CIRRELT

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

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

An Iterated Local Search Heuristic for the Logistics Network Design Problem with Single Assignment

Jean-François Cordeau
Gilbert Laporte
Federico Pasin

June 2007

CIRRELT-2007-20

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

Bureaux de Québec :
Université Laval
Pavillon Palasis-Prince, local 2642
Québec (Québec)
Canada G1K 7P4
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

An Iterated Local Search Heuristic for the Logistics Network Design Problem with Single Assignment

Jean-François Cordeau[1,*] Gilbert Laporte[1], Federico Pasin[2]

1. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Université de Montréal, C.P. 6128, succursale Centre-ville, Montréal, Canada H3C 3J7, and Canada Research Chair in Logistics and Transportation, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

2. Service de l'enseignement de la gestion des opérations et de la logistique, HEC Montréal, 3000 Côte-Sainte-Catherine, Montréal, Canada H3T 2A7

Abstract. In the logistics network design problem (LNDP), decisions must be made regarding the selection of suppliers, the location of plants and warehouses, the assignment of activities to these facilities, and the flows of raw materials and finished products in the network. This article introduces an iterated local search (ILS) heuristic for the LNDP variant arising when each raw material should be supplied by a unique supplier, and each finished product should be produced and distributed by a unique plant and a unique warehouse, respectively. The ILS heuristic exploits the combinatorial nature of the problem and relies on simple moves combined within a descent algorithm. Several perturbation operators are used to allow a broad exploration of the solution space. The performance of the algorithm is evaluated on randomly generated instances, and the solutions are compared with lower bounds computed by solving the LP relaxation of the problem.

Keywords. Iterated local search, logistics, network design.

Acknowledgements. This work was partly supported by a Strategic research grant provided by HEC Montréal, and by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 227837-04 and 39682-05. This support is gratefully acknowledged. We are grateful to two anonymous referees for their valuable comments.

---

* Corresponding author: jean-francois.cordeau@hec.ca

Dépôt légal – Bibliothèque nationale du Québec,
Bibliothèque nationale du Canada, 2007

Introduction

In the *Logistics Network Design Problem* (LNDP), decisions must be made regarding: *i)* the selection of suppliers; *ii)* the location of plants and warehouses; *iii)* the assignment of raw materials to suppliers and of finished products to plants and warehouses (i.e., the activity mix of each facility); and *iv)* the flows of raw materials and finished products through the network. These decisions must be made so as to satisfy customer demand while minimizing the sum of fixed and variable costs associated with procurement, production, warehousing and transportation. This paper is concerned with the deterministic LNDP arising in a single-period and single-country environment.

The LNDP generalizes classical capacitated facility location problems (see, e.g., Aikens (1985), Drezner (1995)) by considering multiple stages (or echelons) connected together by product flows. In addition, the design of a logistics network often involves additional decisions regarding technology acquisition, transportation mode selection or inventory levels. Models for the design of multi-national networks must also take into account factors such as exchange rates, transfer prices, duties, and income taxes (see, e.g., Arntzen et al. (1995), Vidal and Goetschalckx (2001), Hadjinicola and Kumar (2002)). Because they involve strategic decisions over long planning horizons, some models also incorporate stochastic elements (see, e.g., Santoso et al. (2005)).

Following the pioneering work of Geoffrion and Graves (1974) on multi-commodity distribution network design, a large number of models have been proposed to locate facilities by incorporating sourcing, production and transportation aspects. Notable examples are the studies of Brown et al. (1987), Cohen and Lee (1989), Pirkul and Jayaraman (1996), Dogan and Goetschalckx (1999), Lakhal et al. (2001), Paquet et al. (2004) and Martel (2005). For detailed reviews of such models, the reader is referred to the recent surveys of Goetschalckx et al. (2002), Klose and Drexl (2005) and Meixell and Gargeya (2005).

Recently, Cordeau et al. (2006b) have introduced a general formulation of the LNDP for the deterministic, single-country, single-period context. This formulation integrates supplier selection, plant location and warehouse location with product-range assignment, sourcing decisions and transportation mode selection. The authors have described two algorithms for solving the resulting formulation: a simplex-based branch-and-bound algorithm and a Benders decomposition procedure. They have also proposed valid inequalities to strengthen the LP relaxation of the model and accelerate convergence. Both approaches are capable of solving within reasonable CPU time instances with several hundred customers but a modest

number (i.e., a few tens) of products and potential suppliers, plant locations and warehouse locations.

When considering large numbers of suppliers and potential plant and warehouse locations, however, the combinatorial nature of the problem becomes more important, and solving the problem optimally becomes very difficult. This is particularly true when *single assignment constraints* are considered. Under these constraints, each raw material should be supplied by a unique supplier while each finished product should be manufactured and distributed by a unique plant and a unique warehouse, respectively. These constraints are especially relevant in situations where a large number of products have a low but highly variable demand. In this case, it is often desirable to centralize the production and distribution of each product so as to benefit from economies of scale.

For example, the first author has recently studied a network design problem arising in the management of spare parts used by a large public utility in Canada. In this problem, several depots have to be located but each part should be stocked in a single depot to avoid maintaining large inventories of slow-moving items. Obviously, centralization may lead to higher transportation costs but the analysis has shown that the reduction in inventory costs more than offsets the increase in transportation costs.

Under single assignment constraints, the LNDP becomes purely combinatorial and can be reformulated using only binary variables. Solving the resulting formulation exactly through branch-and-bound proves to be extremely difficult because of the large number of variables and the poor lower bounds provided by the linear programming relaxation. This problem can, however, be effectively tackled by means of local search heuristics.

Facility location and logistics network design problems can often be solved satisfactorily by classical mathematical programming techniques such as branch-and-bound, Lagrangian relaxation and Benders decomposition. Nevertheless, numerous heuristic algorithms have been proposed to address different variants of these problems. Since the publication of the Kuehn and Hamburger (1963) heuristic for warehouse location, a large number of approximate algorithms have been devised to address location problems possessing some combinatorial structure. Recent examples are the work of Ghosh (2003), Michel and Van Hentenryck (2004), and Sun (2006) on tabu search for uncapacitated warehouse location. The capacitated case has received far less attention in terms of heuristics. Recently, Ahuja et al. (2002) have devised a very large scale neighborhood search algorithm for the capacitated facility location problem with single-sourcing constraints. Reactive GRASP and tabu search heuristics for this problem were also proposed by Delmaire et al. (1999). Finally, Lapierre et al. (2004) have described a hybrid heuristic combining tabu search

and variable neighborhood search for the design of distribution networks with transshipment centers and general cost functions.

The purpose of this article is to introduce an iterated local search heuristic for the *LNDP with Single Assignment* (LNDPSA). This heuristic is easy to implement and relies on a number of simple yet powerful local search operators. Comparisons with the branch-and-bound approach of Cordeau et al. (2006b) show that the proposed heuristic yields solutions of very high quality in a fraction of the CPU time required for just solving the linear programming relaxation of the problem.

The remainder of the paper is organized as follows. In Section 2, we formally define the LNDPSA and introduce two mathematical formulations of the problem. In Section 3, we then describe the iterated local search heuristic. This is followed by computational results in Section 4, and by the conclusion.

1 Problem definition and mathematical formulation

In this section, we first introduce the notation that will be used throughout the paper. In Section 1.1, we then recall the linear mixed-integer programming formulation of Cordeau et al. (2006b) for the LNDP and explain how it can be modified to model the LNDPSA. We then present in Section 1.2 a more compact, non-linear integer programming formulation.

