Multilayer Variable Neighborhood Search for Two-Level Uncapacitated Facility Location Problems with Single Assignment

Bernard Gendron and Paul-Virak Khuong
Département d'Informatique et de recherche opérationnelle, CIRRELT (Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation), Université de Montréal, CP 6128, succ. Centre-ville, Montreal, Quebec, Canada H3C 3J7

Frédéric Semet
LAGIS, Ecole Centrale de Lille, Cité Scientifique - BP 48 - 59651 Villeneuve d'Ascq Cedex, France

We develop a variant of the variable neighborhood search (VNS) metaheuristic called the multilayer VNS (MLVNS). It consists in partitioning the neighborhood structures into multiple layers. For each layer *l*, a VNS defined on the associated neighborhood structures is invoked, each move being evaluated and completed by a recursive call to the MLVNS at layer *l* − 1. A specific MLVNS is developed to solve approximately a class of two-level uncapacitated facility location problems with single assignment (TUFLPS), when only mild assumptions are imposed on the cost functions. Two special cases are used to illustrate the efficiency of the MLVNS: the classical TUFLPS and a problem with modular costs derived from a real-life case. To assess the efficiency of the MLVNS, computational results on a large set of instances are compared with those obtained by slope scaling heuristic methods and by solving integer programming models using a state-of-the-art commercial solver. **© 2015 Wiley Periodicals, Inc. NETWORKS, Vol. 66(3), 214–234 2015**

Keywords: two-level uncapacitated facility location; variable neighborhood search; slope scaling heuristic; integer programming

1. INTRODUCTION

In this article, we develop solution methods for two-level uncapacitated facility location problems with single assignment constraints (TUFLPS). These problems generalize the uncapacitated facility location problem (UFLP) [22], which consists in selecting a set of depots from potential locations to minimize the sum of fixed costs associated with each depot and transportation costs between depots and customers. In TUFLPS, the single set of depot locations is substituted with two tiers of locations (major and minor depots) and the path to each customer must begin at one major depot and transit by one minor depot. In addition, each minor depot can be connected to at most one major depot; these are the single assignment constraints. These constraints appear in a number of applications, most notably in transportation [37] and telecommunications [9]. Note also that, for a large class of two-level UFLPs for which the single assignment constraints are not explicitly enforced, these constraints can be satisfied in an optimal solution due to the structure of the objective function (see Section 2 for a more detailed discussion).

We consider TUFLPS with nearly arbitrary cost functions. In the *classical TUFLPS*, the cost function includes fixed costs associated with the depots and transportation costs between nodes at each level. In addition to the classical TUFLPS, we study a variant of TUFLPS suggested to us by a multichannel retail company [13]. The company sells a wide range of products (clothes, electronic devices and appliances) via its website, mail-order catalogs and physical stores. One of the main logistical challenges it faces is to adapt its distribution system to demands that vary daily, to meet its service level guarantees (i.e., a maximum delivery period of 24 h) while minimizing total cost. Consolidation is a major concern: most delivery items are small or medium-size parcels. The company addresses this challenge through

Received June 2011; accepted June 2015
Correspondence to: B. Gendron; e-mail: Bernard.Gendron@cirrelt.ca
Contract grant sponsor: NSERC (Canada)
Contract grant sponsor: International Campus on Safety and Intermodality in Transportation
Contract grant sponsor: Nord-Pas-de-Calais Region
Contract grant sponsor: European Community
Contract grant sponsor: Regional Delegation for Research and Technology
Contract grant sponsor: French Ministry of Higher Education and Research
Contract grant sponsor: National Center for Scientific Research
DOI 10.1002/net.21626
Published online 31 July 2015 in Wiley Online Library (wileyonlinelibrary.com).

a multilevel distribution architecture with single sourcing. Its operations are based in a few fixed primary facilities (central warehouses) owned by the company. A fleet of large vehicles delivers parcels from these primary facilities to cross-docking terminals (major depots) rented by subcontractors. Medium-size vehicles depart from these terminals and transfer parcels to smaller, impromptu, cross-docking terminals (minor depots), such as parking lots, where the items are sorted into preassigned tours for final delivery to customers.

We assume that the primary facilities have been located by a prior strategic planning process and that customers are preassigned to routes. We are thus concerned with determining, for each delivery period (i.e., one day), the major depots and minor depots to use, and the path taken by the items delivered in each final tour. We model the transportation costs for individual arcs with modular costs that take into account the distance travelled and the capacity of the vehicles; a fixed cost is associated with the use of each major depot; minor depots only incur costs for each batch of parcels sorted at a given minor depot, which are also represented with modular costs. Note that this problem differs significantly from the ones considered in the literature, since the objective function involves modular costs at some of the arcs (represented by the number of vehicles used on the arcs) and at some of the nodes (corresponding to the number of product batches at the minor depots).

Multilevel location analysis is a central research theme in supply chain management [28]. Most real-life applications lead to more complex problems than those addressed in this article, since they include multiperiod and multicommodity aspects (see, for instance, [2, 17, 36]). If we restrict ourselves to two-level UFLPs (with or without single assignment constraints), there is an abundant literature as illustrated by recent surveys [20, 35]. Two main types formulations are considered: arc-based [26, 32, 33] and path-based models [4, 12, 18, 34]. Most authors propose exact solution methods based on Lagrangean relaxation [5, 27, 32, 33] and on strengthening the models with valid inequalities and facets [1, 9, 23]. Some researchers compare relaxations for the arc-based and path-based formulations [7–9, 27]. Heuristic methods based on Lagrangean or linear programming (LP) relaxations have been developed [6, 32, 33], while other heuristic approaches rely on greedy strategies or on simple neighborhood search methods based on add, drop or exchange moves [4, 29, 31, 42]. To the best of our knowledge, there are very few attempts to solve two-level UFLPs with metaheuristics. A simulated annealing method is presented in [9], while a tabu search algorithm is described in [38].

The real-life variant of the TUFLPS we consider here was introduced in [13]. The arc-based and the path-based formulations for this problem were compared, showing that the LP relaxation of the path-based model provides more effective lower bounds, but also that large-scale instances cannot be solved in a reasonable time by a state-of-the-art commercial solver. This motivates the development of heuristic methods capable of finding provably good solutions to large-scale real instances in short computing times. This article introduces such a heuristic algorithm based on a variant of the variable neighborhood search (VNS) metaheuristic [14, 30] called the *multilayer VNS* (MLVNS).

The contribution of this article is threefold. First, this article introduces the MLVNS, which consists in partitioning the neighborhood structures into multiple layers. For each layer l, a VNS defined on the associated neighborhood structures is invoked, each move being evaluated and completed by a recursive call to the MLVNS at layer $l-1$. Such a decomposition turns out to be quite natural when neighborhood structures can be classified according to their complexity. Simple neighborhood structures are assigned to the first layers, while complex neighborhood structures are divided between the last layers, given that the search in layer 1 is frequently invoked due to the recursive nature of the approach. Second, an adaptation of the MLVNS is proposed to solve efficiently large-scale instances of TUFLPS. We apply the method to the classical TUFLPS and to the TUFLPS with modular costs derived from the real application described above. Third, to assess the performance of the MLVNS, we compare it to the solution of path-based models using a state-of-the-art commercial solver, as well as to slope scaling heuristics that we developed based on the approach presented in [19]. Both the models and the slope scaling heuristics are adapted to each of the two problems we consider, the classical TUFLPS and the TUFLPS with modular costs.

The article is organized as follows. In Section 2, we describe the class of TUFLPS we consider and specify the properties of the cost functions. In Section 3, we first describe the MLVNS and then show how a heuristic method based on MLVNS can be devised for TUFLPS. Section 4 presents the path-based models and the adaptations of the slope scaling heuristics we developed to assess the performance of the MLVNS. Computational results for the classical TUFLPS and for the real-life variant with modular costs are reported in Section 5.

2. PROBLEM DESCRIPTION

We are concerned with problems that might be modeled as TUFLPS with nearly arbitrary cost functions. We denote by I and J the sets of potential major and minor depots, respectively. We denote by L the set of customers, with each customer $l \in L$ having a demand that can be expressed using several measurement units: $d_l^m > 0$, where each $m \in M$ represents a measurement unit (e.g., the industrial application, described in details below, involves two measurement units, the volume and the number of product batches). Each of these sets define a layer of nodes in a three-layered network in which major depots are connected to minor depots, and minor depots to customers. We denote by $R \subseteq I \times J$ and $P \subseteq I \times J \times L$ the sets of arcs from major depots to minor depots and of paths from major depots to minor depots to customers, respectively. Figure 1 represents such a typical network, where $R \subset I \times J$ (i.e., not all connections exist between major and minor depots) and $P \subset I \times J \times L$. For

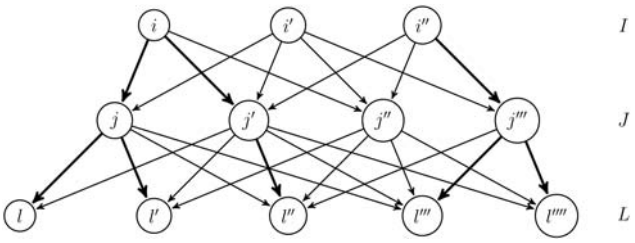

FIG. 1. Typical three-layered network (I: major depots; J: minor depots; and L: customers).

any set S, where S stands for I, J or L, we also denote by S_n the subset of nodes of S that are connected by an arc to node n. For example, in Figure 1, we have $J_i = \{j, j', j''\} \subset J$ and $L_j = \{l, l', l'', l'''\} \subset L$.

Any feasible solution corresponds to a subset of P such that each customer is assigned exactly one path and each minor depot is assigned at most one major depot, that is, a feasible solution corresponds to a forest of directed trees rooted at major depots. In the typical network of Figure 1, the arcs displayed in bold represent a solution that contains major depots i and i'', as well as minor depots j, j', and j''' (i.e., depots i' and j'' are closed in this solution). The corresponding forest consists of two directed trees, one rooted at depot i and the other rooted at depot i''. Any feasible solution induces a flow vector $\vec{x} = (x^m)_{m\in M}$ that can be decomposed as follows: (1) $\vec{x}_{ijl} = (x^m_{ijl})_{m\in M}$: the flow vector on path $(i,j,l) \in P$, with each component x^m_{ijl} equal to d^m_l if (i,j,l) belongs to the feasible solution, and to 0, otherwise; (2) $\vec{x}_{ij} = (x^m_{ij})_{m\in M}$: the flow vector on arc $(i,j) \in R$ from major depot $i \in I$ to minor depot $j \in J$, with each component defined as $x^m_{ij} \equiv \sum_{l\in L_j} x^m_{ijl}$; (3) $\vec{x}_j = (x^m_j)_{m\in M}$: the flow vector through minor depot $j \in J$, with each component defined as $x^m_j \equiv \sum_{i\in I_j} x^m_{ij}$; and (4) $\vec{x}_i = (x^m_i)_{m\in M}$: the flow vector through major depot $i \in I$, with each component defined as $x^m_i \equiv \sum_{j\in J_i} x^m_{ij}$.

We assume that the objective function to minimize is separable into a sum of functions of the flow passing through each node, each arc and each path:

$$f(\vec{x}) = \sum_{i\in I} H^1_i(\vec{x}_i) + \sum_{(i,j)\in R} G^1_{ij}(\vec{x}_{ij}) + \sum_{j\in J} H^2_j(\vec{x}_j) + \sum_{(i,j,l)\in P} G^2_{ijl}(\vec{x}_{ijl}),$$

where H^1_i and H^2_j are the location cost functions associated to each major depot $i \in I$ and to each minor depot $j \in J$, respectively; G^1_{ij} and G^2_{ijl} are the transportation cost functions associated to each depot–depot arc $(i,j) \in R$ and to each path $(i,j,l) \in P$, respectively. Each of these functions is nonnegative and its value at 0 is 0.

A common class of objective functions allows us to impose or relax the single assignment constraints without affecting the optimal value, as has been previously [9, 13] demonstrated for two specific objective functions. Objective functions in this class must satisfy the following three conditions:

1. Functions $H^1_i()$, $G^1_{ij}()$ and $G^2_{ijl}()$ depend only on the flow expressed in terms of one measurement unit. Hence, we use the notations $H^1_i(x_i)$, $G^1_{ij}(x_{ij})$, and $G^2_{ijl}(x_{ijl})$.
2. For each path $(i,j,l) \in P$, the pathwise cost function G^2_{ijl} can be decomposed by arc:
$$G^2_{ijl}(x_{ijl}) = (c^1_{ij} + c^2_{jl})x_{ijl},$$
where c^1_{ij} and c^2_{jl} are nonnegative costs for depot–depot arc and depot-customer arc, respectively.
3. The following inequality is satisfied, for any pair of major depots $i, i' \in I$ and for any minor depot $j \in J_i \cap J_{i'}$:
$$H^1_i(x_i) + H^1_{i'}(x_{i'}) + G^1_{ij}(x_{ij}) + G^1_{i'j}(x_{i'j}) \geq H^1_i(x_i + x_{i'j}) + H^1_{i'}(x_{i'} - x_{i'j}) + G^1_{ij}(x_{ij} + x_{i'j}),$$
where $x_i \geq x_{ij} > 0$ and $x_{i'} \geq x_{i'j} > 0$.

