INFORMS Journal on Computing
Vol. 19, No. 4, Fall 2007, pp. 552–564
ISSN 1091-9856 | EISSN 1526-5528 | 07 | 1904 | 0552

DOI 10.1287/ijoc.1060.0196
© 2007 INFORMS

Primal-Dual Variable Neighborhood Search for the Simple Plant-Location Problem

Pierre Hansen
GERAD and HEC Montréal, Montréal, Quebéc H3T 2A7, Canada, pierre.hansen@gerad.ca

Jack Brimberg
GERAD and Royal Military College of Canada, Kingston, Ontario K7K 5L0, Canada, jack.brimberg@rmc.ca

Dragan Urošević
Mathematical Institute, SANU, Belgrade, Serbia, draganu@mi.sanu.ac.yu

Nenad Mladenović
School of Mathematics, Brunel University, West London,
Uxbridge UB8 3PH, United Kingdom, nenad.mladenovic@brunel.ac.uk

The variable neighborhood search metaheuristic is applied to the primal simple plant-location problem and to a reduced dual obtained by exploiting the complementary slackness conditions. This leads to (i) heuristic resolution of (metric) instances with uniform fixed costs, up to $n = 15{,}000$ users, and $m = n$ potential locations for facilities with an error not exceeding 0.04%; (ii) exact solution of such instances with up to $m = n = 7{,}000$; and (iii) exact solutions of instances with variable fixed costs and up to $m = n = 15{,}000$.

Key words: metaheuristics; variable-neighborhood search; primal-dual methods; branch and bound; simple plant-location problem
History: Accepted by Michel Gendreau, Area Editor for Heuristic Search and Learning; received July 2004; revised February 2005; accepted June 2006. Published online in *Articles in Advance* July 20, 2007.

1. Introduction

The simple plant-location problem (SPLP), also known as the uncapacitated facility-location problem, is one of the fundamental and most studied models in facility-location theory. The objective is to choose, from a set of potential facility-locations on a network, which ones to open to minimize the sum of opening (or fixed) costs and service (or variable) costs to satisfy the known demands from a set of customers. Although the origins of the plant-location problem go back to the pioneering work of Weber (1909), the actual formulation of SPLP may be attributed to Stollsteimer (1963), Kuehn and Hamburger (1963), and Balinski (1965). Since that time numerous articles have been published that deal with the properties and solution of the mathematical model, e.g., see the surveys in Krarup and Pruzan (1983), Labbé et al. (1995), Labbé and Louveaux (1997), and the books by Mirchandani and Francis (1990), Francis et al. (1992), and Daskin (1995). Despite the inherent assumptions of the model that may limit its practicality, the SPLP has gained considerable importance as a basic model in several combinatorial problems dealing, e.g., with vehicle dispatching, set covering, set partitioning, assortment, and, more recently, information retrieval and data mining (Pentico 1976, 1988; Jones et al. 1995; Tripathy et al. 1999). The SPLP is also used in multicriteria extensions (Brimberg and ReVelle 1998, 2000; Myung et al. 1997), and is shown to be an embedded problem in a number of types of location problems (ReVelle and Laporte 1996).

Most articles dealing with SPLP are concerned with solving the mathematical program. An exact branch-and-bound procedure was first proposed by Efroymson and Ray (1966), and later by Khumawala (1972). A dual-based model, the well-known DUALOC algorithm, was developed by Erlenkotter (1978) and a similar version by Bilde and Krarup (1977). The main idea here is to solve a reduced nonlinear form of the dual-based model heuristically by a simple ascent and adjustment procedure that often produces the optimal dual solution, which in turn often corresponds directly to the optimal primal integer solution. Otherwise, a branch-and-bound procedure is implemented to complete the solution. Refinements to the dual approach allowed Körkel (1989) to solve exactly much larger instances than previously attempted with sizes on the order of $1{,}500 \times 1{,}500$. A primal-dual algorithm by Galvão and Raggi (1989) alternates between the primal and dual problems solving each in turn heuristically. Again a branch-and-bound procedure is used as required to close the process. More recently an exact algorithm has been proposed by Goldengorin et al. (2003a) that is based on a pseudo-Boolean representation of the problem due originally to Hammer (1968). A data-correcting algorithm by Goldengorin et al. (2003b) may be used for an exact or approximate solution of the problem; however, the instances tested are relatively small.

A direct solution approach to SPLP involves solving the primal LP relaxation followed by branch and bound on the fractional values as in Morris (1978). The relaxation is known often to yield an integer solution (or zero duality gap), a property referred to as the *integer friendliness* of the model structure (ReVelle 1993). Tests in Brimberg and ReVelle (2000) on problem instances of up to 100 facility nodes × 300 customer nodes revealed that, in the majority of cases where fractional solutions were obtained, the optimal solution was found after branching off a single variable. However, as the SPLP is NP-hard (Cornuéjols et al. 1990), the number of branchings tends to grow exponentially with problem size. Nonetheless, the latest general mixed integer LP solvers such as CPLEX 8.1 are now able to handle problem sizes up to $1{,}000 \times 1{,}000$ by using built-in efficient branch-and-cut routines.

For NP-hard problems such as SPLP, it becomes necessary to use approximate methods to solve large instances. Earlier heuristics typically involve a local search on the primal problem using *add*, *drop*, and *interchange* moves (Kuehn and Hamburger 1963, Feldman et al. 1966, Teitz and Bart 1968, Manne 1964). See Cornuéjols et al. (1977) for an analysis of the worst case behavior of such methods. Later methods include Lagrangian-based heuristics (Beasley 1993), a projection method (Conn and Cornuéjols 1990), and more recently the application of metaheuristics such as Tabu search (Goncharov and Kochetov 1999, 2000; Michel and van Hentenryck 2004; Ghosh 2003), a genetic algorithm (Kratica et al. 2001) and volume algorithms with random rounding (Barahona and Chudak 2000).

In the next section we review the formulation of the SPLP, which is given by a mixed binary integer LP, and discuss several known linear and nonlinear formulations of the relaxed dual that will be required later. In Section 3 the rules of our variable neighborhood search (VNS) applied to the SPLP are outlined, followed by an explanation of the steps of a reduced VNS (RVNS) and the decomposition approach (VNDS), two variants of VNS designed for solving large problem instances. Guaranteed bounds of the heuristic solution may be obtained by solving the relaxed LP, either in the primal or dual space. However, for very large problem instances, it is not possible to store all variables in memory, or solve the LP in reasonable time. We may obtain approximate bounds (Erlenkotter 1978), but these will be less effective when a branch-and-bound phase is used later to find the optimal solution. Hence, in Section 4, we proceed as follows: (i) an initial dual solution (not necessarily feasible) is obtained from the primal VNDS solution using the complementary slackness conditions; (ii) a variable neighborhood descent (VND) is applied in the dual space to improve the solution. Section 5 then solves the problem exactly as follows: (iii) an exact solution of the dual is obtained using the last solution as the starting point and a new *sliding simplex* method developed by us that is able to reduce the size of the dual considerably; (iv) finally, reverting back to the primal problem, a branch-and-bound algorithm is used, strengthened by a tight lower bound from the dual and upper bound from the earlier heuristic (VNDS) solution.

Computational results are reported in Section 6 on a series of randomly-generated problems, including instances that are much larger than previously tested in the literature. The ability to solve very large SPLPs is becoming more important in view of the size of facility-location problems being tackled in practice today, and the fact that the SPLP is finding other applications in such areas as cluster analysis, computer and telecommunications network design, information retrieval, and data mining. Section 7 summarizes the main results and offers possible directions of future research.

2. Mathematical Model

Let $I = \{1, 2, \ldots, m\}$ denote a set of potential facilities, and $J = \{1, 2, \ldots, n\}$ a set of users or customers on a network. The SPLP is

$$\min_{x,y} \quad z_P = \sum_{i\in I} f_i y_i + \sum_{i\in I}\sum_{j\in J} c_{ij} x_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \quad \forall j \in J; \tag{2}$$

$$x_{ij} - y_i \le 0, \quad \forall i \in I, \forall j \in J; \tag{3}$$

$$y_i \in \{0, 1\}, \quad \forall i \in I; \tag{4}$$

$$x_{ij} \ge 0, \quad \forall i \in I, \forall j \in J, \tag{5}$$

where

f_i denotes the fixed cost for opening facility i;

c_{ij} is the distribution cost for satisfying the demand of user j from facility i;

y_i is a boolean variable equal to 1 if facility i is opened, and 0 otherwise;

x_{ij} is the fraction of demand of user j satisfied from facility i.

This problem has $mn + m$ variables and $mn + n$ constraints. The objective function expresses that the sum of fixed costs to open facilities and distribution costs for these facilities to serve the users must be minimized. The f_i and c_{ij} are assumed to be positive. Constraint (2) imposes that the demand at each customer is satisfied. Constraint (3) opens facility i by setting $y_i = 1$ if there is a flow from this facility to any customer j ($x_{ij} > 0$). Once the y_i are fixed, the values of the x_{ij} are easily determined; for each j, $x_{ij} = 1$ for that open facility with minimum distribution cost c_{ij} (ties may be broken arbitrarily); all other $x_{ij} = 0$.

