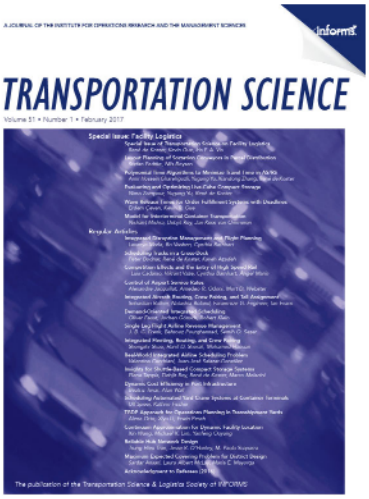

Transportation Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Lagrangian-Based Branch-and-Bound Algorithm for the Two-Level Uncapacitated Facility Location Problem with Single-Assignment Constraints

Bernard Gendron, Paul-Virak Khuong, Frédéric Semet

To cite this article:
Bernard Gendron, Paul-Virak Khuong, Frédéric Semet (2016) A Lagrangian-Based Branch-and-Bound Algorithm for the Two-Level Uncapacitated Facility Location Problem with Single-Assignment Constraints. Transportation Science 50(4):1286-1299.
https://doi.org/10.1287/trsc.2016.0692

TRANSPORTATION SCIENCE
Vol. 50, No. 4, November 2016, pp. 1286–1299
ISSN 0041-1655 (print) | ISSN 1526-5447 (online)

https://doi.org/10.1287/trsc.2016.0692

A Lagrangian-Based Branch-and-Bound Algorithm for the Two-Level Uncapacitated Facility Location Problem with Single-Assignment Constraints

Bernard Gendron, Paul-Virak Khuong
Centre Interuniversitaire de Recherche sur les Réseaux d'Entreprise, la Logistique et le Transport, Montréal, Québec H3T 1J4, Canada; and Département d'informatique et recherche opérationnelle, Université de Montréal, Montréal, Québec H3C 3J7, Canada
{bernard.gendron@cirrelt.ca, paul.khuong@cirrelt.ca}

Frédéric Semet
École Centrale de Lille, UMR 9189, CRIStAL, Centre de Recherche en Informatique, Signal et Automatique de Lille, 59651 Lille, France, frederic.semet@ec-lille.fr

We consider the two-level uncapacitated facility location problem with single-assignment constraints (TUFLP-S), a problem that arises in industrial applications in freight transportation and telecommunications. We present a new Lagrangian relaxation approach for the TUFLP-S, based on solving a single-level uncapacitated facility location problem (UFLP) as the Lagrangian subproblem. We also develop a Lagrangian heuristic that includes a mixed-integer programming–based large neighborhood search heuristic exploring neighborhoods by solving a series of small UFLPs. The dual and primal bounds thus obtained are used within an enumerative algorithm that implements specialized branching rules. Our computational experiments on instances derived from an industrial application in freight transportation as well as on large, hard, artificial instances confirm the efficiency of our specialized branch-and-bound algorithm.

Keywords: two-level uncapacitated facility location; Lagrangian relaxation; Lagrangian heuristic; branch-and-bound algorithm
History: Received: March 2013; revisions received: December 2014, November 2015; accepted: March 2016. Published online in *Articles in Advance* May 31, 2016.

1. Introduction

The two-level uncapacitated facility location problem (TUFLP) (Kaufman, Eede, and Hansen 1977) is an extension of the uncapacitated facility location problem (UFLP) (Krarup and Pruzan 1983). The UFLP consists in locating facilities from a finite set of potential sites and in assigning each customer to one of the selected facility locations to minimize the total costs, which include fixed costs for opening facility locations and assignment costs between customers and facility locations. In the TUFLP, the finite set of potential facility locations is replaced by two levels of such locations, depots at the upper level and satellites at the lower one. The only arcs are between depots and satellites and between satellites and customers. The problem is to decide which depots and satellites to open, and to which depot–satellite pair each customer should be assigned, to satisfy customer demands at minimum cost (Aardal et al. 1996). This problem arises in the design and operation of hierarchical networks that take advantage of economies of scale, most notably in freight transportation (Gendron and Semet 2009), but also in telecommunications (Chardaire, Lutton, and Sutter 1999).

We are concerned with a variant of the TUFLP that forces a single-assignment property on satellites: each satellite can be linked to at most one depot, and fixed costs are imposed on the use of arcs between depots and satellites. The single-assignment constraints ensure that any solution is a forest of trees rooted at depots. This variant of the TUFLP was introduced in Chardaire, Lutton, and Sutter (1999), motivated by an application in the design of telecommunications networks. Our study of the TUFLP with single assignment (TUFLP-S) is motivated by an industrial application in freight transportation, related to the operation of multicommodity distribution systems over a short-term planning horizon (Gendron and Semet 2009). In that industrial problem, depot and satellite locations typically correspond to freight terminals and parking spaces, respectively, for which the locations may vary every day in response to demand fluctuations. The TUFLP-S arises as a subproblem in decomposition and heuristic methods for solving the optimization problem derived from this application.

As illustrated by recent surveys (Klose and Drexl 2005; Melo et al. 2009; Sahin and Süral 2007; Farahani et al. 2014), there is an abundant literature on

multilevel facility location problems, which generalize the TUFLP-S. Mathematical programming models for these problems are usually divided into two classes, arc-based (Marín 2006; Pirkul and Jayaraman 1996, 1998) and path-based (Barros and Labbé 1994; Gao and Robinson 1992; Kaufman, Eede, and Hansen 1977; Ro and Tcha 1984), which have been compared theoretically and experimentally (Bloemhof-Ruwaard, Salomon, and Van Wassenhove 1994, 1996; Chardaire, Lutton, and Sutter 1999; Marín and Pelegrin 1999). Exact solution methods have been developed, based on Lagrangian relaxation (Barros 1995; Marín and Pelegrin 1999; Pirkul and Jayaraman 1996, 1998) and on strengthening the models with valid inequalities and facets (Aardal et al. 1996; Chardaire, Lutton, and Sutter 1999; Landete and Marín 2009). Heuristic methods have been proposed, based on Lagrangian or linear programming (LP) relaxations (Barros, Dekker, and Scholten 1998; Pirkul and Jayaraman 1996, 1998), on greedy strategies or simple neighborhood search methods (Barros and Labbé 1994; Mitropoulos, Giannikos, and Mitropoulos 2009; Narula and Ogbu 1979), on more sophisticated metaheuristics (Chardaire, Lutton, and Sutter 1999; Gendron, Khuong, and Semet 2015; Maric 2010; Ignacio, Filho, and Galvão 2008), and on approximation algorithms (Bumb 2001; Gabor and van Ommeren 2010; Zhang 2006).

Our contribution is threefold. First, we compare a mixed-integer programming (MIP) formulation for the TUFLP-S with previously described formulations for variants of the TUFLP. Then, from that formulation, we derive a Lagrangian relaxation scheme that provides stronger lower bounds than the LP relaxation, as well as an MIP-based large neighborhood search (LNS) heuristic; these bounding algorithms are combined in a Lagrangian heuristic to compute tight lower and upper bounds. Finally, we embed the Lagrangian heuristic within a branch-and-bound algorithm that uses specialized branching schemes. On industrial instances, the Lagrangian heuristic generates tighter bounds at the root node than a state-of-the-art MIP solver, while on large artificial instances, the specialized branch-and-bound method reduces computational times by a factor of three, when compared to a state-of-the-art MIP solver.

Section 2 introduces MIP formulations for variants of the TUFLP and compares them with our formulation for the TUFLP-S. A description of the Lagrangian heuristic follows in Section 3: first, the Lagrangian relaxation scheme; then a heuristic to solve the Lagrangian dual; and finally, the MIP-based LNS heuristic. Section 4 outlines the branch-and-bound algorithm and its branching schemes. Section 5 reports computational results on large industrial instances and on hard artificial instances.

2. TUFLP Formulations

A general model for two-level uncapacitated facility location problems is introduced in Barros and Labbé (1994). In addition to transportation costs for each path from depot to satellite to customer and fixed costs on the use of depots and satellites, the model includes fixed costs for arcs from depots to satellites. The model can be stated as follows.