Let $\mathcal{F}$ and $\mathcal{R}$ denote the sets of finished products and raw materials, respectively. For every $r \in \mathcal{R}$ and every $f \in \mathcal{F}$, let b^{rf} be the quantity of raw material r required in the production of one unit of product f. The set of all suppliers considered by the company is denoted by $\mathcal{S}$, and $\mathcal{S}^r \subseteq \mathcal{S}$ represents the subset of suppliers that can provide raw material $r \in \mathcal{R}$. Let also $\mathcal{P}$ and $\mathcal{W}$ denote the sets of potential locations for plants and warehouses, respectively. For every product $f \in \mathcal{F}$, let $\mathcal{P}^f$ and $\mathcal{W}^f$ denote the subsets of plants and warehouses at which product f can be made and stored, respectively. Finally, let $\mathcal{C}$ be the set of customers, and for every $c \in \mathcal{C}$ and $f \in \mathcal{F}$, let $a_c^f \geq 0$ be the demand of customer c for product f.

For notational convenience, denote by $\mathcal{K} = \mathcal{R} \cup \mathcal{F}$ the set of all commodities represented in the model, and by $\mathcal{O} = \mathcal{S} \cup \mathcal{P} \cup \mathcal{W}$ and $\mathcal{D} = \mathcal{P} \cup \mathcal{W} \cup \mathcal{C}$ the sets of potential origins and destinations for these commodities. Then, for every $k \in \mathcal{K}$, define $\mathcal{O}^k \subseteq \mathcal{O}$ and $\mathcal{D}^k \subseteq \mathcal{D}$ as the sets of potential origins and destinations for commodity k. More specifically, $\mathcal{O}^r = \mathcal{S}^r$ for any raw material $r \in \mathcal{R}$, and $\mathcal{O}^f = \mathcal{P}^f \cup \mathcal{W}^f$ for any product $f \in \mathcal{F}$. Similarly, possible destinations for a raw material r are plants at which products requiring this raw material

can be made, i.e., $\mathcal{D}^r = \cup_{f \in \mathcal{F}^r} \mathcal{P}^f$, where $\mathcal{F}^r = \{f \in \mathcal{F} | b^{rf} > 0\}$. Finally, the set of possible destinations for a product f is defined as $\mathcal{D}^f = \mathcal{W}^f \cup \mathcal{C}^f$, where $\mathcal{C}^f = \{c \in \mathcal{C} | a_c^f > 0\}$.

For every $k \in \mathcal{K}$ and every $o \in \mathcal{O}^k$, let V_o^k be a binary variable equal to 1 if and only if commodity k is assigned to origin o, and let c_o^k be the fixed cost of assigning the commodity to the origin. For instance, variable V_s^r will take the value 1 if supplier s is selected to provide raw material r, and variable V_p^f will take the value 1 if product f is made at plant p. For every origin $o \in \mathcal{O}$, also define a binary variable U_o equal to 1 if and only if this origin is assigned at least one commodity, and let c_o be the fixed cost of selecting this origin. In the case of a supplier $s \in S$, the variable U_s will take the value 1 if the supplier is selected to provide at least one raw material. In the case of a potential plant or warehouse location, the associated variable will take the value 1 if the corresponding location is chosen to site a facility.

For every $o \in \mathcal{O}$, let u_o be the capacity, in equivalent units, of origin o, and for every $k \in \mathcal{K}$, let u_o^k be the amount of capacity required by one unit of commodity k at origin o. An equivalent unit is a common unit of measurement that can be used to compare and add the resource requirements of various products. Typical examples are pallets, kilograms or work hours required. In the case of a plant p, u_p represents the total manufacturing capacity in the planning horizon while u_p^f is the transformation factor to convert real units of product f into equivalent units. For every $k \in \mathcal{K}$ and $o \in \mathcal{O}^k$, let also q_o^k be the capacity of origin o for commodity k.

Finally, for every $k \in \mathcal{K}$, $o \in O^k$ and $d \in D^k$, define a non-negative variable X_{od}^k representing the number of units of commodity k transported from origin o to destination d, and let c_{od}^k be the unit transportation cost. This cost may also include the cost of the activity performed at the origin o (i.e., the cost of acquisition, production or handling of one unit of commodity k).

1.1 A linear mixed-integer programming formulation

The LNDP is formulated by Cordeau et al. (2006b) as the following linear mixed-integer program:

Minimize

$$\sum_{o \in \mathcal{O}} \left[c_o U_o + \sum_{k \in \mathcal{K}} \left[c_o^k V_o^k + \sum_{d \in \mathcal{D}^k} c_{od}^k X_{od}^k \right] \right] \tag{1}$$

subject to

$$\sum_{s\in\mathcal{S}^r} X^r_{sp} - \sum_{f\in\mathcal{F}^r}\sum_{w\in\mathcal{W}^f} b^{rf} X^f_{pw} = 0 \qquad r\in\mathcal{R}; p\in\mathcal{P} \tag{2}$$

$$\sum_{p\in\mathcal{P}^f} X^f_{pw} - \sum_{c\in\mathcal{C}^f} X^f_{wc} = 0 \qquad f\in\mathcal{F}; w\in\mathcal{W}^f \tag{3}$$

$$\sum_{w\in\mathcal{W}^f} X^f_{wc} = a^f_c \qquad f\in\mathcal{F}; c\in\mathcal{C}^f \tag{4}$$

$$\sum_{k\in\mathcal{K}}\sum_{d\in\mathcal{D}^k} u^k_o X^k_{od} - u_o U_o \le 0 \qquad o\in\mathcal{O} \tag{5}$$

$$\sum_{d\in\mathcal{D}^k} X^k_{od} - q^k_o V^k_o \le 0 \qquad k\in\mathcal{K}; o\in\mathcal{O}^k \tag{6}$$

$$X^k_{od} \ge 0 \qquad k\in\mathcal{K}; o\in\mathcal{O}^k; d\in\mathcal{D}^k \tag{7}$$

$$U_o \in \{0,1\} \quad o\in\mathcal{O} \tag{8}$$

$$V^k_o \in \{0,1\} \quad k\in\mathcal{K}; o\in\mathcal{O}^k. \tag{9}$$

The objective function (1) minimizes the sum of all fixed and variable costs. Variable costs c^k_{od} may include not only transportation expenses but also relevant acquisition, production and storage costs. Constraints (2) ensure that the total amount of raw material r shipped to plant p is equal to the total amount required by all products made at this plant, while constraints (3) ensure that all finished products that enter a given warehouse also leave that warehouse. Demand constraints are imposed by equations (4). Constraints (5) impose global capacity limits on suppliers, plants and warehouses, whereas limits per commodity are enforced through (6). The latter constraints can be used, for example, to restrict the total amount of a given raw material purchased from a particular supplier, or the number of units of a finished product made at a particular plant.

To address the LNDPSA, model (1)-(9) can be restricted by imposing the following single assignment constraints for all commodities:

$$\sum_{o\in\mathcal{O}^k} V^k_o = 1 \quad k\in K. \tag{10}$$

As explained by Cordeau et al. (2006b), formulation (1)-(9) can be improved through the introduction of valid inequalities. First, the following forcing constraints link the U_o and V^k_o variables:

$$V^k_o \le U_o \quad k\in\mathcal{K}; o\in\mathcal{O}^k. \tag{11}$$

If the same system of equivalent units is used throughout the network, one may denote by u^k the conversion factor to transform a real unit of commodity k into equivalent units.

