Journal of Global Optimization **25:** 377–406, 2003.

Solving the Simple Plant Location Problem using a Data Correcting Approach

BORIS GOLDENGORIN[1], GERT A. TIJSSEN[1], DIPTESH GHOSH[2] and GERARD SIERKSMA[1]
[1]*Faculty of Economic Sciences, University of Groningen, Groningen, The Netherlands E-mail: B.Goldengorin@eco.rug.nl*
[2]*P&QM Area, Indian Institute of Management, Ahmedabad, India E-mail: diptesh@iimahd.ernet.in*

Abstract. The Data Correcting Algorithm is a branch and bound type algorithm in which the data of a given problem instance is 'corrected' at each branching in such a way that the new instance will be as close as possible to a polynomially solvable instance and the result satisfies an acceptable accuracy (the difference between optimal and current solution). In this paper the data correcting algorithm is applied to determining exact and approximate optimal solutions to the simple plant location problem. Implementations of the algorithm are based on a pseudo-Boolean representation of the goal function of this problem, and a new reduction rule. We study the efficiency of the data correcting approach using two different bounds, the Khachaturov-Minoux bound and the Erlenkotter bound. We present computational results on several benchmark instances, which confirm the efficiency of the data-correcting approach.

AMS 2000 Mathematics Subject Classification: 90B80, 90C59

Key words: Simple plant location problem, Pseudo-Boolean representation, Hammer function, Branch and bound, Preprocessing

1. Introduction

The Simple Plant Location Problem (SPLP) takes a set $I = \{1, 2, \dots, m\}$ of sites in which plants can be located, a set $J = \{1, 2, \dots, n\}$ of clients, each having a unit demand, a vector $F = (f_i)$ of fixed costs for setting up plants at sites $i \in I$, and a matrix $C = [c_{ij}]$ of transportation costs from $i \in I$ to $j \in J$ as input. It computes a set $P^\star, \emptyset \subset P^\star \subseteq I$, at which plants can be located so that the total cost of meeting the demands of all the clients is minimal. The costs involved in meeting the client demands include the fixed costs of setting up plants, and the transportation cost of supplying clients from the plants that are set up. A detailed introduction to this problem has appeared in Cornuejols et al. (1990). The SPLP forms the underlying model in several combinatorial problems, like set covering, set partitioning, information retrieval, simplification of logical Boolean expressions, airline crew scheduling, vehicle despatching Christofides, 1975), assortment (Beresnev, 1978; Goldengorin, 1995; Jones et al., 1995; Pentico, 1976; Pentico, 1988; Tripathy et al., 1999), and is a subproblem for various location analysis problems (Revelle and Laporte, 1996).

The SPLP is $\mathcal{NP}$-hard (Cornuejols, 1990), and several exact and heuristic algorithms for solving it have been discussed in the literature. Most of the exact algorithms are based on a mathematical programming formulation of the SPLP (see for example, Schrage, 1975; Morris, 1978; Held et al., 1974; Cornuejols et al., 1977b; Cornuejols and Thizy, 1982; and Garfinkel et al., 1974). Polyhedral results for the SPLP polytope have been reported in Trubin (1969), Balas and Padberg (1972), Mukendi (1975), Cornuejols et al. (1977a), Krarup and Pruzan (1983), Cho et al. (1983a) and Cho et al. (1983b). In theory, these results allow us to solve the SPLP by applying the simplex algorithm to the strong linear programming relaxation, with the additional stipulation that a pivot to a new extreme point is allowed only when this new extreme point is integral. However, efficient implementations of this pivot rule are not available. Beasley (1993a) reported computational experiments with Lagrangian heuristics for SPLP instances. Körkel (1989) proposed algorithms based on refinements to a dual-ascent heuristic procedure to solve the dual of a linear programming relaxation of the SPLP ((Körkel, 1989)), combined with the use of the complementary slackness conditions to construct primal solutions (Erlenkotter, 1978). An annotated bibliography is available in Labbé and Louveaux (1997). An exact algorithm based on a pseudo-Boolean representation of the problem has been reported in Goldengorin et al. (2001). It uses a preprocessing rule to reduce the size of its input. The preprocessing rule is due to Khumawala (1972).

It is common knowledge that exact algorithms for $\mathcal{NP}$-hard problems in general, and for the SPLP in particular, spend only about 10% of the execution time to find an optimal solution. The remaining time is spent proving the optimality of the solution. In this paper, our aim is to reduce the amount of time spent proving the optimality of the solution obtained. We propose a data correcting algorithm for the SPLP that is designed to output solutions with a pre-specified *acceptable accuracy* α. This means that the difference between the cost of the solution output by the algorithm is no more than α more than the cost of an optimal solution. (Note that $\alpha = 0$ results in an exact algorithm for the SPLP, while $\alpha = \infty$ results in a fast greedy algorithm.) The objective function of the SPLP is supermodular (see Cornuejols et al., 1990) and so, the data correcting algorithm described in Goldengorin et al. (1999) can be used to solve the SPLP. In fact, Goldengorin et al. (1999) contains an example to that effect. However, it can be made much more efficient; for example, by using SPLP-specific bounds (used in Erlenkotter, 1978) and preprocessing rules (used in Khumawala, 1972). The algorithm described here uses a pseudo-Boolean representation of the SPLP, due originally to Hammer (1968) (see also Beresnev (1973)). It uses a new reduction procedure based on data correcting, which is stronger than the preprocessing rules used in Khumawala (1972) to reduce the original instance to a smaller 'core' instance, and then solves it using a procedure based on preliminary preservation and data correcting (see Goldengorin, 1999). Computational experiments with this algorithm on benchmark instances of the SPLP are also described in the paper. We show how the use of preprocessing and bounds specific to the SPLP enhance the performance of the data-correcting algorithm. This algorithm is based on three concepts found in the literature, a pseudo-Boolean representation of the SPLP, data-correcting, and the preliminary preservation procedure. The next section of this paper therefore contains a brief exposure to these concepts. We describe the new algorithm in Section 3 and present the results of our computational experiments with it in Section 4. We finally conclude the paper in Section 5 with a summary of the work presented here and discussions.

2. Preliminaries from the Literature

In this section we describe a pseudo-Boolean representation of the SPLP that we use in our algorithm (Subsection 2.1), an introduction to data correcting (Subsection 2.2), and a description of the preliminary preservation procedure (Subsection 2.3).

2.1. A PSEUDO-BOOLEAN REPRESENTATION OF THE SPLP

The pseudo-Boolean approach to solving the SPLP (Hammer, 1968; Beresnev, 1973) is a penalty-based approach that relies on the fact that any instance of the SPLP has an optimal solution in which each client is supplied by exactly one plant. This implies, that in an optimal solution, each client will be served fully by the plant located closest to it. Therefore, it is sufficient to determine the sites where plants are to be located, and then use a minimum weight matching algorithm to assign clients to plants.

An instance of the SPLP can be described by a m-vector $F = (f_i)$, and a $m \times n$ matrix $C = [c_{ij}]$; $m, n \geqslant 1$. We will use the $m \times (n + 1)$ *augmented matrix* $[F|C]$ as a shorthand for describing an instance of the SPLP. The total cost $f_{[F|C]}(P)$ associated with a subset P of I consists of two components, namely the fixed costs $\sum_{i \in P} f_i$ and the transportation costs $\sum_{j \in J} \min\{c_{ij} | i \in P\}$; i.e.

$$f_{[F|C]}(P) = \sum_{i \in P} f_i + \sum_{j \in J} \min\{c_{ij} | i \in P\},$$

and the SPLP is the problem of finding

$$P^\star \in \arg\min\{f_{[F|C]}(P) : \emptyset \subset P \subseteq I\}. \tag{1}$$

In the remainder of this subsection we describe the pseudo-Boolean formulation of the SPLP due to Hammer (1968) (see also Beresnev (1973)).

A $m \times n$ *ordering matrix* $\Pi = [\pi_{ij}]$ is a matrix each of whose columns $\Pi_j = (\pi_{1j}, \ldots, \pi_{mj})^T$ define a permutation of $1, \ldots, m$. Given a transportation matrix

C, the set of all ordering matrices Π such that $c_{\pi_{1j}j} \leqslant c_{\pi_{2j}j} \leqslant \cdots \leqslant c_{\pi_{mj}j}$ for $j = 1, \ldots, n$, is denoted by $perm(C)$.

Defining

$$y_i = \begin{cases} 0 & \text{if } i \in P \\ 1 & \text{otherwise,} \end{cases} \quad \text{for each } i = 1, \ldots, m \tag{2}$$

we can indicate any solution P by a vector $\mathbf{y} = (y_1, y_2, \ldots, y_m)$. The fixed cost component of the total cost can be written as

$$\mathcal{F}_F(\mathbf{y}) = \sum_{i=1}^{m} f_i(1 - y_i). \tag{3}$$

Given a transportation cost matrix C, and an ordering matrix $\Pi \in perm(C)$, we can denote differences between the transportation costs for each $j \in J$ as

$$\begin{aligned} \Delta c[0, j] &= c_{\pi_{1j}j}, \quad \text{and} \\ \Delta c[l, j] &= c_{\pi_{(l+1)j}j} - c_{\pi_{lj}j}, \quad l = 1, \ldots, m-1. \end{aligned}$$