Let I be the set of potential depot locations, let J be the set of potential satellite locations, let K be the set of customer locations, and let

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

$$z_j = \begin{cases} 1, & \text{if satellite } j \text{ is open,} \\ 0, & \text{otherwise,} \end{cases} \quad j \in J,$$

$$t_{ij} = \begin{cases} 1, & \text{if depot } i \text{ and satellite } j \text{ are operating together,} \\ 0, & \text{otherwise,} \end{cases} \quad (i,j) \in I \times J,$$

and

$$x_{ijk} = \begin{cases} 1, & \text{if customer } k \text{ is served through pair } (i,j), \\ 0, & \text{otherwise,} \end{cases} \quad (i,j,k) \in I \times J \times K.$$

The general TUFLP, called TUFLP-G, can then be formulated as

$$v(\text{TUFLP-G}) = \min \left\{ \sum_{i \in I} f_i y_i + \sum_{j \in J} g_j z_j + \sum_{(i,j) \in I \times J} h_{ij} t_{ij} + \sum_{(i,j,k) \in I \times J \times K} c_{ijk} x_{ijk} \right\}$$

subject to

$$\sum_{(i,j) \in I \times J} x_{ijk} = 1, \quad \forall k \in K, \tag{1}$$

$$x_{ijk} \le t_{ij}, \quad \forall (i,j,k) \in I \times J \times K, \tag{2}$$

$$\sum_{j \in J} x_{ijk} \le y_i, \quad \forall (i,k) \in I \times K, \tag{3}$$

$$\sum_{i \in I} x_{ijk} \le z_j, \quad \forall (j,k) \in J \times K, \tag{4}$$

$$0 \le x_{ijk} \le 1, \quad \forall (i,j,k) \in I \times J \times K,$$

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

$$z_j \in \{0,1\}, \quad \forall j \in J,$$

$$t_{ij} \in \{0,1\}, \quad \forall (i,j) \in I \times J,$$

where f_i, g_j, and h_{ij} are the nonnegative fixed costs for, respectively, each depot $i \in I$, each satellite $j \in J$, and each pair of depot–satellite $(i,j) \in I \times J$, and

where c_{ijk} is the nonnegative total transportation cost for each path from a depot i to a satellite j to a customer k (Barros and Labbé 1994).

Constraints (1) guarantee that the demand for each customer is satisfied exactly, and constraints (2)–(4) ensure that fixed costs are incurred for the use of depot–satellite pairs, depots, and satellites. Note that the integrality of the x_{ijk} variables can be relaxed without affecting the optimal objective value. Thus, TUFLP-G implicitly ensures that the flow to each customer is never split.

Alternative versions of the general model also include the following constraints:

$$x_{ijk} \le y_i, \quad \forall (i,j,k) \in I \times J \times K, \tag{5}$$

$$x_{ijk} \le z_j, \quad \forall (i,j,k) \in I \times J \times K, \tag{6}$$

$$t_{ij} \le y_i, \quad \forall (i,j) \in I \times J, \tag{7}$$

$$t_{ij} \le z_j, \quad \forall (i,j) \in I \times J. \tag{8}$$

However, constraints (5) are dominated by (3), and (6) by (4). Constraints (7) and (8) are redundant given nonnegative fixed costs h_{ij} and constraints (2)–(4). Indeed, $h_{ij} \ge 0$ implies the existence of an optimal solution such that $t_{ij} = \max_{k \in K}\{x_{ijk}\} \equiv x_{ijk^*}$ for each $(i,j) \in I \times J$; thus, $y_i \ge \sum_{j' \in J} x_{ij'k^*} \ge x_{ijk^*} = t_{ij}$, and constraints (7) are implied by (2) and (3). Likewise, constraints (8) are implied by (2) and (4).

Note that an optimal solution to TUFLP-G does not necessarily satisfy the single-assignment property. Transportation costs that vary significantly depending on the depot, for a given customer, may lead to optimal solutions in which the same satellite is linked to multiple depots.

We consider a variant in which we enforce this single-assignment property, expressed with the additional constraints

$$\sum_{i \in I} t_{ij} \le 1, \quad \forall j \in J. \tag{9}$$

The redundant constraints (7) are also added, as they are useful in solving the model. Indeed, the binary nature of the t_{ij} variables along with constraints (7) imply $y_i \in \{0,1\}$ for each $i \in I$. When adding the redundant constraints (7), it is thus possible to relax the integrality of the y_i variables and still maintain feasibility. We have observed that state-of-the-art MIP solvers perform better when these variables are allowed to take fractional values, rather than restricted to binary values. In addition, we also use constraints (7) to strengthen the Lagrangian relaxation described in Section 3.

Constraints (9) allow us to project out the z_j variables by using the equations

$$z_j = \sum_{i \in I} t_{ij}, \quad \forall j \in J.$$

The fixed cost g_j on each satellite j can then be folded in the fixed cost $l_{ij} = g_j + h_{ij}$ for every arc $(i,j) \in I \times J$. Constraints (4) can be eliminated, since they are simple aggregations of constraints (2).

The resulting formulation is TUFLP-S, where the y_i variables are allowed to take fractional values, as explained above

$$v(\text{TUFLP-S}) = \min \left\{ \sum_{i \in I} f_i y_i + \sum_{(i,j) \in I \times J} l_{ij} t_{ij} + \sum_{(i,j,k) \in I \times J \times K} c_{ijk} x_{ijk} \right\}$$

subject to

$$\sum_{(i,j) \in I \times J} x_{ijk} = 1, \quad \forall k \in K,$$

$$\sum_{i \in I} t_{ij} \le 1, \quad \forall j \in J,$$

$$x_{ijk} \le t_{ij}, \quad \forall (i,j,k) \in I \times J \times K,$$

$$\sum_{j \in J} x_{ijk} \le y_i, \quad \forall (i,k) \in I \times K,$$

$$t_{ij} \le y_i, \quad \forall (i,j) \in I \times J,$$

$$0 \le x_{ijk} \le 1, \quad \forall (i,j,k) \in I \times J \times K,$$

$$0 \le y_i \le 1, \quad \forall i \in I,$$

$$t_{ij} \in \{0,1\}, \quad \forall (i,j) \in I \times J.$$

Most studies on two-level uncapacitated location problems without single assignment are concerned with a simplification of the general model TUFLP-G: fixed costs on links between depots and satellites, h_{ij}, are always zero. Two seminal papers (Kaufman, Eede, and Hansen 1977; Ro and Tcha 1984) marked early research on this classical TUFLP: they introduced MIP formulations and specialized lower bounding methods and exploited them in branch-and-bound algorithms. More recent approaches (Aardal et al. 1996; Barros 1995; Landete and Marín 2009) are based on the MIP formulation obtained by eliminating the t_{ij} variables (since $h_{ij} = 0$ for each $(i,j) \in I \times J$) and constraints (2) from TUFLP-G; this does not affect LP relaxation bounds, as values for t_{ij} variables can be assigned easily, without affecting the total cost. We call the resulting formulation TUFLP-C

$$v(\text{TUFLP-C}) = \min \left\{ \sum_{i \in I} f_i y_i + \sum_{j \in J} g_j z_j + \sum_{(i,j,k) \in I \times J \times K} c_{ijk} x_{ijk} \right\}$$

subject to

$$\sum_{(i,j) \in I \times J} x_{ijk} = 1, \quad \forall k \in K,$$

$$\sum_{j \in J} x_{ijk} \le y_i, \quad \forall (i,k) \in I \times K,$$

$$\sum_{i\in I} x_{ijk} \le z_j, \quad \forall (j,k) \in J \times K,$$
$$0 \le x_{ijk} \le 1, \quad \forall (i,j,k) \in I \times J \times K,$$
$$y_i \in \{0,1\}, \quad \forall i \in I,$$
$$z_j \in \{0,1\}, \quad \forall j \in J.$$

It is sometimes further assumed that the transportation costs c_{ijk}, for all $(i,j,k) \in I \times J \times K$, are sums of per-arc costs $d_k c_{ij} + c_{jk}$, thus allowing the use of a more compact, but weaker, arc-based (or two-index) formulation. Such a structure for transportation costs, in conjunction with h_{ij} costs equal to zero, makes it possible to impose or to relax the single-assignment constraints without affecting the optimal value (Chardaire, Lutton, and Sutter 1999). Sets of open depots and satellites obtained from an optimal solution can be converted into a complete solution of TUFLP-S with a greedy procedure (that breaks ties consistently). Thus, even if only for this large class of instances, TUFLP-C and TUFLP-S are equivalent.

Constraints (3) and (4) define facets of the feasible polytope for TUFLP-C (Barros 1995). TUFLP-S preserves nearly all the constraints of TUFLP-C, including (3); the only missing constraints are (4), which are replaced with the stronger constraints (2). Thus, when all three formulations are applicable, *TUFLP-S leads to a stronger LP relaxation than both TUFLP-G and TUFLP-C*. Moreover, the difference is obtained by strictly improving on constraints that define facets of the TUFLP-C polytope. This can only be achieved by expanding the decision variables to include t_{ij} variables and by explicitly considering single-assignment constraints.

Solution methods for TUFLP-G, TUFLP-S, and TUFLP-C can be cast into three classes. Some approaches strengthen the formulation with valid inequalities, many of them facet defining (Aardal et al. 1996; Landete and Marín 2009), leading to large-scale models. Other approaches attempt to decrease solution times by computing approximate LP bound values, via dual ascent or Lagrangian relaxations combined with subgradient methods (Barros 1995; Barros and Labbé 1994; Gao and Robinson 1992). Finally, both metaheuristics (Barros and Labbé 1994) and approximation algorithms (Bumb 2001; Zhang 2006) have been developed to quickly obtain primal solutions.

