Exact Procedures for Solving the Discrete Ordered Median Problem

Natashia Boland [*] Patricia Domínguez-Marín [†] Stefan Nickel [†]
Justo Puerto [‡]

January 15, 2003

Abstract

The Discrete Ordered Median Problem (DOMP) generalizes classical discrete location problems, such as the N-median, N-center and Uncapacitated Facility Location problems. It was introduced by Nickel [16], who formulated it as both a nonlinear and a linear integer program. We propose an alternative integer linear programming formulation for the DOMP, discuss relationships between both integer linear programming formulations, and show how properties of optimal solutions can be used to strengthen these formulations. Moreover, we present a specific branch and bound procedure to solve the DOMP more efficiently. We test the integer linear programming formulations and this branch and bound method computationally on randomly generated test problems.

Keywords: Discrete Location, Integer Programming.

1 Introduction

Discrete location problems have been widely studied, not least because of their importance in practical applications. A number of survey articles and textbooks have been written on these problems, see, for example, [2, 4, 14], and references therein. Discrete location problems typically involve a finite set of *sites* at which facilities can be located, and a finite set of *clients*, which demand requests to be supplied from facilities. Even for the simplest of such problems, such as the Capacitated or Uncapacitated Facility Location

[*]Dept. of Mathematics & Statistics. University of Melbourne. Parkville VIC 3010, Australia, e-mail:natashia@unimelb.edu.au

[†]Dept. of Optimization. Fraunhofer Institut für Techno- und Wirtschaftsmathematik. D-67663 Kaiserslautern, Germany, e-mail:{dominguez}{nickel}@itwm.fhg.de

[‡]Dept. Estadística e I.O. Facultad de Matemáticas, Universidad de Sevilla. C/ Tarfia s/n, 41012 Sevilla, Spain, e-mail:puerto@us.es

Problem, which are relatively tractable, new methods and results continue to emerge (see for example [1, 15]) in an effort to solve larger problems, faster. Variations, such as hub location, are much more challenging, and it is still difficult to solve even modest sized problems to optimality (see for example [5, 6, 13]). Whilst many problem variations have been considered in the literature, we will focus on problems in which a fixed number of facilities must be located at sites chosen from among the given set of candidate sites, and in which a given client can only be supplied from a single facility. For each client-site pair, there is a given cost for meeting the demand of the client from a facility located at the site.

An interesting feature of discrete location problems is the variety of objective functions that have been considered. The *median* objective is to minimize the sum of the costs of supplying all clients from the facilities at the selected sites. The *center* objective is to minimize the maximum cost of supplying a client, from amongst the sites chosen, over all clients. The *centdian* objective is a convex combination of the median and center objectives; it aims to keep both the total cost and largest cost low. These are the three objectives most frequently encountered in the literature. It is worth noting that every problem has its own solution method, including its own algorithmic approach.

The increasing need for discrete location models in strategic supply chain planning, see for example [9], has made it necessary to develop new and flexible location models. To that end, [16] introduced a new type of objective function which generalized the most popular objective functions mentioned above. This objective function applies a penalty to the cost of supplying a client which is dependent on the *position* of that cost relative to the costs of supplying other customers. For example, a different penalty might be applied if the cost of supplying the client was the 5th-most expensive such cost rather than the 2nd-most expensive. This adds a "sorting" problem to the underlying facility location problem, making formulation and solution much more challenging.

For planar and network location problems the generalized model was studied in [7], [17], [18], [19], and [20]. In [16], a formulation of the discrete case, called the Discrete Ordered Median Problem (DOMP), is discussed. A nonlinear integer programming formulation is developed, and a linearization with number of variables and constraints proportional to the number of sites, cubed, is proposed. However, no computation is attempted in [16], and there is no attempt to determine how effective integer programming approaches can be in solving the DOMP.

In this paper we develop two different integer linear programming (ILP) formulations for this new location model, DOMP. Both have $O(M^2)$ constraints, but one uses $O(M^3)$ variables, while the other uses only $O(M^2)$ variables, where M is the number of clients (and sites). For hub location problems, it was found in [5] that using properties of optimal solutions to eliminate variables and add constraints to the formulation could strengthen it significantly. Here we are able to strengthen both formulations using properties of optimal solutions. In order to compare the two formulations we generate test problems from eight representative classes. These eight classes attempt to represent the variety of objective functions possible for DOMP, for example, one of the classes uses a median objective while another class uses a trimmed mean objective. By testing the performance of the two ILP

formulations on problems from these eight classes, we can get some measure of the degree of difficulty of the different classes and can compare the ILP formulations' performance on each class. Finally, we develop a branch and bound (B&B) algorithm designed using the special structure of our model, using relatively simple lower bounds rather than lower bounds determined by solving an LP. We test the B&B algorithm on a large battery of problems, comparing its performance with that of the integer linear programming approach using a standard solver.

Of the three approaches investigated, we would expect the $O(M^3)$ ILP formulation would yield the strongest lower bounds, followed by the $O(M^2)$ ILP formulation, with the easily calculated lower bounds used in the branch and bound procedure weakest. (In fact this was not uniformly the case, as we report in detail in Section 5.) However the order is reversed if we base it on least computational effort required to calculate the bound: the branch and bound method requires only a very easy calculation, followed by the $O(M^2)$ ILP formulation, with the $O(M^3)$ formulation requiring most effort. The three methods thus explore the trade-off between the quality of bound and computational effort required at each node of the branch and bound tree. In the context of hub location, it was found in [5] that an intermediate position gave the best trade-off, but here that is not the case, as we shall discuss in detail in Sections 5 and 7.

The remainder of the paper is organized as follows. In the next section, we present a formal definition of the DOMP and review the nonlinear formulation of [16]. In Section 3 we present the integer linear formulation already introduced in [16]. Moreover, we introduce an alternative integer linear programming formulation for the DOMP. We go on in Section 4 to discuss properties of optimal solutions, which allow us to strengthen the formulations. In Section 5 we describe our random problem generator, and test problem sets, and present the results of computational experiments with all integer linear programming formulations. A branch and bound method is introduced in Section 6 and numerical results comparing two branching schemes are given. Further computational results for this method and comparison of its performance with that of the integer programming linearizations are given in Section 7. Finally, in Section 8, we summarize our conclusions and give suggestions for further research.

2 Problem Definition and Review of Previous Work

Problem Definition

Let A denote the given set of M sites, and identify these with the integers $1, \ldots, M$, so $A = \{1, \ldots, M\}$. Let $C = (c_{ij})_{i,j=1,\ldots,M}$ be the given non-negative $M \times M$ cost matrix, where c_{ij} denotes the cost of satisfying the demand of client i from a facility located at site j. (As is customary for location problems, we assume without loss of generality that the set of candidate sites is identical to the set of clients.) Let $N \leq M$ be the number of facilities to be located, at N of the M candidate sites. A solution to the location problem is given by a set of N sites; we use $X \subseteq A$ with $|X| = N$ to denote a solution. We assume

that each client will be served by a facility located at a site which yields the cheapest cost of satisfying demand, i.e. given a solution X, we assume that each client i will be supplied from a site $j \in X$ such that

$$c_{ij} = c_i(X) := \min_{k \in X} c_{ik}. \tag{1}$$

What distinguishes the DOMP from a usual single-supplier uncapacitated facility location problem is its objective. This objective applies a linear cost, with coefficient $\lambda_i \geq 0$, to the ith cheapest cost of supplying a client, for each $i = 1, \ldots, M$. So in order to calculate the objective, the costs of supplying clients, $c_1(X), \ldots, c_M(X)$, must be ordered. We define σ_X to be a permutation on $\{1, \ldots, M\}$ for which the inequalities

$$c_{\sigma_X(1)}(X) \leq c_{\sigma_X(2)}(X) \leq \cdots \leq c_{\sigma_X(M)}(X) \tag{2}$$

hold. We call any such permutation a *valid permutation for* X. For short we will denote $c_{\leq}(X) = (c_{\sigma_X(1)}(X), \ldots, c_{\sigma_X(M)}(X))$. Let $\lambda = (\lambda_1, \ldots, \lambda_M)$ with $\lambda_i \geq 0$, $i = 1, \ldots, M$. The Discrete Ordered Median Problem (DOMP) is defined as

$$\min_{X \subseteq A\ ,\ |X|=N} \sum_{i=1}^{M} \lambda_i c_{\sigma_X(i)}(X)\ .$$

Note that the linear representation of the DOMP is only pointwise defined, since $c_{\sigma_X(i)}(X)$ depends on X.

For different choices of λ we obtain different types of objective functions. To see that the DOMP objective generalizes well known location objectives, note that taking $\lambda = (1, 1, \ldots, 1)$ makes the DOMP equivalent to the N-median problem; taking $\lambda = (0, 0, \ldots, 0, 1)$ makes the DOMP equivalent to the N-center problem; taking $\lambda = (\mu, \mu, \ldots, \mu, 1)$ for $0 < \mu < 1$ leads to the μ-centdian problem, which is a convex combination of the median and the center objective functions; and taking $\lambda = (0, \ldots, 0, 1, \ldots, 1)$, where the first $M - k$ entries are zero and the last k entries are one, leads to the k-centra problem of minimizing the average cost of the k most expensive clients. Other objectives may also be of practical interest. One example is to take $\lambda = (0, \ldots, 0, 1, \ldots, 1, 0, \ldots, 0)$, where the first k_1 and last k_2 entries are zero, and the middle $M - k_1 - k_2$ entries are one: this leads to a problem in which the k_1 smallest costs and the k_2 largest costs are disregarded and the average of the middle part, the so-called $(k_1 + k_2)$-trimmed mean, which is a robust statistic, is minimized. Another example would be to take $\lambda = (1, \ldots, 1, 0, \ldots, 0, 1, \ldots, 1)$, where the first k_1 entries are one, the next $M - k_1 - k_2$ entries are zero, and the last k_2 entries are one: this leads to the problem of minimizing the sum of the k_1 smallest costs and the k_2 largest costs; the corresponding DOMP searches for a set of N facilities minimizing the average cost for the clients which are very close and very far away. A final example would be to take $\lambda = (2, 0, \ldots, 0, 1)$: this leads to the problem of minimizing the sum of the largest cost and the smallest cost (counted twice), with all other costs ignored. Clearly, classical location problems can easily be modelled. Moreover, new meaningful objective functions are easily derived, as shown above. An example presented by [16] shows the great

impact that the choice of the objective function has on the optimal location of the new facilities.

Observe that the DOMP belongs to the class of NP-hard problems, since it is a generalization of the N-median problem, which is NP-hard (see [10]).

An Illustrative Example

Consider the DOMP with $M = 5$, $N = 2$, and the cost matrix below.

$$C = \begin{pmatrix} 0 & 4 & 5 & 3 & 3 \\ 5 & 0 & 6 & 2 & 2 \\ 7 & 3 & 0 & 5 & 1 \\ 7 & 3 & 3 & 0 & 5 \\ 1 & 3 & 2 & 4 & 0 \end{pmatrix}$$

We show how the objective function value for the case $\lambda = (0, 0, 1, 1, 0)$, which yields the (2+1)-trimmed mean problem, is calculated. The optimal solution has facilities located at sites 1 and 4, with the demand of clients 1 and 5 satisfied by facility 1 and the demand of the remaining clients satisfied by facility 4. The associated cost vector is $(0, 2, 5, 0, 1)$. The sorted cost vector is $(0, 0, 1, 2, 5)$ and the optimal objective function value is thus $\langle \lambda, (0, 0, 1, 2, 5) \rangle = 0 \times 0 + 0 \times 0 + 1 \times 1 + 1 \times 2 + 0 \times 5 = 3$.