The second condition on the per-path costs is usually satisfied by practical transportation costs. It is a necessary condition: in its absence, even a trivial problem without any other cost would exhibit instances for which the single assignment constraints are violated by all optimal solutions. The third condition implies that consolidation is always profitable at the depot–depot level. In other words, given a minor depot $j \in J$ that is used to satisfy some customer demands, it is always more valuable to route these demands from a single major depot $i \in I$, instead of from two major depots $i, i' \in I$.

Theorem 1. *Assume conditions 1, 2, and 3 are satisfied. If we relax the single assignment constraints, there exists an optimal solution to the relaxed problem for which the single assignment constraints are satisfied.*

Proof. Assume we are given an optimal solution to the relaxed problem that induces a flow x, expressed in terms of the measurement unit used to write functions $H^1_i()$, $G^1_{ij}()$, and $G^2_{ijl}()$, for which there is at least one pair of arcs (i,j), (i',j) such that $x_{ij} > 0$ and $x_{i'j} > 0$, called conflicting arcs (with respect to the single assignment constraints). We show this solution can be modified into another, still optimal, solution with one less such conflicting arc. Without loss of generality, we assume that $c^1_{ij} \leq c^1_{i'j}$. We define a feasible solution x' obtained from x by reducing by $x_{i'j}$ the flow on (i',j), so that $x'_{i'j} = 0$, $x'_{i'} = x_{i'} - x_{i'j}$, and $x'_{i'jl} = 0$, $l \in L_j$, and by augmenting it by the same amount on (i,j), so that $x'_{ij} = x_{ij} + x_{i'j}$, $x'_i = x_i + x_{i'j}$, and $x'_{ijl} = x_{ijl} + x_{i'jl}$, $l \in L_j$. As a result, the nonpositive amount (by optimality of x) by which the objective function is modified simplifies to:

$$H^1_i(x_i) + H^1_{i'}(x_{i'}) + G^1_{ij}(x_{ij}) + G^1_{i'j}(x_{i'j}) + \sum_{l\in L_j} (G^2_{ijl}(x_{ijl}) + G^2_{i'jl}(x_{i'jl}))$$

$$- (H_i^1(x_i') + H_{i'}^1(x_{i'}') + G_{ij}^1(x_{ij}') + \sum_{l \in L_j} G_{ijl}^2(x_{ijl}')) \leq 0,$$

which after simplifications, can be divided into two terms:

$$H_i^1(x_i) + H_{i'}^1(x_{i'}) + G_{ij}^1(x_{ij}) + G_{i'j}^1(x_{i'j}) - ((H_i^1(x_i') + H_{i'}^1(x_{i'}') + G_{ij}^1(x_{ij}'))$$

and

$$\sum_{l \in L_j} (G_{ijl}^2(x_{ijl}) + G_{i'jl}^2(x_{i'jl})) - \left(\sum_{l \in L_j} G_{ijl}^2(x_{ijl}') \right).$$

The first term is nonnegative by condition 3. Using condition 2, the second term can be simplified to:

$$\sum_{l \in L_j} (c_{i'j}^1 - c_{ij}^1) x_{i'jl},$$

which is nonnegative by assumption. Hence, the difference in the objective function values between the optimal solution x and x' is nonnegative, which implies that x' is also optimal. By applying the same argument a finite number of times, we eventually eliminate all conflicting arcs and obtain an optimal solution that does not violate the single assignment constraints. ∎

In this article, we consider the classical TUFLPS, as well as an industrial variant of the TUFLPS with modular costs (a well-studied cost structure in the literature on facility location, see for instance [10, 11, 15, 16]). With respect to the classical TUFLPS, the objective function is defined as follows. All cost functions can be expressed in terms of one measurement unit, including $H_j^2()$, so the flows x are written as scalars and the demand for customer l is denoted d_l. Let $h_i^1 \geq 0$ be the fixed cost associated with each major depot location $i \in I$ and $h_j^2 \geq 0$ the fixed cost associated with each minor depot location $j \in J$. The location costs for major depot $i \in I$ and minor depot $j \in J$, respectively, are then given by:

$$H_i^1(x_i) = \begin{cases} h_i^1, & \text{if } x_i > 0, \\ 0, & \text{otherwise,} \end{cases}$$

$$H_j^2(x_j) = \begin{cases} h_j^2, & \text{if } x_j > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Cost functions $G_{ij}^1()$ on depot–depot arcs $(i,j) \in R$ are set to zero. Condition 3 is satisfied by these cost functions. In addition, we assume that condition 2 is satisfied, as in most references in the literature on two-level UFLPs. As a consequence, the single assignment constraints might be imposed without affecting the optimal value.

The industrial variant of the TUFLPS with modular costs might be modeled in our framework by associating cross-docking terminals to major depots, smaller terminals to minor depots and tours to customers. We use two measurement units, the volume and the number of product batches. Functions $H_i^1()$, $G_{ij}^1()$, and $G_{ijl}^2()$ depend only on the volume, while functions $H_j^2()$ depend only on the number of product batches. We use the notations x and x^p to represent the volumetric flows and the number of product batches, respectively, with the demand for customer l being denoted d_l and d_l^p for the two respective measurement units. Thus, $H_i^1(x_i)$, $G_{ij}^1(x_{ij})$, $H_j^2(x_j^p)$, and $G_{ijl}^2(x_{ijl})$ denote the values taken by the functions. Further, for each major depot $i \in I$, let e_i^0 be the cost of operating one large-size vehicle from the closest primary facility to depot $i \in I$, h_i^1 the fixed cost for using that depot and V^0 the volumetric capacity of one large-size vehicle. The location cost for the use of depot $i \in I$ is then given by:

$$H_i^1(x_i) = \begin{cases} h_i^1 + e_i^0 \lceil x_i / V^0 \rceil, & \text{if } x_i > 0, \\ 0, & \text{otherwise.} \end{cases}$$

For each arc $(i,j) \in R$ from a major depot $i \in I$ to a minor depot $j \in J$, let e_{ij}^1 be the cost of operating one medium-size vehicle and V^1 be the volumetric capacity of one medium-size vehicle. The transportation cost on that arc is

$$G_{ij}^1(x_{ij}) = e_{ij}^1 \left\lceil \frac{x_{ij}}{V^1} \right\rceil.$$

The cost of sorting parcels at each minor depot $j \in J$ can also be represented by modular costs. Let B be the number of parcels that can be handled in one batch of sorting, and b_j the cost of sorting one such batch. The location cost function for minor depot $j \in J$ is then

$$H_j^2(x_j^p) = b_j \left\lceil \frac{x_j^p}{B} \right\rceil.$$

Finally, the transportation cost for each path $(i,j,l) \in P$ is simply proportional to the unit cost c_{jl}^2 for starting each tour $l \in L_j$ at minor depot $j \in J$:

$$G_{ijl}^2(x_{ijl}) = c_{jl}^2 x_{ijl}.$$

Note that condition 2 is satisfied, but that condition 3 is not always satisfied by the modular costs $H_i^1()$ and $G_{ij}^1()$. Hence, the single assignment constraints must be imposed in the TUFLPS with modular costs that we consider.

3. MULTILAYER VARIABLE NEIGHBORHOOD SEARCH

The VNS metaheuristic [14, 30] provides a systematic approach for exploiting multiple neighborhood structures for problem (P): $\min_{x \in X \subseteq S} \{f(x)\}$, where X is the set of feasible solutions and S is the search space. Starting from the best known feasible solution $x \in X$ and assuming there are $k_{\max}$ neighborhood structures, VNS first orders them from 1 to $k_{\max}$ and then scans them iteratively in that order. At every iteration k, a new solution $x' \in S$ is generated in the current neighborhood $N_k(x)$ (this solution might then be further

optimized using some search algorithm). Then, there are two cases: (1) $f(x') < f(x)$, in which case the search restarts from x', which is now the best known feasible solution, and is recentered toward the first neighborhood, that is, we set k to 1; (2) $f(x') \geq f(x)$, in which case the attempt to improve solution x using $N_k(x)$ has failed and we move on to the next neighborhood $N_{k+1}(x)$, unless $k+1$ exceeds $k_{\max}$, in which case the loop over k is stopped. Typically, there are randomized components in the selection of x' in $N_k(x)$. This is why this loop is embedded into an outer loop that controls the overall computational effort. Algorithm 1 summarizes the method.

Algorithm 1 VNS(x: current solution)

```
1: Order the k_max neighborhood structures: N_1, ..., N_{k_max}
2: repeat
3:    k ← 1
4:    repeat
5:       Generate x' ∈ N_k(x)
6:       if f(x') < f(x) then
7:          x ← x'
8:          k ← 1
9:       else
10:         k ← k + 1
11:      end if
12:   until k > k_max
13: until some termination condition is satisfied
```

The solution x given as input to the algorithm is typically obtained by some greedy constructive procedure, but can also be derived from more sophisticated methods. In Step 5, one might generate x' at random (the so-called "shaking" step), or find the best solution in $N_k(x)$. When x' is generated at random, one obtains the so-called "reduced" VNS, while if x' is the best solution in $N_k(x)$, one obtains the variable neighborhood descent metaheuristic. Between these two extreme approaches, there is a wide spectrum of possibilities: for instance, find the best solution in a subset of $N_k(x)$ (often randomly generated), or scan the solutions in $N_k(x)$ in some order (often randomly generated) until a first improving solution is found. Often, the final x' is obtained after performing a search algorithm, which can be inspired by VNS itself or by any other metaheuristic, or even by an exact method derived from integer programming or constraint programming. The termination condition of the algorithm typically controls the overall computational effort by imposing limits on the number of iterations or the total time.

The MLVNS (MLVNS) is a variant of VNS based on dividing the neighborhood structures into $l_{\max}$ layers, each layer l having $k^l_{\max}$ associated neighborhood structures. The layers are ordered from 1 to $l_{\max}$ and then scanned in that order. Typically, as in the standard VNS, the neighborhood structures grow in complexity with the layer index. For each layer l, a VNS is invoked, with the generation of x' in a neighborhood $N_k(x)$ consisting in scanning the solutions in a subset of $N_k(x)$ in some (randomly generated) order until a first improving solution is found. To define the subset of $N_k(x)$, we use $m^k_{\max}$ (randomly ordered) subneighborhood structures, resulting from a "natural" decomposition of N_k: $N^1_k, \ldots, N^{m^k_{\max}}_k$. At each step m, one move in $N^m_k(x)$ is performed to generate a candidate solution x'. If $l = 1$, we assume the evaluation of this candidate solution is easy. Otherwise, when $l > 1$, further optimization is necessary to evaluate the impact of the move; for that purpose, MLVNS is called recursively up to layer $l-1$ to produce a final candidate solution x'. The generation of candidate solutions continues until x' improves upon x or $m > m^k_{\max}$, that is, all subneighborhood structures have been explored. The remaining steps are identical to those performed by the standard VNS. Algorithm 2 summarizes the approach.

Algorithm 2 MLVNS(x: current solution, $l_{\max}$: number of layers)

```
1: for l ← 1 to l_max do
2:    Order the k^l_max neighborhood structures: N_1, ..., N_{k^l_max}
3:    repeat
4:       k ← 1
5:       repeat
6:          Order (randomly) the m^k_max sub-neighborhood
            structures: N^1_k, ..., N^{m^k_max}_k
7:          m ← 1
8:          repeat
9:             Generate x' ∈ N^m_k(x)
10:            MLVNS(x', l − 1)
11:            m ← m + 1
12:         until (f(x') < f(x)) OR (m > m^k_max)
13:         if f(x') < f(x) then
14:            x ← x'
15:            k ← 1
16:         else
17:            k ← k + 1
18:         end if
19:      until k > k^l_max
20:   until some termination condition is satisfied
21: end for
```

As for VNS, the initial solution x given as input to the algorithm is often obtained by a greedy constructive procedure. The number of layers $l_{\max}$ and the order in which they will be traversed is also determined in the initialization phase. The search in layer $l > 1$ makes use of the search for the previous layers to evaluate any candidate solution x'. Such a strategy can only be efficient if the complexity of the neighborhood structures increases with the layer index, since the search in layers 1 to l is repeated for all subsequent layers $l+1$ to $l_{\max}$. MLVNS formalizes two approaches that are commonly used in the metaheuristics literature. First, when there are neighborhood structures for which the exact evaluation of each move is so complex that it could be advantageous to perform a heuristic evaluation instead. In that case, the MLVNS

framework represents the situation where this heuristic evaluation is obtained by calling MLVNS itself on previous layers. A second situation occurs when a diversification method is used to complement a VNS, where the diversification steps themselves can be represented by a VNS. We will see examples of these two situations in the heuristic method that we propose to solve our generalized class of problems. It is a three-layer MLVNS approach: layer 1 represents a VNS with simple neighborhood structures; layer 2 is an MLVNS based on complex neighborhood structures for which each move is evaluated using the VNS of layer 1; layer 3 can be seen as a diversification approach that perturbs the current solution with techniques that can be assimilated to particular moves in large-scale neighborhood structures, each move being completed by performing the MLVNS of layer 2.