The TUFLP with single assignment has been studied in Chardaire, Lutton, and Sutter (1999), where the TUFLP-S formulation presented here is introduced. Lower bounds are obtained with a Lagrangian relaxation scheme in which the dual is solved with a subgradient method and upper bounds are computed with a simulated annealing method.

We have shown above that the TUFLP-S formulation leads to stronger LP relaxation bounds than the TUFLP-C formulation, when they can be compared, at the expense of increased formulation size: each constraint (4) is replaced with multiple constraints (2), and the number of binary variables grows from $|I|+|J|$ to $|I \times J|$. Rather than attempting to quickly obtain approximate LP relaxation bounds for this large-scale formulation, we will compute even stronger bounds than the LP relaxation with a Lagrangian relaxation scheme, extract upper bounds with an MIP-based LNS heuristic, and further accelerate a branch-and-bound algorithm with specialized branching schemes.

3. Lagrangian Heuristic

The TUFLP-S is similar to the (single-level) UFLP, a problem for which there exists a large body of literature and efficient solution methods (Cornuéjols, Nemhauser, and Wolsey 1991; Krarup and Pruzan 1983). We exploit this strong foundation with a Lagrangian relaxation scheme in which the subproblems are UFLPs and with an MIP-based LNS heuristic that explores neighborhoods by solving UFLPs. These two components are further combined in a Lagrangian heuristic: the Lagrangian relaxation provides lower bounds and initial solutions, while the MIP-based LNS heuristic repairs and improves these solutions.

3.1. Lagrangian Relaxation

In Barros and Labbé (1994), it is proposed to obtain Lagrangian bounds for TUFLP-G (or the related TUFLP-C) by relaxing constraints (1), (3), and (4). The alternative chosen in Chardaire, Lutton, and Sutter (1999) is to bound TUFLP-S by relaxing (1), (3), and (9). In both cases, the Lagrangian bound is theoretically equal to the LP bound.

We focus on the Lagrangian relaxation obtained by dualizing only constraints (2). As we now show, the resulting Lagrangian subproblem can be converted into a significantly smaller, efficiently solved, UFLP. Moreover, this subproblem does not exhibit the integrality property, and the relaxation thus yields stronger bounds than the LP relaxation of TUFLP-S, both in theory and in practice.

When relaxing constraints (2), the Lagrangian subproblem $S(\lambda)$ can be formulated as follows, where $\lambda \ge 0$ is the vector of Lagrange multipliers:

$$v(S(\lambda)) = \min\left\{\sum_{i\in I} f_i y_i + \sum_{(i,j)\in I\times J}\left(l_{ij} - \sum_{k\in K}\lambda_{ijk}\right)t_{ij} + \sum_{(i,j,k)\in I\times J\times K}(c_{ijk}+\lambda_{ijk})x_{ijk}\right\}$$

subject to

$$\sum_{(i,j)\in I\times J} x_{ijk} = 1, \quad \forall k \in K,$$
$$\sum_{i\in I} t_{ij} \le 1, \quad \forall j \in J,$$

$$\sum_{j\in J} x_{ijk} \le y_i, \quad \forall (i,k) \in I \times K,$$
$$t_{ij} \le y_i, \quad \forall (i,j) \in I \times J,$$
$$0 \le x_{ijk} \le 1, \quad \forall (i,j,k) \in I \times J \times K,$$
$$0 \le t_{ij} \le 1, \quad \forall (i,j) \in I \times J,$$
$$y_i \in \{0,1\}, \quad \forall i \in I.$$

In formulation TUFLP-S, all of the variables are conceptually binary, but the x and y variables are left free to take fractional values. In the Lagrangian subproblem, all of the variables are conceptually binary, but the x and t variables are free to take fractional values, simplifying the formulation. This modification is valid, since the integrality constraints on the y_i variables are redundant in TUFLP-S and can therefore be added to the Lagrangian subproblem; then, the integrality of the t_{ij} variables can be relaxed without changing the optimal value of the Lagrangian subproblem. It can be simplified further, as we now show.

A simple dominance argument confirms that, for each pair of depot $i \in I$ and customer $k \in K$, all but one x_{ijk} variable can be eliminated. If, in a given optimal solution, there is a $j \in J$ such that $x_{ijk} = 1$, that variable can always be substituted with $x_{ij'k}$, where $j' = \arg\min_{j\in J}\{c_{ijk} + \lambda_{ijk}\}$, a variable corresponding to a least-cost (penalized) path from i to k.

By defining

$$\tilde{c}(\lambda)_{ik} = \min_{j\in J}\{c_{ijk} + \lambda_{ijk}\}, \quad \forall (i,k) \in I \times K,$$

the Lagrangian subproblem $S(\lambda)$ can then be solved as a more compact MIP, $S_c(\lambda)$, in which the number of variables scales quadratically (rather than cubically) with the instance size

$$v(S_c(\lambda)) = \min\left\{\sum_{i\in I} f_i y_i + \sum_{(i,j)\in I\times J}\left(l_{ij} - \sum_{k\in K}\lambda_{ijk}\right)t_{ij} + \sum_{(i,k)\in I\times K}\tilde{c}(\lambda)_{ik} w_{ik}\right\}$$

subject to

$$\sum_{i\in I} w_{ik} = 1, \quad \forall k \in K,$$
$$\sum_{i\in I} t_{ij} \le 1, \quad \forall j \in J,$$
$$w_{ik} \le y_i, \quad \forall (i,k) \in I \times K,$$
$$t_{ij} \le y_i, \quad \forall (i,j) \in I \times J,$$
$$0 \le w_{ik} \le 1, \quad \forall (i,k) \in I \times K,$$
$$0 \le t_{ij} \le 1, \quad \forall (i,j) \in I \times J,$$
$$y_i \in \{0,1\}, \quad \forall i \in I.$$

This alternative formulation is easily seen to be equivalent to the previous one by using the dominance argument above. In particular, any solution to $S_c(\lambda)$ can be converted into a solution to $S(\lambda)$, by keeping track of which c_{ijk} corresponds to each $\tilde{c}(\lambda)_{ik}$, for each pair of depot $i \in I$ and customer $k \in K$; subgradients for the Lagrangian subproblem can thus be extracted from optimal solutions to the compact formulation.

The compact Lagrangian subproblem is equivalent to the UFLP (with links only between facilities and customers). Any UFLP instance can be cast as an instance of the compact subproblem, by simply mapping facilities to depots and customers to customers, and letting the set of links between depots and satellites be empty. The compact Lagrangian subproblem itself is also easily reduced to the UFLP, by mapping depots to facilities and satellites and customers to customers. The assignment inequalities, $\sum_{i\in I} t_{ij} \le 1$, $\forall j \in J$, can be turned into strict equalities by allowing every satellite to be linked to an artificial depot with zero costs. The Lagrangian subproblem, like the UFLP, therefore does not have the integrality property.

3.2. Solving the Lagrangian Dual

The Lagrangian dual can be formulated as

$$\max_{\lambda\ge 0} v(S_c(\lambda)).$$

It is well known that the objective function of the Lagrangian dual (i.e., the optimal values $v(S_c(\lambda))$ of the Lagrangian subproblems for all $\lambda \ge 0$) is concave but nondifferentiable (Frangioni 2005). Hence, the Lagrangian dual is generally difficult to solve, even more so, as is the case here, when the Lagrangian subproblem is itself a difficult problem.

We solve the Lagrangian dual with a two-step heuristic that is well suited to being embedded into branch-and-bound methods, as discussed in Section 4. The first step computes a starting point for the Lagrangian dual, by solving the LP of the original (nondualized) model. Lagrange multipliers are read as the dual values corresponding to the relaxed constraints. In the second step, the Lagrangian subproblem (with integer variables) is evaluated once, with multipliers extracted from the LP relaxation. The Lagrangian dual is thus optimized with a fast heuristic.

In the computational experiments reported in Section 5, we compare this heuristic to a state-of-the-art implementation of the bundle method (Frangioni 1996), a nondifferentiable optimization method with stronger convergence properties than simpler approaches like subgradient methods (Frangioni 2005). More precisely, the bundle method is initialized as in the first step of the heuristic described above, to minimize the computational effort dedicated to improving Lagrange multipliers that are far from the optimum. Then, the lower bound is improved with a few iterations of the bundle method. Note that since the

Lagrangian subproblem does not have the integrality property, this approach yields lower bounds that are at least as strong as the LP relaxation bounds of TUFLP-S.