Note that $\Delta c[l, j] \geqslant 0$, even if the transportation cost matrix C contains negative entries. The transportation costs of supplying any client $j \in J$ from any open plant can be expressed in terms of the $\Delta c[\cdot, j]$ values. It is clear that we have to spend at least $\Delta c[0, j]$ in order to satisfy j's demand since this is the cheapest cost of satisfying j. If no plant is located at the site closest to j, i.e., $y_{\pi_{1j}} = 1$, we try to satisfy the demand from the next closest site. In that case, we spend an additional $\Delta c[1, j]$. Continuing in this manner, the transportation cost of supplying $j \in J$ is

$$\begin{aligned} \min\{c_{ij} | i \in P\} &= \Delta c[0, j] + \Delta c[1, j] \cdot y_{\pi_{1j}} + \Delta c[2, j] y_{\pi_{1j}} y_{\pi_{2j}} \\ &\quad + \cdots + \Delta c[m-1, j] y_{\pi_{1j}} \cdots y_{\pi_{(m-1)j}} \\ &= \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \prod_{r=1}^{k} y_{\pi_{rj}}, \end{aligned}$$

so that the transportation cost component of the cost of a solution $\mathbf{y}$ corresponding to an ordering matrix $\Pi \in perm(C)$ is

$$\mathcal{T}_{C,\Pi}(\mathbf{y}) = \sum_{j=1}^{n} \left\{ \Delta c[0, j] + \sum_{k=1}^{m-1} \Delta c[k, j] \prod_{r=1}^{k} y_{\pi_{rj}} \right\}. \tag{4}$$

Combining (3) and (4), the total cost of a solution $\mathbf{y}$ to the instance $[F|C]$ corresponding to the ordering matrix $\Pi \in perm(C)$ is given by the pseudo-Boolean polynomial

$$\begin{aligned} f_{[F|C],\Pi}(\mathbf{y}) &= \mathcal{F}_F(\mathbf{y}) + \mathcal{T}_{C,\Pi}(\mathbf{y}) \\ &= \sum_{i=1}^{m} f_i(1-y_i) + \\ &\quad \sum_{j=1}^{n} \left\{ \Delta c[0,j] + \sum_{k=1}^{m-1} \Delta c[k,j] \prod_{r=1}^{k} y_{\pi_{rj}} \right\}. \end{aligned} \tag{5}$$

It can be shown (Goldengorin et al., 2000) that the total cost function $f_{[F|C],\Pi}(\cdot)$ is identical for all $\Pi \in perm(C)$. We call this pseudo–Boolean polynomial the *Hammer function* $\mathcal{H}_{[F|C]}(\mathbf{y})$ corresponding to the SPLP instance $[F|C]$ and $\Pi \in perm(C)$. In other words

$$\mathcal{H}_{[F|C]}(\mathbf{y}) = f_{[F|C],\Pi}(\mathbf{y}) \text{ where } \Pi \in perm(C). \tag{6}$$

We can now formulate (1) in terms of Hammer functions as

$$\mathbf{y}^\star \in \arg\min\{\mathcal{H}_{[F|C]}(\mathbf{y}) : \mathbf{y} \in \{0,1\}^m, \mathbf{y} \neq \mathbf{1}\}. \tag{7}$$

Notice that if for clients p and q, $\{\pi_{1p}, \pi_{2p}, \ldots, \pi_{kp}\} = \{\pi_{1q}, \pi_{2q}, \ldots, \pi_{kq}\}$ for $k \leqslant n$, then the k^{th} order terms in the Hammer function corresponding to these two clients can be aggregated. This implies that in general, the Hammer function will be a more space-efficient representation of the SPLP than the conventional $[F|C]$ matrix. This representation also makes it easier to construct data structures that allow efficient updating operations in the algorithm presented in Section 3.

As an example, consider the SPLP instance:

$$[F|C] = \left[\begin{array}{c|cccc} 9 & 7 & 12 & 22 & 13 \\ 4 & 8 & 9 & 18 & 17 \\ 3 & 16 & 17 & 10 & 27 \\ 6 & 9 & 13 & 10 & 11 \end{array}\right]. \tag{8}$$

Two possible ordering matrices corresponding to C are

$$\Pi_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{bmatrix} \text{ and } \Pi_2 = \begin{bmatrix} 1 & 2 & 4 & 4 \\ 2 & 1 & 3 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{bmatrix}. \tag{9}$$

The Hammer function can be computed using either Π_1 or Π_2. If we use Π_1 for our calculations, we obtain the Hammer function as $\mathcal{H}_{[F|C]}(\mathbf{y}) = \{9(1-y_1) + 4(1-y_2) + 3(1-y_3) + 6(1-y_4)\} + \{7 + 1y_1 + 1y_1y_2 + 7y_1y_2y_4\} + \{9 + 3y_2 + 1y_1y_2 + 4y_1y_2y_4\} + \{10 + 0y_3 + 8y_3y_4 + 4y_2y_3y_4\} + \{11 + 2y_4 + 4y_1y_4 + 10y_1y_2y_4\} = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 21y_1y_2y_4 + 4y_2y_3y_4$.

If we use Π_2, the contribution of the third client towards the Hammer function is $\{10 + 0y_4 + 8y_3y_4 + 4y_2y_3y_4\}$ instead of $\{10 + 0y_3 + 8y_3y_4 + 4y_2y_3y_4\}$. Clearly, this does not affect the Hammer function.

2.2. FUNDAMENTALS OF DATA CORRECTING

Data correcting is a method in which we alter the data in a problem instance to convert it to an instance that is easily solvable. This methodology was first introduced in Goldengorin (1983). In this subsection we illustrate the method for the SPLP when the instance data is represented by the fixed cost vector and the transportation cost matrix. However it can be applied to a wide variety of optimization problems, and in particular, to the SPLP represented as a Hammer function.

Consider an instance $[F|C]$ of the SPLP. The objective of the problem is to compute a set P, $\emptyset \subset P \subseteq I$, that minimizes $f_{[F|C]}(P)$. Also consider a SPLP instance $[S|D]$ that is known to be polynomially solvable. Let $P^\star_{[F|C]}$ and $P^\star_{[S|D]}$ be optimal solutions to $[F|C]$ and $[S|D]$, respectively. Let us define the proximity measure $\rho([F|C], [S|D])$ between the two instances as

$$\rho([F|C], [S|D]) = \sum_{i \in I} |f_i - s_i| + \sum_{j \in J} \max\{|c_{ij} - d_{ij}| : i \in I\}. \tag{10}$$

We use $\max\{|c_{ij} - d_{ij}| : i \in I\}$ in (10) instead of the expression $\sum_{i \in I} |c_{ij} - d_{ij}|$ since, in an optimal solution, the demand of each client is satisfied by a single facility, only one element in each column in the transportation matrix will contribute to the cost of the optimal solution.

Notice that $\rho([F|C], [S|D])$ is defined only when the instances $[F|C]$ and $[S|D]$ are of the same size. Also note that the value of $\rho(\cdot,\cdot)$ it can be computed in time polynomial in the size of the two instances. The following theorem, which forms the basis of data correcting, shows that $\rho([F|C], [S|D])$ is an upper bound to the difference between the *unknown* optimal costs for the SPLP instances $[F|C]$ and $[S|D]$.

THEOREM 1. *Let $[F|C]$ and $[S|D]$ be two SPLP instances of the same size, and let $P^\star_{[F|C]}$ and $P^\star_{[S|D]}$ be optimal solutions to $[F|C]$ and $[S|D]$ respectively. Then*

$$|f_{[F|C]}(P^\star_{[F|C]}) - f_{[S|D]}(P^\star_{[S|D]})| \leqslant \rho([F|C], [S|D]).$$

Proof. There are two cases to consider.
Case 1: $f_{[F|C]}(P^\star_{[F|C]}) \leqslant f_{[S|D]}(P^\star_{P[S|D]})$, and
Case 2: $f_{[F|C]}(P^\star_{[F|C]}) > f_{[S|D]}(P^\star_{[S|D]})$. We only prove Case 1 here; the proof of

Case 2 is similar to the proof of Case 1.

$$\begin{aligned}
& f_{[F|C]}(P^\star_{[F|C]}) - f_{[S|D]}(P^\star_{[S|D]}) \\
& \leqslant f_{[F|C]}(P^\star_{[S|D]}) - f_{[S|D]}(P^\star_{[S|D]}) \\
& = \sum_{i \in P^\star_{[S|D]}} [f_i - s_i] + \sum_{j \in J} (\min\{c_{ij} : i \in P^\star_{[S|D]}\} - \min\{d_{ij} : i \in P^\star_{[S|D]}\}).
\end{aligned}$$

Let $c_{i_c(j)j} = \min\{c_{ij} : i \in P^\star_{[S|D]}\}$ and $d_{i_d(j)j} = \min\{d_{ij} : i \in P^\star_{[S|D]}\}$. Then

$$\begin{aligned}
& f_{[F|C]}(P^\star_{[F|C]}) - f_{[S|D]}(P^\star_{[S|D]}) \\
& \leqslant \sum_{i \in P^\star_{[S|D]}} [f_i - s_i] + \sum_{j \in J} [c_{i_c(j)j} - d_{i_d(j)j}] \\
& \leqslant \sum_{i \in P^\star_{[S|D]}} [f_i - s_i] + \sum_{j \in J} [c_{i_d(j)j} - d_{i_d(j)j}] \\
& \leqslant \sum_{i \in P^\star_{[S|D]}} [f_i - s_i] + \sum_{j \in J} [\max\{c_{ij} - d_{ij} : i \in P^\star_{[S|D]}\}] \\
& \leqslant \sum_{i \in P^\star_{[S|D]}} |f_i - s_i| + \sum_{j \in J} [\max\{|c_{ij} - d_{ij}| : i \in I\}] \\
& \leqslant \sum_{i \in I} |f_i - s_i| + \sum_{j \in J} [\max\{|c_{ij} - d_{ij}| : i \in I\}] \\
& = \rho([F|C], [S|D]).
\end{aligned}$$

□

Theorem 1 implies that if we have an optimal solution to a SPLP instance $[S|D]$, then we have an upper bound for *all* SPLP instances $[F|C]$ of the same size. This upper bound is actually the distance between the two instances, distances being defined by the accuracy measure (10). Also if the solution to $[S|D]$ can be computed in polynomial time (i.e., $[S|D]$ belongs to a polynomially solvable special case) then an upper bound to the cost of an *as yet unknown* optimal solution to $[F|C]$ can be obtained in polynomial time. If the distance between the instances is not more than a prescribed accuracy α, then the optimal solution of $[S|D]$ is, in fact, a solution to $[F|C]$ within the prescribed accuracy. This theorem forms the basis of data correcting.