To simplify the description of the MLVNS heuristic for our problem, we present a *generic neighborhood search* (GNS) procedure that is used in all subsequent developments. Given a solution x and a nonempty finite neighborhood $N(x)$, the procedure outputs a solution $x' \in N(x) \cup \{x\}$ according to a search strategy that depends on the values of two input parameters: STOP, which determines when to stop the search in $N(x)$, and Δ, which specifies if x' should improve, or not, upon x. The procedure is described in Algorithm 3.

Algorithm 3 GNS(x: current solution, $N(x)$: neighborhood of x, (STOP, Δ), x': new solution)

$x' \leftarrow x$
repeat
  Generate $x'' \in N(x)$ and remove x'' from $N(x)$
  $\Delta(x'', x) \leftarrow f(x'') - f(x)$
  if $\Delta(x'', x) < \Delta$ **then**
    $x' \leftarrow x''$
    $\Delta \leftarrow \Delta(x'', x)$
  end if
until (STOP) OR ($N(x) = \emptyset$)

Four main variants of the GNS procedure are defined by the values of the input parameters STOP and Δ:

1. *best*: finding the best, but not necessarily improving, solution in $N(x)$ is obtained by setting STOP = *false* and $\Delta = \infty$;
2. *best improving*: finding the best improving solution in $N(x)$ (and returning x if no such solution exists) is obtained by setting STOP = *false* and $\Delta = 0$;
3. *first improving*: by setting STOP = ($\Delta < 0$) and $\Delta = 0$, the search stops when a first improving solution in $N(x)$ is found, but if no such solution exists, x is returned;
4. *first improving or best*: by setting STOP = ($\Delta < 0$) and $\Delta = \infty$, the search stops when a first improving solution in $N(x)$ is found, but if no such solution exists, the best solution in $N(x)$ is returned.

In the remainder of the text, these four terms will be used to designate the appropriate combination of values of the two input parameters STOP and Δ.

Before providing the details of the three layers of our MLVNS heuristic, we now describe the structure of feasible solutions to our problem and then present the greedy constructive procedure used to generate the initial solution.

3.1. Solution Space

The structure of any feasible solution x to our problem is a forest containing all customers, and only the open minor depots $J(x)$ and open major depots $I(x)$. Each tree in this forest is rooted at major depot i, which is connected to the open minor depots assigned to major depot i, denoted $J_i(x)$, and each of these open minor depots, say j, is itself connected to the customers assigned to minor depot j, denoted $L_j(x)$. Computing the objective function value $f(x)$ of any such solution x is an easy task, although there are many elements defining the cost: the objective function is decomposed into a sum of functions that are only affected by changes local to a node (customer, minor depot or major depot). We exploit the forest structure of any feasible solution, and the ease in computing the cost of such solution, in the MLVNS method developed to solve TUFLPS.

3.2. Initial Solution by a Greedy Constructive Procedure

At every iteration of the greedy constructive procedure, we are given an infeasible current solution x, that is, some customers are not connected in x. An iteration starts by selecting one such unconnected customer, say l, and by exploring the CONNECTL(l) neighborhood, which contains all solutions that connects l to a depot–depot pair. The CONNECTL(l) neighborhood is explored using the *best* variant; hence, it finds the best way to connect customer l to any pair of minor and major depots, given the cost of the current infeasible solution. Note that connecting customer l to any pair of minor and major depots may lead to assign l to an open minor depot, or to open a minor depot. Also, when a minor depot is open, it may be assigned to an open major depot or it may lead to open a new major depot. In the variants of TUFLPS we consider, multiple components of the cost functions are proportional to the demand at each customer. To select customer l, we simply use the observation that customers with higher demand have more impact on the objective function value. Hence, we initially sort the customers in nonincreasing order of demand (ties being broken randomly) and scan the customers in that order, each time exploring the CONNECTL(l) neighborhood. The greedy constructive procedure is summarized in Algorithm 4.

Algorithm 4 Greedy(x: solution)

Let x be the empty solution (no customers are connected yet)
for all $l \in L$ (in non-increasing order of demand d_l) **do**
  GNS(x, CONNECTL(l), *best*, x')
  $x \leftarrow x'$
end for

3.3. Layer 1: VNS with Exchange and Close Neighborhoods

Four neighborhood structures are used at layer 1 of the MLVNS method: EXCHANGEL, EXCHANGEJ, CLOSEJ and CLOSEI.

In the EXCHANGEL neighborhood structure, the subneighborhood structures are defined by scanning the set of customers in random order. Then, for each customer l, the EXCHANGEL(l) neighborhood is explored: it consists in reassigning customer l, currently assigned to minor depot j, to another open minor depot j'. The EXCHANGEL(l) neighborhood is explored using the *first improving* variant of GNS, with the candidate open minor depots j' being scanned in nondecreasing order of transportation cost (i.e., distance, in most contexts) to l. As soon as an improving solution is found, it is selected as the new current solution, but if no improving solution is obtained, the current solution remains so. Note that, when reassigning customer l from minor depot j to a different minor depot, we may close j (as well as the major depot connected to j), but this is easily taken into account when computing the impact of the move on the objective function. To summarize, steps 6 to 12 of MLVNS are performed as in Algorithm 5 for the EXCHANGEL neighborhood structure.

Algorithm 5 EXCHANGEL(x: current solution)

1: $L' \leftarrow L$
2: **repeat**
3: Randomly select $l \in L'$ and remove l from L'
4: GNS(x, EXCHANGEL(l), *first improving*, x')
5: **until** $(f(x') < f(x))$ OR $(L' = \emptyset)$

In the EXCHANGEJ neighborhood structure, the subneighborhood structures are obtained by scanning the set of open minor depots $J(x)$ in random order. For each open minor depot j, the EXCHANGEJ(j) neighborhood is explored: it consists in reassigning minor depot j, currently assigned to major depot i, to another open major depot i'. The EXCHANGEJ neighborhood is explored using the *first improving* variant of GNS, with the candidate open major depots being scanned in increasing order of transportation costs for the customers linked to j. Steps 6 to 12 of MLVNS are performed as in Algorithm 6 for the EXCHANGEJ neighborhood structure.

Algorithm 6 EXCHANGEJ(x: current solution)

1: $J' \leftarrow J(x)$
2: **repeat**
3: Randomly select $j \in J'$ and remove j from J'
4: GNS(x, EXCHANGEJ(j), *first improving*, x')
5: **until** $(f(x') < f(x))$ OR $(J' = \emptyset)$

The subneighborhood structures in the CLOSEJ neighborhood structure are derived from scanning the set of open minor depots $J(x)$ in random order. For each open satellite j, the CLOSEJ(j) neighborhood is explored by applying the EXCHANGEL(l) neighborhood search for all $l \in L_j(x)$, that is, the customers connected to satellite j in solution x. The exploration of the EXCHANGEL(l) neighborhood is performed using the *first improving or best* variant of the GNS procedure, that is, as soon as customer l can be reassigned to another open minor depot by improving the cost, the move is performed, but otherwise, the best way to reassign customer l to another open minor depot is performed. This approach ensures that all customers currently assigned to minor depot j are reassigned to another minor depot, which effectively closes minor depot j. As in the greedy constructive procedure, the customers in $L_j(x)$ are scanned in nonincreasing order of demand (with ties broken randomly). Algorithm 7 summarizes steps 6 to 12 of MLVNS when the CLOSEJ neighborhood structure is explored.

Algorithm 7 CLOSEJ(x: current solution)

1: $J' \leftarrow J(x)$
2: **repeat**
3: Randomly select $j \in J'$ and remove j from J'
4: $L' \leftarrow L_j(x)$
5: $x'' \leftarrow x$
6: **for all** $l \in L'$ (in non-increasing order of demand d_l) **do**
7: GNS(x'', EXCHANGEL(l), *first improving or best*, x')
8: $x'' \leftarrow x'$
9: **end for**
10: **until** $(f(x') < f(x))$ OR $(J' = \emptyset)$

The CLOSEI neighborhood structure is explored in a similar way. The subneighborhood structures are obtained by scanning the set of open major depots $I(x)$ in random order. For each open major depot i, the CLOSEI(i) neighborhood is explored by searching the EXCHANGEJ(j) neighborhood for all $j \in J_i(x)$, that is, the set of open minor depots connected to major depot i in solution x. Again, the EXCHANGEJ(j) neighborhood is explored using the *first improving or best* variant of the GNS procedure, to make sure that all minor depots connected to major depot i are reassigned, so that major depot i is effectively closed. The EXCHANGEJ(j) neighborhoods are scanned in nonincreasing order of the total demand x_j satisfied through minor depot j in the current solution x. Steps 6 to 12 of MLVNS are performed as in Algorithm 8 when the CLOSEI neighborhood structure is explored.

The four neighborhood structures are explored at layer 1 of the MLVNS algorithm in the following order: EXCHANGEL, CLOSEJ, CLOSEI and EXCHANGEJ. The idea is to perform EXCHANGEL moves as much as possible, before trying the more computationally intensive CLOSEJ and CLOSEI moves, which will also significantly modify the current solution. The less effective EXCHANGEJ neighborhood is explored as a last resort in case the other types of moves did not succeed in improving the solution. Preliminary computational

Algorithm 8 CLOSEI(x: current solution)

1: $I' \leftarrow I(x)$
2: **repeat**
3: Randomly select $i \in I'$ and remove i from I'
4: $J' \leftarrow J_i(x)$
5: $x'' \leftarrow x$
6: **for all** $j \in J'$ (in non-increasing order of total demand x_j) **do**
7: GNS(x'', EXCHANGEJ(j), *first improving or best*, x')
8: $x'' \leftarrow x'$
9: **end for**
10: **until** $(f(x') < f(x))$ OR $(I' = \emptyset)$

Algorithm 9 OPENI(x: current solution)

1: $I' \leftarrow I \setminus I(x)$
2: **repeat**
3: Randomly select $i \in I'$ and remove i from I'
4: GNS(x, OPENI(i), *best*, x')
5: MLVNS(x', 1)
6: **until** $(f(x') < f(x))$ OR $(I' = \emptyset)$

Algorithm 10 OPENJ(x: current solution)

1: $J' \leftarrow J \setminus J(x)$
2: **repeat**
3: Randomly select $j \in J'$ and remove j from J'
4: GNS(x, OPENJ(j), *best*, x')
5: MLVNS(x', 1)
6: **until** $(f(x') < f(x))$ OR $(J' = \emptyset)$

experiments have confirmed the effectiveness of this particular order of exploration. Layer 1 is completed as soon as one loop over the four neighborhood structures did not succeed in improving the current solution, that is, the stopping condition in Step 20 of MLVNS is set to *true*.

3.4. Layer 2: VNS with Open Neighborhoods

At layer 2 of the MLVNS algorithm, two neighborhood structures are used: OPENI and OPENJ.

In the OPENI neighborhood structure, the subneighborhood structures are obtained by scanning the set of closed major depots $I \setminus I(x)$ in random order. For each closed major depot i, it performs the OPENI(i) move, which connects to major depot i several, but not necessarily all, open minor depots in $J(x) \cap J_i$. If *all* such open minor depots were connected to i, some open major depot would not be connected to any minor depot. To avoid this situation, minor depots in $J(x) \cap J_i$ are connected to major depot i, but if an open major depot i' would be bereft of minor depots, one (arbitrarily chosen) minor depot in $J(x) \cap J_i$ instead remains connected to i'. It is only after applying MLVNS at layer 1 to complete the move that the final configuration of open major depots will be determined.

The OPENJ neighborhood structure is explored in a similar way. The subneighborhood structures are derived by scanning the set of closed minor depots $J \setminus J(x)$ in random order. For each closed minor depot j, it performs the OPENJ(j) move by connecting to j most customers in L_j. However, if any open minor depot j' would be "closed" (i.e., left unconnected to any customer), one arbitrarily chosen customer in L_j remains assigned to j'.

Steps 6 to 12 of MLVNS are performed as follows when the OPENI (Algorithm 9) and OPENJ (Algorithm 10) neighborhoods are explored. Note that OPENI(i) and OPENJ(j) can both be seen as neighborhoods containing only one solution; hence, we use GNS with the *best* variant to represent the corresponding moves.