Our computational experiments show that this implementation of the bundle method, with default settings, leads to tighter lower bounds than the proposed heuristic. Unfortunately, the method must perform several iterations before its model of the Lagrangian dual is accurate enough to yield sizeable improvements in bound quality. That is why we propose to solve the Lagrangian subproblem only once, with Lagrange multipliers derived from the LP relaxation. Even in this case, the fact that the Lagrangian subproblem $S_c(\lambda)$ is solved as an integer subproblem ensures that the resulting bound will always be at least as strong as the LP bound. Regardless of the bound value, integer solutions to $S_c(\lambda)$ are also useful to guide the primal heuristic, as we see in Section 3.3.

3.3. Primal Heuristic

The primal heuristic is an MIP-based LNS approach that alternates between two large neighborhoods until no progress is observed between two consecutive iterations. Each neighborhood is explored exactly, through the solution of UFLPs.

The first neighborhood is defined by the set of solutions that preserve a given set of open depots: satellites may be opened, closed, and reconnected to open depots, and each customer may be assigned to any open satellite. Let $I^* \subset I$ be a set of open depots; the first neighborhood, corresponding to solutions in which the depots in I^* are open and those in $I \backslash I^*$ are closed, is explored by solving the following MIP formulation:

$$\min\left\{\sum_{i\in I^*} f_i + \sum_{(i,j)\in I^*\times J} l_{ij}t_{ij} + \sum_{(i,j,k)\in I^*\times J\times K} c_{ijk}x_{ijk}\right\}$$

subject to

$$\sum_{(i,j)\in I^*\times J} x_{ijk} = 1, \quad \forall k \in K, \tag{10}$$

$$\sum_{i\in I^*} t_{ij} \le 1, \quad \forall j \in J, \tag{11}$$

$$x_{ijk} \le t_{ij}, \quad \forall (i,j,k) \in I^* \times J \times K, \tag{12}$$

$$0 \le x_{ijk} \le 1, \quad \forall (i,j,k) \in I^* \times J \times K,$$

$$t_{ij} \in \{0,1\}, \quad \forall (i,j) \in I^* \times J.$$

This formulation corresponds to a UFLP in which the set of open facilities is subject to side conditions, the generalized upper bound (GUB) constraints (11). These side constraints can be expressed in a UFLP with the addition, for each $j \in J$, of an artificial customer k_j. The cost for linking k_j to any $t_{ij}, i \in I^*$, is very negative ($-M$), and zero for all other facilities, and the location costs l_{ij} are all increased by M. Thus, it is only profitable to open a location (i, j) if it can be linked to k_j; in that case, the M values sum to zero, and the objective function is not affected. Moreover, constraints (10) mean that each k_j is linked to exactly one t_{ij}, and thus that at most one t_{ij} is set to one, for any $j \in J$.

The second neighborhood is defined by the set of solutions that preserve a given assignment of customers to satellites: depots can be closed or open, and satellites can be reconnected to different open depots. For each satellite j, let $K^*(j)$ be the set of customers linked to that satellite in a given solution. The second neighborhood for that solution may be directly explored as a UFLP, in which facilities correspond to depots and customers to satellites (for which $K^*(j) \neq \varnothing$). Let $J^* \subset J$ be the set of satellites for which $K^*(j) \neq \varnothing$, and let

$$d_{ij} = l_{ij} + \sum_{k\in K^*(j)} c_{ijk}, \quad \forall (i,j) \in I \times J^*.$$

We then have the following MIP model for the second neighborhood:

$$\min\left\{\sum_{i\in I} f_i y_i + \sum_{(i,j)\in I\times J^*} d_{ij}t_{ij}\right\}$$

subject to

$$\sum_{i\in I} t_{ij} = 1, \quad \forall j \in J^*,$$

$$t_{ij} \le y_i, \quad \forall (i,j) \in I \times J^*,$$

$$0 \le t_{ij} \le 1, \quad \forall (i,j) \in I \times J^*,$$

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

The primal heuristic is completed with a postoptimization procedure that exploits the information from the solutions of the second neighborhood. More specifically, a restricted TUFLP-S model is solved by fixing to 0 the depot location variables that assume value 0 in all optimal solutions of the second neighborhood MIP models.

The primal heuristic must be initialized with a feasible solution. Rather than constructing one, it is possible to repair solutions from the integer Lagrangian subproblem $S_c(\lambda)$. The first neighborhood only requires a set of open depots; such a set can be extracted from any solution to $S_c(\lambda)$.

Tight lower and upper bounds are thus obtained in three steps:

1. Solve the LP relaxation of TUFLP-S.
2. Solve $S_c(\lambda)$, with Lagrange multipliers λ extracted from the previous step.
3. For each integer solution found when solving $S_c(\lambda)$, perform the primal heuristic, starting with the set of open depots in that solution.

4. A Specialized Branch-and-Bound Method

The bounding procedure described in the previous section depends, in part, on the full solution of the LP relaxation. It could be directly embedded within a standard branch-and-bound method, to improve only bound values. We found it more efficient to augment an LP-based branch and bound with the specialized lower and upper bounds, and with branching rules guided by complementary slackness violations in solutions to the Lagrangian subproblem $S_c(\lambda)$.

Note that the evaluation of each node is computationally heavy, and we attempt to minimize the size of the search tree by developing specialized branching schemes and by adapting reliability branching (Achterberg, Koch, and Martin 2005) to our context. Section 4.1 describes the specialized branching schemes. Our adaptation of reliability branching is presented in Section 4.2. We provide an overview of the branch-and-bound algorithm in Section 4.3.

4.1. Branching Schemes

We implemented two branching schemes that exploit the GUB constraints (9): the GUB branching scheme and the polytomic branching scheme.

The GUB scheme branches on sums of variables $\sum_{i\in I} t_{ij}$, for some satellite $j \in J$, forcing them to be equal to 0 or 1, i.e., converting GUB constraints (9) to equalities. The LP relaxation is exploited to consider only those $j \in J$ for which the sum is fractional, which are the *candidate branches*.

The polytomic scheme branches on multiple variables at once. It can be used alone or in conjunction with the GUB branching scheme; it is then also applicable to GUB constraints that were converted to strict equalities by earlier branching steps. Rather than turning GUB constraints into either of two equalities, a child is spawned for each variable in the sum, fixing that variable to one (and the others to zero), and one more in which the sum is set to zero, when the constraint is an inequality. Again, the LP relaxation is exploited to consider as candidate branches only those $j \in J$ such that at least one t_{ij}, for some $i \in I$, is fractional.

The polytomic branching scheme, used alone, ensures the convergence of the branch-and-bound method. However, on many instances, the number of nodes evaluated and the solution time are significantly reduced by initially branching with the GUB scheme and resorting to the polytomic scheme only when necessary, i.e., the node is not fathomed and all of the $\sum_{i\in I} t_{ij}$ are integral.

4.2. Adaptation of Reliability Branching

Reliability branching generalizes pseudocost branching (Benichou et al. 1971) and combines it with strong branching (Applegate et al. 1995). As in pseudocost branching, history-based pseudocosts (i.e., average bound improvements) are used to select the candidate branch. In our adaptation, the particular pseudocosts stored in memory depend on the branching scheme. In the GUB scheme, average bound improvements are computed for each satellite $j \in J$ and for each right-hand side of the GUB constraints (9) (i.e., whether the equality is fixed to 0 or to 1). In the polytomic scheme, we separately track average bound improvements for each variable when it is set to one and when all variables are set to zero (equivalently, when the GUB constraint for satellite j becomes an equality with 0 on the right-hand side).

In reliability branching, a candidate branch is declared *reliable* if its pseudocosts are based on sufficiently many evaluations (more than a reliability parameter $\eta \in \mathbb{N}$). In our adaptation, we proceed as follows. For each reliable candidate branch, a *score* is computed by taking the geometric average of the pseudocosts of all of its potential children (using a small minimum value $\epsilon > 0$). The candidate branch that achieves the maximum score is selected as the current *branching choice*. If all candidate branches are reliable, the procedure terminates and the current branching choice is the final one. In such a case, reliability branching operates in the same way as pseudocost branching.

If some candidate branches are unreliable, these candidate branches are first sorted based on a ranking criterion, to be discussed below. According to this ranking, unreliable candidate branches are iteratively examined to determine their scores. For any candidate branch, each of its potential children is partially evaluated with a small number $\gamma \in \mathbb{N}$ of dual simplex iterations, as in strong branching. The score for each candidate branch is computed by taking the geometric average of the resulting estimated bound increases for all of its potential children (using a small minimum value $\epsilon > 0$). If there is no reliable candidate branch, the current branching choice is the first unreliable candidate branch. Otherwise, the current branching choice was already determined when comparing the reliable candidate branches. To provide a fair basis of comparison with the unreliable candidate branches, this current branching choice is partially evaluated, like the unreliable candidate branches. Subsequently, any candidate branch with a higher score than that of the current branching choice becomes the new current branching choice. The next unreliable candidate branch is examined, and this iterative process is repeated until the current branching choice has not changed for $\lambda \in \mathbb{N}$ iterations (the look-ahead factor). We implemented this method, with $\eta = 8$, $\lambda = 4$ (as suggested in Achterberg, Koch, and Martin 2005), and $\gamma = 200$ (rather than adaptively).