In general, the data correcting procedure works as follows. It assumes that we know a class of polynomially solvable instances of the problem. It starts by choosing a polynomially solvable SPLP instance $[S|D]$ from that class of instances, preferably as close as possible to the original instance $[F|C]$. If $\rho([F|C], [S|D]) \leqslant \alpha$, the procedure terminates and returns an optimal solution to $[S|D]$ as an approximation of an optimal solution to $[F|C]$. The instance $[F|C]$ is said to be 'corrected' to the instance $[S|D]$, which is solved polynomially to generate the solution output by the procedure. Otherwise, the set of feasible solutions for the problem is partitioned into two subsets. For the SPLP, one of these subsets is comprised of solutions that locate a plant at a given site, and the other is comprised of solutions that do not. The two new instances thus formed are perturbed in a way that they both change into instances that are within a distance α from a polynomially solvable instance. The procedure is continued until an instance with a proximity measure not more than α is obtained for all the subsets generated.

2.3. THE PRELIMINARY PRESERVATION PROCEDURE

The preliminary preservation procedure is one that tries to reduce the set of solutions in which to search for optimal solutions to a given instance. It applies to the minimization of supermodular (and maximization of submodular) functions. The function $f_{[F|C]}(P)$ is called *supermodular* on $[P_L, P_U] = \{P : P_L \subseteq P \subseteq P_U\}$ with subsets P_L and P_U of I, such that $\emptyset \subset P_L \subseteq P_U \subseteq I$ if for each $P, Q \in [P_L, P_U]$ it holds that $f_{[F|C]}(P)+f_{[F|C]}(Q) \leqslant f_{[F|C]}(P\cup Q)+f_{[F|C]}(P\cap Q)$. Since the objective function of the SPLP is supermodular, we can apply the procedure to this problem.

Consider P_L as the set of sites where plants will definitely be located in an optimal solution, and P_U as the set of all candidate locations for locating plants in optimal solutions. In other words, plants will definitely *not* be located in any site belonging to $I \setminus P_U$ in an optimal solution. Let $f^{\star}_{[F|C]}[P_L, P_U] = \min\{f_{[F|C]}(P) : P_L \subseteq P \subseteq P_U\}$. The following result is a straightforward application of Theorem 1 in Goldengorin et al. (1999) to the SPLP.

THEOREM 2. *Consider $P_L, P_U \subseteq I$, such that $\emptyset \subset P_L \subseteq P_U \subseteq I$, and let $k \in P_U \setminus P_L$. Then the following assertions hold:*

(a) $f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}] - f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\})$;

(b) $f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U] - f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant f_{[F|C]}(P_U) - f_{[F|C]}(P_U \setminus \{k\})$.

Proof. We will prove part (a) here. The proof of part (b) is similar. Let $P \in [P_L, P_U \setminus \{k\}]$, with $f_{[F|C]}(P \cup \{k\}) = f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U]$. It then follows from the definition of supermodularity that $f_{[F|C]}(P_L \cup \{k\}) + f_{[F|C]}(P) \leqslant f_{[F|C]}(P \cup \{k\}) + f_{[F|C]}(P_L)$, which implies that $f_{[F|C]}(P) \leqslant f_{[F|C]}(P \cup \{k\}) + f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\})$. Hence, $f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant f_{[F|C]}(P \cup \{k\}) + f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\})$. Thus $f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}] - f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\})$, which proves part (a). □

An immediate corollary to Theorem 2 is the following.

COROLLARY 1. *Consider $P_L, P_U \subseteq I$, such that $\emptyset \subset P_L \subseteq P_U \subseteq I$, and let $k \in P_U \setminus P_L$. Then the following preservation rules are valid:*

Preservation Rule 1: *If* $f_{[F|C]}(P_U \setminus \{k\}) \geqslant f_{[F|C]}(P_U)$, *then* $f^{\star}_{[F|C]}[P_L, P_U] = f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}]$.

Preservation Rule 2: *If* $f_{[F|C]}(P_L \cup \{k\}) \geqslant f_{[F|C]}(P_L)$, *then* $f^{\star}_{[F|C]}[P_L, P_U] = f^{\star}_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant f^{\star}_{[F|C]}[P_L \cup \{k\}, P_U]$.

Informally, the Preservation Rule 1 means that with $P_L \setminus \{k\}$ open, the fixed cost of k is less than the transportation savings gained by having k open. The Preservation Rule 2 means that with P_L open, the fixed cost of opening k exceeds the transportation savings realized by opening k.

Using the preservation rules from Corollary 1, we can considerably reduce the search space for any given problem instance. This is done by the preliminary preservation procedure (PP) described below. For a given instance $[F|C]$ of the SPLP, the procedure takes two sets P_L^i and P_U^i ($P_L^i \subseteq P_U^i$) as input, and outputs two sets P_L^o and P_U^o ($P_L^i \subseteq P_L^o \subseteq P_U^o \subseteq P_U^i$), such that $f^{\star}_{[F|C]}[P_L^i, P_U^i] = f^{\star}_{[F|C]}[P_L^o, P_U^o]$. The running time of the procedure is $\mathcal{O}(m^2)$ (Theorem 2, Goldengorin *et al.*, 1999).

Procedure PP (P_L^i, P_U^i)
begin
  $P_L^o \leftarrow P_L^i$; $P_U^o \leftarrow P_U^i$;

  if $P_L^o = P_U^o$ **then**
    return (P_U^o, P_U^o);

  Compute $\delta^+ \leftarrow \min_{k \in P_U^o \setminus P_L^o}\{f_{[F|C]}(P_U^o) - f_{[F|C]}(P_U^o \setminus \{k\})\}$;
  Compute $\delta^- \leftarrow \min_{k \in P_U^o \setminus P_L^o}\{f_{[F|C]}(P_L^o) - f_{[F|C]}(P_L^o \cup \{k\})\}$;

  if $\delta^+ \leqslant 0$ **then** { *Preservation Rule 1* }
  begin
    Compute $r^+ \leftarrow \min\{k : f_{[F|C]}(P_U^o) - f_{[F|C]}(P_U^o \setminus \{k\}) = \delta^+\}$;
    call PP($P_L^o \cup \{r^+\}$, P_U^o);
  end
  else if $\delta^- \leqslant 0$ **then** { *Preservation Rule 2* }
  begin
    Compute $r^- \leftarrow \min\{k : f_{[F|C]}(P_L^o) - f_{[F|C]}(P_L^o \cup \{k\}) = \delta^-\}$;
    call PP(P_L^o, $P_U^o \setminus \{r^-\}$);
  end
  else return (P_L^o, P_U^o);
end;

---

3. The Data Correcting Algorithm

The Data Correcting Algorithm (DCA) that we propose in this paper is one that uses a strong reduction procedure (RP, see Subsection 3.1) to reduce the original instance into a smaller 'core' instance, and then uses a data correcting procedure (DCP, see Subsection 3.2) to obtain a solution to the original instance, whose cost is not more than a pre-specified amount α more than the cost of an optimal solution.

3.1. THE REDUCTION PROCEDURE

The first preprocessing rules for the SPLP involving both fixed costs and transportation costs appeared in Khumawala (1972). In terms of Hammer functions, these rules are stated as follows. We assume (without loss of generality) that we cannot partition I into sets I_1 and I_2, and J into sets J_1 and J_2, such that the transportation costs from sites in I_1 to clients in J_2, and from sites in I_2 to clients in J_1 are not finite. We assume too, that the site indices are arranged in non-increasing order of $f_i + \sum_{j \in J} c_{ij}$ values. Based on the following theorem, we formulate two preprocessing rules, namely, RO and RC.

THEOREM 3. *Let $\mathcal{H}_{[F|C]}(\mathbf{y})$ be the Hammer function corresponding to the SPLP instance $[F|C]$ in which like terms have been aggregated. For each site index k, let a_k be the coefficient of the linear term corresponding to y_k and let t_k be the sum of the coefficients of all non-linear terms containing y_k. Then the following assertion holds.*

RO: *If $a_k \geqslant 0$, then there is an optimal solution $\mathbf{y}^\star$ in which $y_k^\star = 0$.*

RC: *If $a_k + t_k \leqslant 0$, then there is an optimal solution $\mathbf{y}^\star$ in which $y_k^\star = 1$.*

Proof.

RO: Suppose $a_k \geqslant 0$. Let us consider a vector $\mathbf{y}$ for which $y_k = 1$ and a vector $\mathbf{y}'$ for which $y_i' = y_i$ for each $i \neq k$, $y_k' = 0$. Now $\mathcal{H}_{[F|C]}(\mathbf{y}) - \mathcal{H}_{[F|C]}(\mathbf{y}') = a_k \geqslant 0$. Hence $\mathbf{y}'$ is preferable to $\mathbf{y}$. This shows that there exists an optimal solution $\mathbf{y}^\star$ with $y_k^\star = 0$.

RC: Next suppose that $a_k + t_k \leqslant 0$. Note that $a_k + t_k \leqslant 0$ and $t_k > 0$ implies that $a_k < 0$, and t_k cannot be negative for any index k, since the non-linear terms of the Hammer function are all non-negative.