Formulation of the Problem

We now review the integer programming formulation for this problem given in [16]. This nonlinear model is made up of two components. The first has variables and constraints which correspond to the classical N-median location problem. We use the standard variables

$$y_{ij} = \begin{cases} 1, & \text{if the demand of client } i \text{ is satisfied by a facility at site } j \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

for $i, j = 1, \ldots, M$ and

$$x_j = \begin{cases} 1, & \text{if a facility is located at site } j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

for $j = 1, \ldots, M$, and write

$$\begin{aligned} \mathcal{N} = \{(x, y) \in B^M \times \mathbb{R}^{M^2} : \quad & \sum_{j=1}^{M} x_j = N, \\ & \sum_{j=1}^{M} y_{ij} = 1, \quad \forall i = 1, \ldots, M \text{ and} \\ & y_{ij} \leq x_j, \quad \forall i, j = 1, \ldots, M\}, \end{aligned}$$

where $B = \{0, 1\}$. The second component is used to sort the clients, in order of increasing cost. Variables

$$s_{ij} = \begin{cases} 1, & \text{if the cost of supplying client } j \text{ is the } i\text{th cheapest such cost} \\ 0, & \text{otherwise,} \end{cases} \tag{5}$$

for each $i, j = 1, \ldots, M$, are used to indicate the ordering. To be a permutation, these variables must satisfy assignment constraints. We let

$$\begin{aligned} \mathcal{P} = \{s \in B^{M^2} \; : \; & \sum_{i=1}^{M} s_{ij} = 1 \quad \forall j = 1, \ldots, M, \text{ and} \\ & \sum_{j=1}^{M} s_{ij} = 1 \quad \forall i = 1, \ldots, M\} \end{aligned}$$

denote the set of all permutations. Of course, the permutation must be made to order the clients in increasing cost of supply, and the DOMP objective must weight the cost of this supply appropriately. This is accomplished via a nonlinear constraint and nonlinear objective function, as follows.

$$\begin{aligned} & (DOMP) \\ \min_{s \in \mathcal{P}, (x,y) \in \mathcal{N}} \quad & \sum_{i=1}^{M} \sum_{k=1}^{M} s_{ik} \left(\sum_{j=1}^{M} y_{kj} c_{kj} \right) \lambda_i \\ \text{s.t.} \quad & \sum_{k=1}^{M} s_{ik} \left(\sum_{j=1}^{M} y_{kj} c_{kj} \right) \leq \sum_{k=1}^{M} s_{i+1,k} \left(\sum_{j=1}^{M} y_{kj} c_{kj} \right) \quad \forall i = 1, \ldots, M-1 \end{aligned} \tag{6}$$

3 Linearizations

First we recall the linearization given in [16] which is analogous to that of [12] for the quadratic assignment problem. We use binary variables

$$z_{ikj} = s_{ik} y_{kj} \quad i, j, k = 1, \ldots, M, \tag{7}$$

so $z_{ikj} = 1$ if client k is supplied by a facility located at site j and is the ith cheapest client supplied; $z_{ikj} = 0$ otherwise. The linearization of the DOMP is thus as follows.

$$\min_{s\in\mathcal{P},(x,y)\in\mathcal{N}} \quad \sum_{i=1}^{M}\sum_{k=1}^{M}\sum_{j=1}^{M}\lambda_i c_{kj} z_{ikj}$$

$$\text{s.t.} \quad \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z_{ikj} \le \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z_{i+1,kj} \qquad \forall i = 1,\dots,M-1 \qquad (8)$$

$$\sum_{i=1}^{M} z_{ikj} = y_{kj} \qquad \forall k,j = 1,\dots,M \qquad (9)$$

$$\sum_{j=1}^{M} z_{ikj} = s_{ik} \qquad \forall i,k = 1,\dots,M \qquad (10)$$

$$\sum_{k=1}^{M}\sum_{i=1}^{M}\sum_{j=1}^{M} z_{ikj} = M \qquad (11)$$

$$s_{ik} + y_{kj} - 2z_{ikj} \ge 0 \qquad \forall i,j,k = 1,\dots,M \qquad (12)$$

$$z_{ikj} \ge 0 \qquad \forall i,j,k = 1,\dots,M$$

This linearization can be directly improved as follows: constraints (9) and (10) can be used to eliminate the s and y variables respectively, resulting in an equivalent formulation using only x and z variables. In the resulting formulation, the assignment constraints on s, once s is substituted out using (10), imply the sum constraint (11), so this is omitted. The formulation in x and z variables alone is thus as follows.

$(LDOMP_1)$

$$\min_{x\in\{0,1\}^M,\ z\in\{0,1\}^{M^3}} \sum_{i=1}^{M}\sum_{k=1}^{M}\sum_{j=1}^{M}\lambda_i c_{kj} z_{ikj}$$

s.t.

$$\sum_{i=1}^{M}\sum_{j=1}^{M} z_{ikj} = 1 \qquad \forall k = 1,\ldots,M \tag{13}$$

$$\sum_{k=1}^{M}\sum_{j=1}^{M} z_{ikj} = 1 \qquad \forall i = 1,\ldots,M \tag{14}$$

$$\sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z_{ikj} \leq \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z_{i+1,kj} \qquad \forall i = 1,\ldots,M-1 \tag{15}$$

$$\sum_{j=1}^{M} x_j = N \tag{16}$$

$$x_j \geq \sum_{i=1}^{M} z_{ikj} \qquad \forall k, j = 1,\ldots,M \tag{17}$$

It is not difficult to show that (LDOMP$_1$) is a valid integer linear programming model of the DOMP. Note that (LDOMP$_1$) has $O(M^3)$ variables and $O(M^2)$ constraints.

We develop an alternative linearization for (DOMP), which use $O(M^2)$ variables rather than the $O(M^3)$ variables of (LDOMP$_1$). Our linearization is inspired by the mixed integer linear programming formulation of [11], for the quadratic assignment problem. Variables are introduced which take on values of *costs* of supply; in a sense they absorb the objective function.

We define M^2 new real variables w'_{ik} by

$$w'_{ik} = s_{ik}\sum_{j=1}^{M} c_{kj} y_{kj} \quad i,k = 1,\ldots,M,$$

so w'_{ik} is the cost of supplying client k if k is the ith client in order of cost of supply, and zero otherwise. However, it is not necessary to include variables to record the cost of supplying each client k; it suffices to record the ith smallest cost of supply alone. Hence we may replace the variables w'_{ik} with variables w_i defined by

$$w_i = \sum_{k=1}^{M} w'_{ik} = \sum_{k=1}^{M} s_{ik}\sum_{j=1}^{M} c_{kj} y_{kj}$$

for each $i = 1,\ldots,M$, so w_i represents the cost of supplying the client with the ith smallest cost of supply. Using these variables, we get the following linearization of the DOMP, where $\overline{c}_k$ is the Nth largest entry of the cost matrix at row k, i.e. $\overline{c}_k$ is the Nth largest element of the set $\{c_{kj} \mid j = 1,\ldots,M\}$.

$(LDOMP_2)$

$$\min_{s\in\mathcal{P},(x,y)\in\mathcal{N},w\in\mathbb{R}^M_+} \sum_{i=1}^{M} \lambda_i w_i$$

$s.t.$

$$w_i \leq w_{i+1} \qquad \forall i = 1,\ldots,M-1 \tag{18}$$

$$\sum_{i=1}^{M} w_i = \sum_{j=1}^{M}\sum_{k=1}^{M} c_{kj}y_{kj} \tag{19}$$

$$w_i \geq \sum_{j=1}^{M} c_{kj}y_{kj} - \bar{c}_k(1-s_{ik}) \quad \forall i,k = 1,\ldots,M \tag{20}$$

It is not hard to show that (LDOMP_2) is a valid formulation of the DOMP.

4 Strengthening the Formulations

In this section we will present some results to improve the linearizations. These improvements consist of additional constraints, strengthened forms of original constraints, or preprocessing steps, such as fixing some variables to zero, or relaxing integrality requirements on some variables. These help to reduce the computing time required to solve the DOMP, either by reducing the gap between the optimal objective function value and the relaxed LP solution, or by reducing the number of variables for which integrality must be ensured. For some of these properties, we will assume free self-service, i.e. we assume $c_{ii} = 0$, $\forall i = 1,\ldots,M$. For short we will denote this property by (FSS).

Our first strengthening idea makes use of an existing upper bound z_{UB} on the value of the DOMP, and has some resemblance to standard reduced cost variable fixing ideas. Upper bounds are, of course, easy to come by; any set of N locations yields one. *Good* upper bounds might be more difficult to obtain, but for our purposes here, we simply assume that some upper bound is available. Consider a client k. Now either the cost of supplying k is going to be among the largest $M-m$ such costs, or the cost of supplying k is going to be among the smallest m costs, in which case the objective value of the problem must be at least $\sum_{i=m}^{M} \lambda_i$ multiplied by the cost of supplying client k; this value is clearly a lower bound. Obviously we are only interested in solutions to the problem with objective value less than or equal to z_{UB}, so we are only interested in solutions with either client k's cost ranked $m+1$ or higher, or with k supplied from some location j with $c_{kj}\sum_{i=m}^{M}\lambda_i \leq z_{\text{UB}}$, i.e. with $c_{kj} \leq z_{\text{UB}}/\sum_{i=m}^{M}\lambda_i$. We formalize this in the proposition below, in which we make use of the following notation for the total cost of some set of sites X:

$$L(X) = \langle \lambda, c_{\leq}(X)\rangle = \sum_{i=1}^{M} \lambda_i c_{\sigma_X(i)}(X)$$

where σ_X is a valid permutation of X. We also write $\sigma_X^{-1}(k)$ to denote the rank of client k in the cost ordering, so if $\sigma_X(i) = k$ for some $i, k \in \{1, \ldots, M\}$ then $\sigma_X^{-1}(k) = i$. Note that we adopt the convention that if $\sum_{i=m}^{M} \lambda_i = 0$ then $c_{kj} \leq z_{\mathrm{UB}} / \sum_{i=m}^{M} \lambda_i = \infty$ for all k and j.

Proposition 1 *Given a value z_{UB}, any set X of N locations having ordered cost $L(X) \leq z_{UB}$ must have $\sigma_X^{-1}(k) \geq m+1$ for all clients $k = 1, \ldots, M$ and all ranks $m = 1, \ldots, M-1$ for which $c_k(X) > z_{UB} / \sum_{i=m}^{M} \lambda_i$.*

Proof.
Suppose there exist $k \in \{1, \ldots, M\}$ and $m \in \{1, \ldots, M-1\}$ such that $c_k(X) > z_{\mathrm{UB}} / \sum_{i=m}^{M} \lambda_i$ and $\sigma_X^{-1}(k) \leq m$. Then there exists $r \leq m$ such that $\sigma_X^{-1}(k) = r$, i.e. $k = \sigma_X(r)$. Now by the definition of σ_X,

$$c_{\sigma_X(M)}(X) \geq \cdots \geq c_{\sigma_X(m)}(X) \geq c_{\sigma_X(r)}(X) = c_k(X).$$

Since C and λ are both non-negative, we thus have

$$L(X) = \sum_{i=1}^{M} \lambda_i c_{\sigma_X(i)}(X) \geq \sum_{i=m}^{M} \lambda_i c_{\sigma_X(i)}(X) \geq \sum_{i=m}^{M} \lambda_i c_k(X) = c_k(X) \left(\sum_{i=m}^{M} \lambda_i \right) > z_{\mathrm{UB}}.$$

We have proved the contrapositive of the lemma; the result follows. □

In the computational results presented in Section 5 this upper bound (z_{UB}) is computed by a heuristic method based on variable neighbourhood search, see [3].

We state without proof the following proposition, which will be of use later.

Proposition 2 *If (FSS) holds then each open facility can satisfy its demand by itself, at a cost of zero. Thus for any set X of N facilities, we have $c_i(X) = 0$ for all $i \in X$. Furthermore, there exists σ_X, a valid permutation for X, i.e. satisfying (2), that also satisfies*

$$\{\sigma_X(i) \ : \ i = 1, \ldots, N\} = X. \tag{21}$$

In other words, there exists σ_X a valid permutation for X such that the first N elements are the open facilities. Consequently

$$0 = c_{\sigma_X(1)}(X) = \cdots = c_{\sigma_X(N)}(X) \leq c_{\sigma_X(N+1)}(X) \cdots \leq c_{\sigma_X(M)}(X). \tag{22}$$

Notice that the proof is straightforward since we have assumed that costs are non-negative, i.e. $c_{ij} \geq 0$ for all i, j, so $c_i(X) \geq 0$ for all $i \in A$.

After stating these general properties we will show in the following subsections improvements for the different linearizations introduced.

Improvements for (LDOMP$_1$)

In the following we will prove some properties which are fulfilled by optimal solutions of (LDOMP$_1$) and therefore, allow us to derive better formulations.

Lemma 1 *Any optimal solution (x, z) for (LDOMP$_1$) with objective value less than or equal to z_{UB} must satisfy*

$$z_{mkj} = 0 \quad \forall m, k, j \text{ such that } c_{kj} > \frac{z_{UB}}{\sum_{i=m}^{M} \lambda_i}. \tag{23}$$

Proof.
Suppose (x, z) is an optimal solution of $(LDOMP_1)$. Assume further that it has an objective value less than or equal to z_{UB}; and there exist m, j', k' such that $c_{k'j'} > z_{\text{UB}}/\sum_{i=m}^{M} \lambda_i$ and $z_{mk'j'} = 1$. From the definition of $z_{mk'j'}$, see (7), it follows that site j' is in the solution set of sites X induced by (x, z), and that client k' is ranked at position m (i.e. $\sigma_X(m) = k'$). Furthermore, client k' is allocated to the site j', thus $c_{k'}(X) = c_{k'j'} > z_{\text{UB}}/\sum_{i=m}^{M} \lambda_i$. Finally, $L(X)$ is the objective value of (x, z) in (LDOMP$_1$), and so $L(X) \leq z_{\text{UB}}$. Hence by Proposition 1, it must be that $\sigma_X^{-1}(k') \geq m+1$. But $\sigma_X^{-1}(k') = m$, so this is a contradiction, and the result is proved. □

Lemma 2 *If (FSS) holds, then there exists an optimal solution (x, z) of (LDOMP$_1$) satisfying*

$$\sum_{i=1}^{N} z_{ikk} = x_k, \qquad \forall k = 1, \ldots, M. \tag{24}$$

Proof.
Let (x, z') be an optimal solution of (LDOMP$_1$). Let X be the set of facilities induced by x, i.e. $X = \{j \in A : x_j = 1\}$. We have $|X| = N$ by (16). Let σ_X be a valid permutation for X, so σ_X satisfies (2). By Proposition 2, we may assume σ_X also satisfies (21), and so $\sigma_X^{-1}(k) \in \{1, \ldots, N\}$ for each $k \in X$. For each $k \in A \setminus X$, choose $j(k)$ a minimizer of $\min_{j \in X} c_{kj}$. Observe that for each $k \in X$, by (FSS), k itself is a minimizer of $\min_{j \in X} c_{kj}$: set $j(k) = k$ for all $k \in X$. Define z by $z_{ikj} = 1$ if $j = j(k)$ and $k = \sigma_X(i)$, and $z_{ikj} = 0$ otherwise, for each $i, k, j \in A$. By the validity of (LDOMP$_1$), it must be that (x, z) is an optimal solution of (LDOMP$_1$). Now we consider two cases for (24).
Case 1: $k \in X$ In this case $j(k) = k$ and $\sigma_X^{-1}(k) \in \{1, \ldots, N\}$, so $z_{\sigma_X^{-1}(k)kk} = 1$ and $z_{ikk} = 0$ for all $i \in \{1, \ldots, N\}$, $i \neq \sigma_X^{-1}(k)$. Thus $\sum_{i=1}^{N} z_{ikk} = z_{\sigma_X^{-1}(k)kk} = 1 = x_k$ as required.
Case 2: $k \notin X$ In this case $j(k) \neq k$, since $j(k) \in X$ by definition, so $z_{ikk} = 0$ for all i. Thus $\sum_{i=1}^{N} z_{ikk} = 0 = x_k$ as required.
In either case, (24) is satisfied by (x, z), an optimal solution to (LDOMP$_1$). □

Corollary 1 *If (FSS) holds, then there exists an optimal solution to (LDOMP$_1$) satisfying constraints* (23) *and* (24)*.*

Proof.
By Lemma 2 there exists an optimal solution (x, z) satisfying (24). By Lemma 1, every optimal solution satisfies (23), so (x, z) does. Thus (x, z) exists satisfying both (23) and (24). □

Proposition 3 *Suppose (FSS) holds, and let (x, z) solve (LDOMP$_1$) with integrality relaxed on all z_{ikj} variables with $i \in \{1, \ldots, N\}$ and $j, k \in \{1, \ldots, M\}$, and with the additional constraints* (23) *and* (24)*. Then x induces an optimal solution to the DOMP.*

Proof.
We will proceed by showing that there exists z' such that (x, z') is an optimal solution to (LDOMP$_1$). By the validity of (LDOMP$_1$), we conclude that x must induce an optimal solution to the DOMP.