The following constraints can then be added to ensure that sufficient capacity is provided by the selected suppliers, plants and warehouses, respectively:

$$\sum_{s\in\mathcal{S}} u_s U_s \geq \sum_{r\in\mathcal{R}} u^r \sum_{f\in\mathcal{F}} b^{rf} \sum_{c\in\mathcal{C}} a_c^f \tag{12}$$

$$\sum_{p\in\mathcal{P}} u_p U_p \geq \sum_{f\in\mathcal{F}} u^f \sum_{c\in\mathcal{C}} a_c^f \tag{13}$$

$$\sum_{w\in\mathcal{W}} u_w U_w \geq \sum_{f\in\mathcal{F}} u^f \sum_{c\in\mathcal{C}} a_c^f. \tag{14}$$

Valid inequalities (11)-(14) will be added to the formulation when computing the lower bounds used to assess the performance of the heuristic in Section 3.

1.2 A non-linear integer programming formulation

In the presence of single assignment constraints (10), model (1)-(9) can be restated using only binary variables. Indeed, because each commodity $k \in \mathcal{K}$ is assigned to a single location, the flows between the locations are entirely determined by the assignment decisions. It is thus possible to eliminate the X_{od}^k variables and to retain only the binary variables U_o and V_o^k.

For each product $f \in \mathcal{F}$, let $a^f = \sum_{c\in C} a_c^f$ denote the total demand for this product. Similarly, let $a^r = \sum_{f\in\mathcal{F}} b^{rf} a^f$ denote the total requirement for raw material $r \in \mathcal{R}$. For every $f \in \mathcal{F}$, $p \in \mathcal{P}^f$ and $w \in \mathcal{W}^f$, define

$$\tilde{c}_{pw}^f = c_{pw}^f a^f + \sum_{c\in\mathcal{C}^f} c_{wc}^f a_c^f.$$

The parameter $\tilde{c}_{pw}^f$ denotes the total cost of the flow of product f in the network if this product is assigned to plant p and warehouse w. Similarly, for every $r \in \mathcal{R}$, $f \in \mathcal{F}$, $s \in \mathcal{S}^r$ and $p \in \mathcal{P}^f$, define

$$\tilde{c}_{sp}^{rf} = c_{sp} a^f b^{rf}.$$

The parameter $\tilde{c}_{sp}^{rf}$ denotes the total cost of the flow of raw material $r \in \mathcal{R}$ in the network if this raw material is assigned to supplier s while product f is assigned to plant p.

Using this notation, the problem can now be stated as the following non-linear integer program:

Minimize

$$\sum_{o\in\mathcal{O}} c_o U_o + \sum_{k\in\mathcal{K}} \sum_{o\in\mathcal{O}^k} c_o^k V_o^k + \sum_{f\in\mathcal{F}} \sum_{p\in\mathcal{P}} \sum_{w\in\mathcal{W}} \tilde{c}_{pw}^f V_p^f V_w^f + \sum_{r\in\mathcal{R}} \sum_{s\in\mathcal{S}} \sum_{f\in\mathcal{F}} \sum_{p\in\mathcal{P}} \tilde{c}_{sp}^{rf} V_s^r V_p^f \tag{15}$$

subject to

$$\sum_{k\in\mathcal{K}} u_o^k a^k V_o^k - u_o U_o \leq 0 \qquad o \in \mathcal{O} \tag{16}$$

$$\sum_{o\in\mathcal{O}^k} V_o^k = 1 \qquad k \in \mathcal{K} \tag{17}$$

$$U_o \in \{0,1\} \quad o \in \mathcal{O} \tag{18}$$

$$V_o^k \in \{0,1\} \quad k \in \mathcal{K}; o \in \mathcal{O}^k. \tag{19}$$

This model assumes that for all $o \in \mathcal{O}^k$, $a^k \leq q_o^k$, i.e., any facility in $\mathcal{O}^k$ has a capacity q_o^k larger than or equal to the total demand a^k for commodity k. If this is not the case, one may simply remove from the set $\mathcal{O}^k$ the elements o for which $q_o^k < a^k$.

Using the transformation proposed by Frieze and Jadegar (1983) and Padberg and Rijal (1996) for the quadratic assignment problem, formulation (15)-(19) can be linearized by introducing additional binary variables. Indeed, let $Y_{pw}^f = V_p^f V_w^f$ and $Z_{sp}^{rf} = V_s^r V_p^f$. The resulting formulation is the following:

Minimize

$$\sum_{o\in\mathcal{O}} c_o U_o + \sum_{k\in\mathcal{K}}\sum_{o\in\mathcal{O}^k} c_o^k V_o^k + \sum_{f\in\mathcal{F}}\sum_{p\in\mathcal{P}}\sum_{w\in\mathcal{W}} \tilde{c}_{pw}^f Y_{pw}^f + \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}}\sum_{f\in\mathcal{F}}\sum_{p\in\mathcal{P}} \tilde{c}_{sp}^{rf} Z_{sp}^{rf} \tag{20}$$

subject to

$$\sum_{k\in\mathcal{K}} u_o^k a^k V_o^k - u_o U_o \leq 0 \qquad o \in \mathcal{O} \tag{21}$$

$$\sum_{o\in\mathcal{O}^k} V_o^k = 1 \qquad k \in \mathcal{K} \tag{22}$$

$$V_p^f = \sum_{w\in\mathcal{W}} Y_{pw}^f \quad p \in \mathcal{P}, f \in \mathcal{F} \tag{23}$$

$$V_w^f = \sum_{p\in\mathcal{P}} Y_{pw}^f \quad w \in \mathcal{W}, f \in \mathcal{F} \tag{24}$$

$$V_s^r = \sum_{p\in\mathcal{P}} Z_{sp}^{rf} \quad s \in \mathcal{S}, r \in \mathcal{R}, f \in \mathcal{F} \tag{25}$$

$$V_p^f = \sum_{s\in\mathcal{S}} Z_{sp}^{rf} \quad p \in \mathcal{P}, f \in \mathcal{F}, r \in \mathcal{R} \tag{26}$$

$$U_o \in \{0,1\} \qquad o \in \mathcal{O} \tag{27}$$

$$V_o^k \in \{0,1\} \qquad k \in \mathcal{K}; o \in \mathcal{O}^k \tag{28}$$

$$Y_{wp}^f \in \{0,1\} \qquad w \in \mathcal{W}; p \in \mathcal{P}; f \in \mathcal{F} \tag{29}$$

$$Z_{sp}^{rf} \in \{0,1\} \qquad s \in \mathcal{S}; p \in \mathcal{P}, r \in \mathcal{R}; f \in \mathcal{F}. \tag{30}$$

It is worth pointing out that because constraints (23)-(26) define the V_o^k variables in terms of the Y_{pw}^f and Z_{sp}^{rf} variables, the former may be removed from the formulation. Nevertheless, formulation (20)-(30) is tractable only for very small instances. Indeed, for an instance with $|\mathcal{R}| = |\mathcal{F}| = 20$ and $|\mathcal{S}| = |\mathcal{P}| = |\mathcal{W}| = 10$, the formulation would contain a total of 42,030 binary variables (30 U_o variables, $10^2 \cdot 20 = 2000$ Y_{pw}^f variables and $20^2 \cdot 10^2 = 40000$ Z_{sp}^{rf} variables) and 4470 constraints. In contrast, model (1)-(9) would contain only 630 binary variables, 6000 continuous variables and 1230 constraints. In the context of the closely related generalized quadratic assignment problem, the above linearization has also been shown to provide weak linear programming lower bounds (see Cordeau et al. (2006a)).

Although formulations (15)-(19) and (20)-(30) are intractable for instances of realistic size, their structure emphasizes the combinatorial nature of the problem and suggests ways to exploit this structure within local search heuristics. The notation introduced in this section will thus be used to describe our iterated local search heuristic in the next section.

2 Iterated local search heuristic

This section describes the iterated local search heuristic we have developed for the LNDPSA. We first provide a general outline of the method. We then describe its main ingredients in more detail by focusing on the neighborhoods that we use and their evaluation.