Consider two solutions $\mathbf{y}'$ and $\mathbf{y}$, such that $y_i' = y_i$ for each $i \neq k$, $y_k' = 1$, and $y_k = 0$. Then

$$\begin{aligned} &\mathcal{H}_{[F|C]}(\mathbf{y}') - \mathcal{H}_{[F|C]}(\mathbf{y}) \\ &= -f_k y_k' + \sum_{j=1}^{n} \sum_{\substack{p=1 \\ k \in \{\pi_{1j}, \ldots, \pi_{pj}\}}}^{m-1} \Delta c[p,j] \prod_{r=1}^{p} y_{\pi_{rj}}' \end{aligned} \tag{11}$$

which, on separating the linear and non-linear terms, yields that

$$= a_k y'_k + \sum_{j=1}^{n} \sum_{\substack{p=2 \\ k\in\{\pi_{1j},\ldots,\pi_{pj}\}}}^{m-1} \Delta c[p, j] \prod_{r=1}^{p} y'_{\pi_{rj}}. \tag{12}$$

An upper bound to (12) is $(a_k y'_k + t_k y'_k)$ which is obtained by setting y'_i to 1 for each $i \neq k$, since all non-linear terms in the Hammer function have non-negative coefficients. Thus

$$\mathcal{H}_{[F|C]}(\mathbf{y}') - \mathcal{H}_{[F|C]}(\mathbf{y}) \leqslant a_k y'_k + t_k y'_k = a_k + t_k \leqslant 0,$$

which makes $\mathbf{y}'$ preferable to $\mathbf{y}$. This shows that there is an optimal solution $\mathbf{y}^\star$ with $y^\star_k = 1$. Of course, if $y'_i = 1$ for each $i \neq k$, then setting $y'_k = 1$ leads to an infeasible solution. □

Notice that the rules RO and RC primarily try to either open or close sites. If it succeeds, it also changes the Hammer function for the instance, reducing the number of non-linear terms therein. In the remaining portion of this subsection, we describe a completely new reduction procedure (RP), whose primary aim is to reduce the coefficients of terms in the Hammer function, and if we can reduce it to zero, to eliminate the term from the Hammer function. This procedure is based on fathoming rules of branch and bound algorithms and data correcting principles.

Let us assume that we have an upper bound (UB) on the cost of an optimal solution for the given SPLP instance. This can be obtained by running a heuristic on the problem data. Now consider any non-linear term $s \prod_{r=1}^{k} y_{\pi_{rj}}$ in the Hammer function. This term will contribute to the cost of a solution, only if plants are *not* located in any of the sites $\pi_{1j}, \ldots, \pi_{kj}$. Let LB be a lower bound on the optimal solution of the SPLP with respect to the subproblem for which no facilities are located in the sites $\pi_{1j}, \ldots, \pi_{kj}$. If $LB \leqslant UB$, then we cannot make any judgement about this term. On the other hand, if $LB > UB$, then we know that there cannot be an optimal solution with $y_{\pi_{1j}} = \ldots = y_{\pi_{kj}} = 1$. In this case, if we reduce the coefficient s by $LB - UB - \varepsilon$, ($\varepsilon > 0$, small), then the new Hammer function and the original one have identical sets of optimal solutions. If after the reduction, s is non-positive, then the term can be removed from the Hammer function. Such changes in the Hammer function alter the values of t_k, and can possibly allow us to use Khumawala's rules to close certain sites. Once some sites are closed, some of the linear terms in the Hammer function change into constant terms, and some of the quadratic terms change into linear ones. These changes cause changes in both the a_k and the t_k values, and can make further application of Khumawala's rules possible, thus preprocessing some other sites, and making further changes in the Hammer function. A pseudocode of the reduction procedure (RP) is provided below.

Procedure RP ($\mathcal{H}_{[F|C]}(\mathbf{y})$)
begin
  repeat
    Compute an upper bound UB for the instance;
    for each nonlinear term $s\prod_{r=1}^{k} y_{\pi_{rj}}$ in $\mathcal{H}_{[F|C]}(\mathbf{y})$ **do**
    begin
      Compute lower bound LB on the cost of solutions in which plants are not located in sites $\pi_{1j}, \ldots, \pi_{kj}$;
      if $LB > UB$ **then**
        Reduce the coefficient of the term by $\min\{s, LB - UB - \varepsilon\}$;
    Apply Khumawala's rules until no further preprocessing is possible;
    Recompute the Hammer function $\mathcal{H}_{[F|C]}(\mathbf{y})$;
  until no further preprocessing of sites was achieved in the current iteration;
end;

Let us consider the application of all preprocessing rules to the example from Section 2.1 with the Hammer function $\mathcal{H}_{[F|C]}(\mathbf{y}) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 21y_1y_2y_4 + 4y_2y_3y_4$. The values of a_k, t_k and $a_k + t_k$ are as follows:

k :	1	2	3	4
a_k :	−8	−1	−3	−4
t_k :	27	27	12	37
$a_k + t_k$:	19	26	9	33

It is clear that neither RO nor RC is applicable here, since the coefficient of the term $21y_1y_2y_4$ is too large. Therefore, we try to reduce this coefficient by applying the RP.

The upper bound $UB = 51$ to the original problem can be obtained by setting $y_1 = y_4 = 1$ and $y_2 = y_3 = 0$. A lower bound to the subproblem under the restriction $y_1 = y_2 = y_4 = 1$ is 73, since $\mathcal{H}_{[F|C]}(1, 1, 0, 1) = 73$. Note that UB and LB are calculated here for different subproblems. In virtue of RP, we can reduce the coefficient of $21y_1y_2y_4$ by $73 - 51 - \varepsilon = 20$, so that the new Hammer function, with the same set of optimal solutions as the original function, becomes $\mathcal{H}'(\mathbf{y}) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 1y_1y_2y_4 + 4y_2y_3y_4$.

The updated values of a_k, t_k, and $a_k + t_k$ are presented below.

k :	1	2	3	4
a_k :	-8	-1	-3	-4
t_k :	7	7	12	17
$a_k + t_k$:	-1	6	9	13

RC can immediately be applied in this situation to set $y_1 = 1$. After updating $\mathcal{H}'(\mathbf{y})$, we can apply RO and set $y_2 = y_4 = 0$. This allows us to apply RC again to set $y_3 = 1$, yielding the optimal solution $(1, 0, 1, 0)$ with cost 48.

3.2. THE DATA CORRECTING PROCEDURE

Let us suppose that the preliminary preservation (PP) procedure is applied to the SPLP instance $[F|C]$. On termination, it outputs two subsets P_L^o and P_U^o, $\emptyset \subset P_L^o \subseteq P_U^o \subseteq I$. If $P_L^o = P_U^o$, then the instance is said to have been solved by this procedure, and the set P_L^o is an optimal solution. Since the PP procedure is a polynomial time algorithm, instances that it solves to optimality constitute a class of algorithmically defined polynomially solvable instances. We call such instances *PP-solvable*. We use this class of polynomially solvable instances in our algorithm, since it is one of the best among the polynomially solvable cases discussed in Goldengorin (1995).

Next suppose that the given instance is not PP-solvable. In that case we try to extend the idea of the PP procedure to obtain a solution such that the difference between its cost and the cost of an optimal solution is bounded by a pre-defined value α. This is the basic idea behind the data correcting procedure.

We will introduce a few notations to improve the readability of this subsection. Consider the SPLP instance $[F|C]$ and two sets $P_L, P_U \subseteq I$ such that $P_L \subset P_U$. Let

$$\begin{aligned}
\delta_k^- &= f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\}) \text{ for each } k \in P_U \setminus P_L,\\
\delta^- &= \min\{\delta_k^- : k \in P_U \setminus P_L\},\\
r^- &= \min\{k : \delta_k^- = \delta^-,\ k \in P_U \setminus P_L\},\\
\delta_k^+ &= f_{[F|C]}(P_U) - f_{[F|C]}(P_U \setminus \{k\}) \text{ for each } k \in P_U \setminus P_L,\\
\delta^+ &= \min\{\delta_k^+ : k \in P_U \setminus P_L\},\\
r^+ &= \min\{k : \delta_k^+ = \delta^+,\ k \in P_U \setminus P_L\}.
\end{aligned}$$

Note that $\delta_k^- = -a_k$ and $\delta_k^+ = a_k + t_k$ (see Goldengorin, 1995). Therefore, these quantities can be calculated very efficiently when one uses a Beresnev function representation of the SPLP. Lemma 1 is a restatement of the preservation rules in Corollary 1.

LEMMA 1. *Consider $P_L, P_U \subseteq I$, such that $\emptyset \subset P_L \subset P_U \subseteq I$. Let $k \in P_U \setminus P_L$, and let P_A be an arbitrary subset of I. Then the following holds.*

(a) If $\delta_k^- \leqslant 0$ *and* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant \gamma \leqslant \alpha$, *then* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U] \leqslant \gamma \leqslant \alpha$*; and*

(b) if $\delta_k^+ \leqslant 0$ *and* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant \gamma \leqslant \alpha$, *then* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U] \leqslant \gamma \leqslant \alpha$.

In case both δ^- and δ^+ are strictly positive, then Lemma 1 is no longer applicable, and the PP procedure terminates. If $\delta = \min(\delta^-, \delta^+) \leqslant \alpha$, we could however correct the data of the instance so that either the costs of all solutions P, $P_L \subseteq P \subseteq P_U \setminus \{k\}$ increase by δ^-, or the costs of all solutions P, $P_L \cup \{k\} \subseteq P \subseteq P_U$ increase by δ^+, and Lemma 1 becomes applicable again. The solution that we hope to obtain by this correcting procedure will have an accuracy of δ according to Theorem 1. Instead of changing the data in the instance, we may equivalently decrease the allowable accuracy value from α to $\alpha - \delta$. This gives rise to Lemma 2.

LEMMA 2. *Consider* $P_L, P_U \subseteq I$, *such that* $\emptyset \subset P_L \subset P_U \subseteq I$. *Let* $k \in P_U \setminus P_L$, *and let* P_A *be an arbitrary subset of* I. *Then the following holds.*

(a) If $0 \leqslant \delta_k^- \leqslant \alpha$ *and* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant \gamma \leqslant \alpha - \delta_k^-$, *then* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U] \leqslant \gamma + \delta_k^- \leqslant \alpha$*; and*

(b) if $0 \leqslant \delta_k^+ \leqslant \alpha$ *and* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant \gamma \leqslant \alpha - \delta_k^+$, *then* $f_{[F|C]}(P_A) - f^\star_{[F|C]}[P_L, P_U] \leqslant \gamma + \delta_k^+ \leqslant \alpha$.