The two neighborhood structures are explored in the following order: OPENI and OPENJ. The motivation is that OPENI perturbs the solution more significantly than OPENJ, that is, more customers are reassigned when exploring the OPENI neighborhood than when exploring the OPENJ neighborhood. Hence, OPENI tends to diversify the search better than OPENJ and should therefore be performed first. Preliminary computational experiments have confirmed this intuition. Layer 2 is completed when one loop over the two neighborhood structures did not succeed in improving the current solution, that is, the stopping condition in Step 20 of MLVNS is set to *true*.

3.5. Layer 3: Diversification Based on Cost Perturbation

The diversification techniques used at layer 3 of the MLVNS algorithm are based on the following steps: (1) increase the costs associated to subsets of open major and minor depots in solution x, thus attempting to close these depots; (2) starting from x, solve the resulting perturbed problem by performing the MLVNS algorithm at layer 2, thus obtaining a new solution x'; (3) restore the original costs and perform the MLVNS algorithm at layer 2, starting from x'. Steps (1) and (2) correspond to performing one move in a subneighborhood structure, that is, step 9 of the MLVNS algorithm, while step (3) is the recursive call to MLVNS, that is, step 10 of the MLVNS algorithm. Finally, the greedy constructive procedure is used to reinitialize the search when no improvement has been made for some number of consecutive iterations (we use 10 in our experiments). In that case, after step (1), the current solution x is fully replaced with the output of the constructive procedure. This random restart seems particularly useful on small and difficult instances. Note that such cost perturbations are frequently used in metaheuristic approaches such as, for instance, guided local search [39–41] and iterated local search [24, 25].

Four types of cost perturbations are performed, each type being assimilated to a neighborhood structure. The PERTURBNODESJ (PERTURBNODESI) neighborhood structure randomly selects random numbers of open minor (major) depots, and multiplies by a large value M(location) the location cost function of these depots. The PERTURBARCSJ (PERTURBARCSI) neighborhood structure also randomly selects random numbers of open minor (major) depots. When

minor depots are chosen, the transportation costs for all paths going through these depots are multiplied by a large factor $M(\text{transport})$. If, instead, major depots are chosen, the transportation costs on the arcs from these major depots to all connected minor depots are multiplied by the same factor $M(\text{transport})$. Extreme cost perturbations are used instead of explicitly forbidding paths and depots, to preserve feasibility, of the instance itself and of the current solution. Algorithms 11 to 14 summarize steps 6 to 12 of MLVNS for the four neighborhood structures.

Algorithm 11 PERTURBNODESJ(x: current solution)

1: $x' \leftarrow x$
2: Randomly generate a number *NPerturbJ* in $[\lceil p_1|J(x)|\rceil, \lceil p_2|J(x)|\rceil]$
3: Randomly select *NPerturbJ* open minor depots and, for each such $j \in J$, multiply its location cost by $M(\textit{location})$
4: MLVNS(x', 2)
5: Restore all cost functions to their original expressions
6: MLVNS(x', 2)

Algorithm 12 PERTURBNODESI(x: current solution)

1: $x' \leftarrow x$
2: Randomly generate a number *NPerturbI* in $[\lceil p_1|I(x)|\rceil, \lceil p_2|I(x)|\rceil]$
3: Randomly select *NPerturbI* open major depots and, for each such $i \in I$, multiply its location cost by $M(\textit{location})$
4: MLVNS(x', 2)
5: Restore all cost functions to their original expressions
6: MLVNS(x', 2)

Algorithm 13 PERTURBARCSJ(x: current solution)

1: $x' \leftarrow x$
2: Randomly generate a number *NPerturbJ* in $[\lceil p_1|J(x)|\rceil, \lceil p_2|J(x)|\rceil]$
3: Randomly select *NPerturbJ* open minor depots and, for each chosen minor depot $j \in J$, multiply all transportation costs for paths passing through j by $M(\textit{transport})$
4: MLVNS(x', 2)
5: Restore all costs to their original values
6: MLVNS(x', 2)

The order in which these four neighborhood structures are explored is PERTURBARCSJ, PERTURBNODESJ, PERTURBNODESI, PERTURBARCSI. This order is motivated by the same reasons that justify the order selected for layer 1: the PERTURBARCSJ and PERTURBARCSI neighborhood structures are similar to the EXCHANGEL and EXCHANGEJ neighborhood structures, while PERTURBNODESJ and PERTURBNODESI are similar to CLOSEJ and CLOSEI. Layer 3 is completed when a CPU time limit is reached.

Algorithm 14 PERTURBARCSI(x: current solution)

1: $x' \leftarrow x$
2: Randomly generate a number *NPerturbI* in $[\lceil p_1|I(x)|\rceil, \lceil p_2|I(x)|\rceil]$
3: Randomly select *NPerturbI* open major depots and, for each chosen major depot $i \in I$, multiply the transportation costs for arcs originating at i by $M(\textit{transport})$
4: MLVNS(x', 2)
5: Restore all cost functions to their original expressions
6: MLVNS(x', 2)

4. COMPETING APPROACHES

To assess the efficiency of the MLVNS algorithm, we implemented two additional solution methods for each problem addressed: a path-based integer program, solved with the state-of-the-art solver CPLEX (version 12.1) and a slope scaling heuristic [19] based on the same formulation. We describe these approaches in the next subsections.

4.1. Path-Based Integer Programs

For both the classical TUFLPS and the industrial TUFLPS with modular costs, we define the following decision variables:

$$y_i = \begin{cases} 1, & \text{if major depot } i \in I \text{ is used,} \\ 0, & \text{otherwise,} \end{cases}$$

$$z^1_{ij} = \begin{cases} 1, & \text{if depot–depot arc } (i,j) \in R \text{ is used,} \\ 0, & \text{otherwise,} \end{cases}$$

and

$$z^2_{ijl} = \begin{cases} 1, & \text{if path } (i,j,l) \in P \text{ is used,} \\ 0, & \text{otherwise.} \end{cases}$$

Given the notation introduced in Section 2, the formulation of the classical TUFLPS is given by:

$$\min \sum_{i\in I} h^1_i y_i + \sum_{(i,j)\in R} h^2_j z^1_{ij} + \sum_{(i,j,l)\in P} c_{ijl} z^2_{ijl} \tag{1}$$

$$\sum_{j\in J_l} \sum_{i\in I_j} z^2_{ijl} = 1, \qquad l \in L, \tag{2}$$

$$\sum_{j\in J_l \cap J_i} z^2_{ijl} \le y_i, \qquad i \in I, l \in L_i, \tag{3}$$

$$z^2_{ijl} \le z^1_{ij}, \qquad (i,j,l) \in P, \tag{4}$$

$$\sum_{i\in I_j} z^1_{ij} \le 1, \qquad j \in J, \tag{5}$$

$$z^2_{ijl} \in \{0,1\}, \qquad (i,j,l) \in P, \tag{6}$$

$$z^1_{ij} \in \{0,1\}, \qquad (i,j) \in R, \tag{7}$$

$$y_i \in \{0,1\}, \qquad i \in I, \tag{8}$$

where $c_{ijl} = (c^1_{ij} + c^2_{jl})d_l$, for each path $(i,j,l) \in P$, and $L_i = \{l \in L \mid J_l \cap J_i \neq \emptyset\}$. Constraints (2) ensure that exactly one path reaches each customer: not only is the demand then satisfied, but the flow to each customer is also never split. Constraints (3) and (4) force paths to go only through open major and minor depots, respectively. Inequalities (5) are the single assignment constraints, which are redundant, given the cost structure, as seen in Section 2.

Other relevant works (see Section 1) mostly assume that transportation costs are decomposable by arc and exploit an alternative formulation without the single assignment constraints, where fixed costs are assigned to minor depots (using binary location variables) rather than to depot–depot arcs, as in the model above. There is a trivial correspondence between this alternative formulation and the one used here.

Given the formulation above and the notation introduced in Section 2, the industrial variant of the TUFLPS with modular costs can be modeled with additional integer variables: w^0_i represents the number of large-size vehicles serving major depot $i \in I$, w^1_{ij} is the number of medium-size vehicles operating from major depot $i \in I$ to minor depot $j \in J_i$, and w^2_j corresponds to the number of batches of parcels sorted at minor depot $j \in J$. The following integer program was shown to yield strong bounds in [13]:

$$\min \sum_{i\in I} h^1_i y_i + \sum_{i\in I} e^0_i w^0_i + \sum_{(i,j)\in R} e^1_{ij} w^1_{ij} + \sum_{j\in J} b_j w^2_j + \sum_{(i,j,l)\in P} c^2_{jl} d_l z^2_{ijl} \tag{9}$$

subject to constraints (2) to (8) and

$$\sum_{j\in J_i}\sum_{l\in L_j} d_l z^2_{ijl} \leq V^0 w^0_i, \qquad i \in I, \tag{10}$$

$$\sum_{l\in L_j} d_l z^2_{ijl} \leq V^1 w^1_{ij}, \qquad (i,j) \in R, \tag{11}$$

$$\sum_{i\in I_j}\sum_{l\in L_i} d^p_l z^2_{ijl} \leq B w^2_j, \qquad j \in J, \tag{12}$$

$$w^0_i \in \mathbb{N}, \qquad i \in I, \tag{13}$$

$$w^1_{ij} \in \mathbb{N}, \qquad (i,j) \in R, \tag{14}$$

$$w^2_j \in \mathbb{N}, \qquad j \in J. \tag{15}$$

This formulation allows us to obtain near-optimal upper and lower bounds in reasonable time on small instances. However, on realistically sized instances, the gap between the bounds computed by CPLEX remains relatively large even after a few hours of computation (see Section 5).

4.2. *Slope Scaling Heuristics*

We use the integer program (1)–(8) to derive our slope scaling heuristic for the classical TUFLPS. First, the formulation can be simplified by introducing a set of binary variables $y^2_j \equiv \sum_{i\in I_j} z^1_{ij}$, $j \in J$, allowing us to reformulate the objective function as

$$\min \sum_{i\in I} h^1_i y_i + \sum_{j\in J} h^2_j y^2_j + \sum_{(i,j,l)\in P} c_{ijl} z^2_{ijl}.$$

Then, we define a relaxation obtained by dropping the integrality of the y^2_j variables and by aggregating constraints (4) into

$$\sum_{l\in L_j}\sum_{i\in I_j} d_l z^2_{ijl} \leq \left(\sum_{l\in L_j} d_l\right) y^2_j, \quad j \in J.$$

We can then project out the y^2_j variables, since the fixed costs are nonnegative:

$$y^2_j = \frac{1}{(\sum_{l\in L_j} d_l)} \sum_{l\in L_j}\sum_{i\in I_j} d_l z^2_{ijl}.$$

The objective function of the relaxed problem then becomes:

$$\min \sum_{i\in I} h^1_i y_i + \sum_{(i,j,l)\in P} (c_{ijl} + d_l \tilde{h}^2_j) z^2_{ijl},$$

where

$$\tilde{h}^2_j = \frac{h^2_j}{(\sum_{l\in L_j} d_l)}.$$

The relaxed problem can be further reduced. For each pair (i,l), $i \in I, l \in L_i$, if an optimal solution contains a path from i to l, the intermediate node on this path must be a minor depot that minimizes the linearized cost $(c_{ijl} + d_l \tilde{h}^2_j)$. Hence, we define the following linearized cost for each pair (i,l):

$$\tilde{c}_{il} = \min_{j\in J_l \cap J_i} \left\{ c_{ijl} + d_l \tilde{h}^2_j \right\}.$$

Then, we can solve the relaxed problem as an equivalent UFLP, called the *linearized subproblem*:

$$\min \sum_{i\in I} h^1_i y_i + \sum_{i\in I}\sum_{l\in L_i} \tilde{c}_{il} z_{il}$$

$$\sum_{i\in I_l} z_{il} = 1, \qquad l \in L,$$

$$z_{il} \leq y_i, \qquad i \in I, l \in L_i,$$

$$z_{il} \in \{0,1\}, \qquad \forall i \in I, l \in L_i,$$

$$y_i \in \{0,1\}, \qquad \forall i \in I.$$

The UFLP, although NP-hard, tends to be easily solved in practice. In the current case, the instances are also significantly smaller than the original TUFLPS instances, and solving them as integer programs with CPLEX, rather than linear ones, incurs a negligible computational overhead (another

possibility would be to use an efficient heuristic method such as the randomized rounding procedure suggested in [3]).