Iterated local search (ILS) is a neighborhood exploration paradigm that was introduced by Lourenço et al. (2002). Let $\hat{s}$ represent the starting solution for the ILS process. In each cycle, a diversification step is first applied by perturbing $\hat{s}$ to obtain a new solution s'. Intensification is then performed around s' by applying a descent heuristic to produce a new solution $\tilde{s}$. If $\tilde{s}$ satisfies an acceptance criterion, it replaces $\hat{s}$ and the next perturbation is performed from that solution. Otherwise, the search returns to the previous solution $\hat{s}$. The perturbations are aimed at escaping from local optima and exploring other parts of the search space. Let $f(s)$ denote the cost of solution s. The complete process can be summarized by the following pseudo-code:

1. $s_0 \leftarrow$ initial solution; $s^* \leftarrow s_0$;
2. apply local search to s_0 to obtain an improved solution $\hat{s}$;
3. while the termination criterion is not satisfied
   (a) apply a perturbation to $\hat{s}$ to obtain s',

(b) apply local search to s' to obtain a local optimum $\tilde{s}$,

(c) if $\tilde{s}$ satisfies the acceptance criterion, $\hat{s} \leftarrow \tilde{s}$;

(d) if $f(\tilde{s}) < f(s^*)$, $s^* \leftarrow \tilde{s}$;

4. return the best solution found s^*.

It is worth pointing out that the solution $\hat{s}$ perturbed in Step 3a of the algorithm is not necessarily the best solution found since the beginning of the search. Indeed, the acceptance criterion specified in Step 3c may allow solutions that do not improve upon the best known solution s^* but are sufficiently close in terms of cost.

In the following sections, we describe the main steps of the algorithm: initialization, local search, perturbation and acceptance criterion.

2.1 Initialization

Because the algorithm allows infeasible intermediate solutions (i.e., solutions violating the capacity constraints), it can be initialized with any assignment of raw materials to suppliers and finished products to plants and warehouses. In our implementation, we construct an initial solution s_0 as follows:

1. set $U_o = 0$ and $V_o^k = 0$ for all $o \in \mathcal{O}$ and $k \in \mathcal{K}$;

2. for each raw material $r \in \mathcal{R}$, randomly select a supplier s from the set $\mathcal{S}^r$ and set $V_s^r = 1$;

3. for each finished product $f \in \mathcal{F}$, randomly select a plant $p \in \mathcal{P}^f$ and a warehouse $w \in \mathcal{W}^f$ and set $V_p^f = V_w^f = 1$;

4. for each location $o \in \mathcal{O}$ such that $V_o^k = 1$ for at least one commodity $k \in \mathcal{K}$, set $U_0 = 1$.

This solution will clearly satisfy the assignment constraints (17). It may, however, violate the capacity constraints (16). The next section describes how infeasible solutions are handled in the course of the search process.

2.2 Local search

Our local search heuristic performs a descent from a starting solution s' until it reaches a local minimum $\tilde{s}$. At each iteration, this heuristic moves from the current solution s to

the best solution in its neighborhood $N(s)$. We now describe the main ingredients of this heuristic.

2.2.1 Relaxation mechanism

Infeasible solutions are allowed throughout the search by relaxing the capacity constraints (16) and penalizing their violations in the objective function. Let $f(s)$ denote the value of the objective function (15). The local search heuristic considers an augmented objective function $g(s) = f(s) + \alpha v(s)$, where $v(s)$ is the sum of the violations of the capacity constraints and α is a positive parameter equal to 10^6 in our implementation.

2.2.2 Solution neighborhood

An important consideration in the implementation of our algorithm is the fast evaluation of the solutions in the neighborhood $N(s)$ of the current solution s. To this end, we maintain data structures which are updated every time a move is performed on the current solution.

A solution can be fully characterized by the values given to the binary variables U_o and V_o^k. For every commodity $k \in \mathcal{K}$, let $o(k)$ denote the location to which this commodity is currently assigned. In other words, $o(k)$ corresponds to the unique location for which $V_o^k = 1$. Define $\mathcal{K}_o \subset \mathcal{K}$ as the set of commodities currently assigned to location o, i.e., $\mathcal{K}_o = \{k \in \mathcal{K} | V_o^k = 1\}$. For every $o \in \mathcal{O}$, let $\bar{u}_o = \sum_{k|o(k)=o} u_o^k a^k$ denote the capacity usage of location o in the current solution.

For every raw material r and every supplier $s \in \mathcal{S}^r$, let $Add(r, s)$ denote the cost of assigning r to s given the current assignments of products to plants. This value is computed as follows:

$$Add(r,s) = c_s^r + \sum_{f\in\mathcal{F}} \sum_{p\in\mathcal{P}} \tilde{c}_{sp}^{rf} V_p^f + \alpha \min\{u_s^r a^r, \max\{0, \bar{u}_s + u_s^r a^r - u_s\}\}. \tag{31}$$

The first term in this sum is the fixed cost of assigning raw material r to supplier s. The second term is the sum of the flow costs between the supplier s and all plants p that require some amount of raw material r. The third term is the penalty related to the increase in the violation of the capacity constraint at supplier s. This increase is the minimum between the capacity required for raw material r, $u_s^r a^r$, and the total violation of the capacity constraint following the assignment, $\max\{0, \bar{u}_s + u_s^r a^r - u_s\}$.

Similarly, we denote by $Add(f, p)$ the cost of assigning product f to plant p given the current assignments of raw materials to suppliers and products to warehouses. This cost is

computed as follows:

$$Add(f,p) = c_p^f + \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}} \tilde{c}_{sp}^{rf} V_s^r + \sum_{w\in\mathcal{W}} \tilde{c}_{pw}^f V_w^f + \alpha \min\{u_p^f a^f, \max\{0, \bar{u}_p + u_p^f a^f - u_p\}\}. \quad (32)$$

In this case, two sums must be considered: the cost of flows between suppliers and the plant, and the cost of flows between the plant and the warehouses.

Finally, the cost $Add(f,w)$ of assigning product f to warehouse w given the current assignment of products to plants is computed as follows:

$$Add(f,w) = c_w^f + \sum_{p\in\mathcal{P}} \tilde{c}_{pw}^f V_p^f + \alpha \min\{u_w^f a^f, \max\{0, \bar{u}_w + u_w^f a^f - u_w\}\}. \quad (33)$$

For every commodity k, we also compute in a similar fashion the cost $Rem(k,o(k))$ of removing commodity k from its current location $o(k)$.

For every commodity $k \in \mathcal{K}$, let $l(k) \in \mathcal{O}^k \setminus \{o(k)\}$ denote a location for which $Add(k,l)$ is minimized, i.e., $l(k) \in \arg\min_{l\in O^k\setminus\{o(k)\}} Add(k,l)$.

The neighborhood $N(s)$ considered in our local search is defined by the following four types of exchanges:

1. move a commodity $k \in \mathcal{K}$ from its current location $o(k)$ to a different location $o' \in \mathcal{O}^k$, i.e., set $V_{o(k)}^k = 0$ and $V_{o'}^k = 1$;
2. swap two commodities $k_1, k_2 \in \mathcal{K}$ between their respective locations, i.e., set $V_{o(k_1)}^{k_1} = V_{o(k_2)}^{k_2} = 0$ and $V_{o(k_2)}^{k_1} = V_{o(k_1)}^{k_2} = 1$;
3. close a location $o \in \mathcal{O}$ such that $|\mathcal{K}_o| = 1$ and reassign its commodity to a least-cost location, i.e., set $U_o = 0$ and for the commodity k such that $V_o^k = 1$, set $V_o^k = 0$ and $V_{l(k)}^k = 1$;
4. open a location $o \in \mathcal{O}$ and assign to it the commodity k for which the value of $Rem(k,o(k)) + Add(k,o)$ is minimal, i.e., set $U_o = 1$ and set $V_o^k = 1$ and $V_{o(k)}^k = 0$ for a commodity k in $\arg\min_{l\in\mathcal{K}} Rem(l,o(l)) + Add(l,o)$.