Proof. We prove the first part of the lemma. The proof of the second part is similar. There are two cases to be considered.

Case 1: $f^\star_{[F|C]}[P_L, P_U] = f^\star_{[F|C]}[P_L, P_U \setminus \{k\}]$. In this case, the result follows trivially.

Case 2: $f^\star_{[F|C]}[P_L, P_U] = f^\star_{[F|C]}[P_L \cup \{k\}, P_U]$. From Theorem 2(a),

$$\begin{aligned} & f^\star_{[F|C]}[P_L, P_U \setminus \{k\}] - f^\star_{[F|C]}[P_L \cup \{k\}, P_U] \leqslant \delta_k^- \\ \Longleftrightarrow\ & f^\star_{[F|C]}[P_L, P_U \setminus \{k\}] \leqslant f^\star_{[F|C]}[P_L, P_U] + \delta_k^-. \end{aligned}$$

The result follows. □

In case $\delta = \min(\delta^-, \delta^+) > \alpha$, then data correction cannot guarantee a solution within the prescribed allowable accuracy, and hence we need to use a branching procedure.

The data correcting procedure (DCP, see below) in our algorithm takes two sets $P_L, P_U \subseteq I$ ($\emptyset \subset P_L \subset P_U \subseteq I$) and α as input. It outputs a solution P^γ and a bound γ, such that $f_{[F|C]}(P^\gamma) - f_{[F|C]}(P^\star) \leqslant \gamma \leqslant \alpha$, where $P^\star$ is an optimal solution to $[F|C]$. It is a recursive procedure, that first tries to reduce the set $P_U \setminus P_L$ by applying Lemma 1. If Lemma 1 cannot be applied, then it tries to apply Lemma 2 to reduce it. We do not use the reduction procedure at this stage since it increases the computational times substantially without reducing the core problem appreciably. If even this lemma cannot be applied, then the procedure branches on a member $k \in P_U \setminus P_L$ and invokes two instances of DCP, one with sets $P_L \cup \{k\}$

and P_U, and the other with sets P_L and $P_U \setminus \{k\}$. Notice that the solutions searched by the two invocations of DCP are mutually exclusive and exhaustive. A bound is used to remove unpromising subproblems from the solution tree. The choice of the branching variable $k \in P_U \setminus P_L$ in DCP is motivated by the observation that $a_k < 0$ and $t_k + a_k > 0$ for each of these indices. (These are the preconditions of the branching rule.) A plant would have been located in this site in an optimal solution if the coefficient of the linear term involving y_k in the Hammer function would have been increased by $-a_k$. We could have predicted that a plant would not be located there if the same coefficient would have been decreased by $t_k + a_k$. Therefore we could use $\phi_k = \text{average}(-a_k, t_k + a_k) = \frac{t_k}{2}$ as a measure of the chance that we will *not* be able to predict the fate of site k in any subproblem of the current subproblem. If we want to reduce the size of the branch and bound tree by assigning values to such variables, then we can think of a branching function that branches on the index $k \in P_U \setminus P_L$ with the largest ϕ_i value.

Procedure DCP (P_L, P_U, α)
begin
  if $P_L = P_U$ **then**
    return (P_U, 0);
  Compute $\delta^+, \delta^-, r^+, r^-$;
  { *Apply Lemma 1 (Preliminary Preservation)* }
  if $\delta^+ \leqslant 0$ **then** { *Lemma 1(b)* }
    $(P^\gamma, \gamma) \leftarrow \text{DCP}(P_L \cup \{r^+\}, P_U, \alpha)$;
  else if $\delta^- \leqslant 0$ **then** { *Lemma 1(a)* }
    $(P^\gamma, \gamma) \leftarrow \text{DCP}(P_L, P_U \setminus \{r^-\}, \alpha)$;
  { *Apply Lemma 2 (Data Correction)* }
  else if $\delta^+ \leqslant \alpha$ **then** { *Lemma 2(b)* }
  begin
    $(P^\gamma, \gamma) \leftarrow \text{DCP}(P_L \cup \{r^+\}, P_U, \alpha - \delta^+)$;
    $\gamma \leftarrow \delta^+$;
  end
  else if $\delta^- \leqslant \alpha$ **then** { *Lemma 2(a)* }
  begin
    $(P^\gamma, \gamma) \leftarrow \text{DCP}(P_L, P_U \setminus \{r^-\}, \alpha - \delta^-)$;
    $\gamma \leftarrow \delta^-$;
  end
  { *Branch* }
  else
  begin
    select $k \in P_U^o \setminus P_L^o$; { *Branching Rule* }
    if the bound obtained using the sets $P_L \cup \{k\}$ and P_U is better
     than the best solution found so far, **then**
      $(P^{\gamma^+}, \gamma^+) \leftarrow \text{DCP}(P_L \cup \{k\}, P_U, \alpha)$;

if the bound obtained using the sets P_L and $P_U \setminus \{k\}$ is better than the best solution found so far, **then**
$(P^{\gamma^-}, \gamma^-) \leftarrow \text{DCP}(P_L, P_U \setminus \{k\}, \alpha)$;
$P^\gamma \leftarrow \arg\min\{f_{[F|C]}(P^{\gamma^+}), f_{[F|C]}(P^{\gamma^-})\}$;
$\gamma \leftarrow \min\{f_{[F|C]}(P^{\gamma^+}), f_{[F|C]}(P^{\gamma^-})\} - \min\{f_{[F|C]}(P^{\gamma^+}) - \gamma^+, f_{[F|C]}(P^{\gamma^-}) - \gamma^-\}$;
end

return (P^γ, γ);
end;

---

4. Computational Experiments

The execution of the DCA can be divided into two stages, a *preprocessing* stage in which the given instance is reduced to a core instance by using RP; and a *solution* stage in which the core instance is solved using DCP.

In the preprocessing stage we experimented with the following three reduction procedures.

(a) The "delta" and "omega" rules from Khumawala (1972);
(b) Procedure RP with the combinatorial Khachaturov-Minoux bound to obtain a lower bound; and
(c) Procedure RP with the LP dual-ascent Erlenkotter bound (see Erlenkotter, 1978) to obtain a lower bound.

The Khachaturov-Minoux bound lb is a combinatorial bound for general supermodular functions (see Khachaturov, 1968; Minoux, 1977). It can be stated as follows.

If $f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\}) > 0$ and $f_{[F|C]}(P_U) - f_{[F|C]}(P_U \setminus \{k\}) > 0$ for all $k \in P_U \setminus P_L$, then $lb = \max\{lb_1, lb_2\}$ where
$lb_1 = f_{[F|C]}(P_L) - \sum_{k \in P_U \setminus P_L} [f_{[F|C]}(P_L) - f_{[F|C]}(P_L \cup \{k\})]$ and
$lb_2 = f_{[F|C]}(P_U) - \sum_{k \in P_U \setminus P_L} [f_{[F|C]}(P_U) - f_{[F|C]}(P_U \setminus \{k\})]$.

We also experimented with the Khachaturov-Minoux bound and the Erlenkotter bound in the implementation of the DCP.

The effectiveness of the reduction procedure can be measured either by computing the number of free locations in the core instance, or by computing the number of non-zero nonlinear terms present in the Hammer function of the core instance. Note that the number of non-zero nonlinear terms present in the Hammer function is an upper bound on the number of unassigned customers in the core instance. Tables 1 and 2 shows how the various methods of reduction perform on the benchmark SPLP instances in the OR-Library (Beasley (1993b)). The existing preprocessing rules due to Khumawala (1972) and Goldengorin et al. (2000) (i.e. procedure (a), which was used in the SPLP example in Goldengorin et al. (1999))

Table 1. Number of free locations after preprocessing SPLP instances in the OR-Library

Problem	m	n	m after procedure		
			a	b	c
cap71	16	50	4	0	0
cap72	16	50	6	0	0
cap73	16	50	6	3	3
cap74	16	50	2	0	0
cap101	25	50	9	0	0
cap102	25	50	13	3	0
cap103	25	50	14	0	0
cap104	25	50	12	0	0
cap131	50	50	34	32	8
cap132	50	50	27	25	5
cap133	50	50	25	19	10
cap134	50	50	19	0	0

cannot solve any of the OR-Library instances to optimality. However, the variants of the new reduction procedure (i.e., procedures (b) and (c)) solve a large number of these instances to optimality. Procedure (c), based on the Erlenkotter bound is marginally better than procedure (b) in terms of the number of free locations (Table 1), but substantially better in terms of the number of non-zero nonlinear terms in the Hammer function (Table 2).

The information in Tables 1 and 2 can be combined to show that some of the problems that are not solved by these procedures can actually be solved by inspection of the core instances. For example, consider cap74. We see that the core problem (using procedure (a)) has two free variables and one non-linear term. Therefore the Beresnev function of the core instance looks like

$$A + py_u + qy_w + ry_u y_w,$$

where $p, q < 0, r > 0$, $\min\{p + r, p + q\} > 0$ and A is a constant. The minima of such functions are easy to obtain by inspection.

In addition, Tables 1 and 2 demonstrate the superiority of the new preprocessing rule over the “delta” and “omega” rules. Consider for example the problem cap132. The “delta” and “omega” rules reduce the problem size from $m = 50$ and 2389 non-zero nonlinear terms to $m' = 27$ and 112 non-zero nonlinear terms. However, the new preprocessing rule reduces the same problem to one having $m' = 5$ and 3 non-zero nonlinear terms.