A solution to the linearized subproblem can be easily converted into a solution to the TUFLPS. However, doing so yields solutions of poor quality. Instead, we exploit the fact that fixing the set of open major depots based on the solution of the linearized subproblem results in a simpler, single-level facility location problem involving only minor depots and customers. More precisely, fixing the values of the y_i variables in model (1)–(8) incurs instances of the following model, called the *fixed subproblem*, where $I^* \subset I$ is the set of major depots such that y_i is fixed to 1, $R^* \subseteq I^* \times J$ and $P^* \subseteq I^* \times J \times L$ are, respectively, the subsets of R and of P induced by I^*:

$$\min \sum_{i \in I^*} h_i^1 + \sum_{(i,j) \in R^*} h_j^2 z_{ij}^1 + \sum_{(i,j,l) \in P^*} c_{ijl} z_{ijl}^2 \qquad (16)$$

$$\sum_{j \in J_l} \sum_{i \in I^* \cap I_j} z_{ijl}^2 = 1, \qquad l \in L, \qquad (17)$$

$$z_{ijl}^2 \leq z_{ij}^1, \qquad (i,j,l) \in P^* \qquad (18)$$

$$\sum_{i \in I^* \cap I_j} z_{ij}^1 \leq 1, \qquad j \in J, \qquad (19)$$

$$z_{ijl}^2 \in \{0,1\}, \qquad (i,j,l) \in P^*, \qquad (20)$$

$$z_{ij}^1 \in \{0,1\}, \qquad (i,j) \in R^*. \qquad (21)$$

The fixed subproblem can be reduced to a UFLP by relaxing the single assignment constraints (19) and by replacing them with an artificial customer of highly negative (profitable) transportation costs for each minor depot. In our implementation, we did not perform this transformation and rather solve the problem directly with CPLEX.

The slope scaling heuristic iteratively updates the linearized penalties $\tilde{h}_j^2$ to better reflect the original nonlinear costs. Each $\tilde{h}_j^2$ is initialized as above. Then, given an optimal solution $(\tilde{z}^1, \tilde{z}^2)$ to the fixed subproblem, each $\tilde{h}_j^2$ such that $\sum_{l \in L_j} \sum_{i \in I_j} d_l \tilde{z}_{ijl}^2 > 0$ is updated to $h_j^2 / \sum_{l \in L_j} \sum_{i \in I_j} d_l \tilde{z}_{ijl}^2$ (the values of $\tilde{h}_j^2$ are not modified otherwise). The process is then repeated, with the updated linear penalties, until the objective values of two consecutive fixed subproblems are equal, or the objective values of the linearized and fixed subproblems are equal, or some time limit is reached.

For the industrial variant of the TUFLPS, we develop a specialized slope scaling heuristic by linearizing the modular costs, while leaving intact the fixed costs on the major depots. Relaxing the integrality constraints on the w_i^0, w_{ij}^1, and w_j^2 variables in the path-based integer program presented in Section 4.1 allows us to project out these variables and to remove the corresponding constraints (10)–(12) from the formulation. For example, since the costs e_i^0 are nonnegative, we can substitute variable w_i^0 using

$$w_i^0 = \frac{1}{V^0} \sum_{j \in J_i} \sum_{l \in L_j} d_l z_{ijl}^2, \qquad i \in I.$$

As a result, we initialize the linearized penalties as follows:

$$\tilde{e}_i^0 = \frac{e_i^0}{V^0}, \qquad i \in I,$$

$$\tilde{e}_{ij}^1 = \frac{e_{ij}^1}{V^1}, \qquad (i,j) \in R,$$

$$\tilde{b}_j = \frac{b_j}{B}, \qquad j \in J.$$

We then associate a linearized cost to each path $(i,j,l) \in P$: $\tilde{c}_{ijl}^2 = (c_{jl}^2 + \tilde{e}_i^0 + \tilde{e}_{ij}^1) d_l + \tilde{b}_j d_l^p$. The resulting linearized subproblem is:

$$\min \sum_{i \in I} h_i^1 y_i + \sum_{(i,j,l) \in P} \tilde{c}_{ijl}^2 z_{ijl}^2$$

subject to constraints (2) to (8).

The resulting TUFLPS has an objective function that satisfies the conditions that imply the redundancy of the single assignment constraints, given in Section 2. In particular, the per path costs $\tilde{c}_{ijl}^2$ can be decomposed by arc. Therefore, the single assignment constraints (5) can be removed, as well as the z_{ij}^1 variables, since there are no costs (and no more constraints) associated with these variables. The linearized subproblem can be further reduced, since for each pair (i,l), $i \in I, l \in L_i$, if an optimal solution contains a path from i to l, the intermediate node on this path must be a minor depot that minimizes the linearized cost $\tilde{c}_{ijl}^2$. The resulting equivalent model is a UFLP that is solved with CPLEX, in a similar way as in the case of the slope scaling heuristic for the classical TUFLPS. Once an optimal solution to the UFLP is obtained, it is converted into a solution to the TUFLPS. This solution is then improved with a call to layer 1 of the MLVNS heuristic.

The linearized penalties are updated for arcs and nodes through which some flow is circulating in the best solution $(\tilde{y}, \tilde{z}^2)$ obtained after performing the MLVNS-layer 1 heuristic. For example, let major depot $i \in I$ be such that $\sum_{l \in L_i} \sum_{j \in J_i} d_l \tilde{z}_{ijl}^2 > 0$, and let

$$\tilde{w}_i^0 = \frac{1}{V^0} \sum_{j \in J_i} \sum_{l \in L_j} d_l \tilde{z}_{ijl}^2.$$

The linearized penalty $\tilde{e}_i^0$ is then updated so that if the same flow circulates through i in the solution of the next linearized subproblem, then $\tilde{e}_i^0$ corresponds to the per-depot cost of the original problem:

$$\tilde{e}_i^0 = e_i^0 \frac{\lceil \tilde{w}_i^0 \rceil}{\tilde{w}_i^0}$$

(the linearized penalty is not modified if $\tilde{w}_i^0 = 0$). Similar slope scaling updates are applied to $\tilde{e}_{ij}^1$, $(i,j) \in R$, and $\tilde{b}_j$, $j \in J$. The linearized subproblem (formulated as a UFLP) is then iteratively evaluated with the updated penalties, until the objective values from two consecutive calls to MLVNS-layer 1 are equal, or the objective values of the linearized

subproblem and the solution obtained from MLVNS-layer 1 are equal, or some time limit is reached.

5. COMPUTATIONAL RESULTS

The computational results reported in this section were obtained on a 2.8 GHz Intel Xeon X5660 with 24 GB RAM (with Hyper-Threading and Turbo Boost disabled) running Debian 6.0/Linux 3.2 and CPLEX 12.1. All the heuristics were coded in C++ and compiled with g++ at "-O2" optimization settings, and CPLEX was used in single-threaded mode at default settings. The UFLP subproblems in the slope scaling heuristics were also solved with CPLEX, again at default settings. Finally, the MLVNS heuristic, being the only randomized method, was independently executed five times for each instance; the values reported here take into account all the executions of the heuristic.

In the next subsections, we report the results obtained on instances of the classical TUFLPS and of the industrial TUFLPS with modular costs. For both families of instances, the same parameters were used in layer 3 of the MLVNS heuristic: between 10 and 25% of the depots were chosen for cost perturbation ($p_1 = 10\%$ and $p_2 = 25\%$), location costs were increased by $M(\text{location}) = 10^8$, and transportation costs multiplied by $M(\text{transport}) = 100$.

5.1. Classical TUFLPS

Following [23], we derive 90 TUFLPS instances from the Gap A, B and C instance sets for the UFLP [21]. Unlike [23], we treat big-M values in the Gap instances as forbidden arcs. The resulting networks are sparse, and some instances are infeasible; those were not considered. The total number of feasible instances is 74. In Table 1, we report the number of feasible instances in each class next to the class name, in parenthesis. The process yields small ($|I| = |J| = |L| = 50$), but difficult instances: the root integrality gap for the integer programming formulation exceeds 10% in some cases. We also generated 20 larger instances of the same Gap A, B and C families. The corresponding MGap A, B and C TUFLPS instances are larger ($|I| = |J| = |K| = 75$), and even more difficult. Again, infeasible instances were discarded, which leads to 31 large instances in total.

In Table 1, we report average results on each class of instances, while detailed results are provided in the appendix. The upper bounds are expressed in terms of the gap from the exact optimal value (obtained with CPLEX): for a given upper bound $\bar{Z}$ and an optimal value Z^*, the gap is computed as $\bar{Z}/Z^* - 1$. For each column and class of instances, the minimum and maximum gaps are reported (over all independent executions if applicable). The average is a geometric mean of $\bar{Z}/Z^*$ for all executions, from which 1 is subtracted, and the average runtime is an arithmetic mean. We report the upper bounds obtained by executing the first and second layers of the MLVNS (columns "Layer 1" and "Layer 2"), without any time limit, then by performing the third layer (column "Layer 3") with a time limit of 60 s. Columns "CPLEX" give the results obtained using CPLEX to solve the path-based integer formulation (see Section 4.1) at default settings, without any time limit. Finally, columns "Slope Scaling" shows the slope scaling heuristic results when it was executed with a time limit of 24 min (which was almost never reached).

CPLEX solves the path-based model for the difficult TUFLPS instances we generated. The small instances are solved quickly; however, solution times increase significantly on the larger MGap instances. In general, both the MLVNS and the slope scaling heuristics seem to perform reasonably well; however, despite the specialized mathematical heuristic used for the slope scaling, the MLVNS seems preferable overall. Layer 3 fares much better on the smaller instances, regularly converging to optimal solutions. Indeed, over the 74 Gap instances, layer 3 provides systematically the optimal solutions on 39 instances, while the average solution value is within 1% of the optimal one for 32 instances. In comparison, the same figures for the slope scaling heuristic show 7 and 20 instances, respectively. The performance of layer 3 on the MGap instances remains reasonable even with a time limit of 60 s. Over the 31 MGap instances, layer 3 provides systematically the optimal solutions on 7 instances, while the solution value is within 1% of the optimal one for 7 other instances. For the slope scaling heuristic, these figures correspond to 0 and 5 instances, respectively. Layer 3 seems overall preferable to the slope scaling heuristic, for each set of instances, despite the fact that the latter heuristic exploits a specialized, optimally explored, very large neighborhood. The contribution of each layer to the MLVNS performance is also quite explicit: layer l performs better and more consistently than layer $l - 1$.

5.2. Industrial TUFLPS with Modular Costs

We tested the MLVNS on the testbed described in [13]. This testbed is derived from data obtained from a major French mail-order company, which provided us with a typical network comprising 93 potential sites for the major depots, 320 potential sites for the minor depots and 701 customers, as well as realistic estimates of the costs and the capacities. Based on this real-application data, we have generated 32 instances by specifying:

- subsets of the sets of major depots, minor depots and customers, that is, I, J, and L (three subnetworks, large, medium and small, were generated);
- multipliers M_f, M_g, and M_p for, respectively, the fixed costs at the major depots, the unit batch costs at the minor depots and the capacities of the large-size vehicles at the major depots (two values, 1 and 2, were tested for each multiplier).

Every instance is denoted $X(M_f, M_g, M_p)$, with $X = R$, L, M, or S, standing for real-application, large-scale, medium-scale, or small-scale network, M_f, M_g, and M_p denoting the multiplier values. Table 2 summarizes the characteristics of the 32 instances. Column 1 gives the problem name, while the next three columns indicate the number of major depots, minor depots and customers. The next two columns show,

TABLE 1. Performance of MLVNS, slope scaling and CPLEX on classical TUFLPS instances

	gap (min/avg/max) % average runtime (s)														
Instances	Layer 1			Layer 2			Layer 3			CPLEX			Slope Scaling		
GapA (28)	6.57	36.02 0.00	63.83	0.01	14.84 0.03	42.42	0.00	0.01 60.02	0.02	0.00	0.00 7.86	0.00	0.00	5.99 53.53	14.28
GapB (16)	0.00	24.58 0.00	49.58	0.00	11.73 0.02	37.15	0.00	0.01 60.03	0.04	0.00	0.00 4.53	0.00	0.00	3.94 1.92	6.31
GapC (30)	14.00	30.27 0.00	56.80	0.02	14.54 0.02	33.05	0.00	0.23 60.02	6.92	0.00	0.00 31.47	0.00	0.00	4.28 98.34	14.16
MGapA (9)	5.73	34.42 0.00	49.36	0.00	20.06 0.04	39.65	0.00	5.22 60.05	13.22	0.00	0.00 540.84	0.00	0.04	5.33 11.36	12.46
MGapB (12)	26.29	46.09 0.00	78.01	0.00	19.53 0.06	39.58	0.00	5.08 60.05	20.90	0.00	0.00 655.63	0.00	0.02	7.22 129.84	14.11
MGapC (10)	13.24	34.68 0.00	63.63	6.47	18.30 0.05	39.51	0.00	3.52 60.05	14.18	0.00	0.00 1002.10	0.00	0.05	7.23 11.39	19.81

TABLE 2. Set of 32 instances (each row contains 8 instances)

Problem	$\|I\|$	$\|J\|$	$\|L\|$	$\|R\|$	$\|Q\|$	M_f	M_g	M_p
R (M_f, M_g, M_p)	93	320	701	2250	28782	{1,2}	{1,2}	{1,2}
L (M_f, M_g, M_p)	70	240	526	1260	16131	{1,2}	{1,2}	{1,2}
M (M_f, M_g, M_p)	46	160	350	562	6652	{1,2}	{1,2}	{1,2}
S (M_f, M_g, M_p)	23	80	175	167	1807	{1,2}	{1,2}	{1,2}

respectively, the number of arcs between major depots and minor depots, denoted $|R|$, and the number of arcs between minor depots and customers, denoted $|Q|$.