In standard reliability branching, the ranking criterion used to sort the unreliable candidate branches is the pseudocost, which is, however, known to be unreliable. Rather than attempting to compute initial "reliable" pseudocosts (for instance, by using strong branching evaluations), we sort the unreliable candidate branches with respect to complementary slackness violations in the Lagrangian subproblem. For each satellite j, the quantity

$$\sum_{(i,k)\in I\times K} \lambda_{ijk}|x_{ijk} - t_{ij}|$$

is computed, and satellites corresponding to larger values are considered first. In preliminary experiments, this heuristic proved to reduce the number of nodes compared to summing LP-based reduced costs, even on instances for which the integer Lagrangian subproblem does not improve the LP bound value.

4.3. Overview of the Algorithm

The nodes are explored in a best-first order, with respect to the Lagrangian lower bounds. For the current node, a branching choice is determined by reliability branching. The children of the current node are fully evaluated by running the simplex until convergence and by evaluating the Lagrangian integer subproblem once. The children are then adjoined to the best-first search queue.

As in LP-based branch-and-bound approaches, some binary variables are fixed according to their reduced costs. The primal heuristic is executed at each node that is not fathomed. As described in Section 3.3, the solutions to the Lagrangian subproblems are used to initialize the primal heuristic. To make sure a diverse range of initial solutions are provided to the primal heuristic, any feasible solution corresponding to a neighborhood that has already been explored is rejected when the MIP corresponding to each Lagrangian subproblem $S_c(\lambda)$ is solved.

5. Computational Experiments

We compare the strength of the formulations and bounding methods by reporting gaps at the root node. We also compare the performance of our branch-and-bound method with that of the state-of-the-art MIP solver CPLEX by reporting the run time and number of search nodes until optimality is proven (within 0.1%). The tests were performed on four sets of instances.

All of the computations were performed in single-threaded mode on a 2.5 GHz Intel E5-2609 v2 processor with 126 GB RAM (hyperthreading and Turbo Boost were disabled). The solvers were compiled with g++4.9.2 with optimizations, and CPLEX 12.6.1 was used to solve LP and MIP models. The only exceptions were values copied from Landete and Marín (2009), which were computed on an older platform: 2.6 GHz AMD Opteron central processing unit (CPU) with 4 GB RAM and CPLEX 9.1. Note that we always use CPLEX 12.6.1 in default settings mode. Although some parameter settings might perform better on particular classes of instances, the default settings proved robust overall.

Our specialized branch-and-bound method uses CPLEX 12.6.1 to solve the MIP models in the Lagrangian heuristic, i.e., the integer Lagrangian subproblem and the primal heuristic subproblems. As shown in Section 3.1, all of these subproblems can be formulated as UFLPs, and it might be fruitful to exploit robust specialized UFLP solvers (Barahona and Chudak 2005; Beltran-Royo, Vial, and Alonso-Ayuso 2012; Hansen et al. 2007; Körkel 1989; Letchford and Miller 2012; Letchford and Miller 2014; Posta, Ferland, and Michelon 2014) when tackling huge instances of the TUFLP-S.

To our knowledge, it is not possible to embed all components of our specialized branch-and-bound method within the CPLEX environment. Using callback functionalities of CPLEX, the primal heuristic could be embedded within CPLEX branch-and-bound. The Lagrangian subproblem might be appended in a similar way, but then, it is not clear how to combine it with the cut generation features of CPLEX. Finally, CPLEX lacks support for polytomic branching, and it is currently impractical to implement anything but binary or simple multiway branching schemes (by reducing the latter to a series of binary branches). That is why our branch-and-bound implementation uses CPLEX as a subroutine, rather than being embedded within the CPLEX branch-and-bound environment.

The instances in the first set were generated as subproblems in a Lagrangian relaxation used to solve the industrial location problem described in Gendron and Semet (2009). These instances cannot be solved as TUFLP-C, and are relatively large, but exhibit low gaps at the root node; they highlight the effectiveness of the Lagrangian heuristic to compute tight upper and lower bounds.

The second set contains gap instances: artificial, small, and difficult TUFLP (without single assignment) instances constructed from the single-level UFLP gap instances (Kochetov and Ivanenko 2005). These instances exhibit large duality gaps (hence, their name) and are considered difficult for branch-and-bound methods based on LP relaxations.

The instances in the third set are similar, but larger. They are obtained with gap instances generated with the procedures described in Kochetov and Ivanenko (2005). The instances in the second and third sets can be formulated as both TUFLP-C and TUFLP-S, since their transportation costs are sums of per-arc costs.

The instances in the fourth and final set are large-size TUFLP-S instances obtained in a similar way as the

instances in the third set, with the exception that the conversion procedure was also modified to force the explicit consideration of the single-assignment constraints. These large, difficult instances allow us to show the improved scaling properties of the specialized branch-and-bound method (with respect to instance size) compared to CPLEX.

5.1. Industrial Instances

Our interest in the TUFLP-S stems from its appearance as a subproblem in a Lagrangian relaxation method for an application in freight transportation. This subsection reports performance values for 400 TUFLP-S instances derived from that industrial application, 100 on each of four networks: tiny, small, medium, and full (Gendron and Semet 2009). Full instances include 93 depots, 320 satellites, and 701 customers. Medium instances are about three-fourths as large at each level, small ones are about one-half as large, and tiny ones are about one-quarter as large.

5.1.1. Bounds at the Root Node. The industrial instances exhibit a cost structure that cannot be expressed with formulation TUFLP-C. Thus, only formulations and relaxations derived from TUFLP-S are compared. From left to right, the columns in Table 1 report the average gaps (with respect to the optimal value) and CPU times for "LP-S," the LP relaxation of the TUFLP-S formulation; "MIP^0-S," the relaxation derived by CPLEX at the root node, using the TUFLP-S formulation; "Lag," the Lagrangian relaxation with a single execution of the integer Lagrangian subproblem following the solution of the LP relaxation; "Lag/Bdl," which improves the initial Lagrange multipliers derived from the solution of the LP relaxation with up to 300 iterations of the bundle method; "MIP^0-S/H," the primal heuristic performed by CPLEX at the root node, using the TUFLP-S formulation (the reported CPU times are the same as those for MIP^0-S, since CPLEX reports the total time spent at the root node without separating the times spent in computing lower and upper bounds); and "Lag/H," the primal Lagrangian heuristic (to ease the comparison with the CPLEX root node heuristic, the CPU times include those for Lag, in addition to the primal heuristic times).

Table 1 Gaps and Run Times at the Root Node for Industrial Instances

Instances	LP-S	MIP^0-S	Lag	Lag/Bdl	MIP^0-S/H	Lag/H
Tiny						
Gap (%)	0.01	0.00	0.00	0.00	0.00	0.01
Time (s)	0.04	0.08	0.09	0.17	0.08	0.14
Small						
Gap (%)	0.43	0.41	0.36	0.25	0.33	0.26
Time (s)	0.60	2.93	1.85	17.23	2.93	2.92
Medium						
Gap (%)	0.15	0.14	0.08	0.05	0.28	0.09
Time (s)	4.10	11.85	8.12	48.97	11.85	11.70
Full						
Gap (%)	0.22	0.22	0.11	0.05	3.01	0.09
Time (s)	23.24	51.90	38.36	644.11	51.90	44.65

The gaps on these instances seem representative of industrial location problems: across all instances, the gap is lower than 1%, even for the LP relaxation. Solving the Lagrangian subproblem once, with Lagrange multipliers extracted from the LP relaxation, suffices to roughly halve the gap at the root node, and the bundle method further reduces the gap, but significantly increases the computational times. Note that, for any of these instances, the lower bound computed by CPLEX at the root node only slightly improves upon the LP relaxation bound.

The Lagrangian heuristic performs well on these instances, with solutions that are less than 0.3% away from the optimum on average. For small, medium, and full instances, the primal solutions derived by the Lagrangian heuristic are better than those obtained by CPLEX at the root node. In particular, on the full instances (which correspond to the size encountered in practice), our results show a final average gap of 0.09% in 44.65 seconds compared to the CPLEX 3.01% gap obtained in 51.90 seconds.

In summary, for these industrial instances, the Lagrangian heuristic produces high-quality solutions with sufficiently tight lower bounds, so branching can be avoided. Furthermore, for small, medium, and full instances, the two bounds are generally better and computed faster than those found by CPLEX at the root node.

5.1.2. Performance of Enumerative Methods. Recall that industrial instances cannot be solved as TUFLP-C. Thus, three enumerative methods are compared: "MIP-S" corresponds to the TUFLP-S formulation solved by CPLEX, "Lag/Pol" to the specialized branch-and-bound with only polytomic branching, and "Lag/GUB" to the specialized branch-and-bound combining the GUB and polytomic branching schemes. Table 2 displays the CPU times and the average number of nodes for each enumerative method.