Table 2. Number of non-zero nonlinear terms in the Hammer function after preprocessing SPLP instances in the OR-Library

Problem	No. of non-zero terms			
	Before preprocessing	After procedure a	b	c
cap71	699	6	0	0
cap72	699	12	0	0
cap73	699	13	2	2
cap74	699	1	0	0
cap101	1147	24	0	0
cap102	1147	33	2	0
cap103	1147	38	0	0
cap104	1147	29	0	0
cap131	2389	163	135	8
cap132	2389	112	92	3
cap133	2389	101	60	11
cap134	2389	62	0	0

In order to test the effect of bounds in the DCA, we compared the execution times of DCA using the two bounds on some difficult problems of the type suggested in Körkel (1989) (see Subsection 4.4 for more details). The problems were divided into seven sets. Each set consists of five problems, each having 65 sites and 65 clients (see Subsection 4.4 for more details regarding these problems). From Table 3 we see that the Erlenkotter bound reduces the execution time taken by the Khachaturov-Minoux bound (that was used in the SPLP example in Goldengorin et al. (1999)) by a factor more than 100. This is not surprising, since the Khachaturov-Minoux bound is derived for a general supermodular function, while the Erlenkotter bound is specific to the SPLP.

We report our computational experience with the DCA on several benchmark instances of the SPLP in the remainder of this section. The performance of the algorithm is compared with that of the algorithms described in the papers that suggested these instances. We implemented the DCA in PASCAL, compiled it using Prospero Pascal, and ran it on a 733 MHz Pentium III machine. The computation times we report are in seconds on our machine.

4.1. BILDE AND KRARUP-TYPE INSTANCES

These are the earliest benchmark problems that we consider here. The exact instance data is not available, but the process of generating the problem instances is

Table 3. Comparison of bounds used with the DCA on Körkel-type instances with $m = n = 65$

Problem set	Execution time of the DCP (sec)	
	Khachaturov-Minoux bound	Erlenkotter Bound
Set 1	119.078	0.022
Set 2	290.388	0.040
Set 3	458.370	0.056
Set 4	158.386	0.054
Set 9	428.598	0.588
Set 10	542.530	0.998
Set 11	479.092	2.280

Table 4. Description of the instances in Bilde and Krarup (1977)

Type	m	n	f_i	c_{ij}
B	50	100	Discrete uniform (1000, 10000)	Discrete uniform (0, 1000)
C	50	100	Discrete uniform (1000, 2000)	Discrete uniform (0, 1000)
Dq †	30	80	Identical, $1000\times q$	Discrete uniform (0, 1000)
Eq †	50	100	Identical, $1000\times q$	Discrete uniform (0, 1000)

† $q = 1, \ldots, 10$.

described in Bilde and Krarup (1977). There are 22 different classes of instances, and Table 4 summarizes their characteristics.

In our experiments we generated 10 instances for each of the types of problems, and used the mean values of our solutions to evaluate the performance of our algorithm with the one used in Bilde and Krarup (1977). In our implementation, we used reduction procedure (b) and the Khachaturov-Minoux bound in the DCP.

The reduction procedure was not useful for these instances, but the DCA could solve all the instances in reasonable time. The results of our experiments are presented in Table 5. The performance of the algorithm implemented in Bilde and Krarup (1977) was measured in terms of the number of branching operations performed by the algorithm and its execution time in CPU seconds on a IBM 7094 machine. We estimate the number of branching operations by our algorithm as the logarithm (to the base 2) of the number of subproblems it generated. From the table we see that the DCA reduces the number of subproblems generated by the algorithm in Bilde and Krarup (1977) by several orders of magnitude. This is especially interesting because Bilde and Krarup use a bound (discovered in 1967) identical to the Erlenkotter bound in their algorithm (see Körkel, 1989) and we use the

Table 5. Results from Bilde and Krarup-type instances

Problem type	DCA Branching	DCA CPU time	Bilde and Krarup Branching	Bilde and Krarup CPU Ttime†
B	11.72	0.67	43.3	4.33
C	17.17	14.81	⋆	>250
D1	13.80	0.65	216	11
D2	12.13	0.38	218	24
D3	10.87	0.19	169	19
D4	10.25	0.15	141	17
D5	9.24	0.07	106	14
D6	8.99	0.09	101	15
D7	8.79	0.09	83	13
D8	8.60	0.09	55	11
D9	8.15	0.07	47	11
D10	7.29	0.03	43	11
E1	18.66	35.28	1271	202
E2	16.14	8.64	1112	172
E3	14.59	3.81	384	82
E4	13.65	2.74	258	65
E5	12.73	2.01	193	53
E6	11.82	0.90	136	43
E7	10.82	0.53	131	42
E8	10.79	0.68	143	48
E9	10.62	0.76	117	44
E10	10.36	0.69	79	37

† IBM7094 s.

⋆ Could not be solved in 250 s.

Khachaturov-Minoux bound. The CPU time required by the DCA to solve these problems were too low to warrant the use of any $\alpha > 0$.

4.2. GALVÃO AND RAGGI-TYPE INSTANCES

Galvão and Raggi (1989) developed a general 0-1 formulation of the SPLP and presented a three-stage method to solve it. The benchmark instances suggested in this work are unique, in that the fixed costs are assumed to come from a Normal distribution rather than the more commonly used Uniform distribution. The transportation costs for an instance of size $m \times n$ with $m = n$ are computed as follows. A network, with a given arc density δ is first constructed, and the arcs in the network are assigned lengths sampled from a uniform distribution in the range $[1, n]$ (except

Table 6. Description of the instances in Galvão and Raggi (1989)

Problem size	Density	Fixed costs' parameters	
($m = n$)	δ	Mean	Standard deviation
10	0.300	4.3	2.3
20	0.150	9.4	4.8
30	0.100	13.9	7.4
50	0.061	25.1	14.1
70	0.043	42.3	20.7
100	0.025	51.7	28.9
150	0.018	186.1	101.5
200	0.015	149.5	94.4

for $n = 150$, where the range is $[1, 500]$). The transportation cost from i to j is the length of the cheapest path from i to j. The problem characteristics provided in Galvão and Raggi (1989) are summarized in Table 6.

As with the data in Bilde and Krarup (1977), the exact data for the instances are not known. So we generated 10 instances for each problem size, and used the mean values of the solutions for comparison purposes. In our DCA implementation, we used reduction procedure (b) and the Khachaturov-Minoux bound in the DCP. The comparative results are given in Table 7. Since the computers used are different, we cannot make any comments on the relative performance of the solution procedures. However, since the average number of subproblems generated by the DCA is always less than 10 for each of these instances, we can conclude that these problems are easy for our algorithm. In fact they are too easy for the DCA to warrant $\alpha > 0$.

Notice that the average number of opened plants in the optimal solutions to the instances we generated is quite close to the number of opened plants in the optimal solutions reported in Galvão and Raggi (1989). Also notice that the reduction procedure was quite effective — it solved 35 of the 80 instances generated.

4.3. INSTANCES FROM THE OR-LIBRARY

The OR-Library (Beasley, 1993b) has a set of instances of the SPLP. These instances were solved in Beasley (1993a) using an algorithm based on the Lagrangian heuristic for the SPLP. Here too, we used reduction procedure (b) and the Khachaturov-Minoux bound in the DCP. We solved the problems to optimality using the DCA. The results of the computations are provided in Table 8. The execution times suggest that the DCA is faster than the Lagrangian heuristic described in Beasley (1993a). The reduction procedure was also quite effective for these instances, solving four of the 16 instances to optimality, and reducing the number

Table 7. Results from Galvão and Raggi-type instances

Problem	DCA				Galvão and Raggi	
Size ($m = n$)	No. solved by preprocessing	No. of sub-problems†	CPU time†	No. of open plants†	CPU time⋆	No. of open plants
10	6	2.3	<0.001	4.7	<1	3
20	5	2.4	<0.001	9.0	<1	8
30	7	1.8	0.002	13.6	1	11
50	7	2.6	0.002	20.3	2	20
70	2	3.8	0.004	28.8	6	31
100	3	3.5	0.011	41.1	6	44
150	1	7.8	0.010	64.4	25	74
200	4	2.9	0.158	81.8	63	84

† Average over 10 instances.
⋆ IBM 4331 s.

Table 8. Results from OR-Library instances

Problem name	m	n	DCA			CPU time (Beasley)†	No. of open plants
			m after pre-processing	No, of sub-problems	CPU time		
cap71	16	50	⋆	0	<0.01	0.11	11
cap72	16	50	⋆	0	<0.01	0.08	9
cap73	16	50	⋆	0	<0.01	0.11	5
cap74	16	50	⋆	0	<0.01	0.05	4
cap101	25	50	9	6	<0.01	0.18	15
cap102	25	50	13	16	<0.01	0.16	11
cap103	25	50	14	16	<0.01	0.14	8
cap104	25	50	12	7	0.01	0.11	4
cap131	50	50	34	196	0.01	0.31	15
cap132	50	50	27	183	0.02	0.28	11
cap133	50	50	25	71	<0.01	0.29	8
cap134	50	50	19	25	<0.01	0.15	4

⋆ Instance solved by preprocessing only.
† Cray-X-MP/28 s.

of free sites appreciably in the other instances. Once again the use of $\alpha > 0$ cannot be justified, considering the execution times of the DCA.

Table 9. Description of the fixed costs for instances in Körkel (1989)

Problem set	No. of instances	Fixed cost for i^{th} instance
Set 1	5	Identical, set at $141 + 6.6i$
Set 2	5	Identical, set at $174 + 6.6i$
Set 3	5	Identical, set at $207 + 6.6i$
Set 4	5	Identical, set at $174 + 66i$
Set10	5	Identical, set at $7170 + 660i$
Set11	5	Identical, set at $7120.5 + 333.3i$
Set12	5	Identical, set at $8787 + 333.3i$

4.4. KÖRKEL-TYPE INSTANCES WITH 65 SITES

Körkel (1989) described several relatively large Euclidean SPLP instances ($m = n = 100$, and $m = n = 400$) and used a branch and bound algorithm to solve these problems. The bound used in that work is an improvement on a bound based on the dual of the linear programming relaxation of the SPLP due to Erlenkotter (1978) and is extremely effective. The bound due to Erlenkotter (1978) is very effective because, for a large majority of SPLP instances, the optimal solution to the dual of the linear programming relaxation of the SPLP is integral. In this subsection, we use instances that have the same cost structure as the ones in Körkel (1989) but for which $m = n = 65$. Instances of this size were not dealt with in Körkel (1989). We used reduction procedure (b) for the RP, and the Khachaturov-Minoux bound in the DCP.