If the binary constraints in (4) are relaxed to $0 \le y_i \le 1$, $\forall\, i \in I$, one gets the *strong linear programming relaxation*. Another relaxation, called the *weak linear programming relaxation*, may be obtained by also replacing the constraints in (3) by $\sum_{j\in J} x_{ij} \le n y_i$, $\forall\, i \in I$, so that the number of original constraints is reduced from $mn + n$ to $m + n$. But this latter relaxation is less tight than the strong one, and does not appear to be of interest. The strong LP relaxation is known to be *integer-friendly* (ReVelle 1993, Brimberg and ReVelle 2000); i.e., most variables y_i take an integer value in the optimal vector, and thus, the duality gap is also small and often zero.

A particular class of test problems has been studied in depth (Barahona and Chudak 2000) and often used in empirical studies. Potential sites for facilities coincide with the given locations of the users, and are points taken from a uniform distribution on the unit square. Distribution costs c_{ij} are Euclidean distances between i and j; fixed costs are equal for all facilities, and set at $\sqrt{n}/10$, $\sqrt{n}/100$, or $\sqrt{n}/1{,}000$. It is proven that any branch-and-bound algorithm using the strong relaxation (without further cutting planes) will require a number of branches that increases exponentially with m (or n). Nevertheless, near-optimal solutions may be readily obtained for fairly large instances. For example, Barahona and Chudak (2000) recently solved, with an instance-dependent error of at most 1%, problems with $m = n$ up to 3,000.

2.1. Dual Formulations

The dual of the *strong LP relaxation* is

$$\max_{v,\,w,\,t} \left(\sum_{j\in J} v_j - \sum_{i\in I} t_i\right) \tag{6}$$

$$\text{s.t.}\ \sum_{j\in J} w_{ij} - t_i \le f_i, \quad \forall\, i \in I \tag{7}$$

$$v_j - w_{ij} \le c_{ij}, \quad \forall\, i \in I,\ \forall\, j \in J \tag{8}$$

$$t_i,\, w_{ij} \ge 0, \quad \forall\, i \in I,\ \forall\, j \in J. \tag{9}$$

Note that the variables v_j are not restricted in sign. However, because the equality sign in (2) may be replaced by $\ge$ without affecting the optimal solution, the v_j will be nonnegative. This problem has $mn + m + n$ variables and $mn + m$ constraints. Thus, as in the primal, it is large in both dimensions. Fortunately, the dual may be simplified in various ways. First observe that each variable t_i appears only in the objective function, with a negative sign, and in a single constraint in (7). Further examination shows that the t_i may be reduced one at a time without reducing the objective function's value. Using (7) and (9), we have

$$t_i = \max\left\{\sum_{j\in J} w_{ij} - f_i,\, 0\right\} = \left(\sum_{j\in J} w_{ij} - f_i\right)^+, \tag{10}$$

where $a^+ = \max\{a, 0\}$. It follows that in the optimal solution the t_i should all be zero, yielding the simpler LP formulation of the dual that usually appears in the literature:

$$\max_{v,\,w}\ z_D = \sum_{j\in J} v_j \tag{11}$$

$$\text{s.t.}\ \sum_{j\in J} w_{ij} \le f_i, \quad \forall\, i \in I \tag{12}$$

$$v_j - w_{ij} \le c_{ij}, \quad \forall\, i \in I,\ \forall\, j \in J \tag{13}$$

$$w_{ij} \ge 0, \quad \forall\, i \in I,\ \forall\, j \in J. \tag{14}$$

Erlenkotter (1978) observed that for any fixed vector of v_j's, the w_{ij} may be reduced without affecting feasibility, i.e., the w_{ij} may be made as small as possible. Thus, constraints (13) and (14) imply that we should set

$$w_{ij} = \max\{v_j - c_{ij},\, 0\} = (v_j - c_{ij})^+, \quad \forall\, i, j. \tag{15}$$

Now substitute (15) into (12) to get a nonlinear-programming formulation in n variables with m constraints, known as the *restricted dual*:

$$\max_{v}\ \sum_{j\in J} v_j \tag{16}$$

$$\text{s.t.}\ \sum_{j\in J} (v_j - c_{ij})^+ \le f_i, \quad \forall\, i \in I. \tag{17}$$

Another way to obtain a restricted dual is to substitute both (10) and (15) into the standard LP formulation (6)–(9) to get the unconstrained nonlinear program in n variables (Spielberg 1969, Conn and Cornuéjols 1990):

$$\max_{v} F(v) = \sum_{j\in J} v_j - \sum_{i\in I}\left(\max\left\{\sum_{j\in J}(v_j - c_{ij})^+ - f_i,\, 0\right\}\right). \tag{18}$$

Note that this formulation is equivalent to one suggested by Karkazis (1985),

$$\max_{v} F(v) = \sum_{j\in J} v_j + \sum_{i\in I}\left(\min\left\{f_i - \sum_{j\in J}(v_j - c_{ij})^+,\, 0\right\}\right)$$

since $-\max\{x\} = \min\{-x\}$ always holds.

It is well known that $F(v)$ is a piecewise linear concave objective function in n variables, and several nonlinear-programming methods that use specifics of the problem have been suggested to maximize it. For example, in Karkazis (1985) and Klose (1995) subgradient methods have been derived, and in Conn and Cornuéjols (1990) a gradient-projection method is proposed.

3. Solving the Primal Problem by Variable Neighborhood Search

VNS is a recent metaheuristic (see Mladenović and Hansen 1997, and the surveys in Hansen and Mladenović 2001, 2003), whose basic idea is to use systematically different neighborhoods, both in descent phases and to jump out of local optimum traps.

To apply VNS to our problem, the same data structure is used as suggested in Voss (1996) and in Hansen and Mladenović (1997) for the p-median problem:

- Indices of facilities in the current and in the best known solution are stored in arrays s and s^*.
- Indices of closest and second-closest open facilities for each user and associated costs are stored in arrays $\underline{c}$ and $\bar{c}$, respectively.

3.1. A VNS Heuristic for SPLP

To apply VNS, a neighborhood structure must be defined on the solution space. Let S denote any subset of open facilities ($S \subseteq I$); the solution space $\mathscr{X}$ may then be defined as all such possible subsets. The total number of solutions in $\mathscr{X}$ is $2^m - 1$. To define the neighborhoods, we use a distance function. Let S_1, S_2 be any two solutions in $\mathscr{X}$; the distance between them is defined by $\rho(S_1, S_2) = |(S_1 \backslash S_2) \cup (S_2 \backslash S_1)|$. If S_2 is obtained from S_1 by closing one facility $i_1 \in S_1$ and opening an $i_2 \in I \backslash S_1$ (an interchange: $S_2 = (S_1 \backslash \{i_1\}) \cup \{i_2\}$), $\rho(S_1, S_2) = 2$; if $S_2 = S_1 \backslash \{i_1\}$ (a drop) or $S_2 = S_1 \cup \{i_2\}$ (an add), $\rho(S_1, S_2) = 1$. The kth neighborhood of a current solution S is defined as the set of all possible solutions S' derived from S by any combination of exactly k total-interchange, drop, or add moves.

The basic steps of VNS consist of a repetitive sequence of (i) shaking to a kth neighborhood of the incumbent solution; (ii) conducting a local search from the perturbed solution using the first neighborhood; and (iii) moving to a better local optimum if one is found.

(i) **Shaking.** To get a random point S' in the kth neighborhood of the incumbent solution S (which corresponds to distance at most $2k$), the following steps are repeated k times:

- Choose a facility i_1 at random from S equiprobably.
- Choose a facility i_2 at random from $I \backslash S$ equiprobably.
- Generate a uniform random number rnd from the interval $(0, 1)$.
- If $rnd \leq 0.2$, delete i_1 from the solution ($p \leftarrow p-1$); if $rnd \geq 0.8$, add i_2 to the solution ($p \leftarrow p+1$); if $rnd \in (0.2, 0.8)$, interchange positions i_1 and i_2 in s, i.e., close facility i_1 and open facility i_2.
- Update the arrays for first and second closest facilities w.r.t. the new open facilities.

(The values 0.2 and 0.8 used in choosing the move have been obtained empirically by comparing results for threshold values distant of 0.1 at the time. Values 0.2 and 0.8 imply interchanges are more frequent in shaking than opening or closing facilities yet the number of these may vary.)

(ii) **Local search.** A local search is conducted from the perturbed solution S' using the first neighborhood, $\mathcal{N}_1(S')$. In the *best-improvement* version of the local search that we are using, all $p(m-p)+m$ solutions from $\mathcal{N}_1(S')$ are visited, and a move made to the best among them only if its objective-function value is smaller than that of S'. A fast-interchange version, as proposed in Whitaker (1983) and Hansen and Mladenović (1997) for solving the p-median, is applied. The local search is repeated after each downward move until a local minimum is reached.