We firstly let X be the set of facilities induced by x and let σ_X be a valid permutation for X satisfying (21). (Such a permutation exists by Proposition 2.) By (21) it must be that

$$x_{\sigma_X(1)} = x_{\sigma_X(2)} = \cdots = x_{\sigma_X(N)} = 1,$$

i.e. $\sigma_X^{-1}(j) \in \{1, \ldots, N\}$ for all $j \in X$, and $x_j = 0$ for all $j \in \{1, \ldots, M\}$ with $\sigma_X^{-1}(j) > N$. We now define z' by

$$z'_{ikj} = \begin{cases} z_{ikj}, & i > N \\ 1, & i \leq N, k = j = \sigma_X(i) \\ 0, & \text{otherwise} \end{cases}$$

for each $i, k, j \in \{1, \ldots, M\}$.

We show that z' is feasible for (LDOMP$_1$), as follows.

To show z' satisfies (13), we consider two cases.

Case 1: $k \in X$ Since z satisfies (24) and in this case $x_k = 1$, it must be that $\sum_{i=1}^{N} z_{ikk} = 1$. Furthermore, z satisfies (13), and so it must be that $\sum_{i=N+1}^{M} \sum_{j=1}^{M} z_{ikj} = 0$. Hence, and from the definition of z', we have

$$\begin{aligned} \sum_{i=1}^{M} \sum_{j=1}^{M} z'_{ikj} &= \sum_{i=1}^{N} \sum_{j=1}^{M} z'_{ikj} + \sum_{i=N+1}^{M} \sum_{j=1}^{M} z_{ikj} \\ &= z'_{\sigma_X^{-1}(k)kk} + 0 \\ &= 1 \end{aligned}$$

as required.

Case 2: $k \notin X$ We will begin by showing that $\sum_{j \in X} z_{ijj} = 1$ for all $i \in \{1, \ldots, N\}$. Firstly,

we have from (24) that $\sum_{i=1}^{N} z_{ijj} = 1$ for all $j \in X$. Secondly, by (14), we know that $\sum_{j\in X} z_{ijj} \leq 1$ for all $i \in \{1,\ldots,N\}$. Now suppose there exists $i \in \{1,\ldots,N\}$ with $\sum_{j\in X} z_{ijj} < 1$. Then

$$\sum_{i=1}^{N}\sum_{j\in X} z_{ijj} < N = \sum_{j\in X}\sum_{i=1}^{N} z_{ijj}$$

since $|X| = N$. This is obviously a contradiction, so it must be that $\sum_{j\in X} z_{ijj} = 1$ for all $i \in \{1,\ldots,N\}$. Therefore, by (14) again, it must be that $z_{ik'j} = 0$ for all $i \in \{1,\ldots,N\}$, and all $j, k' \in A$ with $k' \neq j$ or $j \notin X$. Thus by (13), and since $k \notin X$, it must be that

$$\sum_{i=N+1}^{M}\sum_{j=1}^{M} z_{ikj} = 1.$$

To conclude, we observe that by the definition of z', and since $k \notin X$, we have that $z'_{ikj} = 0$ for all $i \in \{1,\ldots,N\}$ and all $j \in A$, so

$$\begin{aligned}\sum_{i=1}^{M}\sum_{j=1}^{M} z'_{ikj} &= \sum_{i=1}^{N}\sum_{j=1}^{M} z'_{ikj} + \sum_{i=N+1}^{M}\sum_{j=1}^{M} z_{ikj} \\ &= \sum_{i=1}^{N}\sum_{j=1}^{M} 0 + 1 \\ &= 1\end{aligned}$$

as required.

To show that z' satisfies (14) is much easier. For $i > N$, $z'_{ikj} = z_{ikj}$ for all $j, k \in A$, so

$$\sum_{k=1}^{M}\sum_{j=1}^{M} z'_{ikj} = \sum_{k=1}^{M}\sum_{j=1}^{M} z_{ikj} = 1$$

since z satisfies (14). For $i \leq N$, $z'_{ikj} = 1$ if $k = j = \sigma_X(i)$, and $z'_{ikj} = 0$ otherwise. Therefore (14) must hold.

To show z' satisfies (15) is similar. For all $i > N$, $z'_{ikj} = z_{ikj}$ for all $j, k \in A$, so the results follows since z satisfies (15). For all $i \leq N$,

$$\sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z'_{ikj} = c_{\sigma_X(i)\sigma_X(i)} = 0$$

since (FSS) holds, so (15) holds.

Proving that z' satisfies (17) is similar to showing z' satisfies (13). We have the same two cases.

Case 1: $k \in X$ From earlier arguments, we have in this case that $\sum_{i=N+1}^{M}\sum_{j=1}^{M} z_{ikj} = 0$. Hence

$\sum_{i=N+1}^{M} z_{ikj} = 0$ for all $j \in A$. Now from the definition of z', $z'_{ikj} = 1$ for $i \in \{1, \ldots, N\}$ only if $j = k = \sigma_X(i)$, so we have that if $j \neq k$,

$$\begin{aligned} \sum_{i=1}^{M} z'_{ikj} &= \sum_{i=1}^{N} z'_{ikj} + \sum_{i=N+1}^{M} z_{ikj} \\ &= 0 + 0 = 0 \leq x_j \end{aligned}$$

while if $j = k$, then $x_j = 1$ since $j = k \in X$, and we have

$$\begin{aligned} \sum_{i=1}^{M} z'_{ikj} &= \sum_{i=1}^{N} z'_{ikj} + \sum_{i=N+1}^{M} z_{ikj} \\ &= z'_{\sigma_X^{-1}(k)kk} + 0 \\ &= 1 = x_j \end{aligned}$$

as required.

Case 2: $k \notin X$ In this case, we see from the definition of z' that $z'_{ikj} = 0$ for all $i \in \{1, \ldots, N\}$ and all $j \in A$. Thus

$$\begin{aligned} \sum_{i=1}^{M} z'_{ikj} &= \sum_{i=1}^{N} z'_{ikj} + \sum_{i=N+1}^{M} z_{ikj} \\ &= 0 + \sum_{i=N+1}^{M} z_{ikj} \\ &\leq \sum_{i=1}^{M} z_{ikj} \\ &\leq x_j \end{aligned}$$

since z satisfies (17), as required.

Obviously z' is binary.

Thus z' is feasible for (LDOMP$_1$).

We now show that the objective value of z' in (LDOMP$_1$) is no more than that of z. It is helpful to define that objective, via

$$\zeta_i(z) = \sum_{k=1}^{M} \sum_{j=1}^{M} c_{kj} z_{ikj}$$

for each i, so the objective is

$$\sum_{i=1}^{M} \lambda_i \zeta_i(z).$$

Now for $i > N$, $z'_{ikj} = z_{ikj}$ for all k, j, so

$$\zeta_i(z') = \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z'_{ikj} = \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z_{ikj} = \zeta_i(z).$$

For $i \leq N$, $z'_{ikj} = 1$ only if $j = k = \sigma_X(i)$, and $z'_{ikj} = 0$ otherwise, so

$$\zeta_i(z') = \sum_{k=1}^{M}\sum_{j=1}^{M} c_{kj} z'_{ikj} = c_{\sigma_X(i)\sigma_X(i)} = 0$$

by (FSS), and thus

$$\zeta_i(z') = 0 \leq \zeta_i(z).$$

We have shown that $\zeta_i(z') \leq \zeta_i(z)$ for all $i = 1, \ldots, N$. Since $\lambda \geq 0$ is assumed, the objective value of z' in (LDOMP_1) is

$$\sum_{i=1}^{M} \lambda_i \zeta_i(z') \leq \sum_{i=1}^{M} \lambda_i \zeta_i(z), \tag{25}$$

i.e. no more than the objective value of z.

To conclude, we observe that by Corollary 1, the objective value of z is a lower bound on the value of (LDOMP_1): (LDOMP_1) with additional constraints (23) and (24) has the same optimal objective value as (LDOMP_1) alone, by Corollary 1, and (x, z) solves a relaxation of (LDOMP_1) with additional constraints (23) and (24). Since z' is feasible for (LDOMP_1), the objective value of z' is an upper bound on the optimal value of (LDOMP_1). But by (25), the value of z' is no greater than that of z. Therefore the value of z' must be the optimal value of (LDOMP_1) and (x, z') must be an optimal solution. By the validity of (LDOMP_1), x must induce an optimal solution to DOMP. □

As a consequence of the results above if (FSS) holds we can add constraints (23) and (24) to (LDOMP_1). Moreover, we can replace constraints $z_{ikj} \in \{0, 1\}$ by $0 \leq z_{ikj} \leq 1$ for all $i = 1, \ldots, N$ and all $j, k = 1, \ldots, M$.

Improvements for (LDOMP_2)

In the following we will prove some properties which are fulfilled by optimal solutions of (LDOMP_2).

Lemma 3 *If (x, y, s, w) is an optimal solution for (LDOMP_2) with objective value less than or equal to z_{UB} then*

$$\sum_{j \,:\, c_{kj} \leq \frac{z_{UB}}{\sum_{i=m}^{M} \lambda_i}} x_j \geq \sum_{i=1}^{m} s_{ik} \tag{26}$$

holds for all $k, m = 1, \ldots, M$.

Proof.
Let (x, y, s, w) be an optimal solution for (LDOMP$_2$), with objective value less than or equal to z_{UB}. Let X be the set of facilities induced by x and let σ_X be the permutation of $\{1, \ldots, M\}$ induced by s, i.e. let $\sigma_X(i) = j$ if and only if $s_{ij} = 1$. Then since (LDOMP$_2$) is a correct formulation, $|X| = N$, σ_X is a valid permutation for X and $L(X)$ is the objective value of (x, y, s, w) in (LDOMP$_2$), so $L(X) \leq z_{\text{UB}}$. Take any arbitrary $k, m \in \{1, \ldots, M\}$. By the definition of $\mathcal{P}$, $\sum_{i=1}^{m} s_{ik} \in \{0, 1\}$. Since $x \geq 0$, (26) trivially holds when $\sum_{i=1}^{m} s_{ik} = 0$. Thus, let us assume that $\sum_{i=1}^{m} s_{ik} = 1$. Therefore $s_{rk} = 1$ for some $1 \leq r \leq m$, and so $\sigma_X(r) = k$ and $\sigma_X^{-1}(k) = r \leq m$. Hence, by Proposition 1, we have that $c_k(X) \leq z_{\text{UB}} / \sum_{i=m}^{M} \lambda_i$. By (1), the definition of $c_k(X)$, there must exist $j \in X$ with $c_{kj} = c_k(X)$. Thus there exists j with $x_j = 1$ and with $c_{kj} \leq z_{\text{UB}} / \sum_{i=m}^{M} \lambda_i$, and so

$$\sum_{j \,:\, c_{kj} \leq \frac{z_{\text{UB}}}{\sum_{i=m}^{M} \lambda_i}} x_j \geq 1 = \sum_{i=1}^{m} s_{ik}$$

as required. □

Lemma 4 *Any optimal solution (x, y, s, w) of (LDOMP$_2$) satisfies*

$$w_i \geq \sum_{j=1}^{M} c_{kj} y_{kj} - \overline{c}_k (1 - \sum_{l=1}^{i} s_{lk}), \quad \forall i, k = 1, \ldots, M \tag{27}$$

where $\overline{c}_k$ is the Nth largest entry of the cost matrix at row k.