In Körkel (1989), 120 instances of each problem size are described. These can be divided into 28 sets (the first 18 sets contain five instances each, and the rest contain three instances each). We solved all the 120 instances we generated, and found out that the instances in Sets 1, 2, 3, 4, 10, 11, and 12 are more difficult to solve than others. We therefore used these instances in the experiments in this section. The transportation cost matrix for a Körkel instance of size $n \times n$ is generated by distributing n points in random within a rectangular area of size 700×1300 and calculating the Euclidean distances between them. The fixed cost are computed as in Table 9.

The values of the results that we present for each set is the average of the values obtained for all the instances in that set. Interestingly, the preprocessing rules were found to be totally ineffective for all of these problems. Since the fixed costs are identical for all the sites, the sites are distributed randomly over a region, and the variable cost matrix is symmetric, no site presents a distinct advantage over any other. This prevents our reduction procedure to open or close any site. Table 10

Table 10. Costs of solutions output by the DCA on Körkel-type instances with 65 sites

Problem set	Optimal	Acceptable accuracy* 1%	2%	3%	5%	10%
Set 1	6370.0	6404.8	6450.6	6480.6	6569.2	6781.0
Set 2	6920.6	6952.2	6971.4	7028.4	7123.8	7320.2
Set 3	7707.4	7738.0	7770.2	7797.6	7854.6	8053.8
Set 4	9601.2	9642.4	9680.2	9698.4	9786.6	9932.0
Set10	146691.2	146896.6	146909.6	147543.6	148062.0	151542.2
Set11	168598.4	168858.2	169655.0	170341.6	170597.0	173913.8
Set12	186386.3	186729.7	187112.0	188002.7	188854.2	192528.7

⋆ As a percentage of the optimal cost.

Table 11. Execution times for the DCA on Körkel-type instances with 65 sites

Problem Set	Optimal	Acceptable accuracy* 1%	2%	3%	5%	10%
Set 1	119.078	90.948	70.758	55.494	43.200	20.426
Set 2	290.388	225.108	172.422	145.828	96.240	36.966
Set 3	458.370	339.420	259.022	203.036	150.216	50.378
Set 4	158.386	129.694	109.754	89.666	65.548	30.058
Set10	428.598	370.120	319.804	283.832	230.078	142.090
Set11	542.530	476.350	418.628	408.594	290.338	160.744
Set12	479.092	416.472	370.832	326.572	261.835	149.038

⋆ As a percentage of the optimal cost.

shows the variation in the costs of the solution output by the DCA with changes in α, and Table 11 shows the corresponding decrease in execution times.

The effect of varying the acceptable accuracy α on the cost of the solutions output by the DCA is also presented graphically in Figure 1. We define the *achieved accuracy* β as

$$\beta = \frac{\text{cost of solution output by the DCA} - \text{cost of optimal solution}}{\text{cost of optimal solution}}$$

and the *relative time* τ as

$$\tau = \frac{\text{execution time for the DCA for acceptable accuracy } \alpha}{\text{execution time for the DCA to compute an optimal solution}}$$

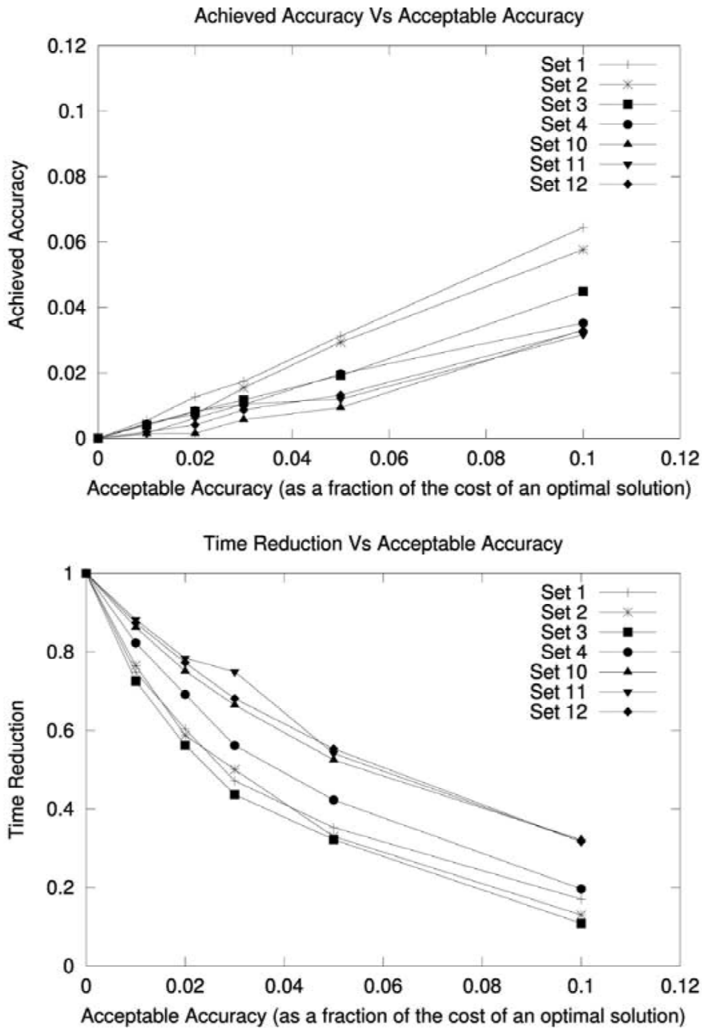

Figure 1. Performance of the DCA for Körkel-type instances with 65 sites.

Note that the achieved accuracy β varies almost linearly with α, with a slope close to 0.5. Also note that the relative time τ of the DCA reduces with increasing α. The reduction is slightly better than linear, with an average slope of -8.

4.5. KÖRKEL INSTANCES WITH 100 SITES

We solved the benchmark instances in Körkel (1989) with $m = n = 100$ to optimality and observed that the instances in Sets 10, 11, and 12 required relatively longer execution times. So we restricted further computations to instances in those sets. The fixed and transportation costs for these problems are computed in the procedure described in Subsection 4.4. Tables 12 and 13 show the results obtained

Table 12. Costs of solutions output by the DCA on Körkel-type instances with 100 sites

Problem Set	Optimal	Acceptable accuracy* 1%	2%	3%	5%	10%
Set10	190782.0	191550.8	192755.4	192080.6	195983.2	203934.2
Set11	219583.4	220438.8	222393.6	221947.2	228467.2	235963.4
Set12	240402.4	241609.6	243336.8	244209.4	247417.6	259168.6

⋆ As a percentage of the optimal cost.

Table 13. Execution times for the DCA on Körkel-type instances with 100 sites

Problem Set	Optimal	Acceptable accuracy* 1%	2%	3%	5%	10%
Set10	133.746	91.774	65.99	65.908	44.2	32.074
Set11	81.564	55.356	39.554	38.348	33.628	17.598
Set12	111.272	85.858	65.608	55.928	61.758	33.014

⋆ As a percentage of the optimal cost.

by running the DCA on these problem instances. In our DCA implementation for solving these instances, we used reduction procedure (c) and the Erlenkotter bound in the DCP.

Figure 2 illustrates the effect of varying the acceptable accuracy α on the cost of the solutions output by the DCA for the instances mentioned above. The nature of the graphs is similar to those in Figure 1. However, in several of the instances we noticed that β reduced when α is increased, and in some other instances τ increased when α was increased.

5. Conclusions

In this paper we tailor the general data correcting algorithm (DCA) for supermodular functions (see Goldengorin et al., 1999) to the simple plant location problem (SPLP). This algorithm consists of two procedures, a reduction procedure to reduce the original instance to a smaller 'core' instance, and a data correcting procedure to solve the core instance.

Theorem 1 can be considered as the basis of data correcting. It states that for two different instances of the SPLP of the same size, the difference between the costs of the *unknown* optimal solutions for these instances is bounded by a polynomially calculated distance between these instances. This distance is used to *correct* one of

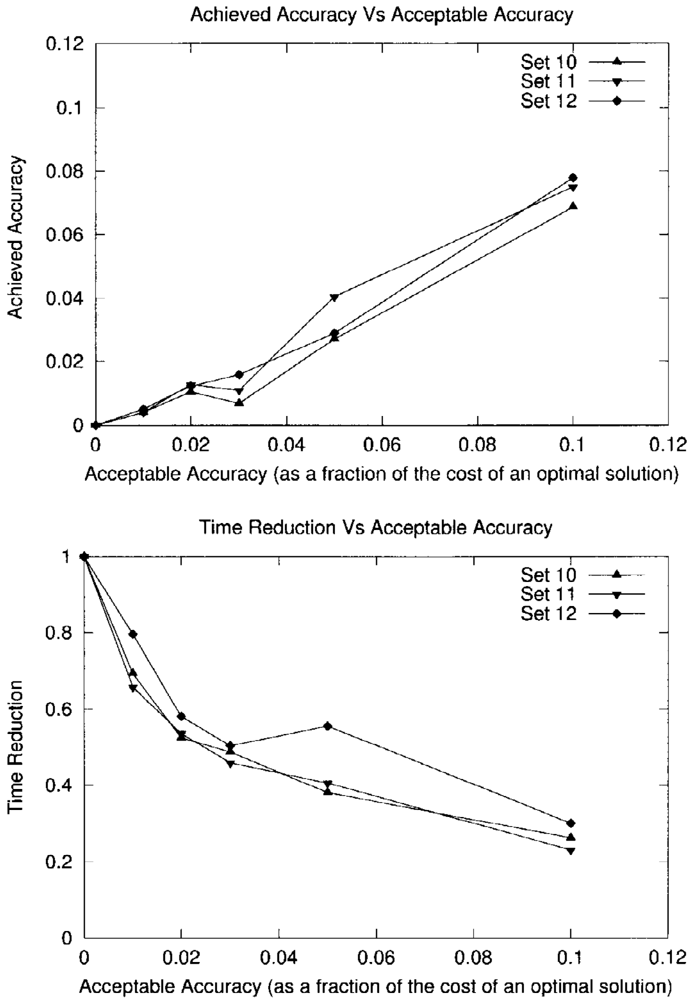

Figure 2. Performance of the DCA for Körkel-type instances with 100 sites.

these instances in an implicit way by just *correcting* the value of the given accuracy parameter in the DCA.