Restricting exchanges of type 3 to locations that are assigned at most one commodity in the current solution allows us to evaluate the impact of each exchange very quickly because this evaluation can be performed directly through the values of $Add(k,o)$ and $Rem(k,o(k))$, which are computed only once at the beginning of each neighborhood evaluation. The same reason motivates us to restrict exchanges of type 4 to the reassignment of a single commodity. More massive reassignments will be performed through the perturbation mechanisms described below.

2.3 Perturbation mechanisms

Five different mechanisms are used to perturb the current solution $\hat{s}$ and obtain a new solution s' in Step 3a of the iterated local search heuristic:

1. close a location $o \in \mathcal{O}$ and reassign each commodity k such that $o(k) = o$ to its least-cost location $l(k)$;
2. open a random location $o \in \mathcal{O}$ with no commodity;
3. open a random location $o \in \mathcal{O}$ and assign to it all commodities k such that $o \in \mathcal{O}^k$;
4. randomly reassign all commodities to open locations;
5. reconstruct the solution by applying the procedure used to generate an initial solution.

Every time a location is opened or closed through a perturbation of type 1, 2 or 3, exchanges that would undo this move are forbidden until the next perturbation is performed. For instance, if a plant is opened through a perturbation of type 2, this plant will remain open until the next perturbation is performed. It is also worth noting that although perturbations of type 1 are similar to exchanges of type 3, they are performed regardless of their impact on the augmented objective function value $g(s)$. This is obviously not the case for exchanges of type 3 which are performed only if they improve $g(s)$ in the descent phase.

Each time a perturbation is performed in Step 3a of the algorithm, a mechanism is chosen randomly (with equal probabiliy) from the set of five mechanisms listed above.

2.4 Neighborhood sampling

In the hope of accelerating the neighborhood evaluation and of performing a larger number of perturbations, only a fraction of the possible exchanges can be considered at each iteration of the local search algorithm. This may also help diversify the search. We use a simple form of neighborhood sampling in which each possible exchange is considered with a given probability π, where π is a user-controlled parameter.

2.5 Acceptance criterion

The solution $\tilde{s}$ obtained at the end of Step 3b is accepted and replaces the current solution $\hat{s}$ if its cost $g(\tilde{s})$ is smaller than or equal to $g(s^*)(1+\phi)$ where s^* is the best solution found since the beginning of the search and $\phi \geq 0$ is a user-set parameter.

2.6 Stopping criterion

The algorithm stops if it fails to improve the incumbent for η successive perturbations in Step 3 of the algorithm.

3 Computational experiments

This section first describes the instances used to evaluate the performance of the heuristic. It then presents a summary of our computational experiments.

3.1 Test instances

To assess the performance of the ILS heuristic, experiments were performed on a set of 50 randomly generated instances. These instances were generated with a procedure similar to that described by Cordeau et al. (2006b) for the general logistics network design problem. We refer the reader to the work of these authors for a complete description of the instance generator.

The size of an instance is given by the number $|\mathcal{S}|$ of suppliers, the number $|\mathcal{P}|$ of potential plant locations, the number $|\mathcal{W}|$ of potential warehouse locations, the number $|\mathcal{C}|$ of customers, the number $|\mathcal{R}|$ of raw materials, and the number $|\mathcal{F}|$ of finished products. For an instance of size n, we have set $|\mathcal{S}| = |\mathcal{P}| = |\mathcal{W}| = |\mathcal{C}| = n$ and $|\mathcal{R}| = |\mathcal{F}| = 2n$. Ten instance sizes were used in our experiments: $n = 5, 10, \ldots, 50$.

In all instances, we assume that $\mathcal{S}^r = \mathcal{S}$ for every $r \in \mathcal{R}$ and that $\mathcal{P}^f = \mathcal{P}, \mathcal{W}^f = \mathcal{W}$ for every $f \in \mathcal{F}$. In other words, all raw materials can be assigned to any of the suppliers and all finished products can be assigned to any plant and any warehouse. Similarly, $\mathcal{C}^f = C$, meaning that all customers have a positive demand for each finished product.

For each commodity $k \in \mathcal{K}$, a unit capacity usage u^k is first generated by drawing a random integer from the set $\{1, \ldots, 10\}$ according to a uniform distribution. For every origin $o \in \mathcal{O}^k$, we assume $u^k_o = u^k$. Let u be the total manufacturing capacity that is required to satisfy the demand for all products and let $\bar{u} = u/|\mathcal{P}|$. The capacity u_p of each plant $p \in \mathcal{P}$ is selected at random from the set $[\alpha\bar{u}, \beta\bar{u}]$ according to a uniform distribution. For all instances, we have set $\alpha = 1$ and $\beta = 10$. The same approach is used to generate u_o values for the suppliers and warehouses.

The number $|\mathcal{R}^f|$ of raw materials that go into each finished product is drawn randomly from the set $\{1, \ldots, |\mathcal{R}|\}$. The amount b^{rf} of raw material $r \in \mathcal{R}^f$ that goes into each unit

of finished product f is then drawn randomly from the set $\{1, \ldots, 10\}$. Finally, the values a_c^f are also drawn randomly in the set $\{1, \ldots, 10\}$ for every finished product $f \in \mathcal{F}$ and every customer $c \in \mathcal{C}^f$.

The cost structure is determined as follows. For each plant $p \in \mathcal{P}$, a fixed cost c_p is first drawn randomly in the interval $[10^5, 10^6]$ according to a uniform distribution. Next, for each product $f \in \mathcal{F}$, an average fixed cost $\bar{c}^f$ is drawn randomly in the interval $[10^4, 10^5]$. Then, for every plant $p \in \mathcal{P}^f$, a fixed cost c_p^f is drawn from the set $[\alpha\bar{c}^f, \beta\bar{c}^f]$, where $\alpha = 0.5$ and $\beta = 5$. For each warehouse $w \in \mathcal{W}$, fixed costs c_w and c_w^f are generated by using the same procedure and drawing values in $[10^4, 10^5]$ and $[10^3, 10^4]$, respectively. In the case of suppliers, the corresponding intervals are $[10^3, 10^4]$ and $[10^2, 10^3]$.

For every variable X_{od}^k, the variable cost c_{od}^k is composed of two distinct terms: the unit transportation cost of commodity k from o to d and the unit purchase, production or warehousing cost of commodity k at the origin o. For every commodity k, every origin $o \in \mathcal{O}^k$ and every destination $d \in \mathcal{D}^k$, an average unit transportation cost $\bar{t}_{od}^k$ is first generated by multiplying the Euclidean distance between o and d by a random number drawn according to a uniform distribution in the interval $[1, 10]$. For every location, Euclidean coordinates are themselves drawn randomly in the unit square $[0, 1] \times [0, 1]$. Next, for every raw material $r \in \mathcal{R}$ and every finished product $f \in \mathcal{F}$, an average unit purchase, production or warehousing cost $\bar{e}^k$ is drawn randomly in the interval $[1, 10]$. Then, for every origin $o \in \mathcal{O}^k$, a unit cost e_o^k is drawn in the interval $[\alpha\bar{e}^k, \beta\bar{e}^k]$ where $\alpha = 0.5$ and $\beta = 5.0$. Finally, the cost c_{od}^k is obtained by setting $c_{od}^k = t_{od}^k + e_o^k$.