(iii) **Move or not.** The simplest acceptance criterion for basic VNS is used. A move is made only if the local search in (ii) obtains a better solution than the incumbent S. Each time a move is made, k is reset to $k_{\min}$ (a parameter typically 1); otherwise k is changed (typically augmented by one until a parameter $k_{\max}$ is reached, after which it is reset to $k_{\min}$), and the cycle is repeated. The search is terminated after a stopping criterion, such as a limit on execution time or on the number of iterations without an improvement, is reached.

3.2. Variable Neighborhood Decomposition Search

Reduced variable neighborhood search (RVNS) and variable neighborhood decomposition search (VNDS) are two variants of VNS devoted to solving large problem instances. In RVNS, we simply skip the local-search phase of the basic VNS.

Our procedure first obtains an initial solution with RVNS. Two parameters are specified for RVNS: the maximum neighborhood distance, $k'_{\max}$, for the shaking operation, and a stopping criterion based on the maximum number of iterations $i_{\max}$ allowed between two improvements. A suitable compromise between speed and quality was found experimentally to be $k'_{\max} = 2$ and $i_{\max} = 30$ or 20 seconds elapsed time. This shows that shaking must remain moderate to get an imprived solution without a descent phase. Once RVNS is executed, we proceed with our decomposition heuristic as outlined below:

1. **Initialization.** Choose values for the two parameters $\ell_{\max}$ (maximum number of open facilities to

be selected from the incumbent solution) and $t_{\max}$ (maximum computing time for the heuristic). Set the incumbent solution S to be the set of open facilities obtained by RVNS, and let $p = |S|$. Set the size of the decomposed problem, $\ell = 2$.

2. **Constructing the decomposed problem.** (i) Determine the $(p-1) \times p$ matrix $R = [r_{ij}]$ of ordered network distances where column j is assigned the jth facility listed in S, row i is reserved for the ith closest facility in S to each facility j, $i = 2, \ldots, p$, and r_{ij} is the corresponding network distance. (Note: Also save facility indices in R.)

(ii) Determine $r_{\ell j^*} = \min_{1 \le j \le p}\{r_{\ell j}\}$.

(iii) The subproblem and its initial solution (D) are defined as follows:

- The open facilities are given by the facility assigned column j^* and the 2nd, 3rd, …, ℓth closest facilities to it (the first $\ell - 1$ entries in column j^*).
- The subset of users consists of the n' ones assigned in the incumbent solution S to the subset of ℓ open facilities just identified.
- Additional potential facility sites are added by a subroutine.

3. **Solving the decomposed problem.** If the total number of facility sites (open or closed) in the subproblem, $m' \le 1{,}000$, solve it by VNS; if $1{,}000 < m' \le 1{,}500$, solve it by RVNS; else set $\ell = 2$ and return to step 2 (the decomposed problem is too big).

4. **Move or not.** If the new solution D' is better than D, proceed to step 5; else if $\ell = \ell_{\max}$, set $\ell = 2$; else set $\ell \leftarrow \ell + 1$. If $t < t_{\max}$, return to step 2; else stop.

5. **Adjusting for boundary effect.** Add the new decomposed solution D' to the fixed portion of S ($S \leftarrow (S \backslash D) \cup D'$). Conduct a local search from the new solution to obtain a local optimum S'. Set $S = S'$. If $t < t_{\max}$, set $\ell = 2$ and return to step 2; else stop.

Note that the new set of facilities obtained in the subproblem may influence users not considered in the subproblem, i.e., some users may change assignment with respect to this new solution. Such a *boundary effect* is accounted for by updating arrays $\underline{c}$ and $\bar{c}$ in the whole space each time a new improved solution in the subproblem is found.

4. A VNS Heuristic for the Dual

4.1. Initial Dual Solution

Guaranteed performance of the primal heuristic may be determined if a lower bound on the objective function value is known. To that end the standard approach is to relax the integrality condition on the y_i variables. The well-known integer-friendliness property ensures that the strong LP relaxation for the SPLP gives a small duality gap between the optimal integer and relaxed solutions. We may then evaluate the existing gap as a percentage: $100 \times (z_h - z_r)/z_r$, where z_h denotes the solution obtained by the heuristic and z_r the solution of the strong LP relaxation, to determine the maximum error obtained by the heuristic solution.

Let us consider the dual of the strong LP relaxation, (11)–(14). For large problems (say $n = m = 1{,}500$) finding the exact solution of the primal or dual by some general LP solver such as CPLEX would be impossible or, at best, very time consuming. Thus, we first develop some procedures that will take into account the primal solution (y, x) found by the heuristic, and avoid solving completely the dual problem at this stage.

The *complementary-slackness* conditions for the SPLP are

$$v_j\left(\sum_{i\in I} x_{ij} - 1\right) = 0, \quad \forall\, j \in J \tag{19}$$

$$w_{ij}(y_i - x_{ij}) = 0, \quad \forall\, i \in I,\ \forall\, j \in J \tag{20}$$

$$\left(f_i - \sum_{j\in J} w_{ij}\right) y_i = 0, \quad \forall\, i \in I \tag{21}$$

$$(c_{ij} - v_j + w_{ij}) x_{ij} = 0, \quad \forall\, i \in I,\ \forall j \in J, \tag{22}$$

where (y, x) and (v, w) denote the associated primal and dual solutions (feasible or not), respectively.

The strong duality theorem ($z_P^* = z_D^*$) is obtained by summing first each of (19)–(22) and then summing their left- and right-hand sides. In this proof all four complementary slackness conditions (19)–(22) are needed. However, (20), (21), and (22) are not necessarily true if we add integrality constraints on the primal variables y_i, and that is the source of the duality gap $z_P - z_D$. Since the primal solution is feasible, (19) is automatically satisfied.

Mladenović et al. (2006) prove following proposition. Here I^+ denotes the set of open facilities, and $I^- = I \backslash I^+$, the set of closed ones, in the heuristic solution (y, x).

Proposition 1. *If $|I^+| \ge 2$ and a feasible primal solution is such that*

$$\sum_{j\in J} (\bar{c}_j - c_{ij})^+ \le f_i, \quad \forall\, i \in I^-, \tag{23}$$

then (y, x) is an optimal solution of the strong LP relaxation of SPLP.

Therefore, having a set of open facilities I^+ and associated vector of second-closest distances $\bar{c}$ obtained by VNDS, we first check if (23) is satisfied, and if that is the case, the incumbent solution solves SPLP optimally and no further work is required.

Otherwise, our next objective is to derive an approximate dual solution from the primal (heuristic) solution. What makes our procedure new is that the dual solution does not have to be feasible. In effect, we take advantage of two expected conditions: (a) the primal VNDS solution is very close to optimum; (b) the duality gap is small. Thus, by finding a dual solution with the same objective-function value as the primal, we expect to be close in the dual solution space to the optimal (feasible) dual solution. To accomplish this, the complementary-slackness conditions must be satisfied.

Proposition 2. *For a given primal solution y, let v be a corresponding dual solution such that*

$$\sum_{j\in J_i} v_j = f_i + \sum_{j\in J_i} \underline{c}_j, \quad \forall\, i \in I^+, \tag{24}$$

where J_i denotes the subset of users assigned to open facility i. Then $z_D(v) = z_P(y)$.

Proof.

$$z_D(v) = \sum_{j\in J} v_j = \sum_{i\in I^+}\sum_{j\in J_i} v_j = \sum_{i\in I^+}\left(f_i + \sum_{j\in J_i}\underline{c}_j\right) = z_P(y). \quad \square$$

We use the previous proposition to find an initial dual solution v that belongs to the intersection of the $p = |I^+|$ hyperplanes in (24). If, in addition, we impose the condition

$$\underline{c}_j \le v_j \le \bar{c}_j, \quad \forall\, j \in J, \tag{25}$$

it follows that all the complementary slackness conditions will be satisfied (see Mladenović et al. 2006). Thus, a dual solution that satisfies (24) and (25) also satisfies the four complementary slackness conditions (19)–(22). Also note that conditions (21) and (22) are usually used exclusively to get the primal solution for a given dual (Erlenkotter 1978), and that the transformation of a primal to a corresponding dual is usually done by setting $v_j = \underline{c}_j$, which is known to produce feasible but bad initial dual solutions (Galvão and Raggi 1989). Proposition 2 is capable of providing a better initial dual, but, since such a solution is not unique, alternative procedures must be investigated. We consider the following approaches.

(i) **Proportional formula.**

$$v_j = \underline{c}_j + \frac{f_i(\bar{c}_j - \underline{c}_j)}{\sum_{\ell\in J_i}(\bar{c}_\ell - \underline{c}_\ell)}, \quad \forall\, j \in J,\ i = i_j^+, \tag{26}$$

where i_j^+ denotes the closest open facility to j (i.e., the facility assigned to j). Summing the left and right sides of (26) over $j \in J_i$, it follows that (24) holds, $\forall\, i \in I^+$. We may also show that (25) is satisfied. Since the primal solution is a local minimum, we have (Mladenović et al. 2006) $\sum_{j\in J_i}(\bar{c}_j - \underline{c}_j) \ge f_i$, $\forall\, i \in I^+$ and thus, $\underline{c}_j < v_j \le \bar{c}_j$, $\forall\, j \in J$.

