seconds)		UB GAP (%)		LB GAP (%)	
	Cplex	LNS	Min	Avg.	Min	Avg.
s6	909	69	1.13	1.84		
	1097	77	0.45	0.86		
	430	80	1.45	2.41		
	2826	78	1.91	2.52		
s7	4942	89	1.57	1.93		
	3534	101	0.64	1.27		
	7395	101	0.83	1.37		
	3115	97	0.90	1.30		
s8	1443	119	0.39	1.38		
	*	136	1.10	1.88	1.43	2.22
	7659	140	1.10	1.46		
	3905	135	0.48	1.24		
s9	4786	142	0.83	1.58		
	6265	169	0.79	1.10		
	*	161	1.07	1.79	1.15	1.88
	6180	165	1.41	1.93		
s10	*	194	0.60	1.62	1.73	2.76
	5029	213	0.65	1.00		
	*	252	1.06	1.47	1.67	2.09
	4464	212	0.51	0.90		
Average			0.94	1.54	1.50	2.23

Table 7: Comparison between Cplex and the LNS (sets s11 to s15)

Set	CPU (seconds)		UB GAP (%)		LB GAP (%)	
	Cplex	LNS	Min	Avg.	Min	Avg.
s11	*	348	1.14	1.44	4.51	4.83
	*	406	0.97	1.80	1.85	2.69
	*	410	1.43	1.92	1.60	2.09
	5481	381	0.49	1.14		
s12	*	606	0.73	1.80	3.23	4.33
	*	473	0.87	1.03	2.77	2.93
	*	500	1.28	1.61	1.47	1.80
	*	498	1.08	2.01	1.55	2.49
s13	*	569	-0.42	0.65	5.03	6.16
	*	638	0.28	0.88	4.01	4.64
	*	608	-0.21	0.91	3.48	4.64
	*	627	0.60	0.88	4.70	4.99
s14	*	765	1.13	1.62	4.02	4.52
	*	757	1.05	1.38	1.68	2.00
	*	705	0.81	1.26	2.92	3.38
	*	751	0.68	1.21	0.70	1.23
s15	*	907	0.05	1.11	4.15	5.26
	*	861	0.17	0.66	4.08	4.58
	*	908	-0.86	0.27	3.41	4.59
	*	923	-1.91	-1.63	2.13	2.42
Average			0.47	1.10	3.02	3.66

5 is 0.97% and 1.50%, respectively.

As an illustration, Figure 2 represents how the gap is reduced during the search process for test instance s15, pattern 4. The initial solution has a 6.78% gap from the lower bound. After 2000 iterations, the LNS outperforms the upper bound provided by Cplex. After 10500 iterations, the LB gap is 2.2%.

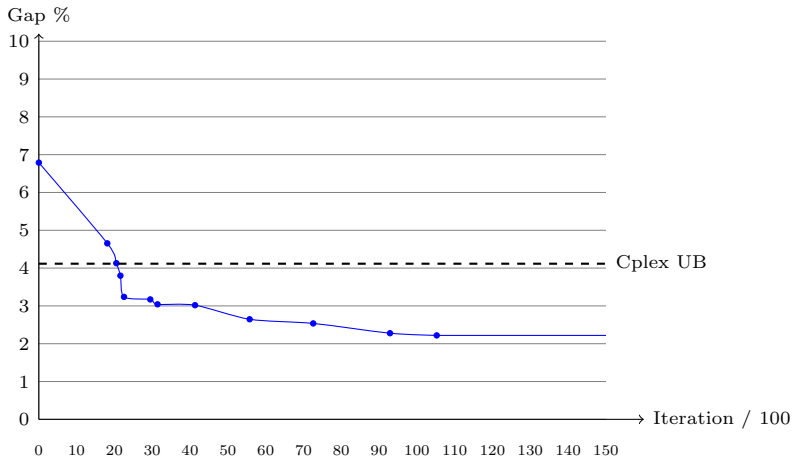

Figure 2: Example of LNS iterations (Test set s15, pattern 4)

It can be observed from Tables 6–7 that the LB gap of the LNS ranges from 0.97% to 4.98% with an average value of 2.83%. For instance, in test set $s13$, pattern 1, the LB gap is 4.98%, due to the difficulty of finding good lower bounds with Cplex. The UB gap ranges from -2.00% to 2.13% with an average value of 0.84%, which shows the efficiency of the proposed solution method. It is worth mentioning that the best and average results (columns 5–6, columns 7–8) and the non-significant difference between the results for each pattern are good indicators of the stability of our heuristic.

The maximum running time of the LNS is 923 seconds with an average of 269 seconds, which shows the ability of the heuristic to find good results within an acceptable time. Moreover, it can be observed that the CPU time of the LNS is not influenced by the pattern of data. For instance, Cplex could solve some test problems such as $s8$ and $s9$ to optimality, while it could not do so for other instances of the same size.