Proof.
Let (x, y, s, w) be an optimal solution of (LDOMP$_2$) and let $i, k \in \{1, \ldots, M\}$. Then by constraints (18) and (20) we have for any $l \in \{1, \ldots, i\}$ that

$$w_i \geq w_l \geq \sum_{j=1}^{M} c_{kj} y_{kj} - \overline{c}_k (1 - s_{lk}). \tag{28}$$

Now since $s \in \mathcal{P}$ we have to consider only two cases: either $s_{lk} = 0$ for all $l \in \{1, \ldots, i\}$, or there exists $l' \in \{1, \ldots, i\}$ such that $s_{l'k} = 1$ and $s_{lk} = 0$ for all $l \in \{1, \ldots, i\} \setminus \{l'\}$. In both cases, $\sum_{l=1}^{i} s_{lk} = s_{l'k}$ for some $l' \in \{1, \ldots, i\}$, and by (28) with $l = l'$ we get that (27) is satisfied, as required. □

Note that (27) is a direct strengthening of (20), and so the former can replace the latter in the formulation of (LDOMP$_2$).

In what follows, we find it convenient to define an *(LDOMP$_2$)-feasible point* induced by a set of facilities $X \subseteq \{1, \ldots, M\}$.

Definition 1 *Assume (FSS) holds. We say that* (x, y, s, w) *is an* (LDOMP$_2$)-feasible point induced by X *if* x, y, s *and* w *are defined as follows. Let* x *binary be defined by* $x_j = 1$ *if and only if* $j \in X$, *for each* $j \in \{1, \ldots, M\}$. *Choose* σ_X *to be a valid permutation for* X *satisfying* (21). *(Such a permutation exists by Proposition 2.) Let* s *binary be defined by* $s_{ik} = 1$ *if and only if* $\sigma_X(i) = k$, *for each* $i, k \in \{1, \ldots, M\}$. *For each* $k \in \{1, \ldots, M\}$, *choose* $j(k) \in \arg\min_{j \in X} c_{kj}$, *and let binary* y *be defined by* $y_{kj} = 1$ *if and only if* $j = j(k)$, *for each* $j \in \{1 \ldots, M\}$. *Set* $w_i = c_{\sigma_X(i)}(X)$ *for all* $i = 1, \ldots, M$.

Of course, to apply the definition, we require $|X| = N$. In this case, it is easy to show that the name is justified, and that (x, y, s, w) must indeed be a feasible point for (LDOMP$_2$).

Lemma 5 *If (FSS) holds then there exists an optimal solution* (x, y, s, w) *of (LDOMP*$_2$*) satisfying*

$$w_1 = w_2 = \cdots = w_N = 0, \tag{29}$$

$$\sum_{i=1}^{N} s_{ij} = x_j, \qquad \forall j = 1, \ldots, M, \tag{30}$$

and

$$w_{N+1} \geq \underline{c} x_{j^*} + \underline{d}(1 - x_{j^*}), \tag{31}$$

where $\underline{c} = \min_{\substack{k,j=1,\ldots,M,\\ k \neq j}} c_{kj} = c_{k^* j^*}$ *and* $\underline{d} = \min_{\substack{k,j=1,\ldots,M,\\ k \neq j,\ j \neq j^*}} c_{kj}$.

Proof.
Let X be an optimal solution of DOMP and let (x, y, s, w) be an (LDOMP$_2$)-feasible point induced by X. Let σ_X be the permutation chosen in defining this (LDOMP$_2$)-feasible point. Now

$$\sum_{i=1}^{M} \lambda_i w_i = \sum_{i=1}^{M} \lambda_i c_{\sigma_X(i)}(X) = L(X)$$

so by the correctness of (LDOMP$_2$), and since X is optimal for DOMP, it must be that (x, y, s, w) is optimal for (LDOMP$_2$). By (22), and since $w_i = c_{\sigma_X(i)}(X)$ for all $i = 1, \ldots, M$, by definition, we have that $w_1 = w_2 = \cdots = w_N = 0$, and (29) holds. Furthermore, by (21), and the definition of s, we have, for each $j = 1, \ldots, M$, that if $x_j = 1$, so $j \in X$, then $s_{ij} = 1$ if and only if $i \in \{1, \ldots, N\}$, whilst if $x_j = 0$, so $j \notin X$, then $s_{ij} = 0$ for all $i \in \{1, \ldots, N\}$. Thus, and since $\sum_{i=1}^{M} s_{ij} = 1$, it must be that (30) holds.

Now by (21), and since $|X| = N$, we have that $\sigma_X(N+1) \notin X$. Now $c_{\sigma_X(N+1)}(X) = c_{\sigma_X(N+1),j}$ for some $j \in X$, by definition, and so it must be that $\sigma_X(N+1) \neq j$. Thus $c_{\sigma_X(N+1)}(X) \geq \underline{c}$. Now $w_{N+1} = c_{\sigma_X(N+1)}(X)$, by definition of w, so we have $w_{N+1} \geq \underline{c}$. Thus (31) holds in the case $x_{j^*} = 1$. In the case $x_{j^*} = 0$, we have $j^* \notin X$, and $c_{\sigma_X(N+1)}(X) = c_{\sigma_X(N+1),j}$ for some $j \neq j^*$, where of course $j \neq \sigma_X(N+1)$ as well. Thus $w_{N+1} = c_{\sigma_X(N+1)}(X) \geq \underline{d}$ as required. □

We now give our final result. Note that its proof makes use of a property of sorted vectors. Since this property is quite general, we relegate its statement and proof to Appendix A (see Lemma 6).

Proposition 4 *Suppose (FSS) holds and let (x, y, s, w) solve (LDOMP$_2$) with additional constraints* (29)*,* (30) *and* (31)*, but with integrality relaxed on all variables s_{ij} with $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$. Then x induces an optimal solution to DOMP.*

Proof.
Let (x, y, s, w) be an optimal solution of (LDOMP$_2$), satisfying (29), (30) and (31), with integrality relaxed on all variables s_{ij} with $i \in \{1, \ldots, N\}$ and $j \in \{1, \ldots, M\}$. Let X be the set of facilities induced by x, and choose σ_X to be a valid permutation for X satisfying (22). (Such a permutation exists, by Proposition 2, and since (FSS) holds.) Also let $\hat{y}$, $\hat{s}$ and $\hat{w}$ be constructed, using σ_X, so that $(x, \hat{y}, \hat{s}, \hat{w})$ is an (LDOMP$_2$)-feasible point induced by X. Now for $i \in \{N+1, \ldots, M\}$, if $s_{ij} = 1$ then it must be $j \notin X$, since if $j \in X$ then $x_j = 1$, so by (30), $\sum_{i=1}^{N} s_{ij} = 1$, and hence $\sum_{i=N+1}^{M} s_{ij} = 1 - \sum_{i=1}^{N} s_{ij} = 0$, since $s \in \mathcal{P}$. Furthermore, if $j \notin X$, then $x_j = 0$, so by(30), $\sum_{i=1}^{N} s_{ij} = 0$, and hence $\sum_{i=N+1}^{M} s_{ij} = 1 - \sum_{i=1}^{N} s_{ij} = 1$, i.e. there is some $i \in \{N+1, \ldots, M\}$ such that $s_{ij} = 1$. Thus s_{ij} for $i \in \{N+1, \ldots, M\}$, $j \in \{1, \ldots, M\}$, binary, assigns a position in the ordering of $N+1$ or higher to each client $j \notin X$. Now $\{\sigma_X(i) \ : \ i = 1, \ldots, N\} = X$, so σ_X assigns a position in the ordering of N or lower for each $j \in X$. Thus setting

$$s'_{ij} = \begin{cases} s_{ij}, & i \geq N+1 \\ 1, & i \leq N \text{ and } \sigma_X(i) = j \\ 0, & \text{otherwise} \end{cases}$$

for each $i, j \in \{1, \ldots, M\}$ must yield $s' \in \mathcal{P}$. Let σ be the permutation of $\{1, \ldots, M\}$ induced by s'. Then for $i \in \{1, \ldots, N\}$, $\sigma(i) = \sigma_X(i) \in X$, and for $i \in \{N+1, \ldots, M\}$, $\sigma(i) = k$ if and only if $s_{ik} = 1$, in which case $k \notin X$. Now for each $i = N+1, \ldots, M$,

$$\begin{aligned}
w_i &\geq \sum_{j=1}^{M} c_{\sigma(i)j} y_{\sigma(i)j} \quad \text{by (20) and since } s_{i\sigma(i)} = 1 \\
&= \sum_{j \in X} c_{\sigma(i)j} y_{\sigma(i)j} \quad \text{since } (x, y) \in \mathcal{N}, \text{ so } y_{kj} \leq x_j \text{ for all } k, j, \text{ and } x_j = 0 \text{ if } j \notin X \\
&\geq c_{\sigma(i)}(X) \sum_{j \in X} y_{\sigma(i)j} \quad \text{since by definition, } c_{\sigma(i)}(X) \leq c_{\sigma(i)j} \text{ for all } j \in X \\
&\geq c_{\sigma(i)}(X) \quad \text{since } (x, y) \in \mathcal{N}, \text{ so } y \geq 0 \text{ and } \textstyle\sum_{j=1}^{M} y_{\sigma(i)j} = 1 \\
&= \hat{w}_{\sigma_X^{-1}(\sigma(i))}
\end{aligned}$$

by the definition of $\hat{w}$. Also, for each $i = 1, \ldots, N$, we have $w_i = 0$, by (29), and $\sigma(i) \in X$, so $c_{\sigma(i)}(X) = 0$, as (FSS) holds. Thus

$$w_i = 0 = c_{\sigma(i)}(X) = \hat{w}_{\sigma_X^{-1}(\sigma(i))},$$

again by the definition of $\hat{w}$. So

$$w_i \geq \hat{w}_{\sigma_X^{-1}(\sigma(i))}, \quad \forall i = 1, \ldots, M.$$

Observe that w satisfies (45), since it satisfies (18), and that $\hat{w}$ satisfies (46), by its definition, and the definition of σ_X. Furthermore w and $\hat{w}$ satisfy (47) for the permutation $\sigma_X^{-1}\sigma$. Thus the conditions of Lemma 6 (in Appendix A) are met, and we deduce that

$$\hat{w}_i \leq w_i, \quad \forall i = 1, \ldots, M.$$

Now we have $(x, \hat{y}, \hat{s}, \hat{w})$ feasible for (LDOMP$_2$), with objective value

$$\sum_{i=1}^{M} \lambda_i \hat{w}_i \leq \sum_{i=1}^{M} \lambda_i w_i,$$

since $\lambda \geq 0$. But the optimal objective value of (LDOMP$_2$) is equal to the optimal objective value of (LDOMP$_2$) with additional constraints (29), (30) and (31), by Lemma 5. Since (x, y, s, w) solves a relaxation of this, it must be that the optimal objective value of (LDOMP$_2$) is no less than $\sum_{i=1}^{M} \lambda_i w_i$. Therefore $(x, \hat{y}, \hat{s}, \hat{w})$ must be an optimal solution of (LDOMP$_2$), and so by the correctness of the formulation, x must induce an optimal solution to DOMP. □

From the above results, we conclude that if (FSS) holds, then we can add constraints (26), (29), (30) and (31) to (LDOMP$_2$), and we can replace constraints (20) by (27). Moreover, we can relax constraints $s_{ij} \in \{0, 1\}$, substituting them with $0 \leq s_{ij} \leq 1$, for all $i = 1, \ldots, N$ and all $j = 1, \ldots, M$.

5 Comparing the Linearizations

In order to get a good comparison between the linearizations (LDOMP$_1$) and (LDOMP$_2$) we have developed an experimental design with the following factors and levels:

- Size of the problem: As we have seen the number of sites, M, determines the dimensions of the cost matrix (C) and the λ-vector. Moreover, it is an upper bound of the number of facilities (N) to be located. For these reasons we consider M as a factor in our design and we propose four levels: $M = 8, 10, 12, 15$.

- New facilities: N is the second factor with four levels. To obtain equivalent levels, they are presented as proportions of the M values. These levels are: $\lceil \frac{M}{4} \rceil$, $\lceil \frac{M}{3} \rceil$, $\lceil \frac{M}{2} \rceil$, and $\lceil \frac{M}{2} + 1 \rceil$. Therefore for $M = 8$ we consider $N = 2, 3, 4, 5$, for $M = 10$, $N = 3, 4, 5, 6$, for $M = 12$, $N = 3, 4, 6, 7$ and finally for $M = 15$, $N = 4, 5, 8, 9$.

- Type of problem: Each λ-vector is associated with a different objective function, i.e. it models different types of problems. As these problems are unlikely to have the same difficulty of being solved, λ is also considered as a factor. Its levels were constructed depending on the value of M as follows:

1. T1: λ-vector corresponding to the N-median problem, i.e. $\lambda = (\underbrace{1, \ldots, 1}_{M})$.
2. T2: λ-vector corresponding to the N-center problem, i.e. $\lambda = (\underbrace{0, \ldots, 0}_{M-1}, 1)$.
3. T3: λ-vector corresponding to the k-centra problem, i.e. $\lambda = (0, \ldots, 0, \underbrace{1, \ldots, 1}_{k})$, where $k = \lfloor \frac{M}{3} \rfloor$.
4. T4: λ-vector corresponding to the $k_1 + k_2$-trimmed mean problem, i.e. $\lambda = (\underbrace{0, \ldots, 0}_{k_1}, 1, \ldots, 1, \underbrace{0, \ldots, 0}_{k_2})$, where $k_1 = N + \lceil \frac{M}{10} \rceil$ and $k_2 = \lceil \frac{M}{10} \rceil$.
5. T5: λ-vector with binary entries alternating both values and finishing with an entry equal to 1, i.e. $\lambda = (0, 1, 0, 1, \ldots, 0, 1, 0, 1)$ if M is even and $\lambda = (1, 0, 1, \ldots, 0, 1, 0, 1)$ if M is odd.
6. T6: As T5, but finishing with an entry equal to 0, i.e. $\lambda = (1, 0, 1, 0, \ldots, 1, 0, 1, 0)$ if M is even and $\lambda = (0, 1, 0, \ldots, 1, 0, 1, 0)$ if M is odd.
7. T7: λ-vector generated by the repetition of the sequence $0, 1, 1$ from the end to the beginning, i.e. $\lambda = (\ldots, 0, 1, 1, 0, 1, 1)$.
8. T8: λ-vector generated by the repetition of the sequence $0, 0, 1$ from the end to the beginning, i.e. $\lambda = (\ldots, 0, 0, 1, 0, 0, 1)$.