An important contribution of this paper is a new reduction procedure, which when implemented in the DCA yields to a substantial reduction in the size of the original instance. This reduction procedure is much more powerful than the “delta” and “omega” reduction rules in Khumawala (1972). It also incorporates the Erlenkotter bound specific to the SPLP (see Erlenkotter, 1978), which is more computationally efficient than the bound used in Goldengorin et al. (1999). The strength of the new reduction procedure based on the Erlenkotter bound is made obvious by the observation that none of the instances in the OR-Library could be solved by the “delta" and “omega" rules to optimality, but the new reduction procedure solves 75% of them to optimality, and preprocesses at least twice the number of sites as the "delta" and "omega" rules for the remaining 25% of the instances. Another contribution of the paper is the incorporation of the Erlenkotter bound to the recursive branch-and-bound type data correcting procedure.

We have compared the performance of the Erlenkotter bound implemented in an usual branch-and-bound type algorithm (see Bilde and Krarup, 1977) and the Khachaturov-Minoux bound implemented in the DCP for the new reduction rule and for fathoming subproblems created by the DCP. On the instances in Bilde and Krarup (1977), the number of subproblems created by the branch-and-bound type algorithm with Erlenkotter bound is found to be more than 1000 times the number of subproblems created by the DCP based on the Khachaturov-Minoux bound.

We have tested the DCA on a broad range of different classes of instances available in the literature (Bilde and Krarup, 1977; Galvão and Raggi, 1989; OR-Library, Körkel, 1989). The striking computational result is the ability of the DCA to find exact solutions for many relatively large instances within fractions of a second. For example, an exact global optimum of the 200×200 instances from Galvão and Raggi (1989) was found within 0.2 s on a PC with a 733 MHz processor.

In all of our implementations for the DCA with Khachaturov-Minoux and Erlenkotter bounds we have used data structures induced by pseudo–Boolean representations of the SPLP due to Hammer (1968) (see also Beresnev (1973)). These data structures are conducive to efficient updating for the current subproblems in the DCA and sometimes show that a current subproblem remaining after application of the new reduction procedure has relatively small numbers of linear and non-linear terms in the corresponding Hammer function and therefore can be solved by any branch-and-bound type algorithm for the SPLP.

We have found that for all instances in Körkel (1989) the "delta" and "omega" reduction rules were totally ineffective since none of the sites presented any distinct advantage over any other (the fixed costs are almost identical for all sites, the sites are distributed randomly over a region, and the transportation costs matrix is symmetric). Anyway, the DCA has solved to optimality all the instances with $m = n = 100$ within fractions of a second except for the instances in Sets 10, 11 and 12 which required relatively longer execution times. On these sets of instances we have studied the behavior of the execution time and calculated the accuracy for acceptable values of α. When the acceptable value of α increases, we see that the costs of the solutions output by the DCA generally worsen, but the execution times also decrease.

In summary, our computational experience with the DCA on several benchmark instances known in the literature suggest that the algorithm compares well with other algorithms known for the problem. However, like any other branch-and-bound algorithm, DCA depends heavily on the quality of the bounds used. We believe that this algorithm merits serious consideration as a solution tool for the SPLP.

6. Acknowledgements

The authors thank the anonymous referees and Prof. Panos Pardalos for their useful comments which improved the presentation of the results in this paper. The first author gratefully acknowledges the financial support of DIMACS, granted in December 2000.

References

Balas, E. and Padberg, M.W. (1972), On the set covering problem, *Operations Research* 20: 1152–1161.

Beasley, J.E. (1993a), Lagrangian heuristics for location problems, *European Journal of Operational Research* 65: 383–399.

Beasley, J.E. (1993b), OR-Library, http://mscmga.ms.ic.ac.uk/info.html

Bilde, O. and Krarup, J. (1977), Sharp lower bounds and efficient algorithms for the simple plant location problem, *Annals of Discrete Mathematics* 1: 79–97.

Beresnev, V.L. (1973), On a Problem of mathematical Standardization theory, *Upravliajemyje Sistemy* 11: 43–54 (in Russian).

Beresnev, V.L., Gimadi, E.Kh. and Dementyev, V.T. (1978), *Extremal Standardization Problems*, Nauka, Novosibirsk (in Russian).

Cho, D.C., Johnson, E.L., Padberg. M.W. and Rao M.R. (1983a), On the uncapacitated plant location problem, I: Valid inequalities and facets, *Mathematics of Operations Research* 8: 579–589.

Cho, D.C., Padberg, M.W. and Rao, M.R. (1983b), On the Uncapacitated plant location problem, II: facets and lifting theorems, *Mathematics of Operations Research* 8: 590–612.

Christofides, N. (1975), *Graph Theory: An Algorithmic Approach*, Academic Press, London.

Cornuejols, G., Fisher, M.L. and Nemhauser, G.L. (1977a), On the uncapacitated location problem, *Annals of Discrete Mathematics* 1: 163–177.

Cornuejols, G., Fisher, M.L. and Nemhauser, G.L. (1977b), Location of bank accounts to optimize float: An analytic study of exact and approximate algorithms, *Management Science* 23: 789–810.

Cornuejols, G. and Thizy, J.M. (1982), A primal approach to the simple plant location problem, *SIAM Journal on Algebraic and Discrete Methods* 3: 504–510.

Cornuejols, G., Nemhauser, G.L. and Wolsey, L.A. (1990), The uncapacitated facility location problem. In: Mirchandani, P.B. and Francis, R.L. (Eds.) *Discrete Location Theory*, Wiley-Interscience, New York, pp. 119–171.

Erlenkotter, D. (1978), A dual-based procedure for uncapacitated facility location, *Operations Research* 26: 992–1009.

Galvão, R.D. and Raggi, L.A. (1989), A method for solving to optimality uncapacitated location problems, *Annals of Operations Research* 18: 225–244.

Garfinkel, R.S., Neebe, A.W. and Rao, M.R. (1974), An algorithm for the M-median plant location problem, *Transportation Science* 8, 217–236.

Goldengorin, B. (1983), A correcting algorithm for solving some discrete optimization problems, *Soviet Math. Doklady* 27: 620–623.

Goldengorin, B. (1995), *Requirements of Standards: Optimization Models and Algorithms*, ROR, Hoogezand, The Netherlands.

Goldengorin, B., Sierksma, G., Tijssen, G.A. and Tso, M. (1999), The data-correcting algorithm for minimization of supermodular functions, *Management Science* 45: 1539–1551.

Goldengorin, B., Ghosh, D. and Sierksma, G. (2000), Equivalent instances of the simple plant location problem, *SOM Research Report-00A54*, University of Groningen, The Netherlands.

Goldengorin, B., Ghosh, D. and Sierksma, G. (2001), Improving the efficiency of branch and bound algorithms for the simple plant location problem, In Proceedings of the 5th International Workshop on Algorithm Engineering (WAE2001), *Lecture Notes in Computer Science* 2141: 106–117.

Hammer, P.L. (1968), Plant Location — a pseudo–boolean approach, *Israel Journal of Technology* 6: 330–332.

Held, M.P., Wolfe, P. and Crowder, H.P. (1974), Validation of subgradient optimization, *Mathematical Programming* 6: 62–88.

Jones, P.C., Lowe, T.J., Muller, G., Xu, N., Ye, Y. and Zydiak, J.L. (1995), Specially structured uncapacitated facility location problems, *Operations Research* 43: 661–669.

Khachaturov, V.R. (1968), Some problems of the consecutive calculation method and its applications to location problems, Ph.D thesis, Central Economics & Mathematics Institute, Russian Academy of Sciences, Moscow (in Russian).

Khumawala, B.M. (1972), An efficient branch and bound algorithm for the warehouse location problem, *Management Science* 18: B718–B731.

Körkel, M. (1989), On the exact solution of large-scale simple plant location problems. *European Journal of Operational Research* 39: 157–173.

Krarup, J. and Pruzan, P.M. (1983), The simple plant location problem: A survey and synthesis, *European Journal of Operational Research* 12: 36–81.

Labbé, M. and Louveaux, F.V. (1997), Location problems. In: Dell'Amico, M. Maffioli, F. and Martello, S. (eds.), *Annotated Bibliographies in Combinatorial Optimization*, John Wiley & Sons, New York pp. 264–271.

Minoux, M. (1977), Accelerated greedy algorithms for maximizing submodular set functions. In: Stoer J. (ed.), Actes Congres IFIP, Springer, Berlin, pp. 234–243.

Morris, J.G. (1978), On the extent to which certain fixed charge depot location problems can be solved by LP, *Journal of the Operational Research Society* 29: 71–76.

Mukendi, C. (1975), *Sur l'Implantation d'Équipement dans un Réscau: Le Problème de m-Centre,* PhD Thesis, University of Grenoble, France.

Pentico, D.W. (1976), The assortment problem with nonlinear cost functions, *Operations Research* 24: 1129–1142.

Pentico, D.W. (1988), The discrete two-dimensional assortment problem, *Operations Research* 36: 324–332.

Revelle, C.S. and Laporte, G. (1996), The plant location problem: New models and research prospects, *Operations Research* 44: 864–874.

Schrage, L. (1975), Implicit representation of variable upper bounds in linear programming, *Mathematical Programming Study* 4: 118–132.

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

Trubin, V.A. (1969), On a method of solution of integer programming problems of a special kind, *Soviet Math. Doklady* 10: 1544–1546.