(ii) **Projection formula.** Given any dual solution $v'_j \in \mathbb{R}^n$, we may find its closest point that belongs to the manifold defined in (24) by

$$v_j = v'_j - \frac{1}{|J_i|}\left(\sum_{\ell\in J_i} v'_\ell - f_i - \sum_{\ell\in J_i}\underline{c}_\ell\right), \quad j \in J,\ i = i_j^+.$$

For example, we could select $v'_j = (\bar{c}_j + \underline{c}_j)/2$, $\forall\, j \in J$; i.e., take a point (v'_j) in the middle of the hypercube $H = \prod_{j=1}^n [\underline{c}_j, \bar{c}_j]$. Another possibility would be $v'_j = \max\{\underline{c}_j, \min\{\tilde{c}_j, \bar{c}_j\}\}$, $\forall\, j \in J$. In the last expression, $\tilde{c}_j$ is defined as $\tilde{c}_j = \min_{i\in I^-}\{c_{ij}\}$, $\forall\, j \in J$ (see Mladenović et al. 2006 for details).

4.2. Improving the Dual Solution

As seen above, the initial dual solution is easily obtained by closed formula; however, it will most likely be infeasible. To reduce this infeasibility, we consider the unconstrained dual function in (20):

$$F(v) = \sum_{j\in J} v_j - \sum_{i\in I}\left(\sum_{j\in J}(v_j - c_{ij})^+ - f_i\right)^+,$$

where the second term in the right side is the sum of infeasibilities. To maximize this function, we devise a powerful local search that uses variable neighborhood descent (VND) rules and four neighborhood structures designed for this purpose.

The first two neighborhoods represent *windows* around the current v_j in the ranked matrix $[c_{ij}]$. Letting i_j denote the lower index of the window, we obtain $c_{i_j, j} \le v_j \le c_{i_j+1, j}$, $\forall\, j \in J$. To simplify the notation, denote the last inequalities that define the window around the current dual value by $a_j \le v_j \le b_j$. The first neighborhood $N_1(v)$ is constructed by replacing v_j with a_j, i.e.,

$$N_1(v) = \{(a_1, v_2, \ldots, v_n), (v_1, a_2, \ldots, v_n), \ldots, (v_1, v_2, \ldots, a_n)\}.$$

In the same way, neighborhood $N_2(v)$ is obtained by replacing v_j with its upper window (one at the time):

$$N_2(v) = \{(b_1, v_2, \ldots, v_n), (v_1, b_2, \ldots, v_n), \ldots, (v_1, v_2, \ldots, b_n)\}.$$

The cardinality of each of these two neighborhoods equals n.

In the third neighborhood $N_3(v)$, the value of some variable v_j is increased by $\Delta v_j = \min\{b_j - v_j, \min_{i\in I,\, c_{ij}\le v_j}\Delta f_i\}$, where $\Delta f_i = (f_i - \sum_{j\in J}(v_j - c_{ij})^+)^+$. A move in N_3 will improve $F(v)$ without increasing the infeasibility of the solution.

In $N_4(v)$ the value of some variable v_j is decreased by

$$\Delta v_j = \min\left(\min_{i:\, v_j > c_{ij}}\left(\sum_{j\in J}(v_j - c_{ij})^+ - f_i\right)^+, v_j - \underline{c}_j\right). \tag{27}$$

The last formula needs to be explained in more detail. When v_j is reduced by some amount, then, in order to get a larger $F(v)$, we need to reduce at least two members of the sum $\sum_{i\in I}(\sum_{j\in J}(v_j - c_{ij})^+ - f_i)^+$. Those two members should then satisfy the conditions $v_j > c_{ij}$ and $\sum_{j=1}^{n}(v_j - c_{ij})^+ > f_i$. Thus, it may be possible to increase $F(v)$ by decreasing v_j according to (27).

The VND procedure first makes best improvement moves in the N_1 neighborhood of the current solution by examining all n points in that neighborhood. Once stalled, the procedure moves to the next neighborhood in the sequence (N_2, N_3, N_4), always reverting to N_1 when an improvement is found. The iterations end when no improvement is found consecutively in each of the four neighborhoods. Note that the output solution may still be infeasible.

5. Exact Solution Methods

5.1. Sliding Simplex for Exact Dual Solution

Earlier we saw that Erlenkotter's observation in (17) leads to a restricted dual ((18), (19)) with only n variables v_j and m constraints. However, the constraints are nonlinear due to the max operator. Combining a vector $v = (v_j)$ with relation (17) gives a dual solution (v, w) satisfying constraints (15) and (16), but not necessarily (14). Instead of trying to solve the restricted dual, we rewrite the original (linear) dual in a reduced form by taking advantage of the fact that (a) many of the constraints in (15) are nonbinding and may be eliminated; (b) for those that are binding, the w_{ij} may be eliminated by direct substitution. The reduced linear dual and its solution by a new sliding simplex approach is discussed next.

5.1.1. Reduced Dual. Suppose that

$$\underline{c}_j \le v_j \le \bar{c}_j, \quad \forall j \in J. \tag{28}$$

Let us then divide the set of users J into three subsets for each facility $i \in I$:

$$J_{i1} = \{j \in J \mid c_{ij} < \underline{c}_j\}, \qquad J_{i2} = \{j \in J \mid \underline{c}_j \le c_{ij} \le \bar{c}_j\},$$
$$J_{i3} = \{j \in J \mid \bar{c}_j < c_{ij}\}.$$

Using (15), it is immediately seen that $w_{ij} = 0$ for all users $j \in J_{i3}$. Also from $w_{ij} = (v_j - c_{ij})^+$, it holds that $w_{ij} = v_j - c_{ij}$, for all $j \in J_{i1}$, $i \in I^-$, since $v_j \ge \underline{c}_j$. Therefore, model (11)–(14) is reduced as follows:

$$\max_{v,\,w} \; z_D = \sum_{j\in J} v_j \tag{29}$$

$$\text{s.t.} \; \sum_{j\in J_{i1}} v_j + \sum_{j\in J_{i2}} w_{ij} \le f_i + \sum_{j\in J_{i1}} c_{ij}, \quad \forall i \in I \tag{30}$$

$$v_j - w_{ij} \le c_{ij}, \quad \forall i \in I,\; j \in J_{i2} \tag{31}$$

$$w_{ij} \ge 0, \quad \forall i \in I,\; j \in J_{i2}. \tag{32}$$

In our sliding simplex method, the w_{ij} variables corresponding $c_{ij} \notin [\underline{c}_j, \bar{c}_j]$ are removed with their constraints as in the above formulation, but now the bounds $\underline{c}_j$ and $\bar{c}_j$ are allowed to vary during the solution process to move towards the optimal solution while keeping a reasonable dimension on the problem size. To this end, it is necessary to rank the c_{ij} by nondecreasing values for each j. Using a second-level index for ranking, we have $c_{i_1 j} \le c_{i_2 j} \le \cdots \le c_{i_m j}$, $\forall j \in J$. Consider a value of $v_j \in [c_{i_1 j}, c_{i_m j}]$, and let k denote the largest index such that $c_{i_k j} \le v_j$. Then, we define the *ℓ-interval* of v_j to be

$$[c_{i_{k-\ell} j}, c_{i_{k+\ell} j}], \tag{33}$$

which contains the following values of the c_{ij}: $c_{i_{k-\ell} j}$, $c_{i_{k-\ell+1} j}, \ldots, c_{i_k j}, c_{i_{k+1} j}, \ldots, c_{i_{k+\ell} j}$. Note that there may be an adjustment necessary for border effect; i.e., some end terms are obviously omitted if $k - \ell < 1$ or $k + \ell > m$.

Setting $\underline{c}_j = c_{i_{k-\ell} j}$ and $\bar{c}_j = c_{i_{k+\ell} j}$ $\forall j \in J$, one gets the reduced ℓ-dual associated with vector v, as given in (29)–(32) and (28) with the subsets J_{i1}, J_{i2}, J_{i3}, $i = 1, \ldots, m$, updated appropriately.

The steps of the sliding simplex are as follows:

1. **Initialization.** For each $j \in J$ rank the c_{ij} in order of nondecreasing values (ties being broken arbitrarily). Record the values and corresponding indices as $c_{i_p j}$ and i_p for $p = 1, \ldots, |I|$ and all $j \in J$. Choose a value for parameter ℓ.
2. **Initial solution.** Obtain a vector v which corresponds to a feasible or infeasible solution (v, w) of the dual. Set up the first reduced ℓ-dual from v.
3. **Solution of the ℓ-dual.** Solve the current ℓ-dual using the simplex algorithm (e.g., with CPLEX) and the latest dual solution as starting solution to obtain a vector v^*.
4. **Optimality test.** Check for each $j \in J$, that the following condition holds: $v_j^* = c_{i_1 j}$ or $c_{i_{k-\ell} j} < v_j^* < c_{i_{k+\ell} j}$ or $v_j^* = c_{i_m j}$. If for some j it is not the case, go to step 5; otherwise go to step 6.
5. **Updating of the reduced ℓ-dual.** Update the index k and window in (33) for each j as required; reformulate the ℓ-dual accordingly and return to step 3.
6. **Output.** An optimal solution of the dual is given by (v^*, w^*) where $w_{ij}^* = \max\{v_j^* - c_{ij}, 0\}$, $\forall i \in I$, $\forall j \in J$; its value is $\sum_{j\in J} v_j^*$.