- Linearization: This factor has two levels corresponding to the two linearizations we would like to test: (LDOMP$_1$) and (LDOMP$_2$).

In this way we have a linear model with four factors (M with four levels, N with four levels, λ with eight levels and two linearizations) therefore, we get 256 different combinations. For each combination we do 15 replications, thus this design leads to 3840 problems to be solved. We have considered two dependent variables in the model: the integrality gap and the computing time. All the factors were tested to be simultaneously meaningful performing a multivariate analysis of the variance applied to the model. Additionally, principal component analysis was also applied in order to check whether some factors may be removed from consideration. Once more, we get the conclusion that none of them is meaningless.

Numerical Comparisons

In this section we present computational results for solving the DOMP using either (LDOMP$_1$) or (LDOMP$_2$). The DOMP was solved with the commercial package ILOG CPLEX 6.6 using the C++ modelling library ILOG Planner 3.3 (see [8]). These computational results were obtained using a Pentium III 800 Mhz with 1 GB RAM. The upper bounds used in constraints, as discussed in the previous section, were obtained by a heuristic method based on variable neighbourhood search (see [3]).

We first comment briefly on the effect of the improvements presented in Section 4 on the linear formulations. Both linearizations (LDOMP_1) and (LDOMP_2) are greatly affected. Some small examples ($M = 8$) were tested to compare the performance of each linearization, with and without the strengthening provided by results in Section 4. The results are reported in Table 1. We found that for all types of problems (except type T1, for which the bound was unaffected) the integrality gap provided by the improved linearizations was significantly smaller than that given by the original formulation. The effect was more pronounced for (LDOMP_2), which had weaker bounds to begin with. The gap for this formulation was reduced by a factor of up to more than twenty, (which was achieved on type T2 problems), with the average reduction being a factor of around 4.6. For (LDOMP_1) the greatest reduction was observed on type T4 problems, with the gap given by the improved formulation just over one eighth of the gap reported by the original formulation, i.e. an improvement by a factor of almost eight. The average gap reduction for (LDOMP_1) across all problem types was by a factor of about 2.6. Improvements in computing time were even more dramatic. Even for problems of type T1, for which no improvement in root node bound was reported, the computing time was decreased by an order of magnitude for (LDOMP_1) and cut in about four for (LDOMP_2). Indeed for every problem type, the computing time for (LDOMP_1) was reduced by at least one order of magnitude, with a reduction of two orders of magnitude for two of the eight problem types. For (LDOMP_2), the computing time on all types of problems except T1 was reduced by at least one order of magnitude, with two problem types showing reductions of two orders of magnitude, and three types having computing time around three orders of magnitude less. Thus we see that the strengthening provided by results in Section 4 are highly effective in improving both formulations.

Formulation	Problem Type	Gap (%) original	Gap (%) improved	Ratio original/ improved	Computing Time (s) original	Computing Time (s) improved	Ratio original/ improved
	T1	0.71	0.71	1.00	0.63	0.07	9.00
	T2	68.44	39.27	1.74	24.31	0.10	243.10
	T3	63.05	13.88	4.54	15.93	0.10	159.30
(LDOMP_1)	T4	58.40	7.46	7.83	3.19	0.07	45.57
	T5	20.90	15.92	1.31	1.69	0.12	14.08
	T6	21.55	13.82	1.56	0.79	0.09	8.78
	T7	18.36	9.22	1.99	1.10	0.13	8.46
	T8	32.72	26.75	1.22	1.97	0.14	14.07
	T1	0.71	0.71	1.00	0.40	0.11	3.64
	T2	68.44	3.22	21.25	169.88	0.05	3397.60
	T3	63.05	17.89	3.52	125.04	0.08	1563.00
(LDOMP_2)	T4	100	38.81	2.58	158.96	0.19	836.63
	T5	20.90	12.69	1.65	6.71	0.15	44.73
	T6	100	30.71	3.26	73.52	0.17	432.47
	T7	18.36	11.65	1.58	2.98	0.15	19.87
	T8	32.72	15.79	2.07	14.46	0.12	120.50

Table 1: Gaps and computing times for the (LDOMP_1) and (LDOMP_2) formulations, with and without the improvements provided by Section 4, on problems with $M = 8$. Each row of the table represents an average taken over problems with $N = 2, 3, 4$ and 5, where for each value of N, fifteen instances of the stated problem type were solved and the results averaged.

Our next step is to compare the two linearizations. To this end, we show the number of nodes in the branch and bound tree, the gap between the optimal solution and the linear relaxation objective function value, and the computing time in seconds. Each row represents the result of 15 replications of each combination (M, N). For this reason we compute the average, minimum and maximum of the number of nodes, integrality gap and computing time. We first give detailed results of the experiments with problems having $M = 15$, and then describe the results of our analysis for all 3840 test problems.

In Tables 2-9 we summarize the results of our experiments on problems with $M = 15$. We denote as L1 and L2 the results corresponding to the linearization (LDOMP_1) and (LDOMP_2), respectively.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	697.53	166	1577	2.39	0.00	18.80	8.40	2.42	18.94
		5	311.87	6	894	1.19	0.00	10.81	2.99	0.36	7.19
		8	12.53	0	121	0.00	0.00	0.00	0.31	0.19	1.06
		9	3.80	0	19	0.14	0.00	2.08	0.22	0.16	0.31
L2	15	4	2210.33	164	9350	2.39	0.00	18.80	25.41	5.03	114.16
		5	727.40	92	2570	1.19	0.00	10.81	8.79	2.83	21.41
		8	33.07	2	103	0.00	0.00	0.00	0.95	0.28	2.17
		9	19.93	0	63	0.14	0.00	2.08	0.63	0.17	1.61

Table 2: Computational results corresponding to the T1 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	5107.20	548	12181	50.55	31.60	66.16	58.85	4.98	117.09
		5	2242.07	114	8196	53.26	28.00	73.75	17.81	2.00	74.00
		8	42.73	10	129	40.50	26.19	57.14	0.41	0.28	0.70
		9	31.53	0	150	39.65	25.00	60.00	0.38	0.22	0.88
L2	15	4	1.20	0	7	16.21	0.00	40.72	1.03	0.61	1.59
		5	2.67	0	15	16.61	0.00	59.92	0.84	0.23	1.69
		8	0.00	0	0	0.00	0.00	0.00	0.26	0.08	0.41
		9	0.40	0	2	12.06	0.00	45.45	0.29	0.09	0.61

Table 3: Computational results corresponding to the T2 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	2026.87	405	7079	32.25	16.86	50.00	22.54	3.72	96.09
		5	976.93	138	4620	25.44	16.37	45.90	7.26	1.08	27.08
		8	31.13	0	129	12.63	0.00	22.08	0.41	0.19	0.94
		9	11.87	0	47	6.21	0.00	13.04	0.28	0.17	0.47
L2	15	4	544.40	91	1091	33.79	26.95	45.26	9.41	4.08	14.25
		5	467.27	69	1570	28.90	20.37	43.48	6.51	2.39	14.50
		8	100.60	11	358	13.55	4.76	19.19	1.55	0.53	3.86
		9	64.73	5	188	8.11	4.44	13.62	0.98	0.33	1.95

Table 4: Computational results corresponding to the T3 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	369.00	28	1054	12.47	1.08	17.29	4.37	1.13	10.92
		5	108.73	12	385	12.77	5.10	19.20	1.70	0.53	5.17
		8	1.67	0	7	10.32	0.00	26.67	0.32	0.20	0.44
		9	1.60	0	5	10.83	0.00	33.33	0.30	0.19	0.39
L2	15	4	41956.87	271	178289	82.65	70.83	91.25	578.05	7.36	2168.08
		5	4386.73	151	34872	76.62	53.85	87.76	63.86	6.22	472.94
		8	34.47	2	209	50.25	25.00	76.92	1.55	0.28	3.72
		9	8.53	0	26	34.51	0.00	75.00	0.62	0.19	1.89

Table 5: Computational results corresponding to the T4 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	2357.20	593	5758	13.26	7.46	26.15	17.35	3.66	45.47
		5	1459.87	230	6206	12.15	4.41	26.67	9.04	1.59	37.64
		8	36.67	0	125	8.33	0.00	18.18	0.37	0.22	0.94
		9	37.73	0	147	12.32	0.00	29.17	0.36	0.17	0.70
L2	15	4	15986.47	166	46286	13.39	7.46	26.15	108.06	4.78	358.00
		5	4810.87	85	28586	12.15	4.41	26.67	28.99	2.64	150.23
		8	80.13	4	357	8.33	0.00	18.18	1.42	0.42	3.78
		9	108.60	3	435	12.19	0.00	27.19	1.29	0.33	3.16

Table 6: Computational results corresponding to the T5 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	1097.73	51	3414	11.96	4.41	23.61	10.63	1.45	28.84
		5	413.80	68	1727	12.03	4.29	20.97	3.62	0.98	10.20
		8	32.73	0	160	12.32	0.00	29.17	0.58	0.25	1.27
		9	5.67	0	13	8.06	0.00	21.43	0.26	0.14	0.41
L2	15	4	29733.13	894	93126	82.36	75.00	87.95	266.90	14.45	719.80
		5	19748.87	612	115575	78.00	64.29	85.55	201.05	7.20	1304.75
		8	261.80	37	785	62.63	25.00	81.25	4.62	1.52	12.42
		9	35.47	0	149	44.73	0.00	70.00	1.37	0.08	3.02

Table 7: Computational results corresponding to the T6 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	1564.67	320	3960	11.96	7.63	23.61	12.87	2.45	30.77
		5	835.73	142	1744	12.15	5.21	20.83	5.04	1.22	9.66
		8	36.20	0	108	9.95	0.00	15.48	0.37	0.16	0.72
		9	22.13	0	93	9.41	0.00	21.48	0.31	0.19	0.56
L2	15	4	9981.33	225	86922	12.01	8.15	23.81	54.38	4.75	413.48
		5	1692.80	205	9298	11.85	5.21	20.83	12.12	3.84	56.42
		8	106.80	3	399	9.30	0.00	14.96	1.52	0.30	3.20
		9	55.87	0	172	9.33	0.00	21.67	0.90	0.23	2.17

Table 8: Computational results corresponding to the T7 type problem.

	Example		# nodes			gap(%)			CPU(s)		
Lin	M	N	aver	min	max	aver	min	max	aver	min	max
L1	15	4	5000.20	493	21037	21.81	13.77	32.14	46.95	2.52	162.47
		5	1400.93	233	3861	21.13	7.84	34.34	7.75	1.22	30.31
		8	66.33	0	263	16.88	0.00	29.17	0.50	0.20	1.22
		9	43.40	0	142	19.13	0.00	39.74	0.37	0.22	0.70
L2	15	4	18360.47	235	134087	21.85	14.49	31.91	104.12	4.31	654.41
		5	2366.33	131	10157	20.66	7.84	31.31	18.40	3.20	68.56
		8	59.87	6	264	15.54	0.00	22.22	1.39	0.42	3.33
		9	43.53	0	301	17.50	0.00	39.74	0.76	0.13	2.64

Table 9: Computational results corresponding to the T8 type problem.

Finally, our statistical analysis for all 3840 test problems is as follows. The comparison between (LDOMP$_1$) and (LDOMP$_2$) was made in two steps. First, we compare the results within each type of problem to test whether the integrality gap and/or the computing time show a statistically similar behavior in (LDOMP$_1$) and (LDOMP$_2$). To perform this test we use the Mann-Whitney U statistic. With respect to the gap, all types of problems but $T1$ are different under (LDOMP$_1$) and (LDOMP$_2$) with a confidence level above 90%. For the computing time, all types of problems are different with a confidence level above 88%. Second, we check which linearization performs better in each problem using robust statistics (trimmed mean) and confidence intervals for the mean value. The conclusions, calculated using all four levels of M, are:

		T1	T2	T3	T4	T5	T6	T7	T8
L1 vs L2	p-value	1.000	0.000	0.001	0.000	0.097	0.000	0.091	0.000
L1	trimmed mean	0.443	43.922	16.736	9.987	13.875	12.743	10.484	23.558
L2	trimmed mean	0.443	1.892	20.700	49.266	12.511	52.763	11.413	18.639

Table 10: p-values and trimmed means obtained for the integrality gap.

		T1	T2	T3	T4	T5	T6	T7	T8
L1 vs L2	p-value	0.000	0.000	0.011	0.000	0.000	0.000	0.000	0.115
L1	trimmed mean	0.519	5.481	0.951	0.348	1.020	0.688	0.977	1.461
L2	trimmed mean	1.147	0.223	1.020	6.173	2.453	6.799	1.602	2.092

Table 11: p-values and trimmed means obtained for the computing time.

From Table 10 we observe that (LDOMP_1) provides stronger lower bounds for (DOMP), for problem types T3, T4, T6, T7 and (LDOMP_2) bounds are stronger for T2, T5, T8.