For medium and full instances, the specialized branch-and-bound method with polytomic branching proves optimality in fewer nodes than CPLEX on model MIP-S. In particular, on full instances, the specialized branch-and-bound method explores significantly fewer nodes (more than 14 times fewer on average) than CPLEX on MIP-S. However, CPLEX nevertheless executes three to five times faster on average on full and medium instances. Small instances seem more difficult than others for the specialized branch-and-bound method. In particular, the specialized method with only polytomic branching,

Table 2 Run Times and Node Counts for Enumerative Methods on Industrial Instances

Instances	MIP-S	Lag/Pol	Lag/GUB
Tiny			
Time (s)	0.08	0.16	0.16
Nodes	1.03	0.49	0.56
Small			
Time (s)	4.95	239.07	4,021.88
Nodes	56.89	77.53	582.39
Medium			
Time (s)	23.97	115.13	1,176.68
Nodes	37.18	4.42	81.68
Full			
Time (s)	321.71	1,036.06	1,606.06
Nodes	162.51	11.46	36.50

Lag/Pol, is significantly (close to 50 times) slower than MIP-S on average.

5.2. Gap Instances

Landete and Marín (2009) describe a simple procedure to construct TUFLP-C instances from small and hard UFLP instances (Kochetov and Ivanenko 2005). We used the same procedure, on the same input, to obtain the same set of 90 instances with 50 depots, 50 satellites, and 50 customers. These instances have transportation costs that are sums of per-arc costs. Therefore, they can also be modeled as TUFLP-S. This allows us to compare the bounds obtained with our methods with those in Landete and Marín (2009). The only difference is that we consider the UFLP instances as sparse graphs, while Landete and Marín (2009) directly sums "big-M" costs. This way, 16 instances are made infeasible, leaving 28 instances derived from Gap A, 16 from Gap B, and 30 from Gap C.

5.2.1. Bounds at the Root Node. Table 3 reports the average gap (with respect to the optimal value) and CPU times at the root node for various bounding methods on Gap A, B, and C instances. These instances are derived from hard UFLP instances and are expected to exhibit huge integrality gaps on all practical formulations. In order, the columns are "Land," the formulation with specialized facet-defining constraints developed in Landete and Marín (2009) (result tables in Landete and Marín (2009) do not report solution times at the root node); "LP-C," the LP relaxation of the TUFLP-C formulation; "MIP^0-C," the relaxation derived by CPLEX at the root node, using the TUFLP-C formulation; "LP-S," the LP relaxation of the TUFLP-S formulation; "MIP^0-S," the relaxation derived by CPLEX at the root node, using the TUFLP-S formulation; "Lag," the integer Lagrangian subproblem solved only once, with Lagrange multipliers extracted from TUFLP-S; "Lag/Bdl," the Lagrangian relaxation with up to 300 iterations of the bundle method; "MIP^0-C/H," the primal heuristic performed by CPLEX at the root node using the TUFLP-C formulation; "MIP^0-S/H," the primal heuristic performed by CPLEX at the root node using the TUFLP-S formulation; and "Lag/H," the primal Lagrangian heuristic.

For each instance, LP-S provides a better bound than LP-C, and Lag/Bdl a better bound than LP-S. However, Lag almost always obtains the exact same bound as LP-S: for gap instances, forcing the t_{ij} variables to take integer values only improves the lower bound when the Lagrange multipliers are optimized with a bundle method. Overall, the valid inequalities introduced in Landete and Marín (2009) improve LP-C and yield bounds that are comparable with (and generally better than) those derived with LP-S; they too are weaker than the bounds obtained with Lag/Bdl. In general, the lower bounds computed by CPLEX at the root node improve on their corresponding LP relaxation bounds. It is interesting to note that even though LP-S provides better lower bounds than LP-C, the lower bounds obtained by CPLEX at the root node are sometimes better with MIP^0-C than with MIP^0-S (this is true in particular for Gap B instances for which MIP^0-C generates better lower bounds on average).

The LP-S formulation takes more time to solve than the more compact LP-C, by a factor of 2 to 3. Solving one Lagrangian subproblem adds reasonable overhead, roughly doubling the run time compared to LP-S, but rarely improves the bound value for gap instances. Optimizing the Lagrange multipliers with

Table 3 Gaps and Run Times at the Root Node for Gap Instances

Instances	Land	LP-C	MIP^0-C	LP-S	MIP^0-S	Lag	Lag/Bdl	MIP^0-C/H	MIP^0-S/H	Lag/H
Gap A										
Gap (%)	10.44	11.62	10.66	11.06	10.45	11.06	8.38	3.45	3.99	11.06
Time (s)		0.04	0.30	0.10	0.55	0.26	153.40	0.30	0.55	0.34
Gap B										
Gap (%)	6.88	8.00	7.15	7.84	7.31	7.84	5.24	2.66	3.02	7.21
Time (s)		0.04	0.24	0.09	0.51	0.22	155.03	0.24	0.51	0.26
Gap C										
Gap (%)	12.15	13.47	12.74	12.82	12.48	12.82	9.82	3.45	4.30	9.83
Time (s)		0.05	0.41	0.13	0.69	0.31	235.90	0.41	0.69	0.38

up to 300 iterations of the bundle method improves the lower bound significantly, but requires too much CPU time to be practical.

The "Lag/H" column shows the optimality gap after performing the primal heuristic. The execution of the primal heuristic took negligible time compared to the Lagrangian subproblem, but the average gaps are modest, between 7% and 12%, depending on the instance sets. The primal solutions computed by CPLEX at the root node are generally better, displaying average gaps between 2% and 4% for MIP^0-C/H, and between 3% and 5% for MIP^0-S/H. It is noteworthy that the former performs generally better, even though it is based on a weaker (but more compact) LP relaxation.

5.2.2. Performance of Enumerative Methods. In Table 4, the columns "Land," "MIP-C," and "MIP-S" correspond, respectively, to the formulation with specialized valid inequalities in Landete and Marín (2009), the TUFLP-C formulation, and the TUFLP-S formulation, solved by CPLEX. The column "Lag/Pol" reports run times and node counts when only polytomic branching is used, while the column "Lag/GUB" reports values for the specialized branch-and-bound method combining the two branching schemes, GUB and polytomic.

The difference in run times between our results and those of Landete and Marín (2009) is so wide that the averages are of limited usefulness. One reason for the difference might be their use of "big-M" values, instead of a sparse formulation that implicitly filters forbidden paths. Improvements in CPU performance and in CPLEX, from versions 9 to 12, are other possible explanations for such a huge difference in CPU times.

Formulation TUFLP-C (without single-assignment constraints) achieves the lowest run times. It seems likely that a more sophisticated implementation of the formulation with additional facet-defining inequalities described in Landete and Marín (2009) would achieve slightly lower run times: the formulation size is similar, and the facet-defining inequalities help decrease the number of nodes explored. TUFLP-S comprises more variables and constraints than TUFLP-C for the same instance, and this is reflected in increases in CPU time and number of nodes.

The specialized branching schemes (Lag/Pol and Lag/GUB) significantly reduce the node count compared to all of the other formulations. This is even more marked for Lag/GUB, which exploits the structure of GUB constraints to convert such constraints to equalities before executing polytomic branching. However, run times are longer than those for the MIP-C formulation solved by CPLEX, while they remain comparable to those for the MIP-S formulation solved by CPLEX.

Table 4 Run Times and Node Counts for Enumerative Methods on Gap Instances

Instances	Land	MIP-C	MIP-S	Lag/Pol	Lag/GUB
Gap A					
Time (s)	97.03	1.63	5.82	13.88	6.89
Nodes	99.79	191.00	484.61	29.89	18.82
Gap B					
Time (s)	69.83	1.12	3.56	8.44	4.71
Nodes	80.63	153.19	338.94	19.88	14.38
Gap C					
Time (s)	171.8	3.93	17.92	43.01	14.72
Nodes	260.17	476.27	1,325.83	79.87	36.20

5.3. Large Gap Instances

Using the procedure described in Landete and Marín (2009), we reimplemented the generators in Kochetov and Ivanenko (2005) to produce 30 UFLP instances each of the A, B, and C classes of size 150×150 and converted them to TUFLP-S instances of size $75 \times 75 \times 75$ ("Large A," "Large B," and "Large C"). Like gap instances, these instances have transportation costs that are sums of per-arc costs. Therefore, they can be modeled as TUFLP-C as well as TUFLP-S.