Table 1 summarizes the main characteristics of model (1)-(9) for each instance size. A total of 50 test instances were finally obtained by generating 5 random instances of each size.

It is worth pointing out that increasing the number of customers has a very negligible impact on the computational effort required by the heuristic. Indeed, the set of customers is only considered at the beginning of the algorithm when computing the cost of assigning a product to a warehouse. This is obviously not true when solving model (1)-(9) by branch-and-bound since any increase in the number of customers leads to an increase in the number of flow variables X_{od}^k.

3.2 Summary of computational experiments

We first ran some experiments to determine appropriate values for the three parameters controlling the ILS heuristic: the sampling fraction π, the acceptance criterion ϕ, and the stopping criterion η. The best values for these parameters are of course interdependent. In

Table 1: Characteristics and size of problem instances

n	$\|\mathcal{S}\|$, $\|\mathcal{P}\|$, $\|\mathcal{W}\|$, $\|\mathcal{C}\|$	$\|\mathcal{R}\|$, $\|\mathcal{F}\|$	Number of variables U_o	V_o^k	X_{od}^k	Number of constraints
5	5	10	15	150	750	315
10	10	20	30	600	6,000	1,230
15	15	30	45	1,350	20,250	2,745
20	20	40	60	2,400	48,000	4,860
25	25	50	75	3,750	93,750	7,575
30	30	60	90	5,400	162,000	10,890
35	35	70	105	7,350	257,250	14,805
40	40	80	120	9,600	384,000	19,320
45	45	90	135	12,150	546,750	24,435
50	50	100	150	15,000	750,000	30,150

initial experiments, however, we have noticed that the performance of the heuristic is not very sensitive to the values of π and ϕ, and that using values in the interval $[0.125; 1.0]$ for π and $[0.0125; 0.10]$ for ϕ produced good results on most instances. Setting $\pi = 0.5$ and $\eta = 1000$, we first performed experiments by varying the value of ϕ. To this end, we used one instance of each size. The results of these experiments are reported in Table 2. Solution quality varies only slightly but for most instances, the best solution is obtained with $\phi = 0.025$. We have thus retained this value for all further experiments.

Table 2: Solution costs obtained with different values of ϕ

	Value of ϕ			
n	0.0125	0.025	0.05	0.1
5	1814367.84	1801620.68	1802080.05	1801620.68
10	3553332.15	3553332.15	3553332.15	3553332.15
15	5831649.09	5831649.09	5831979.04	5831649.09
20	9617584.51	9612246.79	9739113.98	9612246.79
25	12441090.95	12440863.83	12441090.95	12804686.87
30	19627137.15	19753067.43	19753067.43	19789078.90
35	27401795.54	27398144.86	27334986.03	27464642.30
40	35428372.25	35439159.14	35535641.22	35413980.24
45	48744625.49	46463159.45	46670189.51	46485241.47
50	71314197.69	71224354.81	71240726.74	71315081.83
Sum	235774152.66	233517598.23	233902207.10	234071560.32

Using $\phi = 0.025$, we have then let the value of π vary in the interval $[0.125; 1.0]$. The

results of these experiments, reported in Table 3, show that the best performance is obtained with $\pi = 0.5$. We have thus used this value in further experiments.

Table 3: Solution costs obtained with different values of π

	Value of π			
n	0.125	0.25	0.5	1.0
5	1802080.05	1814367.84	1801620.68	1801620.68
10	3553332.15	3553332.15	3553332.15	3553332.15
15	5860700.03	5832732.21	5831649.09	5860700.03
20	9621647.09	9871752.37	9612246.79	9612246.79
25	12473117.99	12485001.77	12440863.83	12442808.19
30	19661064.09	20040665.57	19753067.43	19770266.59
35	27431696.71	27402070.11	27398144.86	27411987.16
40	35487445.29	35472568.08	35439159.14	35480558.51
45	46884297.17	46836262.95	46463159.45	46620993.15
50	71546688.23	71448310.20	71224354.81	71416056.85
Sum	234322068.80	234757063.25	233517598.23	233970570.10

In the third set of experiments, we have let the value of η vary while holding the values of π and ϕ fixed. Three values were considered for η: 10, 100 and 1000. In all cases, we also impose an upper limit of 10^5 on the total number of iterations performed in order to control the CPU time. Results of these experiments are reported in Table 4. The last column of this table indicates the CPU time when using $\eta = 1000$. Computing times for $\eta = 100$ and $\eta = 10$ are roughly 10 times and 100 times smaller, respectively. As can be seen from these results, a large improvement is obtained when going from $\eta = 10$ to $\eta = 100$. The improvement is less significant for $\eta = 1000$. In column 10×100, we also report the cost of the best solution obtained by running the algorithm 10 times with $\eta = 100$, thus using a similar total CPU time to the case where $\eta = 1000$. These results show that, on average, slightly better results are obtained by performing one execution with $\eta = 1000$ than performing 10 executions with $\eta = 100$. Finally, one can observe that for the largest instance considered, the CPU time is below two hours. These computing times are acceptable considering that the LNDPSA is a strategic planning problem.

In Table 5, we also indicate for the case $\eta = 1000$ the total number of perturbations performed in step 3a of the algorithm, the total number of iterations performed in the descent phases (step 3b of the algorithm), the average number of iterations performed in each descent, and the total number of exchanges of each type (see Section 2.2.2) performed in the descent phases. These results show that for instances of size $n = 30, 40, 45$ and 50,

Table 4: Solution costs obtained with different values of η

	Value of η				CPU time
n	10	100	1000	10×100	(minutes)
5	1808948.73	1801620.68	1801620.68	1801620.68	0.00
10	3553332.15	3637887.66	3553332.15	3553332.15	0.06
15	5869631.22	5843900.40	5831649.09	5843900.40	0.72
20	9906493.78	9766667.74	9612246.79	9616115.38	3.83
25	13661890.06	12786584.84	12440863.83	12448978.35	11.04
30	20220229.65	19798728.62	19753067.43	19643015.04	12.65
35	31314231.58	27394259.39	27398144.86	27394259.39	30.03
40	36148942.69	35416322.85	35439159.14	35398397.45	28.70
45	52066643.91	46623355.91	46463159.45	46621653.02	65.95
50	72026193.77	71362327.39	71224354.81	71360956.64	94.94
Sum	246576537.54	234431655.48	233517598.23	233682228.50	

the algorithm was stopped because the total number of iterations performed reached 10^5. Considering all instances, the average number of iterations performed in each descent phase was 28.9, but this number obviously increases with instance size. The results also indicate that exchanges of types 1 and 2 (moving a commodity to a different location or swapping two commodities) are performed the most often, with exchanges of types 3 and 4 (closing or opening a facility) being less frequent. The latter results are to be expected since the number of potential exchanges of types 1 and 2 is much larger than the number of potential exchanges of types 3 and 4.

Table 5: Number of perturbations, iterations and exchanges performed with $\eta = 1000$

	Perturbation	Descent	Iterations /	Exchange type			
n	cycles	iterations	descent	1	2	3	4
5	1167	6471	5.54	3007	1267	349	680
10	1083	15063	13.91	8458	2694	1097	1730
15	1627	25973	15.96	14433	5671	1998	2243
20	2731	58383	21.38	31547	15073	4212	4819
25	2883	74762	25.93	41196	19438	6413	4831
30	3300	100000	30.30	52405	27784	9439	7193
35	2527	87116	34.47	46710	26381	6560	4937
40	2356	100000	42.44	54962	28708	9229	4806
45	2071	100000	48.29	50941	32236	9540	5376
50	1987	100000	50.33	52009	32687	8840	4580
Avg.	2173	66777	28.86	35567	19194	5768	4120

Using the best parameter settings determined above, we have then run the heuristic ten times on each of the 50 instances to study the stability of the solutions produced. Table 6 indicates, for each instance, the cost of the best and worst solutions obtained at the end of these ten executions. We also report the average solution cost and the standard deviation as a percentage of the average cost. These results show that the heuristic is very stable and that the standard deviation of the cost is on average only 0.31% of the average cost. It is also interesting to observe that the variability of the results increases only very slightly with the size of the instance.