The sliding simplex method bears some resemblance to the BOXSTEP method of Marsten et al. (1975). Indeed, BOXSTEP proceeds by solving a sequence of problems with additional constraints defining a box around the current solution; if the solution of the current such problem is on the boundary, the box is translated. The sliding simplex method has some differences too: (i) it adds interval constraints on a small part of the variables only; (ii) the role of these constraints is to eliminate a large part of the original constraints and variables; and (iii) the problem obtained is a linear program solved by simplex instead of a nonlinear one solved by subgradient optimization.

Theorem 1. *The sliding simplex algorithm solves the dual of SPLP.*

Proof. From sensitivity analysis, one may add to the dual (11)–(14) without changing the optimal solution the set of constraints

$$c_{i_1 j} \le v_j \le c_{i_m j}, \quad \forall\, j \in J \tag{34}$$

as at least one y_i must be equal to 1 in any feasible solution. Then the current reduced ℓ-dual is equivalent to this problem with the additional constraints

$$c_{i_{k-\ell} j} \le v_j \le c_{i_{k+\ell} j}, \quad \forall\, j \in J, \tag{35}$$

where we assume that redundant constraints obtained when $c_{i_{k-\ell} j} = c_{i_1 j}$ or $c_{i_{k+\ell} j} = c_{i_m j}$ are omitted.

The optimal solution (v^*, w^*) of (11)–(14), (34), (35) is such that none of the constraints (35) are tight, as otherwise the condition of step 4 would not hold. So it remains optimal if those constraints are removed, i.e., for (11)–(14), (34) and hence for (11)–(14).

Furthermore, since there are a finite number of combinations of windows for v, and each combination may be encountered at most once, the algorithm must terminate after a finite number of iterations. □

5.2. Exact Primal Solution

At this stage we have a primal solution obtained by the VNDS heuristic that provides an upper bound, and an exact dual solution by sliding simplex that provides a lower bound for the optimal solution of the SPLP. If the two bounds are equal, the VNDS solution must be optimal. Otherwise, a classical branch-and-bound procedure is initiated. The tightness of the upper and lower bounds will be useful in keeping the number of branchings to a minimum. The main features of the branch-and-bound algorithm are as follows:

(a) For branching, the fractional primal variables (duals of the dual) are first identified, and those that correspond to open facilities in the heuristic solution are closed, one at a time, by a depth-first strategy.

(b) At each node of the branch-and-bound tree, a relaxed dual is solved by the sliding simplex method, described in the previous section using the solution of the parent node as the starting point.

(c) As in Erlenkotter (1978), to keep facility i closed, the fixed cost f_i is temporarily set to $+\infty$; to keep facility i open, the fixed cost f_i is set to 0.

(d) An elementary backtracking scheme with last-in, first-out is applied.

(e) Pruning of nodes in the branch-and-bound tree is either by bounding (relaxed solution is worse than upper bound) or by obtaining a primal integer solution (no fractional dual values of the dual problem).

Because the sliding simplex method may call the LP solver many times at each node as the windows on the v_j change, it is advisable after each call to check if the node can be fathomed by bounding, before proceeding further to the exact lower bound.

6. Computational Experience

In this section we first explain what type of test instances are used. They are available in the Online Supplement to this paper on the journal's website. Then we verify the capacity of the latest CPLEX 8.1 as a general mixed integer solver applied to the SPLP. Our main computer results on the specialized heuristics and algorithms described above are then given.

6.1. Barahona-Chudak Instances

Our procedure is tested on similarly-constructed instances from Barahona and Chudak (2000). That is, both facility and user points are assumed to be the same random uniformly-distributed set of vertices in the unit square. The fixed costs are the same for all facilities, and the transportation costs correspond to the Euclidean distances separating pairs of points in the plane. Several interesting properties of such test problems are described in Ahn et al. (1988), e.g.: (i) for $n \le 500$, the problems are easy to solve; (ii) when n is large, any enumerative method based on LP relaxation requires the exploration of an exponentially-increasing number of solutions; and (iii) the value of the LP relaxation is about 0.998 of the optimal value.

Three different types of instances based on different magnitudes of fixed cost are considered. These problems provide a wide range of structural diversity: (i) *Type I*, $f_i = \sqrt{n}/10$, $\forall\, i \in I$; (ii) *Type II*, $f_i = \sqrt{n}/100$, $\forall\, i \in I$; and (iii) *Type III*, $f_i = \sqrt{n}/1{,}000$, $\forall\, i \in I$. To avoid numerical problems, all data entries are made integer by rounding them to four significant digits. As noted in Barahona and Chudak (2000), the DUALOC heuristic seems to benefit most from such rounding. However, their results show that their *volume algorithm* together with *random rounding* (V&RRWC for short) outperforms significantly the *dual ascent* and *dual adjustment* heuristics of Erlenkotter (1978) on Type I and Type II instances. On Type III instances *dual adjustment* (DA) was the best. Experiments were performed on relatively large instances for the time, with $m = n \le 3{,}000$. For example, comparing the % gap (or guaranteed instance-dependent bounds) of DA and V&RRWC on all three types of instances for the maximum problem size considered $m = n = 3{,}000$, the following average results were reported: Type I: 16.79% for DA vs. 0.71% for V&RRWC; Type II: 4.27% vs. 0.93%; Type III: 0.62% vs. 0.85%.

Table 1 Testing CPLEX

f_i	n	z_{mip^*}	t_{mipopt}	z_{VNDS}	t_{VNDS}
$\sqrt{n}/10$	100	209,805.00	1.26	209,805.00	0.09
	200	361,531.00	7.19	361,531.00	0.75
	300	511,252.00	26.97	511,428.00	1.55
	400	660,444.00	67.03	660,444.00	6.72
	500	799,791.00	141.55	802,841.00	24.40
	600	926,829.00	241.20	926,829.00	31.59
	700	1,058,487.00	720.65	1,058,487.00	46.20
$\sqrt{n}/100$	100	70,769.00	0.84	70,769.00	0.16
	200	138,674.00	5.05	138,674.00	0.89
	300	203,000.00	24.13	203,000.00	2.70
	400	267,729.00	34.32	267,729.00	4.80
	500	328,235.00	62.64	328,235.00	9.95
	600	388,733.00	205.90	388,734.00	10.97
	700	447,089.00	117.75	447,089.00	16.55
	800	503,200.00	200.73	503,200.00	24.47
	900	557,946.00	443.44	557,953.00	30.43
	1,000	611,110.00	372.52	611,110.00	52.14
	1,100	665,303.00	1,209.21	665,303.00	57.48
$\sqrt{n}/1,000$	100	9,959.00	0.87	9,959.00	0.19
	200	27,806.00	5.10	27,806.00	0.23
	300	49,687.00	15.72	49,687.00	0.20
	400	74,711.00	31.77	74,711.00	0.26
	500	99,794.00	56.00	99,794.00	0.30
	600	124,479.00	83.39	124,479.00	0.45
	700	150,446.00	117.75	150,446.00	0.59
	800	175,042.00	167.02	175,042.00	0.83
	900	199,145.00	233.64	199,145.00	1.05
	1,000	223,206.00	274.58	223,206.00	1.30
	1,100	246,267.00	584.98	246,267.00	1.56

6.2. CPLEX Integer Solver

One solution approach, of course, is to use a general off-the-shelf mixed integer program solver such as CPLEX. Table 1 shows the results obtained by the latest CPLEX 8.1 on various sizes of the three types of problems considered (in the usual form (1)–(5)). Columns 1 and 2 give the fixed cost and problem size, respectively, of each instance; the next two columns provide the optimal solution value and the execution time obtained by running CPLEX on a PC Pentium 4; the last two columns provide the corresponding results of our VNDS heuristic on the same machine.

The experiments show that execution time and memory requirements of CPLEX increase rapidly for moderately-sized problems. Type I problems also appear the hardest to solve. It is clear that the solution of large problems with CPLEX is still not a viable option. Meanwhile, comparing the results in the table from VNDS, we see that high-quality solutions are obtained in a fraction of the time by this heuristic. The optimal solution was found in all Type III instances, in 9 out of 11 Type II, and 5 out of 7 Type I. The Type III instances appeared very easy, requiring VNDS on average around 1 sec. to find the optimal solution.

6.3. Main Computational Results

Here we examine the computational results obtained on problem instances generated by the procedure of Barahona and Chudak (2000) described earlier. We cover the same range of instances, and go far beyond, testing problem instances as large as $15,000 \times 15,000$. All programs are coded in C++ and run on two machines depending on problem size: for $n \leq 7,000$, we use a 1,800 MHz PC Pentium 4, and for $n > 7,000$, a SUN Enterprise 10,000 (with 400 Mhz clock and 64 gigabyte of RAM) that is slower but has sufficient memory to handle the larger problems.