The behaviour of the computing time is substantially different. From Table 11 we can observe that the computing time needed for solving the problem using (LDOMP_1) is significantly shorter than using (LDOMP_2). Only for T2 does (LDOMP_2) solve the problem faster.

As we can see from the computational experiments, large problems with smaller numbers of facilities cannot be solved to optimality. Therefore, we develop a branch and bound (B&B) algorithm that takes advantage of the structure of the problem.

6 A Branch and Bound Method

The driving variables for the DOMP are the binary x_j variables, indicating which sites have been selected for facility location. Once these are known, the objective value is easy to calculate. All the other variables are in the integer linear programming formulations to enable the costs to be calculated. It thus makes sense to build a branch and bound (B&B) method based entirely on the x_j variables, i.e. on decisions of whether or not a site is selected for facility location.

We develop a B&B in which each node represents a disjoint pair of sets of sites: a set of sites at which facilities will be opened and a set of sites at which facilities will *not* be opened. We refer to these as the set of *open* and *closed* sites respectively. Recall $A = \{1, \ldots, M\}$ denotes the set of sites. For a given node, we may let $F \subseteq A$ denote the set of open sites and $\overline{F} \subseteq A \setminus F$ denote the set of closed sites. We refer to the sites indexed by $A \setminus (F \cup \overline{F})$ as *undecided*. The node in the B&B tree is represented by the pair $(F, \overline{F})$. Of course, the node is a leaf node if either $|F| \geq N$ or $|\overline{F}| \geq M - N$. Otherwise, we calculate a lower bound on the cost function of the problem defined by $(F, \overline{F})$. Our lower bound is relatively simple to calculate; it does not require solution of a linear program. We discuss our lower bound in detail in Section 6.1.

Because of the nature of the cost function, making the decision to open a facility gives us very little information which could impact on a lower bound. It is only making the decision *not* to open a facility which restricts choices and forces the objective up. We thus develop a branching rule which is strong, and which ensures that on each branch some site is closed. We discuss our branching rule in detail in Section 6.2.

In Section 6.3 we compare the performance of our B&B method with that of the best integer linear programming formulation computationally.

6.1 Combinatorial Lower Bounds

A node in our B&B tree is represented by the disjoint pair of sets $(F, \overline{F})$ of sites open and closed, respectively. At each node which is not a leaf node of the B&B tree, we need to calculate a lower bound on the value of the cost function. Let $\hat{F} = A \setminus \overline{F}$ denote the set of sites which are either open, or undecided.

For any set of sites $R \subseteq \{1, \ldots, M\}$ we define $L(R)$ to be the cost of having facilities open at precisely those sites. So if we define

$$c_i(R) = \min_{j \in R} c_{ij} \tag{32}$$

to be the minimum cost assignment for a customer at site i to a facility in R, and define σ^R to be a permutation of $1, \ldots, M$ such that

$$c_{\sigma^R(1)}(R) \leq \cdots \leq c_{\sigma^R(M)}(R)$$

then we have that

$$L(R) = \sum_{i=1}^{M} \lambda_i \, c_{\sigma^R(i)}(R). \tag{33}$$

This notation is identical to that defined earlier for a set of sites X, with $|X| = N$; it is obviously not difficult to extend these definitions to arbitrary sets of sites.

We now derive two lower bounds, based on somewhat different ideas. The second only applies in the case self-service is cheapest (or free). As we show with an example, either one can be stronger, depending on the problem data. As both bounds are very easy to calculate, the lower bound we use in our B&B method is the maximum of the two.

In the following proposition we present a lower bound on the objective function value of any feasible solution having facilities in $\overline{F} \subseteq A$ closed.

Proposition 5 *Given $\overline{F} \subseteq A$ a set of closed sites with $|\overline{F}| < M - N$, let S be any set of N facilities not intersecting facilities in $\overline{F}$, i.e. let $S \subseteq \hat{F}$ and $|S| = N$, where $\hat{F} = A \setminus \overline{F}$. Then*

$$L(S) \geq L(\hat{F}),$$

i.e. $L(\hat{F})$ is a lower bound on the objective function value of any feasible solution having facilities in $\overline{F}$ closed.

Proof.

We know from (33) that $L(S) = \sum_{i=1}^{M} \lambda_i \, c_{\sigma^S(i)}(S)$ and $L(\hat{F}) = \sum_{i=1}^{M} \lambda_i \, c_{\sigma^{\hat{F}}(i)}(\hat{F})$. Since we assume $\lambda \geq 0$, we just have to show that $c_{\sigma^S(i)}(S) \geq c_{\sigma^{\hat{F}}(i)}(\hat{F})$ for all $i = 1, \ldots, M$ to show that $L(S) \geq L(\hat{F})$.

Observe that by (32), and since $S \subseteq \hat{F}$, it must be that $c_i(S) \geq c_i(\hat{F})$ for all $i = 1, \ldots, M$. Taking $r = s = M$, $p_i = c_i(S)$ and $q_i = c_i(\hat{F})$ for all $i = 1, \ldots, M$, $\sigma^P = \sigma^S$

and $\sigma^Q = \sigma^{\hat{F}}$, we see that Theorem 1 (Appendix A) may be applied to deduce that $p_{\sigma^P(i)} = c_{\sigma^S(i)}(S) \geq q_{\sigma^Q(i)} = c_{\sigma^{\hat{F}}(i)}(\hat{F})$ for all $i = 1, \ldots, M$, as required. □

Observe that if the self-service is free (FSS), i.e. $c_{ii} = 0$, for all $i \in A$, this lower bound is likely to be very weak, unless $|\hat{F}|$ is not too much greater than N, i.e. unless a relatively large number of facilities have been closed by branching. This will not occur until relatively deep in the B&B tree, which closes one more site at each level (see Section 6.2 for details of our branching rule). Thus, if (FSS) holds, we consider another lower bound, which may be somewhat more effective higher in the tree. In fact, the lower bound we propose applies more generally to the case that self-service is cheapest, i.e. $c_{ii} \leq c_{ij}$, for all $i, j = 1, \ldots, M$ with $j \neq i$. This condition, which we refer to as *cheapest self-service*, or *(CSS)*, can be assumed in most of the facility location problems without any capacity constraint.

The idea is that for any feasible set of columns $S \subseteq \hat{F}$, the feasible cost vector will consist of N diagonal elements and $M - N$ off-diagonal row minima, taken over S. Hence, using the vector consisting of the N smallest diagonal elements and the $M - N$ smallest off-diagonal row minima, taken over $\hat{F} \supseteq S$, we obtain a sorted cost vector which is in every component no more than the sorted cost vector given by S, and therefore provides a valid lower bound.

To introduce this lower bound, we define $D^{\hat{F}}$ to be the vector of diagonal elements of the cost matrix, taken over columns in $\hat{F}$, and let $d_1, \ldots, d_N$ be the N smallest elements in $D^{\hat{F}}$, ordered so that

$$d_1 \leq \cdots \leq d_N. \tag{34}$$

Furthermore, we define the vector $H^{\hat{F}} \in \mathbb{R}^M$ via

$$H_i^{\hat{F}} = \min_{j \in \hat{F},\ j \neq i} c_{ij}, \quad \forall i \in A, \tag{35}$$

to be the vector of cheapest off-diagonal elements in each row, over the columns in $\hat{F}$, and let $h_1, \ldots, h_{M-N}$ be the $M - N$ smallest elements in $H^{\hat{F}}$ ordered so that

$$h_1 \leq \cdots \leq h_{M-N}. \tag{36}$$

Finally, we define the vector $K^{\hat{F}} = (d_1, \ldots, d_N, h_1, \ldots, h_{M-N})$ and let $k_1, \ldots, k_M$ be the M elements of $K^{\hat{F}}$ ordered so that

$$k_1 \leq \cdots \leq k_M. \tag{37}$$

Now we define

$$B(\hat{F}) = \sum_{i=1}^{M} \lambda_i \, k_i. \tag{38}$$

Note that if self-service is in fact free, i.e. if (FSS) holds, then $d_1 = \cdots = d_N = 0$, $k_i = 0$ for $i = 1, \ldots, N$ and $k_i = h_{i-N}$ for $i = N + 1, \ldots, M$, and thus

$$B(\hat{F}) = \sum_{i=1}^{M-N} \lambda_{N+i} \, h_i. \tag{39}$$

In the proposition below, we prove that $B(\hat{F})$ is a valid lower bound on the objective value $L(S)$ for any feasible set $S \subseteq \hat{F}$ with $|S| = N$, if (CSS) holds. Note that the proof relies on a relatively simple, general result concerning sorted vectors: a vector of r real numbers that is componentwise no less than r elements chosen from a vector Q of s real numbers, $s \geq r$, is, when sorted, no less than (componentwise) the vector of the r smallest real numbers in Q, sorted. This general result is given in Appendix A, Lemma 7.

Proposition 6 *Given $\overline{F} \subseteq A$ a set of closed sites with $|\overline{F}| < M - N$, let S be any set of N facilities not intersecting facilities in $\overline{F}$, i.e. let $S \subseteq \hat{F}$ and $|S| = N$, where $\hat{F} = A \setminus \overline{F}$. Then, if (CSS) holds,*

$$L(S) \geq B(\hat{F}),$$

i.e. $B(\hat{F})$ is a lower bound on the objective function value of any feasible solution having facilities in $\overline{F}$ closed.

Proof.

From (33) we know that $L(S) = \sum_{i=1}^{M} \lambda_i \, c_{\sigma^S(i)}(S)$. Thus, since $\lambda \geq 0$ is assumed, to show that $B(\hat{F})$ is a lower bound on the objective function value of any feasible solution having facilities in $\overline{F}$ closed, we only need to show that $c_{\sigma_S(i)}(S) \geq k_i$, for all $i = 1, \ldots, M$. To do this, we need to consider diagonal and off-diagonal costs separately.

Firstly, we observe that the cost $c_i(S)$ is the diagonal cost matrix element in row i if $i \in S$; otherwise it is an off-diagonal element in a column in S, in row i. Thus the vector of all costs $c_i(S)$, $i = 1, \ldots, M$, which we denote by $c(S)$, has N elements which are diagonal cost matrix elements and $M - N$ elements which are off-diagonal cost matrix elements, and where every elements is taken from a column of the cost matrix which is in S. Let $D^S \in \mathbb{R}^N$ be the vector consisting of the N diagonal cost elements of $c(S)$, sorted in increasing cost order, i.e. chosen so that

$$D^S_1 \leq \cdots \leq D^S_N.$$

Since $S \subseteq \hat{F}$, for all $j = 1, \ldots, N$ there exists a unique $i(j) \in \hat{F}$ such that $D^S_j = D^{\hat{F}}_{i(j)}$. Then by Lemma 7 (in Appendix A) we have that $D^S_j \geq d_j$ for all $j = 1, \ldots, N$.

Similarly, if we let $H^S \in \mathbb{R}^{M-N}$ be the vector consisting of the $M - N$ off-diagonal cost elements of $c(S)$, sorted in increasing cost order, i.e. chosen so that

$$H^S_1 \leq \cdots \leq H^S_{M-N},$$

then for each $j = 1, \ldots, M - N$ there must exist a unique $i(j) \in \hat{F}$ such that $H^S_j \geq H^{\hat{F}}_{i(j)}$. To see this, we note that for each $j = 1, \ldots, M - N$ there must exist a unique $i(j) \in A \setminus S$ such that $H^S_j = c_{i(j)}(S)$, by the definition of H^S. Now since $S \subseteq \hat{F}$ and $i(j) \notin S$ we know that $c_{i(j)}(S) = \min_{i' \in S} c_{i(j)i'} \geq \min_{i' \in \hat{F},\ i' \neq i(j)} c_{i(j)i'} = H^{\hat{F}}_{i(j)}$ by the definition of $H^{\hat{F}}$, and so $H^S_j \geq H^{\hat{F}}_{i(j)}$ as required. Then by Lemma 7 we have that $H^S_j \geq h_j$ for all $j = 1, \ldots, M - N$.

Now we define $K^S = (D_1^S, \ldots, D_N^S, H_1^S, \ldots, H_{M-N}^S)$ and observe that we have shown K^S is (componentwise) no less than $K^{\hat{F}}$, i.e. $K_j^S \geq K_j^{\hat{F}}$ for all $j = 1, \ldots, M$. (This is obvious, since if $j \leq N$ then $K_j^S = D_j^S$ and $K_j^{\hat{F}} = d_j$, and we have shown above that $D_j^S \geq d_j$. Similarly if $j > N$ then $K_j^S = H_{j-N}^S$ and $K_j^{\hat{F}} = h_{j-N}$, and we have shown above that $H_{j-N}^S \geq h_{j-N}$.) Note also that K^S is simply a permutation of $c(S)$ and hence the ith component of K^S when sorted must be $c_{\sigma_S(i)}(S)$. Thus by Theorem 1 it must be that $c_{\sigma_S(i)}(S) \geq k_i$ for all $i = 1, \ldots, M$, (recall that k_i is, by definition, the ith component of $K^{\hat{S}}$ when sorted), as required.

Thus, as a consequence, since we assume $\lambda \geq 0$, we have that $B(\hat{F})$ is a lower bound on the cost of any feasible solution having facilities in $\overline{F}$ closed. □

The following example illustrates the two lower bounds, and demonstrates that their relative strength depends not only on how many sites have been closed by branching, but also on the value of λ.