5.3.1. Bounds at the Root Node. Table 5 reports the average gaps (with respect to the optimal values) and the average CPU times for the three classes of instances, and for the same methods as in Table 3, except Land. These instances display significantly larger gaps than the gap instances. In particular, LP-C

Table 5 Gaps and Run Times at the Root Node for Gap Instances

Instances	LP-C	MIP^0-C	LP-S	MIP^0-S	Lag	Lag/Bdl	MIP^0-C/H	MIP^0-S/H	Lag/H
Gap A									
Gap (%)	18.26	17.94	17.28	17.13	17.28	16.18	9.14	24.33	13.33
Time (s)	0.38	2.88	1.18	5.22	2.71	2,673.40	2.88	5.22	3.05
Gap B									
Gap (%)	21.32	21.06	19.51	19.42	19.51	19.05	8.99	24.11	11.38
Time (s)	0.49	3.35	1.35	5.43	2.87	2,394.60	3.35	5.44	3.31
Gap C									
Gap (%)	21.01	20.74	19.31	19.22	19.31	18.83	8.96	26.47	12.27
Time (s)	0.53	3.53	1.79	6.83	3.25	2,563.18	3.53	6.83	3.56

and LP-S show average gaps between 17% and 22%, depending on the instance sets. We observe that the Lagrangian bound, Lag, never improves on the LP relaxation LP-S for any of these instances. Performing the bundle method for 300 iterations improves on these modest gaps, at the expense of excessive computational times. In general, LP-S provides a better bound than LP-C by about 1% to 2%, and Lag/Bdl a better bound than LP-S by about 1%. The best lower bounds are obtained by Lag/Bdl, which slightly improves upon the lower bounds produced by CPLEX at the root node, but the average gaps still range in the 16%–20% interval.

The primal heuristic, Lag/H, finds solutions with gaps that often exceed 10%, but identifies significantly better solutions than CPLEX on formulation LP-S at the root node, which computes primal solutions displaying gaps around 25%. Performing the CPLEX root node heuristic on formulation LP-C gives the best upper bounds, but the average gaps are relatively high, around 9%.

5.3.2. Performance of Enumerative Methods. Table 6 reports average CPU times and node counts for the TUFLP-C and the TUFLP-S formulations, MIP-C and MIP-S, respectively, as well as for the specialized branch-and-bound method combining GUB and polytomic branching schemes, Lag/GUB. The latter was shown above to be preferable to Lag/Pol for instances in the gap family. Indeed, Lag/Pol, the Lagrangian branch-and-bound with only polytomic branching, failed to solve all but a few of these large gap instances.

On average, across all instances, the Lagrangian-based branch-and-bound method explores significantly fewer nodes than CPLEX on MIP-S (about 50 times fewer, on average) and MIP-C (more than six times fewer, on average). When compared with CPLEX on MIP-S, the specialized branch-and-bound method is three times faster, but three times slower than CPLEX on model MIP-C. These results indicate that on these instances, the lower bound improvement from TUFLP-C to TUFLP-S does not compensate for the increase in the number of variables.

Table 6 Run Times and Node Counts for Enumerative Methods on Large Gap Instances

Instances	MIP-C	MIP-S	Lag/GUB
Large A			
Time (s)	343.51	3,288.34	1,189.90
Nodes	5,459.17	35,860.69	816.41
Large B			
Time (s)	399.65	3,740.50	1,115.20
Nodes	4,651.60	38,768.73	653.87
Large C			
Time (s)	448.32	4,492.01	1,500.79
Nodes	5,072.63	36,225.87	801.13

5.4. Large Gap Instances with Single-Assignment Constraints

To obtain additional large instances that consider the single-assignment constraints explicitly, we used the same procedure as the one performed to generate large gap instances, with one difference: each transportation cost was incremented by $(7i + 3j + k)$ mod 10, where i is the rank of the depot, j is the rank of the satellite, and k is the rank of the customer. This way, 90 instances of size $75 \times 75 \times 75$ are generated, 30 in each of three classes, "Large A-S," "Large B-S," and "Large C-S."

5.4.1. Bounds at the Root Node. Table 7 reports the average gaps (with respect to the optimal values) and the average CPU times for the three classes of instances for the same methods as in Table 1. These instances appear as difficult as the large gap instances of Section 5.3. In particular, LP-S shows average gaps between 17% and 20%, and Lag never improves on the LP relaxation gap for any of these instances. Performing CPLEX at the root node only slightly improves on these gaps, while Lag/Bdl provides the best lower bounds (with average gaps between 16% and 19%) at the expense of excessive computational times. The Lagrangian-based primal heuristic, Lag/H, identifies significantly better solutions than the root node heuristic of CPLEX, but the gaps exceed 10% on average.

5.4.2. Performance of Enumerative Methods. Table 8 reports average CPU times and node counts

Table 7 Run Times and Gaps at the Root Node for Large Gap Instances with Single Assignment

Instances	LP-S	MIP^0-S	Lag	Lag/Bdl	MIP^0-S/H	Lag/H
Large A-S						
Gap (%)	17.15	17.01	17.15	16.27	22.24	9.52
Time (s)	1.19	5.35	2.79	2,581.82	5.35	3.22
Large B-S						
Gap (%)	19.36	19.26	19.36	18.92	23.73	13.20
Time (s)	1.38	5.51	2.92	2,479.71	5.51	3.25
Large C-S						
Gap (%)	19.14	19.03	19.14	18.76	25.75	12.46
Time (s)	1.81	6.80	3.16	2,566.94	6.80	3.39

Table 8 Run Times and Node Counts for Enumerative Methods on Large Gap Instances with Single Assignment

Instances	MIP-S	Lag/GUB
Large A-S		
Time (s)	3,622.68	1,293.26
Nodes	38,163.90	798.14
Large B-S		
Time (s)	4,009.84	1,203.34
Nodes	41,169.53	660.47
Large C-S		
Time (s)	4,892.40	1,593.91
Nodes	38,857.73	760.80

until completion when solving these large artificial instances of the TUFLP-S. The TUFLP-C formulation is not equivalent to TUFLP-S on these instances, so we only consider MIP-S solved by CPLEX and Lag/GUB. On average, the Lagrangian-based branch-and-bound method explores more than 50 times fewer nodes and uses one-third as much time as CPLEX on MIP-S.

One might attribute the superior performance of the specialized branch-and-bound algorithm on these instances to better upper bounds than those obtained by CPLEX, as shown by the results at the root node. However, by analyzing the evolution of the upper bounds when MIP-S is solved by CPLEX, we found that CPLEX quickly reaches (typically within one minute) the best upper bound computed by Lag/GUB. Since Lag lower bounds are dominated by those obtained by CPLEX at the root node, the explanation for the superiority of Lag/GUB lies in its aggressive search strategy, including the combination of GUB and polytomic branching schemes, the adaptation of reliability branching, and the best-first search strategy (the last two features make use of the information derived from the Lagrangian subproblem at each node). In particular, we compared the Lagrangian-based sorting of unreliable candidates with an LP-based one (see Section 4.2), and we measured a CPU time improvement of 15% on average on the large gap instances. Note that on industrial and gap instances, it was shown above, in Sections 5.1.2 and 5.2.2, that such an aggressive search strategy significantly reduces the number of nodes, compared to what CPLEX achieves on MIP-S. However, when CPLEX generates only a few hundred nodes, as is the case for most industrial and gap instances, this reduction in the number of nodes does not translate into an improvement in the computational time. For the difficult large gap instances, with or without explicit single-assignment constraints, CPLEX on formulation TUFLP-S generates more than 35,000 nodes, while Lag/GUB explores less than 1,000 nodes on average; this significant reduction now translates into a net gain in terms of the CPU time needed to prove the optimality of the solutions.

6. Conclusion

We addressed the two-level uncapacitated facility location problem with single-assignment constraints, a problem that arises in industrial applications in freight transportation (Gendron and Semet 2009) and in telecommunications (Chardaire, Lutton, and Sutter 1999). The problem can also be used to model two-level uncapacitated facility location problems without single-assignment constraints, where transportation costs are sums of per-arc costs and there are no assignment costs between depots and satellites (TUFLP-C). We showed that the LP relaxation of TUFLP-S is stronger than the LP relaxation of the usual MIP model used in this case, at the expense of increasing the size of the formulation.

We presented a Lagrangian relaxation approach for which the Lagrangian subproblem reduces to a single-level uncapacitated facility location problem. The Lagrangian dual is solved with a fast two-step heuristic; in the first step, the LP relaxation is solved, while the second step solves a single Lagrangian subproblem with Lagrange multipliers initialized with the LP-optimal dual solution.

We also developed a Lagrangian heuristic that includes an MIP-based LNS heuristic that solves a series of small UFLPs. The dual and primal bounds thus obtained were embedded within a specialized branch-and-bound method that implements two branching strategies: the GUB branching strategy and the polytomic branching strategy. The latter can be used alone or combined with the first strategy.