Tables 2–4 summarize our computational results for the three types of problems considered and the problem sizes ($n = m$) indicated in the first column; for a given size the same problem is used but the fixed cost (f_i) at each node is adjusted according to the type I, II, or III. The value p gives the number of open facility sites in the optimal (if available) or best-known solution. The next four columns give objective-function values obtained, respectively, by our branch-and-bound (optimal value), sliding simplex for exact solution of the dual, reduced variable neighborhood search (RVNS) for the first stage of the heuristic procedure, and variable neighborhood decomposition search (VNDS) for the final stage. The computation times are reported in sequence in the next four columns, followed by time totals, best representing the total time spent until the best heuristic solution was found and all, the actual time spent. The last two columns give the gap calculated as a percentage as follows:

$$Gap(B\&B) = \left(\frac{z_h - z_P^*}{z_P^*}\right) \times 100,$$

$$Gap(Sliding) = \left(\frac{z_h - z_D^*}{z_D^*}\right) \times 100,$$

where z_h, z_P^*, and z_D^* are the objective-function values obtained, respectively, by VNDS, B&B, and sliding simplex. Optimal solutions of the primal problem are not shown where computation times of the B&B exceeded an imposed limit resulting in premature termination of the algorithm.

Parameter settings selected for the variable neighborhood search procedures are noted as follows (also see Section 3):

VNDS: $l_{\max} = \min\{p, 25\}$, where p is the number of open facilities in the current solution; stopping condition: 20 sequences of $\ell_{\max}$ iterations without improvement (Type III), 10 (Type I and II).

VNS subroutine: $k_{\min} = 1$, step size $= 1$, $k_{\max} = 20$ (Type III), 10 (Type I and II); stopping condition: 20 sequences of $k_{\max}$ iterations without improvement (Type III), 10 (Type I and II).

Table 2 Main Results: Type I Instances

		Objective values				Time (sec.)				Time total		% gap	
n	p	B&B	Sliding	RVNS	VNDS	B&B	Sliding	RVNS	VNDS	Best	All	B&B	Sliding
1,000	15	1,431,038	1,431,013.5	1,524,403	1,431,038		35.3	0.1	15.9	51.3	156.7	0.00	0.0017
1,100	16	1,555,369	1,555,027.5	1,666,104	1,555,369	6,829.2	35.3	0.1	21.8	57.2	105.9	0.00	0.0017
1,300	18	1,789,843	1,789,021.3	1,906,447	1,789,843	49.9	38.2	0.1	14.2	52.5	93.4	0.00	0.0459
1,400	18	1,904,195	1,903,679.5	2,077,133	1,905,380	42.9	35.3	0.1	39.5	74.9	115.6	0.06	0.0892
1,500	18		2,023,878.0	2,141,113	2,024,911		205.8	0.1	94.0	299.9	387.7		0.0510
2,000	20		2,581,964.9	2,790,620	2,590,631		842.5	0.1	98.0	940.6	1,505.7		0.3356
2,500	21		3,101,007.7	3,422,644	3,106,197		1,923.0	0.6	56.0	1,979.6	2,504.3		0.1673
3,000	23		3,602,388.1	3,904,718	3,606,160		3,073.1	0.3	153.3	3,226.7	4,078.5		0.1047
3,500	23		4,099,448.7	4,504,614	4,116,586		7,943.7	0.3	113.9	8,057.9	10,982.2		0.4181
4,000	25		4,581,458.1	4,879,807	4,599,619		15,927.2	0.5	245.1	16,172.8	18,342.0		0.3964
4,500	26		5,047,959.0	5,476,685	5,077,153		31,329.1	0.9	183.5	31,513.5	34,624.9		0.5783
5,000	29		5,517,681.8	6,099,060	5,548,718		52,734.9	1.2	118.4	52,854.5	55,923.6		0.5625

Table 3 Main Results: Type II Instances

		Objective values				Time (sec.)				Time total		% gap	
n	p	B&B	Sliding	RVNS	VNDS	B&B	Sliding	RVNS	VNDS	Best	All	B&B	Sliding
500	62	328,235	328,235.0	355,279	328,235	1.0	1.0	1.2	24.4	26.6	51.2	0.0000	0.0000
1,000	77	611,110	611,110.0	694,078	611,110	5.9	4.9	0.2	11.6	16.7	96.7	0.0000	0.0000
1,500	85	872,278	872,216.0	983,339	872,434	124.4	9.8	0.3	171.6	181.7	345.1	0.0179	0.0250
2,000	92	1,122,577	1,122,498.6	1,248,881	1,123,159	495.8	24.2	0.6	138.3	163.1	440.3	0.0518	0.0588
2,500	104		1,366,092.2	1,495,801	1,366,643		105.5	1.2	485.7	592.4	965.7		0.0403
3,000	107	1,595,895	1,595,895.0	1,748,782	1,595,896	75.5	63.3	1.4	315.6	380.3	905.8	0.0001	0.0001
3,500	111		1,819,686.8	1,999,865	1,820,639		357.4	1.9	1,106.8	1,466.1	2,274.7		0.0524
4,000	113		2,042,313.4	2,296,471	2,043,218		500.2	0.9	2,366.6	2,867.7	4,030.2		0.0443
4,500	119		2,255,880.7	2,526,011	2,256,254		502.4	1.5	1,767.6	2,271.5	3,668.4		0.0166
5,000	124		2,466,883.6	2,768,239	2,467,480		1,054.5	2.3	1,360.2	2,417.0	4,095.1		0.0242

Table 4 Main Results: Type III Instances

		Objective values				Time (sec.)				Time total		% gap	
n	p	B&B	Sliding	RVNS	VNDS	B&B	Sliding	RVNS	VNDS	Best	All	B&B	Sliding
500	347	99,794	99,794.0	107,984	99,794	0.1	0.1	0.2	0.3	0.6	1.5	0.0000	0.0000
1,000	391	223,206	223,206.0	247,790	223,206	1.1	1.1	0.7	1.3	3.1	7.4	0.0000	0.0000
1,500	410	332,750	332,744.0	382,516	332,764	8.6	2.6	1.0	3.0	6.6	17.3	0.0042	0.0060
2,000	453	438,574	438,568.5	496,630	438,578	68.0	4.8	2.7	30.7	38.2	55.6	0.0009	0.0023
2,500	498	542,203	542,182.5	614,880	542,267	86.1	7.3	3.2	34.8	45.3	70.9	0.0118	0.0157
3,000	519	642,321	642,309.0	736,939	642,321	78.3	8.8	4.2	152.1	165.1	203.5	0.0000	0.0018
3,500	542	741,097	741,057.3	848,933	741,126	555.9	13.3	5.6	87.1	106.0	158.2	0.0039	0.0092
4,000	570	839,922	839,909.5	972,883	839,942	205.3	21.1	5.8	162.2	189.1	255.9	0.0024	0.0039
4,500	582	932,428	932,361.5	1,069,821	932,597	5,156.1	23.4	7.3	113.7	144.4	231.7	0.0181	0.0253
5,000	600	1,028,249	1,028,235.0	1,217,007	1,028,255	1,266.3	32.2	6.9	239.6	278.7	394.8	0.0006	0.0019
6,000	637	1,211,889	1,211,861.0	1,384,204	1,211,932	4,030.7	47.6	14.3	761.4	823.3	981.9	0.0035	0.0058
7,000	668	1,392,127	1,392,099.0	1,593,475	1,392,232	28,547.7	87.8	14.9	715.5	818.8	1,044.9	0.0075	0.0096
8,000	701		1,569,292.0	1,791,174	1,569,767		1,718.6	22.0	1,043.8	2,784.4	3,090.1		0.0303
9,000	724		1,742,075.5	2,022,581	1,742,388		2,270.7	20.2	1,528.0	3,818.9	4,182.9		0.0180
10,000	752		1,915,065.1	2,163,915	1,915,562		2,695.2	36.3	1,530.4	4,261.9	4,740.1		0.0259
11,000	768		2,079,253.9	2,391,519	2,079,616		3,916.6	33.4	1,144.2	5,094.2	5,686.6		0.0175
12,000	789		2,245,167.0	2,551,364	2,245,526		4,630.4	40.4	3,113.3	7,784.1	8,504.6		0.0215
13,000	802		2,411,167.2	2,702,463	2,411,335		5,507.2	80.9	6,266.9	11,855.0	12,716.4		0.0109
14,000	827		2,570,329.3	2,913,820	2,570,792		7,023.2	74.1	3,919.3	11,016.6	11,935.4		0.0180
15,000	844		2,733,373.3	3,134,626	2,733,979		8,181.9	61.3	5,103.4	13,346.6	14,500.1		0.0221