Example 6.1 *Let $A = \{1, \ldots, 5\}$ be the set of sites and assume that we are interested in building $N = 2$ new facilities. Let the cost matrix be as follows:*

$$C = \begin{pmatrix} 0 & 4 & 5 & 6 & 4 \\ 5 & 0 & 6 & 2 & 2 \\ 7 & 1 & 0 & 5 & 1 \\ 7 & 4 & 3 & 0 & 5 \\ 1 & 3 & 5 & 4 & 0 \end{pmatrix}.$$

Suppose we are at a node of the branch and bound tree represented by the pair $(F, \overline{F})$ with $F = \emptyset$ and $\overline{F} = \{1\}$. So site 1 has been closed by branching and $\hat{F} = \{2, 3, 4, 5\}$. Observe that (FSS), and hence (CSS), holds in this example, since $c_{ii} = 0$ for all $i \in A$. Therefore both $C(\hat{F})$ and $B(\hat{F})$ are lower bounds on the value of the (DOMP) at this node.

To calculate the bound $C(\hat{F})$ we determine the vector of row minima over columns in $\hat{F}$, ($c_i(\hat{F})$ for all $i \in A$), to be $(4, 0, 0, 0, 0)$ yielding the sorted vector $(0, 0, 0, 0, 4)$.

To calculate $B(\hat{F})$, we have to determine the off-diagonal cost matrix row minima over columns in $\hat{F}$, i.e. we calculate $H^{\hat{F}} = (4, 2, 1, 3, 3)$. Thus the $M - N = 3$ smallest off-diagonal row minima are $h_1 = 1$, $h_2 = 2$ and $h_3 = 3$, and, since the diagonal costs are all zero, we get a lower bound based on the cost vector $k = (0, 0, 1, 2, 3)$.

Which of $C(\hat{F})$ or $B(\hat{F})$ yields the better bound depends on the value of λ. For instance, $\lambda = (1, 0, 0, 0, 1)$ means that $C(\hat{F}) = 0 + 4 = 4$ is better than $B(\hat{F}) = 0 + 3 = 3$. However $\lambda = (0, 0, 1, 1, 1)$ implies $B(\hat{F}) = 1 + 2 + 3 = 6$ which is better than $C(\hat{F}) = 0 + 0 + 4 = 4$.

Notice that both lower bounds can easily be computed. Hence, we propose when (CSS) holds, and in particular when (FSS) holds, we use the lower bound given by

$$\max\{C(\hat{F}), B(\hat{F})\} \tag{40}$$

at a node in the B&B tree identified by sets F and $\overline{F}$, with $\hat{F} = A \setminus \overline{F}$.

Observe that this lower bound is not trivial at the root node, represented by the pair $(\emptyset, \emptyset)$ giving $\hat{F} = A$, so we can compute a gap between the optimal objective function value (z^*) and $\max\{C(A), B(A)\}$ at the root node as follows:

$$\text{gap at root node} = \frac{z^* - \max\{C(A), B(A)\}}{z^*} \times 100. \tag{41}$$

In calculating the lower bound, we may have an opportunity to find a feasible solution of the same value, and so be able to prune the node. In calculating $C(\hat{F})$, the row minimum was found for each row, over columns in $\hat{F}$: let $m(i) \in \hat{F}$ denote the column in which the row minimum for row i was found. In case of a tie with a row minimum occurring in a column in F, $m(i)$ is chosen to be in F. Let $V(F, \overline{F}) = \{m(i) \ : \ i \in A\} \setminus F$ be the set of columns in which the selected row minima occur, outside of F. Now if $|V(F, \overline{F})| + |F| \leq N$, then any set $S \subseteq \hat{F}$ with $|S| = N$ and $S \supseteq V(F, \overline{F}) \cup F$ must be an optimal set for the subproblem at that node, and the lower bound $C(\hat{F})$ will be equal to the upper bound obtained from S. In this case, either the value of S is better than the current upper bound, which can thus be updated to the value of S, or the lower bound (equal to the value of S) is not better than the current upper bound; in either case the node can be pruned.

Similarly, in calculating $B(\hat{F})$ in the case that (CSS) holds, the off-diagonal row minimum was found for each row, over columns in $\hat{F}$: let $o(i) \in \hat{F}$ denote the column in which the off-diagonal row minimum for row i was found. In case of a tie with a row minimum occurring in a column in F, $o(i)$ is chosen to be in F. Let $V'(F, \overline{F}) = \{o(i) \ : \ i \in A \setminus F\} \setminus F$ be the set of columns in which off-diagonal row minima occur, outside of F, for rows not in F. Now if $|V'(F, \overline{F})| + |F| \leq N$, then any set $S \subseteq \hat{F}$ with $|S| = N$ and $S \supseteq V'(F, \overline{F}) \cup F$ must be an optimal set for the subproblem at that node, and the lower bound $B(\hat{F})$ will be equal to the upper bound obtained from S, the node can be pruned and if the value $B(\hat{F})$ is better than the current upper bound, the upper bound can be set to this value.

6.2 Branching

Since the lower bound presented in the previous section is based on row minima of the cost matrix calculated over columns corresponding to open or undecided sites, making the decision to open a site will not affect the lower bound. Closing a site, however, would be likely to increase the lower bound. We thus design a branching rule so that a (different) site is closed on each branch. We also ensure that the branching is strong, i.e. partitions the solution space.

We begin by describing the generic form of the branching rule, which assumes that an ordering of the undecided sites is given. We then describe two different orderings, and compare the two computationally.

Consider a node of the branch and bound tree, defined by the pair of sets $(F, \overline{F})$ with $|F| < N$ and $|\overline{F}| < M - N$. We may also assume that $|V(F, \overline{F})| > N - |F|$ and, if (CSS) holds, that $|V'(F, \overline{F})| > N - |F|$; otherwise the node would have been pruned, as discussed at the end of the previous section. Set $\hat{F} = A \setminus \overline{F}$. Suppose an ordering is given for the undecided sites, defined to be $U = \hat{F} \setminus F$. Say $\beta : \{1, \ldots, |U|\} \to U$ defines

the given ordering, so $\beta(i)$ is the index of the ith undecided site in the order. Note that $|U| = M - |\overline{F}| - |F| > N - |F|$ since $|\overline{F}| < M - N$. Our branching rule creates child nodes with the ith child node having site $\beta(i)$ closed and sites $\beta(1), \ldots, \beta(i-1)$ open. Now a node with more than N sites open is infeasible, so at most $N - |F| + 1$ child nodes need be created. Furthermore $|U| > N - |F|$ and so $N - |F| + 1$ child nodes can be created. In other words, the child nodes are defined by the pairs of sets $(F^i, \overline{F}^i)$ for $i = 1, \ldots, N - |F| + 1$, where $\overline{F}^i = \overline{F} \cup \{\beta(i)\}$ and $F^i = F \cup \{\beta(1), \ldots, \beta(i-1)\}$, with $F^1 = F$.

We now describe the two orderings we considered. The first is simply the site index ordering, i.e. we take β so that

$$\beta(1) \leq \cdots \leq \beta(N - |F| + 1).$$

We refer to the resulting branching rule as the *index-order* branching rule. The second ordering attempts to maximize the impact of the branching on the lower bound, and is much more complicated.

Recall $|V(F, \overline{F})| > N - |F|$. We wish to branch in a way which will have the most impact on the lower bound. We can do this by "eliminating" the column which will have the biggest impact on a row minima. Arguably, this will be a column containing the smallest row minimum. Thus we define for each $j \in V(F, \overline{F})$ the set of rows which have their row minimum in column j to be $W(j) = \{i \in A \ : \ m(i) = j\}$, (where $m(i)$ is as defined at the end of the previous section), and define the smallest row minimum in column j to be

$$v_j = \min_{i \in W(j)} c_{ij}.$$

Let σ^V denote a permutation of $V(F, \overline{F})$ which sorts the vector v in increasing order, i.e. such that

$$v_{\sigma^V(1)} \leq v_{\sigma^V(2)} \leq \cdots \leq v_{\sigma^V(|V(F,\overline{F})|)}.$$

Observe that when the self-service is free, i.e. (FSS) holds, there is little or nothing to differentiate the v values, unless $|\hat{F}|$ is not too much greater than N, i.e. unless a relatively large number of facilities have been closed by branching. This will not occur until relatively deep in the B&B tree. Thus, a secondary key could be used in sorting, such as the second-smallest row costs. For each row i, let u_i denote the second-smallest cost over columns in $\hat{F}$, and let w_j denote the largest difference between the second-smallest and smallest row element in $W(j)$, i.e. set

$$w_j = \max_{i \in W(j)} (u_i - c_{ij})$$

for each $j \in V(F, \overline{F})$. Now we may choose σ^V so that whenever $v_{\sigma^V(j)} = v_{\sigma^V(j')}$ for $j < j'$ it must be that $w_{\sigma^V(j)} \geq w_{\sigma^V(j')}$.

In either case, we take $\beta(i) = \sigma^V(i)$ for $i = 1, \ldots, N - |F| + 1$ to be the second ordering we consider. A similarly ordering can be used if the lower bound was achieved by $B(\hat{F})$, in the case that (CSS) holds. The only difference would be that everything should be based on off-diagonal row minima rather than row minima.

The branching rule resulting from this second ordering can be viewed as seeking to close sites in order of decreasing "maximum regret", i.e. maximizing the cost impact of the decision. Thus we refer to it as the *max-regret* branching rule. The example below illustrates the use of this rule.

Example 6.2 *Consider the data presented in Example 6.1. Assume that the current node is defined by* $F = \{4\}$ *and* $\overline{F} = \{1\}$*, and so* $\hat{F} = \{2, 3, 4, 5\}$*. Note that we expect to have* $N - |F| + 1 = 2 - 1 + 1 = 2$ *branches from this node.*

On the one hand, if the lower bound is given by $C(\hat{F})$*, we have to focus on the row minima. These are achieved in columns* $m(1) = 2$*,* $m(2) = 2$*,* $m(3) = 3$*,* $m(4) = 4$*,* $m(5) = 5$*, so* $V(F, \overline{F}) = \{2, 3, 5\}$*,* $W(2) = \{1, 2\}$*,* $W(3) = \{3\}$ *and* $W(5) = \{5\}$*, with* $v_2 = v_3 = v_5 = 0$*. Note that, in this case, we need the secondary key for sorting. The second-smallest cost over columns in* $\hat{F}$*, for rows not in* F*, are* $u_1 = 4$*,* $u_2 = 2$*,* $u_3 = 1$ *and* $u_5 = 3$*. Then* $w_2 = \max\{u_1 - c_{12}, u_2 - c_{22}\} = \max\{4-4, 2-0\} = 2$*,* $w_3 = u_3 - c_{33} = 1-0 = 1$ *and* $w_5 = u_5 - c_{55} = 3 - 0 = 3$*. Therefore* $\sigma^V(1) = 5$*,* $\sigma^V(2) = 2$ *and* $\sigma^V(3) = 3$*. Thus the two child nodes are defined by the pairs* $(\{4\}, \{1, 5\})$ *and* $(\{4, 5\}, \{1, 2\})$*.*

On the other hand, if the lower bound is achieved by $B(\hat{F})$*, now we have to focus on the off-diagonal row minima. These are achieved in columns* $o(1) = 2$*,* $o(2) = 4$*,* $o(3) = 2$*,* $o(4) = 3$*,* $o(5) = 2$*, so* $V'(F, \overline{F}) = \{2, 3\}$*,* $W(2) = \{1, 3, 5\}$ *and* $W(3) = \{4\}$*, with* $v_2 = 1$ *and* $v_3 = 3$*. Therefore* $\sigma^V(1) = 2$ *and* $\sigma^V(2) = 3$ *and the secondary key is not required. Thus the two child nodes are defined by the pairs* $(\{4\}, \{1, 2\})$ *and* $(\{4, 2\}, \{1, 3\})$*.*

6.3 Numerical Comparison of the Branching Rules

We implemented the B&B method using the programming language C++, with the upper bound initialized by a heuristic method based on variable neighbourhood search (see [3]). In both cases, the tree search strategy used was best bound. The method was run on a Pentium III 800 Mhz with 1 GB RAM, and tested on problem instances with the structure described in Section 5. As will be reported in more detail in Section 7, the B&B method performed very well, and much larger instances could be solved. Here we show the results of running the B&B algorithm using each branching rule on problems with $M = 30$, averaged over fifteen instances for each value of $N = 8, 10, 15, 16$. All cost matrices were randomly generated so that (FSS) holds. The results are presented in Table 12.

Problem Type	# of B& B Nodes		Ratio i.-ord./ max-r.	Computing Time (s)		Ratio i.-ord./ max-r.
	index-order	max-regret		index-order	max-regret	
T1	1727594.63	578787.85	2.99	912.34	235.92	3.87
T2	156211.15	17841.50	8.76	82.51	7.44	11.09
T3	772448.15	265769.78	2.91	417.62	107.04	3.90
T4	2774061.28	840401.43	3.30	1640.40	339.68	4.83
T5	1306657.1	428017.95	3.05	704.83	179.73	3.92
T6	2093979.28	633977.18	3.30	1143.49	256.07	4.47
T7	1388014.55	463225.33	3.00	730.87	190.62	3.83
T8	981157.38	310314.05	3.16	517.87	129.53	4.00
Average	1400015.44	442291.88	3.81	768.74	180.75	4.99

Table 12: Numbers of B&B nodes and computing times for the B&B method using either the index-order or max-regret branching rule on problems with $M = 30$ and $N = 8, 10, 15, 16$, for which (FSS) holds. Results are averages taken over fifteen problem instances for each value of N.