The time limit for both MLVNS and slope scaling was set to 24 min, and we report values for the path-based integer program solved with CPLEX under time limits of 24 min and 2 h. Note that, as the termination criteria are only tested at the end of each iteration of the outer loop, the total CPU time for MLVNS will always exceed 24 min (1440 s), usually by a few seconds. The gaps are computed relative to the lower bound obtained by executing CPLEX for up to 2 h (or until more than 24 GB of RAM was required); this lower bound corresponds to the optimal value only for the smallest instances ("S" subnetwork), since CPLEX then reports final gaps on the order of .1%.

Table 3 reports the performance of each layer in the MLVNS (columns "Layer 1," "Layer 2," and "Layer 3"), that of CPLEX, solving the integer formulation with a time limit of 24 min and 2 h (columns "CPLEX (24 min)" and "CPLEX (2h)"), and that of the slope scaling heuristic (column "Slope Scaling"). For each set of instances, the minimum, average and maximum gaps are shown, along with the average runtime. Detailed results are provided in the appendix.

These results indicate that the MLVNS algorithm is competitive with solving the path-based formulation with CPLEX on the smaller subnetworks, and outperforms it on the larger subnetworks. In fact, even when the time limit is extended to 2 h, CPLEX computes weak upper bounds, with gaps on the order of 50%, the best upper bound on instances derived from the actual application being 11.91%. In contrast, the MLVNS consistently converges to solutions at most 6% worse than the optimum. Of particular interest is instance R(1,1,1), which corresponds to the full network with the original costs obtained from the French mail-order company. On this instance, the solution found with CPLEX after 2 h exhibits a gap of 57%, while the solutions obtained with the MLVNS are always within 5% of the best lower bound.

The MLVNS also performs better, overall, than the slope scaling heuristic, although the difference is weaker on the instances derived from the real subnetwork. The table of detailed results (see Appendix) reveals that this can be attributed to the better performance of the slope scaling heuristic on instances in which location costs for the major depots are scaled up from their actual value. Note that these costs are not linearized in the slope scaling heuristic, which explains why the heuristic performs better when location costs weigh more heavily in the objective function.

Again, it is clear that each layer contributes both to the quality and consistency of the method. Not only are average, minimum and maximum gaps reduced in layer l, compared to layer $l-1$, but so are the differences between minimum and maximum gaps.

6. CONCLUSIONS

In this article, we have presented a heuristic method for solving a large class of TUFLPS constraints (TUFLPS). The heuristic algorithm is based on a variant of the VNS metaheuristic, which we call the MLVNS. Three layers of neighborhood structures are used and ordered in increasing complexity. In addition, each time a move is performed in a neighborhood at layer $l > 1$, this move is completed and evaluated by a recursive call to MLVNS up to layer $l-1$. The heuristic was tested on two specific problems, the classical TUFLPS, and a multiechelon location-distribution problem

TABLE 3. Performance of MLVNS, slope scaling and CPLEX on instances of the industrial TUFLPS

	gap (min/avg/max) % average runtime (s)																	
Instances	Layer 1				Layer 2			Layer 3			CPLEX (24 min)			CPLEX (2h)			Slope scaling	
Small	9.08	10.81 0.00	12.45	0.06	3.20 0.29	5.59	0.01	0.23 1440.35	0.69	0.01	0.01 722.54	0.03	0.01	0.01 3602.44	0.01	0.34	2.05 360.09	4.04
Medium	5.84	8.86 0.01	12.20	1.53	3.18 5.92	5.35	0.40	0.96 1443.95	1.72	0.18	0.51 1440.08	0.90	0.07	0.24 7200.07	0.38	2.68	3.57 548.50	5.05
Large	10.85	12.55 0.03	13.61	2.76	5.22 23.77	11.58	1.76	2.61 1454.11	3.65	10.14	38.01 1440.17	66.14	1.44	2.72 7200.18	4.50	3.46	3.95 1241.08	4.37
Real	12.10	14.16 0.09	16.13	5.22	6.53 59.17	9.39	3.42	4.87 1482.26	6.59	74.16	114.67 1440.43	294.26	11.91	46.61 7200.34	87.61	4.15	5.14 1304.99	6.26

arising from an actual application in fast delivery service. The computational results on large sets of instances of the two problems show the efficiency of our adaptation of the MLVNS approach to TUFLPS with complex cost functions.

These results also confirm the quality of the lower bounds obtained by solving the path-based integer programming formulations. A promising research avenue is to explore the development of exact algorithms based on decomposition approaches applied to the path-based models. These algorithms would benefit from the integration of the heuristic methods developed in this article. Other research avenues include the study of extensions to our problem. In particular, both the location-distribution and the routing decisions could be handled simultaneously within a multiperiod version of the problem.

ACKNOWLEDGMENTS

The authors are thankful to two anonymous referees whose comments helped us write a better article.

REFERENCES

[1] K. Aardal, M. Labbé, J.M.Y. Leung, and M. Queyranne, On the two-level uncapacitated facility location problem, INFORMS J Comput 8 (1996), 289–301.

[2] H. Amrani, A. Martel, N. Zufferey, and P. Makeeva, A variable neighborhood search heuristic for the design of multicommodity production distribution-networks with altenative facility configurations, OR-Spektrum 33 (2011), 989–1007.

[3] F. Barahona and F. Chudak, Near-optimal solutions to large-scale facility location problems, Discrete Optim 2 (2005), 35–50.

[4] I. Barros and M. Labbé, A general model for the uncapacitated facility and depot location problem, Location Sci 2 (1994), 173–191.

[5] I. Barros, Discrete and fractional programming techniques for location models. Combinatorial optimization, Vol. 3, Kluwer Academic Publishers, Dordrecht, Boston, London, 1998.

[6] A.I. Barros, R. Dekker, and V. Scholten, A two-level network for recycling sand: A case study, Eur J Oper Res 110 (1998), 199–214.

[7] J.M. Bloemhof-Ruwaard, M. Salomon, and L.N. Van Wassenhove, On the coordination of product and by-product flows in two-level distribution networks: Model formulations and solution procedures, Eur J Oper Res 79 (1994), 325–339.

[8] J.M. Bloemhof-Ruwaard, M. Salomon, and L.N. Van Wassenhove, The capacitated distribution and waste disposal problem, Eur J Oper Res 88 (1996), 490–503.

[9] P. Chardaire, J.-L. Lutton, and A. Sutter, Upper and lower bounds for the two-level simple plant location problem, Ann Oper Res 86 (1999), 117–140.

[10] I. Correia and M.E. Captivo, A Lagrangean heuristic for a modular capacitated location problem, Ann Oper Res 122 (2003), 141–161.

[11] I. Correia and M.E. Captivo, Bounds on the single-source modular capacitated plant location problem, Comput Oper Res 33 (2006), 2991–3003.

[12] L.-L. Gao and E.P. Robinson, A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. Naval Res Logist 39 (1992), 191–212.

[13] B. Gendron and F. Semet, Formulations and relaxations for a multi-echelon capacitated location-distribution problem. Comput Oper Res 36 (2009), 1335–1355.

[14] P. Hansen and N. Mladenovic, "Variable neighborhood search," Handbook of metaheuristics (Glover F., Kochenberger G., Gary A. Eds.), Chap. 8, international series in operations research and management science, Vol. 57, 2003, 145–184.

[15] K. Holmberg, Solving the staircase cost facility location problem with decomposition and piecewise linearization, Eur J Oper Res 75 (1994), 41–61.

[16] K. Holmberg and J. Ling, A Lagrangean heuristic for the facility location problem with staircase costs, Eur J Oper Res 97 (1997), 63–74.

[17] Y. Huang, C.-W. Chen, and Y. Fan, Multistage optimization of the supply chains of biofuels, Transport Res Part E: Logist Transport Rev 46 (2010), 820–830.

[18] L. Kaufman, M.V. Van Eede, and P. Hansen, A plant and warehouse location problem, Oper Res Q 28 (1977), 547–554.

[19] D. Kim and P. Pardalos, A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure, Oper Res Lett 24 (1999), 195–203.

[20] A. Klose and A. Drexl, Facility location models for distribution system design, Eur J Oper Res 162 (2005), 4–29.

[21] Y. Kochetov and D. Ivanenko, "Computationally difficult instances for the uncapacitated facility location problem," Metaheuristics: Progress as real problem solvers, (Ibaraki T., Nonube K., Yagiura M. Eds.), Chap. 16, Operations research, computer science interfaces series, Vol. 32, 2005, 351–367.

[22] J. Krarup and P.M. Pruzan, The simple plant location problem: Survey and synthesis, Eur J Oper Res 12 (1983), 36–81.

[23] M. Landete and A. Marín, New facets for the two-stage uncapacitated facility location polytope, Comput Optim Appl 44 (2009), 487–519.

[24] H.R. Lourenço, O. Martin, and T. Stützle, "Iterated local search," Handbook of Metaheuristics, The Kluwer International Series in Operations Research and Management Science, Chapter 11, F. Glover and G.A. Kochenberger (Editors), Kluwer Academic Publishers Group, London, Dordrecht, Boston, 2003, pp. 321–353.

[25] H.R. Lourenço, O. Martin, and T. Stützle, "Iterated local search: Framework and applications," Handbook of metaheuristics, The Kluwer international series in operations research and management science, Chapter 12, Kluwer Academic Publishers Group, London, Dordrecht, Boston, 2010, pp. 363–397.

[26] A. Marin, Lower bounds for the two-stage uncapacitated facility location problem, Eur J Oper Res 179 (2006), 1126–1142.

[27] A. Marin and B. Pelegrin, Applying Lagrangian relaxation to the resolution of two-stage location problems, Ann Oper Res 86 (1999), 179–198.

[28] M.T. Melo, S. Nickel, and F. Saldanha da Gama, Facility location and supply chain management–A review, Eur J Oper Res 196 (2009), 401–412.

[29] P. Mitropoulos, I. Giannikos, and I. Mitropoulos, Exact and heuristic approaches for the locational planning of an integrated solid waste management system, Oper Res 9 (2009), 329–347.

[30] N. Mladenovic and P. Hansen, Variable neighborhood search, Comput Oper Res 24 (1997), 1097–1100.

[31] S.C. Narula and U.I. Ogbu, An hierarchal location–allocation problem, Omega 7 (1979), 137–143.

[32] H. Pirkul and V. Jayaraman, Production, transportation and distribution planning in a multi-commodity tri-echelon system, Transport Sci 30 (1996), 291–302.

[33] H. Pirkul and V. Jayaraman, A multi-commodity multiplant capacitated facility location problem: Formulation and efficient heuristic solution, Comput Oper Res 25 (1998), 869–878.

[34] H.-B. Ro and D.-W. Tcha, A branch and bound algorithm for the two-level uncapacitated facility location problem with some side constraints, Eur J Oper Res 18 (1984), 349–358.

[35] G. Sahin and H. Süral, A review of hierarchical facility location models, Comput Oper Res 34 (2007), 2310–2331.

[36] P.N. Thanh, O. Péton, and N. Bostel, A linear relaxation-based heuristic approach for logistics network design, Comput Ind Eng 59 (2010), 964–975.

[37] S. Tragantalerngsak, J. Holt, and M. Rönnqvist, Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem, Eur J Oper Res 102 (1997), 611–625.

[38] A.A. Vilcapoma Ignacio, V.J. Martins Ferreira Filho, and R. Dieguez Galvao, Lower and upper bounds for a two-level hierarchical location problem in computer networks, Comput Oper Res 35 (2008), 1982–1998.

[39] C. Voudouris and E. Tsang, Guided local search and its application to the traveling salesman problem, Eur J Oper Res 113 (1999), 469–499.

[40] C. Voudouris and E. Tsang, "Guided local search," Handbook of metaheuristics, the kluwer international series in operations research and management science, Chapter 7, F. Glover and G.A. Kochenberger (Editors), Kluwer Academic Publishers Group, London, Dordrecht, Boston, 2003, pp. 185–218.

[41] C. Voudouris, E. Tsang, and A. Alsheddy, "Guided local search," Handbook of metaheuristics, the Kluwer international series in operations research and management science, Chapter 11, M. Gendreau and J.-Y. Potvin (Editors), Kluwer Academic Publishers Group, London, Dordrecht, Boston, 2010, pp. 321–362.

[42] J. Zhang, Approximating the two-level facility location problem via a quasi-greedy approach, Math Prog 108 (2006), 159–176.