Table 5 Main Results: Type IV Instances

		Objective values				Time (sec.)				Time total		% gap	
n	p	B&B	Sliding	RVNS	VNDS	B&B	Sliding	RVNS	VNDS	Best	All	B&B	Sliding
500	46	368,408	368,408	643,882	368,408	2.34	2.24	0.25	20.06	20.31	484.28	0.0000	0.0000
1,000	77	578,740	578,740	1,130,397	578,740	5.16	5.13	0.61	20.05	20.66	974.34	0.0000	0.0000
1,500	109	740,870	740,870	1,058,235	740,870	30.85	30.73	1.60	527.12	528.72	1,361.93	0.0000	0.0000
2,000	128	915,155	915,155	2,098,798	915,155	65.75	65.44	1.62	344.37	345.99	1,667.10	0.0000	0.0000
2,500	144	1,071,962	1,071,962	1,547,638	1,071,962	109.81	109.46	3.44	134.94	138.38	1,683.46	0.0000	0.0000
3,000	178	1,193,847	1,193,847	1,864,527	1,193,847	107.86	107.71	3.59	1,191.94	1,195.53	1,909.35	0.0000	0.0000
3,500	198	1,334,569	1,334,569	2,056,067	1,334,620	150.51	150.36	4.85	1,045.48	1,050.33	1,490.51	0.0038	0.0038
4,000	206	1,438,336	1,438,304	2,218,219	1,438,336	250.12	249.89	6.34	932.58	938.92	1,313.91	0.0000	0.0022
4,500	231	1,541,880	1,541,880	2,593,945	1,542,339	298.16	297.89	6.58	1,064.58	1,071.16	1,445.19	0.0298	0.0298
5,000	246	1,693,821	1,693,782	2,616,424	1,695,554	394.28	369.74	8.22	1,490.67	1,498.89	2,123.97	0.0009	0.0010
5,500	255	1,813,342	1,813,324	2,941,636	1,813,342	468.12	462.57	8.02	2,824.83	2,832.85	3,576.60	0.0000	0.0000
6,000	281	1,919,477	1,919,477	3,108,396	1,917,477	811.12	810.89	8.73	2,617.86	2,626.59	3,389.80	0.000	0.0000
6,500	306	2,016,429	2,016,284	3,519,048	2,016,678	812.23	551.15	8.60	4,389.96	4,398.56	5,988.45	0.0123	0.0196
7,000	305	2,154,829	2,152,592	3,960,288	2,154,829	1,130.82	747.30	8.11	4,627.29	4,635.40	6,125.82	0.0000	0.1039
8,000	331	2,320,081	2,320,081	5,652,921	2,320,944	1,261.99	1,260.93	20.01	6,895.36	7,015.37	7,892.23	0.0000	0.0000
9,000	362	2,529,390	2,529,390	6,554,965	2,529,575	1,364.11	1,362.94	20.03	3,510.40	3,530.43	8,773.16	0.0081	0.0081
10,000	384	2,737,359	2,737,350	7,528,204	2,737,359	4,224.59	1,592.29	20.02	5,261.97	5,281.99	9,676.13	0.0000	0.0003
11,000	411	2,934,323	2,934,323	8,149,233	2,934,428	1,896.45	1,893.23	20.01	5,812.11	5,832.12	9,005.78	0.0036	0.0036
12,000	430	3,110,714	3,110,714	8,068,905	3,110,714	2,571.31	2,567.24	20.05	4,930.31	4,950.36	9,682.77	0.0000	0.0000
13,000	458	3,300,155	3,300,155	8,975,232	3,300,155	3,126.11	3,122.78	20.16	9,530.48	9,550.64	9,941.00	0.0000	0.0000
14,000	473	3,461,208	3,461,208	9,091,698	3,461,208	4,764.81	4,759.12	20.18	8,168.80	8,188.98	10,184.80	0.0000	0.0000
15,000	490	3,645,572	3,645,340	9,787,616	3,645,990	19,217.56	5,429.82	20.20	4,898.03	4,918.23	12,025.06	0.0115	0.0178

From the summary results in Tables 2–4, we see:

- The VNDS heuristic provides high-quality solutions over a wide range of problem sizes and types. This includes much larger problem instances than currently considered in the literature. For Type III instances up to $15{,}000 \times 15{,}000$, the largest gap obtained is on the order of 0.03%. For Type II, the maximum gap is 0.06% for problem sizes up to $5{,}000 \times 5{,}000$, and Type I, 0.58%. These results present a significant improvement in the state-of-the-art given in Barahona and Chudak (2000), where gaps of 1% are reported on problem sizes up to $3{,}000 \times 3{,}000$. Meanwhile the computation time for VNDS is very reasonable considering the problem sizes investigated. For example, problems up to $3{,}000 \times 3{,}000$ take only a few minutes; the largest one $(15{,}000 \times 15{,}000)$ ran for 1.8 hours. Type III instances, the easiest for our VNDS, were the hardest for Barahona and Chudak's V&RRWC.
- The sliding simplex method is capable of solving the dual exactly for the large problems investigated. This is quite impressive considering that the largest problem solved has $n + mn = 225{,}015{,}000$ dual variables. By obtaining a tight starting solution (which may be infeasible), and then using our sliding simplex method, a substantial reduction in problem size and number of simplex iterations is obtained. Computation times for sliding simplex are also seen to be reasonable, although Type I instances took significantly longer.
- By using the entire package proposed here, namely, a heuristic solution of the primal problem by VNDS, followed by exact solution of the dual with sliding simplex, and then closing the gap with branch and bound, exact solution of large SPLPs is achieved. Our largest problem solved $(7{,}000 \times 7{,}000)$ set a new record (soon to be beaten, as shown below).

A referee suggested that we test the exact algorithm on instances with different fixed costs. Consequently, we combined a series of instances of type IV, in which fixed costs are drawn randomly from a uniform distribution on the interval $[\sqrt{n}/1{,}000, \sqrt{n}/10]$. Results are presented in Table 5; they show the following:

- Instances with different fixed costs are easier to solve than are those with uniform fixed costs. Indeed, all problems with sizes up to 15,000 could be solved exactly, again setting a new record.
- Reduced VNS does not give good results if it is allocated a small computing time as in the previous experiments; VNDS, with the new stopping rule as before, takes more time but obtains excellent results, close to or equal to those of the sliding simplex algorithm; the value of the LP relaxation obtained by the sliding simplex algorithm is optimal in 12 cases out of 22, including those with 12,000, 13,000, and 14,000 users; the branch-and-bound algorithm has less work to do than with instances of type III, II, and especially I, although some computing time is required even when there is no duality gap to obtain an integer solution.

Finally, Table 6 studies the effect of the window size ℓ in the sliding simplex. Here an instance of size $n = 1{,}500$ is generated, and the three problem types investigated. As expected, a smaller window results

Table 6 **Efficiency of Sliding Simplex for Different Values of ℓ and $n = 1{,}500$**

Type	z_D	ℓ	# CPLEX calls	# iter	Time
I	2,023,878	1	49	182,590	205.72
		2	20	301,999	497.28
		3	13	334,159	843.37
		4	11	433,068	1,363.54
		5	8	411,637	1,526.79
II	872,216	1	12	26,688	9.97
		2	8	33,278	11.26
		3	5	34,034	14.55
		4	5	50,983	29.11
		5	4	55,738	44.79
III	332,744	1	6	11,559	2.73
		2	4	12,507	2.60
		3	3	13,085	2.65
		4	2	10,322	2.18
		5	2	10,980	2.38

in more subproblems or calls to CPLEX. However, the effect on computation time appears to be the reverse for types I and II; computation time appears to be insensitive to the parameter ℓ for type III.

7. Conclusions

This paper develops a new methodology for solving the SPLP. In the first stage a heuristic based on variable neighborhood search (VNS) is used to obtain a near-optimal solution. We show that VNS with decomposition is a very powerful technique for large-scale problems, up to 15,000 facilities × 15,000 users. In the second phase, our approach is to find an exact solution of the relaxed dual problem. This is accomplished in three stages: (i) find an initial dual solution (generally infeasible) using the primal heuristic solution and complementary slackness conditions; (ii) improve the solution by applying VNS on the unconstrained nonlinear form of the dual; and (iii) finally, solve the dual exactly using a customized sliding simplex algorithm that applies windows on the dual variables to reduce the size of the problem substantially. In all problems tested, including instances much larger than previously reported, our procedure was able to find the exact dual solution in reasonable computing time. In the third and final phase, armed with tight upper and lower bounds obtained respectively from the heuristic primal solution in phase one and the exact dual solution in phase two, we apply a standard branch-and-bound algorithm to find an optimal solution of the original problem. The lower bounds are updated with the dual sliding simplex method and the upper bounds whenever new integer solutions are obtained at the nodes of the branching tree. In this way we were able to solve exactly problem instances with up to $7{,}000 \times 7{,}000$ for uniform fixed costs and $15{,}000 \times 15{,}000$ otherwise. This advances the record considerably.

Future directions include further experimenting with, and fine-tuning of, our primal-dual variable neighborhood search methodology. Adapting the exact solution method proposed here to extensions of the SPLP should also be investigated.

Acknowledgments

The first author was supported by NSERC Grant 105574-02 and the second author by NSERC Grant OGP205041. The third and fourth authors were partially supported by Serbian Ministry of Science, Project 1583. The authors thank Yuri Kotchetov for useful remarks that helped to improve this paper's presentation.