Table 6: Results obtained by running the algorithm 10 times on each instance

Instance	Best	Worst	Average	% St.Dev.
5-1	1801620.68	1814367.84	1809268.98	0.36
5-2	1883625.86	1883625.86	1883625.86	0.00
5-3	2119688.44	2119688.44	2119688.44	0.00
5-4	2040172.94	2040172.94	2040172.94	0.00
5-5	2010293.74	2010293.74	2010293.74	0.00
10-1	3553332.15	3637887.66	3561787.70	0.75
10-2	4198965.70	4198965.70	4198965.70	0.00
10-3	3675831.47	3675831.47	3675831.47	0.00
10-4	3705195.00	3705575.69	3705309.21	0.00
10-5	3902731.51	3903547.76	3902813.14	0.01
15-1	5831649.09	5847306.70	5834505.97	0.10
15-2	5037894.82	5039768.32	5038285.64	0.02
15-3	5430692.87	5430692.87	5430692.87	0.00
15-4	5866317.96	5866317.96	5866317.96	0.00
15-5	5671336.61	5672156.84	5671992.79	0.01
20-1	9612246.79	9640694.65	9623734.55	0.12
20-2	9679158.17	9701624.15	9691482.71	0.09
20-3	9431694.38	9497659.03	9453438.95	0.32
20-4	8394520.82	8426653.48	8410683.66	0.11
20-5	8007238.59	8092871.02	8042198.80	0.40
25-1	12440863.83	12804166.91	12511484.83	1.19
25-2	12833465.03	12967270.43	12888430.09	0.42
25-3	12140851.88	12171249.77	12157844.54	0.08
25-4	12335638.64	12366276.52	12341862.91	0.10
25-5	12088834.15	12141094.62	12118362.53	0.15
30-1	19626916.20	20013946.80	19776336.23	0.69
30-2	19161622.91	19295629.21	19214311.11	0.25
30-3	17249596.14	17277853.04	17264389.01	0.05

Continued on next page

Instance	Best	Worst	Average	% St.Dev.
30-4	18626308.11	19217279.32	18740790.61	1.28
30-5	19596807.68	20090834.11	19788238.26	1.06
35-1	27314096.09	27381282.25	27343862.09	0.08
35-2	29629540.16	29903544.54	29774835.10	0.38
35-3	28725913.27	28853902.44	28796036.00	0.14
35-4	25898208.08	26611297.49	26382975.66	0.74
35-5	29684344.93	29807184.04	29729950.02	0.14
40-1	35325745.25	35539369.90	35428838.74	0.21
40-2	34696451.21	35945819.58	34952710.36	1.49
40-3	36901639.28	37680093.35	37006982.47	0.64
40-4	38305006.46	39335669.70	38470359.71	0.79
40-5	37287750.96	37896768.86	37487268.53	0.68
45-1	46384395.31	46742059.95	46595698.71	0.26
45-2	50321946.34	50546153.66	50444418.81	0.15
45-3	45835983.79	45990360.63	45951322.37	0.10
45-4	51436271.04	52302030.96	51622513.05	0.59
45-5	50653540.95	51055045.38	50932457.08	0.32
50-1	71283418.52	71412116.83	71307732.80	0.05
50-2	60701772.87	61013818.53	60827126.88	0.18
50-3	63369633.98	64417642.56	63624409.61	0.46
50-4	62518134.37	63429809.40	62773134.82	0.45
50-5	59933934.56	60159886.85	60010347.69	0.10
Avg.				0.31

Finally, to evaluate the performance of the heuristic in terms of solution quality, we have tried solving formulation (1)-(9) with the branch-and-bound of CPLEX. The CPLEX branch-and-bound was successful on instances of size $n = 5$ and $n = 10$ but failed to solve any of the larger instances when given a maximum of two hours of computing time. We then tried to solve the LP relaxation of formulation (1)-(9) for all remaining instances. For instances of size $n \geq 35$, this was again impossible with two hours of computing time.

In Table 7 we first report the value of the LP relaxation for all instances with $n \leq 30$. For instances of size $n = 5$ and $n = 10$, we also indicate the cost of the optimal integer solution computed by CPLEX (column "IP"). For instances of size $n = 15$ and $n = 20$, we stopped the branch-and-bound search after two hours of CPU time. We thus report in column "IP" the value of the best integer solution identified within that time limit. In column "H", we then indicate the cost of the best solution identified by the heuristic (this information is identical to that reported in column "Best" of Table 6). Column "Int. gap" indicates the percentage integrality gap between the optimal integer solution and the value of the LP relaxation, computed as 100(IP - LP)/LP. Column "Tot. gap" indicates the total

gap between the heuristic solution and the value of the LP relaxation. This gap is the sum of the true integrality gap and the optimality gap (both unknown quantities). Finally, column "Improv." indicates the percentage improvement of the heuristic solution with respect to the CPLEX solution. This improvement was computed as -100(H - IP)/IP.

Table 7: Comparison with LP relaxation and optimal solution values

Instance	LP	IP	H	Int. gap (%)	Tot. gap (%)	Improv. (%)
5-0	1779010.11	*1801620.68	1801620.68	1.27	1.27	0.00
5-1	1883216.60	*1883625.86	1883625.86	0.02	0.02	0.00
5-2	2119688.44	*2119688.44	2119688.44	0.00	0.00	0.00
5-3	2038262.55	*2040172.94	2040172.94	0.09	0.09	0.00
5-4	2000746.79	*2010293.74	2010293.74	0.48	0.48	0.00
10-0	3455529.77	*3553332.15	3553332.15	2.83	2.83	0.00
10-1	4080404.55	*4198965.70	4198965.70	2.91	2.91	0.00
10-2	3611933.64	*3675831.47	3675831.47	1.77	1.77	0.00
10-3	3593498.44	*3705195.00	3705195.00	3.11	3.11	0.00
10-4	3739652.91	*3902731.51	3902731.51	4.36	4.36	0.00
15-0	5487504.24	5831649.09	5831649.09		6.27	0.00
15-1	4776425.20	5037894.82	5037894.82		5.47	0.00
15-2	5011175.66	5430692.87	5430692.87		8.37	0.00
15-3	5591494.83	5866317.96	5866317.96		4.92	0.00
15-4	5110369.76	5672319.23	5671336.61		10.98	0.02
20-0	8671992.44	9897571.07	9612246.79		10.84	2.88
20-1	8894963.08	9766087.82	9679158.17		8.82	0.89
20-2	8216891.58	9554016.81	9431694.38		14.78	1.28
20-3	7440934.15	8409654.03	8394520.82		12.82	0.18
20-4	7237136.90	8057327.04	8007238.59		10.64	0.62
25-0	11417242.89		12440863.83		8.97	
25-1	11509929.09		12833465.03		11.50	
25-2	11243712.57		12140851.88		7.98	
25-3	11303032.81		12335638.64		9.14	
25-4	11001150.10		12088834.15		9.89	
30-0	17129018.41		19626916.20		14.58	
30-1	17181982.25		19161622.91		11.52	
30-2	15570310.21		17249596.14		10.79	
30-3	16668973.12		18626308.11		11.74	
30-4	17366397.93		19596807.68		12.84	

These results show that the heuristic has identified the optimal solution for all instances with $n \leq 10$. For instances of size $n = 15$ or $n = 20$, the heuristic has identified in less than 4 minutes (see Table 4) a solution that is better than or equal to the best solution

identified by CPLEX in two hours of CPU time. In fact, the heuristic found significantly better solutions for all instances of size $n = 20$. For the remaining instances, just solving the LP relaxation took a very long time and CPLEX could often not even find a feasible integer solution within the allowed CPU time. In this case, it is difficult to evaluate the quality of the solution produced by the heuristic. Looking at the results for the instances with $n \leq 10$, however, it appears that the integrality gaps grow rather quickly with instance size, going from an average of 0.37% for $n = 5$ to an average of 2.99% for $n = 10$. As a result, the total gap reported in the last column of the table is likely to be very close to the actual integrality gap. Finally, it is worth pointing out that for instances with $n = 35$, the heuristic requires only 30 minutes of computing time whereas CPLEX could not even solve the LP relaxation in two hours. The comparison is of course even more impressive for larger instances.