APPENDIX: DETAILED COMPUTATIONAL RESULTS

TABLE A1. Detailed numerical results for small GapA instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
432GapA	6.57	6.57	6.57	6.56	6.56	6.56	0.00	0.00	0.01	0.00	0.00
		0.00			0.01			60.03		2.71	0.94
532GapA	39.72	39.72	39.72	0.04	2.69	13.26	0.00	0.00	0.00	0.00	6.70
		0.00			0.04			60.02		12.63	3.00
632GapA	31.08	31.08	31.08	6.17	16.10	30.98	0.02	0.02	0.02	0.00	6.21
	0.00			0.02			60.03			2.25	2.36
732GapA	49.68	49.68	49.68	0.02	9.94	21.30	0.00	0.00	0.00	0.00	0.05
		0.00			0.03			60.02		16.43	2.95
832GapA	24.85	24.85	24.85	6.27	12.46	18.65	0.02	0.02	0.02	0.00	6.25
		0.00			0.02			60.03		18.96	1.40
932GapA	52.93	52.93	52.93	13.16	18.50	26.48	0.00	0.00	0.00	0.00	6.66
		0.00			0.03			60.03		9.30	2.83
1032GapA	42.49	42.49	42.49	0.01	16.97	28.33	0.00	0.00	0.00	0.00	7.06
		0.00			0.03			60.03		1.25	1.66
1132GapA	26.43	26.43	26.43	13.16	13.18	13.19	0.02	0.02	0.02	0.00	6.58
		0.00			0.02			60.02		3.81	1.77
1232GapA	26.48	26.48	26.48	6.61	7.96	13.23	0.02	0.02	0.02	0.00	13.27
		0.00			0.03			60.03		5.63	1.76
1332GapA	28.47	28.47	28.47	7.10	7.11	7.12	0.02	0.02	0.02	0.00	0.02
		0.00			0.04			60.02		8.04	1.76
1432GapA	42.45	42.45	42.45	14.09	18.39	21.24	0.01	0.01	0.01	0.00	7.14
		0.00			0.03			60.03		3.14	1.52
1532GapA	42.36	42.38	42.39	21.12	26.78	35.26	0.01	0.01	0.01	0.00	14.10
		0.00			0.03			60.02		9.77	1.97
1632GapA	28.27	28.28	28.28	7.00	15.50	21.17	0.00	0.01	0.01	0.00	14.07
		0.00			0.03			60.01		4.38	2.01
1832GapA	31.05	31.05	31.05	12.43	18.62	24.81	0.00	0.00	0.00	0.00	6.15
		0.00			0.02			60.03		7.27	2.58
1932GapA	63.82	63.82	63.83	7.14	17.04	21.35	0.01	0.01	0.01	0.00	7.20
		0.00			0.04			60.02		2.44	1.94
2032GapA	35.44	35.44	35.44	21.22	22.66	28.33	0.00	0.00	0.00	0.00	0.00
		0.00			0.03			60.01		2.48	1440.03
2132GapA	35.46	35.46	35.46	7.04	11.31	21.24	0.00	0.00	0.00	0.00	14.28
		0.00			0.03			60.04		11.66	1.33
2232GapA	33.03	33.03	33.03	6.58	11.84	19.73	0.00	0.00	0.00	0.00	6.72
		0.00			0.03			60.02		8.55	1.38
2332GapA	33.16	33.16	33.16	13.28	22.51	26.46	0.00	0.00	0.00	0.00	0.11
		0.00			0.03			60.02		11.54	1.28
2432GapA	42.66	44.07	49.73	0.01	5.66	7.09	0.01	0.01	0.01	0.00	0.14
		0.00			0.04			60.02		8.58	1.43
2532GapA	42.61	45.45	49.72	7.05	11.35	14.23	0.00	0.00	0.00	0.00	0.00
		0.00			0.04			60.02		3.74	2.29
2632GapA	21.19	21.19	21.19	21.19	21.19	21.19	0.00	0.00	0.00	0.00	7.05
		0.00			0.01			60.01		4.26	1.96
2732GapA	63.65	63.65	63.65	35.36	41.00	42.42	0.00	0.00	0.00	0.00	14.18
		0.00			0.03			60.02		4.97	4.25
2832GapA	49.54	49.54	49.54	14.15	18.40	35.37	0.00	0.00	0.00	0.00	0.00
		0.00			0.02			60.02		1.94	1.92
2932GapA	46.30	46.30	46.31	6.67	17.20	32.98	0.00	0.00	0.00	0.00	13.23
		0.00			0.03			60.02		2.71	4.58
3032GapA	42.52	42.52	42.52	7.06	16.97	28.33	0.01	0.01	0.01	0.00	7.08
		0.00			0.04			60.04		22.57	3.20
3132GapA	33.10	34.42	39.72	6.57	10.58	19.81	0.00	0.00	0.00	0.00	0.04
		0.00			0.03			60.02		14.31	3.18
3232GapA	6.68	6.68	6.68	0.01	4.00	6.66	0.00	0.00	0.00	0.00	6.71
		0.00			0.02			60.03		14.83	1.44

TABLE A2. Detailed numerical results for small GapB instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
431GapB	23.37	23.37	23.37	11.69	14.01	17.50	0.00	0.00	0.00	0.00	0.05
		0.00			0.02			60.02		3.18	1.61
531GapB	27.59	27.59	27.59	10.98	16.52	22.02	0.02	0.02	0.02	0.00	5.53
		0.00			0.03			60.03		6.64	1.82
931GapB	27.57	27.57	27.57	0.00	13.21	16.53	0.00	0.00	0.00	0.00	5.54
		0.00			0.02			60.03		1.63	0.33
1031GapB	29.20	29.21	29.21	11.65	11.66	11.67	0.00	0.00	0.00	0.00	5.90
		0.00			0.04			60.03		9.58	1.45
1231GapB	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.00			0.01			60.02		1.95	2.28
1331GapB	15.72	15.76	15.77	0.03	1.07	5.24	0.02	0.02	0.02	0.00	5.30
		0.00			0.02			60.03		16.70	3.26
1431GapB	23.39	23.40	23.40	5.89	11.70	17.53	0.00	0.00	0.00	0.00	5.85
		0.00			0.02			60.03		3.46	2.10
1731GapB	23.41	23.41	23.41	5.86	7.03	11.68	0.00	0.00	0.00	0.00	5.97
		0.00			0.03			60.03		5.63	1.43
2031GapB	16.56	19.87	22.08	10.99	11.02	11.04	0.00	0.00	0.00	0.00	5.51
		0.00			0.02			60.03		2.72	0.92
2331GapB	24.75	24.75	24.75	6.20	12.38	24.75	0.00	0.00	0.00	0.00	6.31
		0.00			0.02			60.01		1.55	3.05
2431GapB	31.05	31.05	31.05	6.22	11.17	18.63	0.02	0.02	0.03	0.00	0.07
		0.00			0.03			60.02		3.73	1.87
2731GapB	19.95	19.95	19.95	6.63	7.96	13.24	0.02	0.02	0.02	0.00	0.02
		0.00			0.03			60.02		4.44	3.64
2831GapB	22.10	24.30	27.62	5.57	9.95	11.05	0.02	0.03	0.04	0.00	5.57
		0.00			0.02			60.04		1.99	2.04
2931GapB	49.58	49.58	49.58	24.75	30.95	37.15	0.00	0.00	0.00	0.00	0.00
		0.00			0.03			60.02		1.37	1.28
3131GapB	43.32	43.32	43.32	6.15	14.81	18.52	0.02	0.02	0.02	0.00	6.16
		0.00			0.04			60.03		7.01	2.23
3231GapB	17.50	17.50	17.50	17.49	17.49	17.49	0.02	0.02	0.02	0.00	5.84
		0.00			0.01			60.02		0.96	1.41

TABLE A3. Detailed numerical results for small GapC instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
333GapC	14.00	14.00	14.00	6.88	8.29	13.90	0.00	4.12	6.87	0.00	13.99
		0.00			0.01			60.02		2.24	3.60
433GapC	14.03	14.03	14.03	14.00	14.01	14.03	0.01	1.39	6.92	0.00	7.05
		0.00			0.01			60.02		107.01	4.14
533GapC	33.12	33.12	33.12	13.25	17.22	19.85	0.02	0.02	0.02	0.00	6.72
		0.00			0.03			60.02		31.41	1.96
633GapC	39.71	39.71	39.71	19.86	22.50	26.48	0.02	0.02	0.02	0.00	0.05
		0.00			0.02			60.02		5.52	1440.04
733GapC	31.15	31.15	31.16	0.07	9.99	18.71	0.02	0.02	0.02	0.00	0.06
		0.00			0.03			60.02		87.39	2.31
833GapC	26.49	31.79	33.12	26.44	26.46	26.48	0.00	0.00	0.00	0.00	6.60
		0.00			0.03			60.02		7.54	2.45
933GapC	19.86	19.86	19.86	0.04	4.00	6.66	0.00	0.00	0.00	0.00	0.07
		0.00			0.02			60.02		5.37	1.09
1033GapC	55.94	55.94	55.94	6.18	19.90	31.06	0.00	0.00	0.00	0.00	6.29
		0.00			0.03			60.02		12.38	4.65
1133GapC	37.35	37.35	37.35	6.26	9.99	12.47	0.02	0.02	0.02	0.00	0.11
		0.00			0.04			60.02		29.23	2.34
1233GapC	28.24	28.24	28.24	14.09	14.09	14.09	0.01	0.01	0.01	0.00	7.11
		0.00			0.01			60.03		6.34	2.63
1333GapC	18.63	18.63	18.63	12.50	12.50	12.50	0.02	0.02	0.02	0.00	0.06
		0.00			0.01			60.02		81.97	3.86
1433GapC	19.88	19.88	19.88	13.26	13.26	13.27	0.02	0.02	0.02	0.00	6.59
		0.00			0.02			60.03		96.85	2.45
1533GapC	19.91	19.91	19.91	6.73	11.95	13.31	0.00	0.00	0.00	0.00	6.69
		0.00			0.02			60.02		13.17	1.74
1633GapC	28.34	28.34	28.34	7.08	9.91	21.25	0.00	0.00	0.00	0.00	14.16
		0.00			0.02			60.02		19.40	1.15
1733GapC	28.38	32.64	35.49	14.17	14.18	14.19	0.02	0.02	0.02	0.00	0.10
		0.00			0.03			60.02		46.68	3.47
1833GapC	39.69	39.69	39.69	19.80	23.79	33.05	0.02	1.33	6.60	0.00	6.61
		0.00			0.03			60.01		51.91	1.74
1933GapC	29.13	29.13	29.13	5.82	10.43	11.59	0.02	0.04	0.07	0.00	0.02
		0.00			0.03			60.03		7.20	1.58
2033GapC	31.08	31.08	31.08	0.02	8.71	18.66	0.02	0.02	0.02	0.00	0.06
		0.00			0.04			60.02		98.58	1.52
2133GapC	26.45	26.45	26.45	13.16	13.19	13.21	0.02	0.02	0.02	0.00	6.60
		0.00			0.02			60.03		10.69	3.83
2233GapC	56.80	56.80	56.80	7.15	17.05	21.32	0.01	0.01	0.01	0.00	7.08
		0.00			0.04			60.03		5.17	4.10
2333GapC	28.32	28.32	28.32	28.30	28.30	28.30	0.00	0.00	0.00	0.00	0.00
		0.00			0.02			60.02		33.57	1440.09
2433GapC	19.93	19.93	19.93	6.63	9.26	13.22	0.02	0.02	0.02	0.00	6.67
		0.00			0.02			60.02		11.51	1.86
2533GapC	42.41	42.41	42.41	14.26	19.81	21.25	0.00	0.00	0.00	0.00	7.04
		0.00			0.03			60.02		2.12	1.90
2633GapC	18.65	18.65	18.65	0.02	4.98	6.24	0.01	0.01	0.01	0.00	0.06
		0.00			0.02			60.01		10.58	0.68
2733GapC	26.46	31.75	33.07	0.06	11.90	19.88	0.00	0.00	0.00	0.00	6.63
		0.00			0.03			60.03		33.25	2.20
2833GapC	39.69	39.69	39.69	13.21	21.14	26.45	0.01	0.01	0.01	0.00	6.65
		0.00			0.03			60.03		12.48	2.58
2933GapC	35.44	35.44	35.45	14.13	18.42	21.33	0.02	0.02	0.02	0.00	0.02
		0.00			0.03			60.02		1.80	2.40
3033GapC	37.25	37.25	37.25	12.37	16.10	18.61	0.00	0.00	0.00	0.00	0.05
		0.00			0.04			60.02		28.37	2.04
3133GapC	28.36	28.36	28.36	0.06	14.19	28.33	0.01	0.01	0.01	0.00	0.05
		0.00			0.02			60.03		81.40	2.51
3233GapC	30.41	30.41	30.41	15.16	15.17	15.20	0.00	0.00	0.00	0.00	7.61
		0.00			0.01			60.02		3.00	3.26