As stated before, few models in the literature incorporate transportation modes in designing a supply chain network. However, in real-life problems, using suitable transportation modes influences the logistics network efficiency. We ran the algorithm over a subset of instances with two options. Firstly, there is only one mode available between each pair of nodes. Secondly, it is possible to choose a suitable mode. The LNS heuristic was run 10 times for each test instance. The results presented in Table 8 show that considering different transportation modes has a strong influence on the network configuration. Columns 2 and 3 represent the percentage of different facilities between each scenario. The results show that the locations of 19% of plants and 24% of DCs have been changed.

6 Conclusion

In this paper, we studied a 4-layer multi-product supply chain network design model including production plant and distribution center location as well as selection of the transportation modes. The goal was to design the supply chain to satisfy the demand while minimizing the total fixed and variable costs over a single period of time.

Table 8: Influence of transportation modes on network configuration

Test Problems	*Network configuration*	
	% of plants changed	*% of DCs changed*
1	-	-
2	-	20
3	-	40
4	33	17
5	-	43
6	50	-
7	-	22
8	25	30
9	20	17
10	20	38
11	50	36
12	33	27
13	29	24
14	0	22
15	29	32
Average	19	24

An LNS approach was been designed to solve the proposed model. The LNS framework was mainly used to fix the location decisions whereas a greedy heuristic was called upon to select the appropriate transportation modes and determine the product flows. Once the network configuration had been determined, a post-optimization step using the simplex algorithm determined the optimal product flows.

The performance of the proposed method was tested on a variety of randomly-generated instances with various sizes and practical characteristics. We provide extensive comparisons with optimal solutions or bounds obtained with Cplex. The numerical results show the stability of the LNS algorithm and its efficiency, in terms of both quality of solution and computation time.

This study shows that the principles of the LNS can be used successfully for supply chain network design problems. Further research will include the adaptation of our algorithm to more complex models, for example with multiple periods or multiple objectives.

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.

References

References

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

[2] Y. Cardona-Valdés, A. Álvarez, and J. Pacheco. Metaheuristic procedure for a bi-objective supply chain design problem with uncertainty. *Transportation Research Part B: Methodological*, 60(0):66 – 84, 2014.

[3] Dick Carlsson and Mikael Rönnqvist. Supply chain management in forestry—case studies at Södra Cell {AB}. *European Journal of Operational Research*, 163(3):589 – 616, 2005. Supply Chain Management and Advanced Planning.

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

[5] J.-F. Cordeau, F. Pasin, and M.M. Solomon. An integrated model for logistics network design. *Annals of Operations Research*, 144:59–82, 2006.

[6] Mark S. Daskin, Lawrence V. Snyder, and Rosemary Berger. Facility location in supply chain design. In André Langevin and Diane Riopel, editors, *Logistics systems: design and optimization*, pages 39–65. Springer US, 2005.

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

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

[9] M. Eskandarpour, E. Nikbakhsh, and S.H. Zegordi. Variable neighborhood search for the bi-objective post-sales network design problem: A fitness landscape analysis approach. *Computers & Operations Research*, In Press.

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

[11] Erdem Eskigun, Reha Uzsoy, Paul V. Preckel, George Beaujon, Subramanian Krishnan, and Jeffrey D. Tew. Outbound supply chain network design with mode selection, lead times and capacitated vehicle distribution centers. *European Journal of Operational Research*, 165(1):182 – 206, 2005.

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

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

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

[15] A. Klose and A. Drexl. Facility location models for distribution system design. *European Journal of Operational Research*, 162(1):4–29, 2005.

[16] Mary J. Meixell and Vidyaranya B. Gargeya. Global supply chain design: A literature review and critique. *Transportation Research Part E: Logistics and Transportation Review*, 41(6):531–550, 2005.

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

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

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

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

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

[22] D. Pisinger and S. Ropke. Large neighborhood search. In Michel Gendreau and Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer, 2010.

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

[24] C.S. ReVelle and H.A. Eiselt. Location analysis: A synthesis and survey. *European Journal of Operational Research*, 165(1):1–19, 2005.

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

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

[27] Güvenç Sahin and Haldun Süral. A review of hierarchical facility location models. *Computers & Operations Research*, 34(8):2310–2331, 2007.

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

[29] Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, *Principles and Practice of Constraint Programming – CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin Heidelberg, 1998.

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

[31] M.K. Tiwari, N. Raghavendra, S. Agrawal, and S.K. Goyal. A hybrid Taguchi-immune approach to optimize an integrated supply chain design problem with multiple shipping. *European Journal of Operational Research*, 203(1):95–106, 2010.

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

[33] Wilbert Wilhelm, Dong Liang, Brijesh Rao, Deepak Warrier, Xiaoyan Zhu, and Sharath Bulusu. Design of international assembly systems and their supply chains under {NAFTA}. *Transportation Research Part E: Logistics and Transportation Review*, 41(6):467 – 493, 2005.

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

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

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