References

Ahn, S., C. Cooper, G. Cornuéjols, A. M. Frieze. 1988. Probabilistic analysis of a relaxation for the *p*-median problem. *Math. Oper. Res.* **13** 1–31.

Balinski, M. 1965. Integer programming: methods, uses, computation. *Management Sci.* **12** 253–313.

Barahona, F., F. Chudak. 2000. Solving large scale uncapacitated facility-location problems. P. Pardalos, ed. *Approximation and Complexity in Numerical Optimization*. Kluwer Academic Publishers, Norwell, MA, 48–62.

Beasley, J. E. 1993. Lagrangean heuristics for location problems. *Eur. J. Oper. Res.* **65** 383–399.

Bilde, O., J. Krarup. 1977. Sharp lower bounds and efficient algorithms for the simple plant-location problem. *Ann. Discrete Math.* **3** 79–97.

Brimberg, J., C. ReVelle. 1998. A bi-objective plant-location problem: Cost vs. demand served. *Location Sci.* **6** 121–135.

Brimberg, J., C. ReVelle. 2000. The maximum return-on-investment plant-location problem. *J. Oper. Res. Soc.* **51** 729–735.

Conn, A. R., G. Cornuéjols. 1990. A projection method for the uncapacitated facility-location problem. *Math. Programming* **46** 273–298.

Cornuéjols, G., M. L. Fisher, G. L. Nemhauser. 1977. Location of bank accounts to optimize float: An analytical study of exact and approximate algorithms. *Management Sci.* **23** 163–177.

Cornuéjols, G., G. L. Nemhauser, L. A. Wolsey. 1990. The uncapacitated facility-location problem. P. B. Mirchandani, R. L. Francis, eds. *Discrete Location Theory*. John Wiley, New York, 119–171.

Daskin, M. 1995. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley, New York.

Efroymson, M. A., T. L. Ray. 1996. A branch-and-bound algorithm for plant-location. *Oper. Res.* **14** 361–368.

Erlenkotter, D. 1978. A dual-based procedure for uncapacitated facility location. *Oper. Res.* **26** 992–1009.

Feldman, E., F. A. Lehrer, T. L. Ray. 1966. Warehouse location under continuous economies of scale. *Management Sci.* **12** 670–684.

Francis, R., L. F. McGinnis, Jr., J. A. White. 1992. *Facility Layout and Location: An Analytical Approach*, 2nd ed. Prentice Hall, Englewood Cliffs, NJ.

Galvão, R. D., L. A. Raggi. 1989. A method for solving to optimality uncapacitated location problems. *Ann. Oper. Res.* **18** 225–244.

Ghosh, D. 2003. Neighborhood search heuristics for the uncapacitated facility-location problem. *Eur. J. Oper. Res.* **150** 150–162.

Goldengorin, B., D. Ghosh, G. Sierksma. 2003a. Branch and peg algorithms for the simple plant-location problem. *Comput. Oper. Res.* **30** 967–981.

Goldengorin, B., G. A. Tijssen, D. Ghosh, G. Sierksma. 2003b. Solving the simple plant location problem using data correcting approach. *J. Global Optim.* **25** 377–406.

Goncharov, E., Y. Kochetov. 1999. Behavior of the probabilistic greedy algorithms for the multistage uncapacitated facility-location problem. *Discrete Anal. Oper. Res.* **6** 12–32 (in Russian).

Goncharov, E., Y. Kochetov. 2000. Probabilistic tabu search algorithm for the multi-stage uncapacitated facility-location problem. *Oper. Res. Proc.*, Springer, Berlin, Germany, 65–70.

Hammer, P. L. 1968. Plant-location—pseudo-boolean approach. *Israel J. Tech.* **6** 330–332.

Hansen, P., N. Mladenović. 1997. Variable neighborhood search for the *p*-median. *Location Sci.* **5** 207–226.

Hansen, P., N. Mladenović. 2001. Variable neighborhood search: Principles and applications. *Eur. J. Oper. Res.* **130** 449–467.

Hansen, P., N. Mladenović. 2003. Variable neighborhood search. F. Glover, G. Kochenberger, eds. *Handbook of Metaheuristics*. Kluwer Academic Publishers, Boston, MA, 145–184.

Hansen, P., N. Mladenović, D. Perez-Brito. 2001. Variable neighborhood decomposition search. *J. Heuristics* **7** 335–350.

Jones, P., T. J. Lowe, G. Muller, N. Xu, Y. Ye, J. L. Zydiak. 1995. Specially structured uncapacitated facility-location problems. *Oper. Res.* **43** 661–669.

Karkazis, J. 1985. Principal direction search: A new method of search for unconstrained LP formulations. *Eur. J. Oper. Res.* **20** 352–362.

Khumawala, B. M. 1972. An efficient branch and bound algorithm for the warehouse location problem. *Management Sci.* **18** 718–731.

Klose, A. 1995. A comparison between the Erlenkotter algorithm and a branch and bound algorithm based on subgradient optimization to solve the uncapacitated facility-location problem. U. Derigs, A. Bachem, A. Drexl, eds. *Operations Research Proceedings 1994*, Springer, Berlin, Germany, 335–339.

Körkel, M. 1989. On the exact solution of large-scale simple plant-location problem. *Eur. J. Oper. Res.* **39** 157–173.

Krarup, J., P. M. Pruzan. 1983. Simple plant location problem: Survey and synthesis. *Eur. J. Oper. Res.* **12** 36–81.

Kratica, J., D. Tosic, V. Filipovic, I. Ljubic. 2001. Solving the simple plant-location problem by genetic algorithms. *RAIRO—Oper. Res.* **35** 127–142.

Kuehn, A. A., M. J. Hamburger. 1963. A heuristic program for locating warehouses. *Management Sci.* **9** 643–666.

Labbé, M., F. V. Louveaux. 1997. Location problem. N. Dell'Amico, F. Maffioli, S. Martello, eds. *Annotated Bibliographies in Combinatorial Optimization*. John Wiley, New York, 264–271.

Labbé, M., D. Peeters, J. F. Thisse. 1995. Location on networks. M. Ball, T. Magnanti, C. Monma, G. Nemhauser, eds. *Handbook of Operations Research and Management Science: Networks*. North-Holland, Amsterdam, The Netherlands, 551–624.

Manne, A. S. 1964. Plant-location under economies-of-scale: Decentralization and computation. *Management Sci.* **11** 213–235.

Marsten, R. E., W. W. Hogan, J. W. Blankenship. 1975. The boxstep method for large-scale optimization. *Oper. Res.* **23** 389–405.

Michel, L., P. Van Hentenryck. 2004. A simple tabu search for warehouse location. *Eur. J. Oper. Res.* **157** 576–591.

Mirchandani, P. B., R. L. Francis. 1990. *Discrete Location Theory*. Wiley-Interscience, New York.

Mladenović, N., P. Hansen. 1997. Variable neighborhood search. *Comput. Oper. Res.* **24** 1097–1100.

Mladenović, N., J. Brimberg, P. Hansen. 2006. A note on duality gap in the simple plant-location problem. *Eur. J. Oper. Res.* **174** 11–22.

Morris, J. G. 1978. On the extent to which certain fixed charge depot location problems can be solved by LP. *J. Oper. Res. Soc.* **29** 71–76.

Myung, V. S., H. Kim, D. Tcha. 1997. A bi-objective uncapacitated facility-location problem. *Eur. J. Oper. Res.* **100** 608–616.

Pentico, D. W. 1976. The assortment problem with nonlinear cost functions. *Oper. Res.* **24** 1129–1142.

Pentico, D. W. 1988. The discrete two-dimensional assortment problem. *Oper. Res.* **36** 324–332.

ReVelle, C. 1993. Facility siting and integer friendly programming. *Eur. J. Oper. Res.* **65** 147–158.

ReVelle, C., G. Laporte. 1996. The plant-location problem: New models and research prospects. *Oper. Res.* **44** 864–874.

Spielberg, K. 1969. Algorithms for the simple plant-location problem with some side conditions. *Oper. Res.* **17** 85–111.

Stollsteimer, J. F. 1963. A working model for plant numbers and locations. *J. Farm. Econom.* **45** 631–645.

Teitz, M. B., P. Bart. 1968. Heuristic methods for estimating the generalized vertex median of a weighted graph. *Oper. Res.* **16** 955–961.

Tripathy, A., H. Süral, Y. Gerchak. 1999. Multidimensional assortment problem with an application. *Networks* **33** 239–245.

Voss, S. 1996. A reverse elimination approach for the *p*-median problem. *Stud. Locational Anal.* **8** 45–58.

Weber, A. 1909. *Ueber den Standort der Industrien*. Mohr, Tübingen (English Translation: C. I. Friedrich, translator. 1929. *Theory of the Location of Industries*. University of Chicago Press, Chicago, IL).

Whitaker, R. 1983. A fast algorithm for the greedy-interchange for large-scale clustering and median location problems. *INFOR* **21** 95–108.