TABLE A4. Detailed numerical results for larger MGapA instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
MGapA0	31.00	32.24	37.19	24.71	24.72	24.74	12.38	12.38	12.38	0.00	0.11
		0.00			0.04			60.07		1818.95	17.46
MGapA3	40.90	40.91	40.91	11.71	14.05	17.56	0.07	0.07	0.07	0.00	0.04
		0.00			0.07			60.03		621.89	15.79
MGapA5	43.37	43.37	43.37	18.53	22.28	30.93	0.04	0.04	0.04	0.00	12.46
		0.00			0.05			60.04		425.73	7.62
MGapA6	32.85	32.85	32.85	16.34	16.35	16.35	5.35	5.37	5.38	0.00	5.41
		0.00			0.03			60.06		161.69	15.21
MGapA9	46.29	46.29	46.29	32.95	34.33	39.65	13.20	13.21	13.22	0.00	6.63
		0.00			0.06			60.05		190.81	6.73
MGapA10	49.35	49.36	49.36	24.68	30.83	37.01	6.18	11.08	12.31	0.00	6.17
		0.00			0.05			60.03		23.21	9.40
MGapA12	21.97	21.97	21.97	10.95	15.34	21.92	0.00	0.00	0.00	0.00	5.56
		0.00			0.04			60.07		326.17	5.07
MGapA13	5.73	5.73	5.73	0.00	3.38	5.73	0.00	0.00	0.00	0.00	5.88
		0.00			0.02			60.05		908.03	12.62
MGapA17	43.23	43.23	43.24	18.48	22.20	30.83	6.09	6.09	6.09	0.00	6.26
		0.00			0.04			60.02		391.05	12.37

TABLE A5. DetailStrabed numerical results for larger MGapB instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
MGapB0	39.71	39.71	39.71	6.65	14.60	19.92	0.02	0.03	0.04	0.00	0.16
		0.00			0.06			60.03		356.64	8.00
MGapB2	33.14	33.14	33.14	13.20	18.54	26.47	0.00	0.01	0.04	0.00	6.67
		0.00			0.04			60.03		2449.02	9.40
MGapB3	63.60	63.60	63.60	21.08	23.91	28.15	14.02	14.02	14.02	0.00	14.11
		0.00			0.09			60.07		109.80	1440.11
MGapB4	52.67	52.67	52.67	19.60	24.91	39.44	6.36	6.36	6.36	0.00	0.02
		0.00			0.07			60.05		259.86	8.27
MGapB5	49.21	49.21	49.21	27.94	30.81	35.02	13.88	15.28	20.90	0.00	13.98
		0.00			0.07			60.05		56.50	3.95
MGapB6	39.58	39.58	39.58	13.06	17.04	19.75	0.00	3.82	6.43	0.00	6.57
		0.00			0.06			60.02		349.88	12.99
MGapB9	66.06	66.06	66.06	19.74	29.03	39.58	0.02	2.64	6.58	0.00	13.19
		0.00			0.07			60.07		682.00	12.96
MGapB10	32.92	32.92	32.92	13.10	14.43	19.69	6.49	6.50	6.54	0.00	13.19
		0.00			0.04			60.04		186.10	13.50
MGapB11	78.01	78.01	78.01	14.31	28.42	35.52	14.20	14.20	14.20	0.00	7.20
		0.00			0.09			60.03		630.15	8.22
MGapB14	43.49	43.49	43.49	6.18	12.37	18.54	0.00	0.00	0.00	0.00	0.03
		0.00			0.05			60.07		1634.07	12.22
MGapB16	26.29	26.29	26.29	0.00	9.16	13.13	0.00	0.00	0.00	0.00	6.64
		0.00			0.04			60.04		587.66	15.07
MGapB19	37.14	37.15	37.15	0.00	13.60	24.70	0.00	0.00	0.00	0.00	6.35
		0.00			0.06			60.08		565.93	13.43

TABLE A6. Detailed numerical results for larger MGapC instances

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)	
Instances	Layer 1			Layer 2			Layer 3			CPLEX	Slope Scaling
MGapC0	31.10	33.59	37.32	12.41	14.89	18.62	0.05	0.05	0.05	0.00	6.21
		0.00			0.08			60.06		1677.23	8.24
MGapC1	24.88	24.88	24.88	12.44	16.17	18.67	0.04	0.04	0.04	0.00	6.42
		0.00			0.03			60.07		2987.68	12.64
MGapC3	52.94	52.94	52.94	19.82	22.49	26.49	0.06	0.07	0.13	0.00	13.28
		0.00			0.07			60.06		1389.84	12.73
MGapC4	53.16	53.16	53.16	13.37	22.59	26.59	6.75	6.75	6.75	0.00	6.76
		0.00			0.08			60.03		2153.55	10.07
MGapC5	26.33	26.33	26.33	6.47	10.53	19.75	0.00	0.00	0.00	0.00	0.11
		0.00			0.04			60.05		502.36	9.86
MGapC7	21.27	21.27	21.27	21.26	21.26	21.26	7.19	12.78	14.18	0.00	7.17
		0.00			0.02			60.07		591.97	23.55
MGapC10	63.63	63.63	63.63	14.15	25.40	28.23	7.09	7.09	7.09	0.00	7.19
		0.00			0.07			60.05		19.02	6.68
MGapC11	13.24	13.24	13.24	6.65	9.24	13.12	0.00	0.00	0.00	0.00	6.62
		0.00			0.03			60.05		536.81	7.48
MGapC12	46.16	46.16	46.16	13.13	26.37	39.51	6.52	6.53	6.54	0.00	19.81
		0.00			0.07			60.05		58.69	10.04
MGapC16	21.14	21.15	21.15	7.00	15.44	21.08	0.00	2.74	6.92	0.00	0.05
		0.00			0.06			60.05		103.81	12.61

TABLE A7. Detailed numerical results for the industrial problem

	gap (min/avg/max) % average runtime (s)									gap% runtime (s)		
Instances	Layer 1				Layer 2			Layer 3		CPLEX (24 min)	CPLEX (2 h)	Slope Scaling
S(1,1,1)	9.57	9.65	9.76	1.00	2.95	4.67	0.15	0.49	0.69	0.01	0.01	4.01
		0.00			0.32			1440.33		1440.01	7200.01	0.12
S(1,1,2)	10.66	10.74	10.85	0.08	2.48	4.30	0.03	0.04	0.05	0.01	0.01	0.96
		0.00			0.30			1440.37		7.22	6.60	0.11
S(1,2,1)	9.08	9.12	9.13	0.84	2.86	4.24	0.43	0.44	0.44	0.01	0.01	3.59
		0.00			0.24			1440.36		1440.02	7200.01	0.14
S(1,2,2)	10.36	10.37	10.37	0.12	3.30	4.69	0.01	0.03	0.03	0.01	0.01	1.08
		0.00			0.29			1440.17		4.67	5.07	0.13
S(2,1,1)	11.36	11.43	11.51	0.83	3.48	5.59	0.17	0.43	0.58	0.03	0.01	4.04
		0.00			0.32			1440.44		1440.01	7200.01	1440.00
S(2,1,2)	12.29	12.35	12.45	0.06	3.10	5.29	0.02	0.03	0.05	0.01	0.01	0.34
		0.00			0.30			1440.38		4.66	3.94	0.09
S(2,2,1)	10.87	10.89	10.90	0.70	3.38	5.18	0.35	0.36	0.37	0.01	0.01	1.98
		0.00			0.23			1440.40		1440.02	7200.01	1440.01
S(2,2,2)	11.96	11.97	11.97	0.10	4.08	5.57	0.02	0.03	0.04	0.01	0.01	0.52
		0.00			0.29			1440.35		3.67	3.83	0.10
M(1,1,1)	6.04	6.56	6.87	1.98	3.05	4.68	1.06	1.34	1.72	0.22	0.11	5.05
		0.01			5.95			1445.48		1440.03	7200.09	1440.03
M(1,1,2)	8.63	8.88	9.20	2.68	2.78	2.90	0.40	0.54	0.66	0.38	0.32	4.06
		0.00			5.13			1442.84		1440.12	7200.04	3.83
M(1,2,1)	5.84	6.49	7.43	2.57	3.02	3.30	0.94	1.13	1.34	0.60	0.22	3.33
		0.01			6.85			1446.05		1440.12	7200.09	1440.58
M(1,2,2)	8.34	8.55	8.90	2.15	2.75	3.29	0.65	0.70	0.75	0.74	0.35	4.51
		0.01			6.29			1441.69		1440.03	7200.07	1440.10
M(2,1,1)	8.57	9.02	9.29	2.46	3.34	3.63	0.79	1.08	1.45	0.18	0.07	3.16
		0.01			6.62			1442.16		1440.11	7200.05	17.10
M(2,1,2)	11.69	11.91	12.20	4.21	4.48	5.35	0.62	0.94	1.17	0.70	0.33	2.88
		0.00			4.47			1444.28		1440.11	7200.11	5.78
M(2,2,1)	7.22	8.21	9.67	2.11	2.91	3.96	0.90	1.17	1.36	0.36	0.16	2.68
		0.01			5.54			1445.00		1440.04	7200.05	31.31
M(2,2,2)	11.19	11.38	11.69	1.53	3.13	4.26	0.56	0.82	1.04	0.90	0.38	2.90
		0.01			6.51			1444.07		1440.07	7200.03	9.32
L(1,1,1)	11.98	12.33	12.69	4.04	5.23	6.20	2.05	2.46	2.81	27.40	2.56	3.94
		0.03			25.23			1451.13		1440.10	7200.19	1440.41
L(1,1,2)	11.29	11.67	12.01	2.76	5.61	11.58	1.76	1.99	2.14	43.17	2.82	4.16
		0.03			19.81			1447.65		1440.24	7200.08	1440.51
L(1,2,1)	11.56	12.31	12.71	4.44	4.97	5.51	2.32	3.02	3.65	26.36	3.35	4.37
		0.03			34.97			1461.73		1440.18	7200.26	1440.49
L(1,2,2)	12.12	12.30	12.46	4.52	5.00	5.52	2.42	2.53	2.79	52.16	4.50	3.89
		0.02			21.47			1450.31		1440.28	7200.24	1441.00
L(2,1,1)	12.51	12.82	13.15	4.14	5.02	6.09	2.38	2.56	2.74	38.69	1.59	3.76
		0.03			21.01			1458.52		1440.14	7200.09	1440.71
L(2,1,2)	12.49	12.83	13.14	3.40	4.85	6.53	2.05	2.40	2.98	47.82	1.44	3.46
		0.03			19.55			1462.06		1440.07	7200.24	574.33
L(2,2,1)	10.85	12.67	13.32	4.34	5.90	7.58	2.41	2.91	3.27	10.14	2.80	4.01
		0.03			27.43			1446.48		1440.13	7200.20	1440.28
L(2,2,2)	13.31	13.47	13.61	4.30	5.19	5.99	2.54	2.99	3.49	66.14	2.72	4.01
		0.02			20.70			1455.02		1440.21	7200.14	710.91
R(1,1,1)	12.10	12.74	14.27	5.22	5.68	5.99	3.42	3.94	4.33	74.16	57.17	5.21
		0.10			56.32			1466.50		1440.58	7200.26	1440.01
R(1,1,2)	14.33	14.84	15.29	5.33	5.76	6.07	4.06	4.27	4.62	294.26	78.37	4.62
		0.09			78.21			1485.24		1440.10	7200.65	1440.29
R(1,2,1)	12.36	13.04	14.06	5.93	6.40	6.95	3.89	4.35	5.05	79.38	56.96	5.39
		0.10			42.42			1520.18		1440.49	7200.20	1440.54
R(1,2,2)	14.66	14.96	15.34	6.13	7.16	9.39	4.31	4.77	5.28	112.53	87.61	6.26
		0.08			45.38			1480.06		1440.19	7200.46	1218.47
R(2,1,1)	13.40	13.79	14.41	5.59	6.11	6.55	4.55	5.10	5.96	86.19	67.45	4.15
		0.10			67.08			1475.84		1440.60	7200.50	1440.61
R(2,1,2)	14.70	15.06	15.36	6.53	7.02	7.37	4.83	5.35	6.08	111.21	19.03	5.06
		0.10			77.00			1465.79		1440.61	7200.19	578.64
R(2,2,1)	12.78	13.33	13.91	6.09	6.79	7.52	4.83	5.14	5.58	86.88	15.89	4.84
		0.10			40.68			1480.86		1440.47	7200.27	1440.62
R(2,2,2)	15.15	15.52	16.13	6.91	7.31	7.64	5.25	6.07	6.59	134.46	11.91	5.59
		0.09			66.27			1483.59		1440.44	7200.21	1440.77