The model introduced in this paper assumes a single-country environment and a single-period planning horizon. It can, however, be generalized to a global context by considering aspects such as transfer pricing and cost allocation (see, e.g., Vidal and Goetschalckx, 2001). Following the work of Hamer-Lavoie and Cordeau (2006), the model could also be extended to a multiple-period horizon while taking seasonal, cyclic and safety stocks into account. Finally, additional considerations such as single sourcing constraints and multiple transporation modes with different fixed and variable costs can be handled as explained by Cordeau et al. (2006b)

4 Conclusion

This article has introduced an iterated local search heuristic for an important variant of the logistics network design problem. This heuristic is both easy to implement and fast. It is also flexible in the sense that it could be adapted to handle additional constraints such as single sourcing constraints which require each customer to receive all products from the same warehouse. Computational experiments show that the heuristic clearly outperforms the branch-and-bound of CPLEX both in computation times and solution quality.

Acknowledgements

This work was partly supported by a Strategic research grant provided by HEC Montréal, and by the Canadian Natural Science and Engineering Research Council under grants 227837-04 and 39682-05. This support is gratefully acknowledged. We are grateful to two anonymous referees for their valuable comments.

References

R. Ahuja, J. Orlin, S. Pallottino, M. Scaparra and M. Scutellà. "A multi-exchange heuristic for the single source capacitated facility location problem." Technical Report 0211, Dipartimento di Informatica, University of Pisa, 2002.

C.H. Aikens. "Facility Location Models for Distribution Planning." *European Journal of Operational Research*, **22**:263–279 (1985).

B.C. Arntzen, G.G. Brown, T.P. Harrison and L.L. Trafton. "Global Supply Chain Management at Digital Equipment Corporation." *Interfaces*, **25**(1):69–93 (1995).

G.G. Brown, G.W. Graves and M.D. Honczarenko. "Design and Operation of a Multicommodity Production/Distribution System Using Primal Goal Decomposition." *Management Science*, **33**:1469–1480 (1987).

M.A. Cohen and H.L. Lee. "Resource Deployment Analysis of Global Manufacturing and Distribution Networks." *Journal of Manufacturing and Operations Management*, **2**:81–104 (1989).

J.-F. Cordeau, M. Gaudioso, G. Laporte and L. Moccia. "A Memetic Heuristic for the Generalized Quadratic Assigment Problem." *INFORMS Journal on Computing*, **18**:433–443 (2006a).

J.-F. Cordeau, F. Pasin and M.M. Solomon. "An Integrated Model for Logistics Network Design." *Annals of Operations Research*, **144**:59–82 (2006b).

H. Delmaire, J.A. Díaz, E. Fernández and M. Ortega. "Reactive GRASP and tabu search based heuristics for the single source capacitated plant location problem." *INFOR*, **37**:194–225 (1999).

K. Dogan and M. Goetschalckx. "A primal decomposition method for the integrated design of multi-period production-distribution systems." *IIE Transactions*, **31**:1027–1036 (1999).

Z. Drezner, editor. *Facility Location*. Springer-Verlag, New York, 1995.

A. Frieze and J. Jadegar. "On the Quadratic Assignment Problem." *Discrete Applied Mathematics*, **5**:89–98 (1983).

A.M. Geoffrion and G.W. Graves. "Multicommodity Distribution System Design by Benders Decomposition." *Management Science*, **20**:822–844 (1974).

D. Ghosh. "Neighborhood Search Heuristics for the Uncapacitated Facility Location Problem." *European Journal of Operational Research*, **150**:150–162 (2003).

M. Goetschalckx, C.J. Vidal and K. Dogan. "Modeling and Design of Global Logistics Systems: A Review of Integrated Strategic and Tactical Models and Design Algorithms." *European Journal of Operational Research*, **143**:1–18 (2002).

G.C. Hadjinicola and K.R. Kumar. "Modeling Manufacturing and Marketing Options in International Operations." *International Journal of Production Economics*, **75**:287–304 (2002).

G. Hamer-Lavoie and J.-F. Cordeau. "Un modèle pour la conception d'un réseau de distribution avec localisation, affectation et stocks." *INFOR*, **44**:1–18 (2006).

A. Klose and A. Drexl. "Facility Location Models for Distribution System Design." *European Journal of Operational Research*, **162**:4–29 (2005).

A.A. Kuehn and M.J. Hamburger. "A heuristic program for locating warehouses." *Management Science*, **9**:643–666 (1963).

S. Lakhal, A. Martel, O. Kettani and M. Oral. "On the Optimization of Supply Chain Networking Decisions." *European Journal of Operational Research*, **129**:259–270 (2001).

S.D. Lapierre, A. Ruiz and P. Soriano. "Designing distribution networks: Formulations and solution heuristic." *Transportation Science*, **38**:174–187 (2004).

H.R. Lourenço, O.C. Martin and T. Stützle. "Iterated Local Search." In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, pages 321–353. Kluwer, Boston, 2002.

A. Martel. "The Design of Production-Distribution Networks: A Mathematical Programming Approach." In J. Geunes and P.M. Pardalos, editors, *Supply Chain Optimization*. Springer, New York, 2005.

M.J. Meixell and V.B. Gargeya. "Global Supply Chain Design: A Literature Review and Critique." *Transportation Research Part E*, **41**:531–550 (2005).

L. Michel and P. Van Hentenryck. "A simple tabu search for warehouse location." *European Journal of Operational Research*, **157**:576–591 (2004).

M.W. Padberg and M. P. Rijal. *Location, Scheduling, Design and Integer Programming*. Kluwer, Boston, 1996.

M. Paquet, A. Martel and G. Desaulniers. "Including Technology Selection Decisions in Manufacturing Network Design Models." *International Journal of Computer Integrated Manufacturing*, **17**:117–125 (2004).

H. Pirkul and V. Jayaraman. "Production, Transportation, and Distribution Planning in a Multi-Commodity Tri-Echelon System." *Transportation Science*, **30**:291–302 (1996).

T. Santoso, S. Ahmed, M. Goetschalckx and A. Shapiro. "A Stochastic Programming Approach for Supply Chain Network Design under Uncertainty." *European Journal of Operational Research*, **167**:96–115 (2005).

M. Sun. "Solving the Uncapacitated Facility Location Problem Using Tabu Search." *Computers & Operations Research*, **33**:2563–2589 (2006).

C.J. Vidal and M. Goetschalckx. "A Global Supply Chain Model with Transfer Pricing and Transportation Cost Allocation." *European Journal of Operational Research*, **129**:134–158 (2001).