We presented and analyzed computational results on four sets of instances. On instances derived from a freight transportation application (Gendron and Semet 2009), the Lagrangian heuristic, without any branching, provides lower and upper bounds that are within 1% of optimality on average. On these instances, the Lagrangian lower bound improves on the (already strong) LP bound. The Lagrangian heuristic computes better bounds (both lower and upper) than CPLEX at the root node, and in less time. On these instances, the specialized branch-and-bound method reduces significantly the number of nodes compared with CPLEX, but its CPU times are nevertheless higher. On difficult artificial instances, the combined polytomic/GUB branching strategy performs well: compared with CPLEX on the TUFLP-S formulation, the number of nodes is significantly reduced, and the CPU times comparable, if not shorter. Since these instances can be cast as TUFLP-C, our experiments showed that a weaker, but smaller, formulation for the problem is solved more efficiently by CPLEX than both CPLEX on the TUFLP-S model and our specialized branch-and-bound method. Finally, on large, even more difficult, artificial instances that cannot be cast as TUFLP-C instances, our specialized branch-and-bound method outperforms CPLEX on the TUFLP-S model, both in terms of the number of nodes and the CPU times.

The performance of the specialized branch-and-bound method could be improved by adaptive parameter tuning and branching strategy choice, along with further refinements. In particular, it is likely that a specialized UFLP solver would be beneficial, as nearly half the run time of the specialized branch-and-bound methods is used to solve such problems. Nevertheless, our specialized branch-and-bound method exhibits better scaling properties to large and difficult instances than CPLEX on the TUFLP-S model.

Acknowledgments

Financial support for this project was provided by the Natural Sciences and Engineering Research Council of Canada and by the Fonds de recherche du Québec en nature et technologies. This research was also partly supported by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais region, the European Community, the Regional Delegation for Research and Technology, the French Ministry of Higher Education and Research, and the National Center for Scientific Research. The authors thank Antonio Frangioni for providing his bundle code. The authors also thank two anonymous referees whose comments helped improve this paper.

References

Aardal K, Labbé M, Leung JMY, Queyranne M (1996) On the two-level uncapacitated facility location problem. *INFORMS J. Comput.* 8(3):289–301.

Achterberg T, Koch T, Martin A (2005) Branching rules revisited. *Oper. Res. Lett.* 33(1):42–54.

Applegate D, Bixby RE, Chvátal V, Cook W (1995) Finding cuts in the TSP. DIMACS Technical report 95-05, Rutgers University, Piscataway, NJ.

Barahona F, Chudak F (2005) Near-optimal solutions to large-scale facility location problems. *Discrete Optim.* 2(1):35–50.

Barros AI (1995) Discrete and fractional programming techniques for location models. Unpublished doctoral thesis, Erasmus University Rotterdam, Rotterdam, Netherlands.

Barros AI, Labbé M (1994) A general model for the uncapacitated facility and depot location problem. *Location Sci.* 2(3):173–191.

Barros AI, Dekker R, Scholten V (1998) A two-level network for recycling sand: A case study. *Eur. J. Oper. Res.* 110(2):199–214.

Beltran-Royo C, Vial J-P, Alonso-Ayuso A (2012) Semi-Lagrangian relaxation applied to the uncapacitated facility location problem. *Comput. Optim. Appl.* 51(1):387–409.

Benichou M, Gauthier JM, Girodet P, Hentges G, Ribiere G, Vincent O (1971) Experiments in mixed-integer linear programming. *Math. Programming* 1(1):76–94.

Bloemhof-Ruwaard JM, Salomon M, Van Wassenhove LN (1994) On the coordination of product and by-product flows in two-level distribution networks: model formulations and solution procedures. *Eur. J. Oper. Res.* 79(2):325–339.

Bloemhof-Ruwaard JM, Salomon M, Van Wassenhove LN (1996) The capacitated distribution and waste disposal problem. *Eur. J. Oper. Res.* 88(3):490–503.

Bumb A (2001) An approximation algorithm for the maximization version of the two level uncapacitated facility location problem. *Oper. Res. Lett.* 29(4):155–161.

Chardaire P, Lutton JL, Sutter A (1999) Upper and lower bounds for the two-level simple plant location problem. *Ann. Oper. Res.* 86:117–140.

Cornuéjols G, Nemhauser GL, Wolsey LA (1991) The uncapacitated facility location problem. Michandani PB, Francis RL, eds. *Discrete Location Theory* (John Wiley and Sons, New York), 119–171.

Farahani R, Hekmatfar M, Fahiminia B, Kazemzadeh N (2014) Hierarchical facility location problem: Models, classifications, techniques, and applications. *Comput. Indust. Engrg.* 68(1): 104–117.

Frangioni A (1996) Solving semidefinite quadratic problems within nonsmooth optimization algorithms. *Comput. Oper. Res.* 23(11):1099–1118.

Frangioni A (2005) About Lagrangian methods in integer optimization. *Ann. Oper. Res.* 139(1):163–193.

Gabor AF, van Ommeren J-KCW (2010) A new approximation algorithm for the multilevel facility location problem. *Discrete Appl. Math.* 158(5):453–460.

Gao LL, Robinson EP Jr (1992) A dual-based optimization procedure for the two-echelon uncapacitated facility location problem. *Naval Res. Logist.* 39(2):191–212.

Gendron B, Semet F (2009) Formulations and relaxations for a multi-echelon capacitated location-distribution problem. *Comput. Oper. Res.* 36(5):1335–1355.

Gendron B, Khuong PV, Semet F (2015) Multilayer variable neighborhood search for two-level uncapcaitated facility location problems with single assignment. *Networks* 66(3):214–234.

Hansen P, Brimberg J, Urosevic D, Mladenovic N (2007) Primal-dual variable neighborhood search for the simple plant-location problem. *INFORMS J. Comput.* 19(4):552–564.

Ignacio AAV, Filho VJMF, Galvão RD (2008) Lower and upper bounds for a two-level hierarchical location problem in computer networks. *Comput. Oper. Res.* 35(6):1982–1998.

Kaufman L, Eede MV, Hansen P (1977) A plant and warehouse location problem. *Oper. Res. Quart.* 28(3):547–554.

Klose A, Drexl A (2005) Facility location models for distribution system design. *Eur. J. Oper. Res.* 162(1):4–29.

Kochetov Y, Ivanenko D (2005) Computationally difficult instances for the uncapacitated facility location problem. Ibaraki T, Nonobe K, Yagiura M, eds. *Metaheuristics: Progress as Real Problem Solvers*, Oper. Res./Comput. Sci. Interfaces Series, Vol. 32 (Springer Science+Business Media, New York), 351–367.

Körkel M (1989) On the exact solution of large-scale simple plant location problems. *Eur. J. Oper. Res.* 39(2):157–173.

Krarup J, Pruzan PM (1983) The simple plant location problem: Survey and synthesis. *Eur. J. Oper. Res.* 12(1):36–81.

Landete M, Marín A (2009) New facets for the two-stage uncapacitated facility location polytope. *Comput. Optim. Appl.* 44(3): 487–519.

Letchford AN, Miller SJ (2012) Fast bounding procedures for large instances of the simple plant location problem. *Comput. Oper. Res.* 39(5):985–990.

Letchford AN, Miller SJ (2014) Fast bounding procedures for large instances of the simple plant location problem. *Eur. J. Oper. Res.* 234(3):674–682.

Maric M (2010) An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem. *Comput. Informatics* 29(2):183–201.

Marín A (2006) Lower bounds for the two-stage uncapacitated facility location problem. *Eur. J. Oper. Res.* 179(3):1126–1142.

Marín A, Pelegrin B (1999) Applying Lagrangian relaxation to the resolution of two-stage location problems. *Ann. Oper. Res.* 86:179–198.

Melo MT, Nickel S, Saldanha da Gama F (2009) Facility location and supply chain management—A review. *Eur. J. Oper. Res.* 196(2):401–412.

Mitropoulos P, Giannikos I, Mitropoulos I (2009) Exact and heuristic approaches for the locational planning of an integrated solid waste management system. *Eur. J. Oper. Res.* 9(3):329–347.

Narula SC, Ogbu UI (1979) An hierarchical location—Allocation problem. *Omega* 7(2):137–143.

Pirkul H, Jayaraman V (1996) Production, transportation and distribution planning in a multi-commodity tri-echelon system. *Transportation Sci.* 30(4):291–302.

Pirkul H, Jayaraman V (1998) A multi-commodity multi-plant capacitated facility location problem: Formulation and efficient heuristic solution. *Comput. Oper. Res.* 25(10):869–878.

Posta M, Ferland JA, Michelon P (2014) An exact cooperative method for the simple plant location problem. *Math. Programming Comput.* 6(3):199–231.

Ro HB, Tcha DW (1984) A branch and bound algorithm for the two-level uncapacitated facility location problem with some side constraints. *Eur. J. Oper. Res.* 18(3):349–358.

Sahin G, Süral H (2007) A review of hierarchical facility location models. *Comput. Oper. Res.* 34(8):2310–2331.

Zhang J (2006) Approximating the two-level facility location problem via a quasi-greedy approach. *Math. Programming A* 108(1):159–176.