As can be seen from the table, using the max-regret branching rule reduces the number of B&B nodes by a factor of about 3.8 on average and reduces the computing time by a factor of about 5. The effect was more pronounced for problems of type T2 (i.e. N-center problems) for which the number of nodes required by the B&B algorithm with the max-regret rule was less than one eighth of the number of nodes provided by the algorithm with the intuitive branching rule. Furthermore, for this type of problems the computing time was decreased by an order of magnitude.

It is clear that the more sophisticated max-regret rule is much more effective than the simple index-order rule. Furthermore, the computing time for solving the instances with $M = 30$ is 180.75 seconds on average, and so problems with even more sites could be expected to be solved. However, after attempting problems with $M = 35$, we found that although problems could be solved to optimality, the computing times were in most cases around a factor of ten more than for $M = 30$. This indicates that the B&B method is not going to be particularly effective for problems much bigger than those with $M = 35$.

7 Computational Results

In this section we compare the computational performance of the B&B method described in Section 6, with the max-regret branching rule, with that of the best linearization (for type T2 problems this is (LDOMP$_2$) and for all other problem types it is (LDOMP$_1$)). Again all codes were run on a Pentium III 800 Mhz with 1 GB RAM, and tested on problem instances with the structure described in Section 5. Upper bounds for the B&B method were initialized by a heuristic method based on variable neighbourhood search (see [3]). The same upper bounds were used, where needed, in constraints defining the linearizations.

The performance of the B&B method was consistently better than that of the best linearization, with the former out-performing the latter by a significant margin. To illus-

trate the type of performance we observed, we give results for problems with $M = 18$ and $N = 5$, as well as $N = 10$, in Table 13. We give results for these extreme values of N as the performance of the linearizations generally improved as N increased; the performance of the B&B method was, by contrast, relatively consistent. Intermediate results values of N can be roughly interpolated between these extremes.

	Problem	Best Linearization			B&B Method		
N	Type	gap(%)	# nodes	CPU(s)	gap(%)	# nodes	CPU(s)
5	T1	2.8	1999.1	40.77	48.4	2102.5	0.57
	T2	24.6	3.8	3.37	54.4	473.0	0.13
	T3	34.2	17469.6	370.32	51.7	1477.3	0.40
	T4	12.1	3568.2	53.76	39.5	916.4	0.25
	T5	11.1	19169.1	187.03	49.0	2054.3	0.56
	T6	10.9	12797.7	190.53	44.8	1419.5	0.38
	T7	10.5	24350.6	289.50	49.2	1723.7	0.56
	T8	16.0	36343.93	408.60	49.0	1723.7	0.47
10	T1	0.0	42.1	0.75	20.4	1395.2	0.33
	T2	17.0	0.73	0.95	29.3	222.4	0.05
	T3	11.3	90.8	1.16	22.3	1030.3	0.24
	T4	9.8	11.1	0.64	14.3	1760.5	0.42
	T5	10.2	318.9	1.75	22.7	968.8	0.23
	T6	7.8	38.2	0.99	14.3	1124.5	0.27
	T7	7.9	63.1	0.81	21.2	1107.0	0.26
	T8	17.8	421.1	1.99	23.6	819.5	0.19

Table 13: Numerical results for problems with $M = 18$ and the extreme values of N tested. All results are averages taken over 15 problem instances.

We see that the B&B method always requires less CPU time than using the best linearization, and for some of the $N = 5$ problems more than two orders of magnitude less. The B&B method shows less variation in the number of nodes needed across problem type, and across different values of N. We report the average (root node) gap, but note that whilst this may be indicative of the quality of an integer programming formulation, it is less meaningful for the B&B method, where the bounds are very weak high in the tree but improve rapidly deeper in the tree.

To demonstrate that the B&B method can solve larger problems, we also provide detailed numerical results for problems with $M = 30$, in Table 14. As in Sections 5 and 6.3, each row of the tables contains results averaged over fifteen instances having the same parameter values.

8 Conclusions

We have presented two integer linear programming formulations for the DOMP, one having $O(M^3)$ variables and the other having $O(M^2)$ variables (both have $O(M^2)$ constraints). We go on to prove a variety of properties of optimal solutions, which allow us to strengthen

Problem		# nodes			gap(%)			CPU(s)		
Type	N	aver	min	max	aver	min	max	aver	min	max
T1	8	376661.7	90639	920625	50.8	40.4	58.3	167.19	40.92	408.73
	10	698401.1	50974	1678543	43.1	31.8	51.8	303.11	21.75	762.08
	15	710577.9	71558	1444065	28.2	15.2	38.2	274.74	28.03	562.25
	16	529510.7	107189	1112917	25.0	12.9	36.7	198.65	41.19	417.09
T2	8	30463.8	5043	100375	55.9	36.4	66.7	13.54	2.14	44.19
	10	15012.3	2565	38291	47.0	22.2	62.5	6.39	1.06	16.59
	15	12473.1	0	37005	37.2	0.0	50.0	4.81	0.00	14.84
	16	13416.8	275	35905	35.9	0.0	60.0	5.03	0.09	14.23
T3	8	195498.0	87519	447017	54.1	43.5	65.8	86.44	39.11	197.08
	10	275646.3	58988	867397	45.6	28.3	55.6	117.02	25.20	370.39
	15	326233.7	38961	630901	32.3	13.8	44.8	125.19	15.36	246.95
	16	265701.1	56589	585759	28.7	11.5	42.3	99.51	21.83	217.92
T4	8	124932.5	11396	275910	44.9	30.5	54.3	56.37	5.06	125.25
	10	354196.3	5306	1108572	36.7	20.5	47.8	154.46	2.31	476.23
	15	1606801.0	52119	3474800	21.3	7.1	33.3	649.56	20.81	1457.81
	16	1275675.9	110028	3970039	18.6	6.3	33.3	498.32	42.88	1460.81
T5	8	351434.6	98789	729440	51.8	42.0	60.0	160.75	45.42	332.02
	10	550405.8	58142	1364222	43.8	30.6	52.6	242.00	25.72	601.64
	15	533272.8	38551	1279337	28.9	14.3	40.9	209.68	15.55	502.16
	16	276958.6	47709	606337	26.4	12.5	40.0	106.49	18.88	239.69
T6	8	292640.9	34600	597933	48.3	35.7	54.3	131.82	15.64	272.22
	10	582688.3	29094	1265889	40.7	28.1	48.6	253.11	12.58	561.84
	15	839755.2	65764	1927705	26.4	12.5	40.0	327.44	26.34	777.64
	16	820824.3	50294	2022796	22.0	9.1	35.7	311.89	19.88	792.27
T7	8	380210.3	89852	825887	51.8	41.8	58.3	170.57	40.69	368.23
	10	625290.4	36296	1621612	43.1	30.4	53.3	269.65	15.44	712.63
	15	466872.1	31681	1160957	28.9	13.0	40.0	179.30	12.47	442.06
	16	380528.5	75834	809368	25.8	13.6	38.1	142.95	29.02	313.39
T8	8	337418.9	85926	710147	52.0	41.7	59.0	151.38	38.84	328.09
	10	436277.5	26389	1174336	42.5	28.0	51.7	188.97	11.33	503.84
	15	263030.8	2277	1139116	28.9	11.1	40.0	101.19	0.91	436.17
	16	204529.0	47135	479781	25.8	9.1	36.4	76.59	18.30	173.09

Table 14: Computational results for the B&B method with the max-regret branching rule for problems with $M = 30$.

the formulations, via either additional constraints or preprocessing (in particular, fixing the values of some variables). After defining eight problem classes, according to objective function, we compare the performance of the two formulations computationally. From our statistical study of the results, we can observe that in all but one of the eight problem classes, the $O(M^3)$ formulation performs better; only on the problem class corresponding to the N-center problem is the $O(M^2)$ formulation better. We believe the success of the $O(M^3)$ formulation is in part due to the fact that many of the properties of optimal solutions found could be implemented in this formulation using preprocessing, whereas additional constraints were needed to implement the same properties in the $O(M^2)$ formulation.

Even when using the better formulation, solution times for problems with M much larger than 20 were prohibitive. Thus we developed a specialized branch and bound method, based on combinatorial lower bounds that were very quick to calculate. We found that a "max-regret" branching rule performed very substantially better than a naive branching rule, and the resulting branch and bound method outperformed the integer programming approaches by a substantial margin. Problems with M smaller than or equal to 35 can be solved in reasonable time using this method.

We note that the aim of the DOMP is not only to unify all classical discrete facility location problems but also to model new problems not previously formulated as integer programs (e.g. the $k_1 + k_2$ trimmed mean problem). Our objective is to develop a common solution method for all these problems, not to develop a method which competes against other well-known approaches for specific discrete location problems. Thus we do not necessarily expect any of the methods we present here to outperform specialized methods already developed for some of the special cases of the DOMP. Nevertheless, we believe our general approaches could be improved, as we discuss below.

In future work, we plan to develop a hybrid branch and bound method, that makes use of the strengths of both the integer programming and branch and bound approaches we have developed. The integer programming formulations would provide lower bounds and preprocessing information high in the tree, (where they are clearly stronger than the combinatorial bounds), combinatorial lower bounds would be used lower in the tree, (to speed up the solution process), branching would be based only on the facility location variables, (as is the case in our branch and bound method), and the "max-regret" branching rule together with the ideas we present for detecting feasible solutions and early pruning of nodes in the branch and bound method would be used throughout. We believe this hybrid method would capitalize on the strengths of the methods we have presented here.

We also believe that more needs to be understood about the less well-known special cases of the DOMP objective. There is a vast literature on N-median problems, but very little is known about other instances of discrete location models presented in this paper. Our work here goes some way to classifying some key problem classes, and assessing their relative difficulty, for example we find that the N-median problem class is far from the hardest. But it is obvious much more can be done in this direction.

It should be noted that some of the ideas we present have the potential to be applied in other discrete location contexts, such as hub location or QAP.

Cutting planes are another avenue for future research. With respect to polyhedral analysis, it is interesting that the DOMP includes as special cases both the N-median problem, which is known to be "integer-friendly", and the N-center problem, for which not much is known about its facets. DOMP formulations could provide a helpful bridge in the study of such polyhedra.

Finally, it is obvious that good heuristics will be necessary if large DOMPs are to be tackled. Although a variable neighbourhood search, presented in [3], has been found to be effective, there is clearly room for more exploration in this direction.

References

[1] K. Aardal. Capacitated facility location: separation algorithms and computational experience. *Mathemathical Programming*, 81:149–175, 1998.

[2] M.S. Daskin. *Network and Discrete Location: Models, Algorithms, and Applications*. Wiley, 1995.

[3] P. Domínguez-Marín, S. Nickel, P. Hansen, and N. Mladenović. Heuristic procedures for solving the Discrete Ordered Median Problem. ITWM Bericht 46, Fraunhofer Institut für Techno– und Wirtschaftsmathematik (ITWM), Kaiserslautern, Germany, 2003.

[4] Z. Drezner and H.W. Hamacher (Eds.). *Facility Location: Applications and Theory*. Springer Berlin Heidelberg New York, 2002.

[5] J. Ebery, M. Krishnamoorthy, A. Ernst, and N. Boland. The capacitated multiple allocation hub location problem: formulations and algorithms. *European Journal of Operational Research*, 120:614–631, 2000.

[6] A. Ernst and M. Krishnamoorthy. Solution algorithms for the capacitated single allocation hub location problem. *Annals of Operations Research*, 86:141–159, 1999.

[7] R.L. Francis, T.J. Lowe, and A. Tamir. Aggregation error bounds for a class of location models. *Operations Research*, 48:294–307, 2000.

[8] ILOG Optimization Suite. ILOG, Inc., Incline Village, Nevada, 2000. http://www.ilog.com/products/optimization.

[9] J. Kalcsics, T. Melo, S. Nickel, and V. Schmid-Lutz. Facility location decisions in supply chain management. In K. Inderfurth, G. Schwödiauer, W. Domschke, F. Juhnke, P. Kleinschmidt, and G. Wäscher, editors, *Operations Research Proceedings 1999*, pages 467–472. Springer, 2000.

[10] O. Kariv and S.L. Hakimi. An algorithmic approach to network location problems. II: The p-medians. *SIAM Journal on Applied Mathematics*, 37:539–560, 1979.

[11] L. Kaufman and F. Broeckx. An algorithm for the quadratic assignment problem using Benders' decomposition. *European Journal of Operational Research*, 2:204–211, 1978.

[12] E.L. Lawler. The Quadratic Assignment Problem. *Management Science*, 9:586–599, 1963.

[13] G. Mayer and B. Wagner. Hublocator: an exact solution method for the multiple allocation hub location problem. *Computers and Operations Research*, 29:715–739, 2002.

[14] P.B. Mirchandani and R.L. Francis. *Discrete Location Theory*. Wiley, New York, NY, 1990.

[15] P. Neame, N. Boland, and D. Ralph. An outer approximate subdifferential method for piecewise affine optimization. *Mathemathical Programming*, 87:57–86, 2000.

[16] S. Nickel. Discrete Ordered Weber problems. In B. Fleischmann, R. Lasch, U. Derigs, W. Domschke, and U. Rieder, editors, *Operations Research Proceedings 2000*, pages 71–76. Springer, 2001.

[17] S. Nickel and J. Puerto. A unified approach to network location problems. *Networks*, 34:283–290, 1999.

[18] J. Puerto and F.R. Fernández. The symmetrical single facility location problem. Technical Report, Faculty of Mathematics, University of Sevilla, 1995.

[19] J. Puerto and F.R. Fernández. Geometrical properties of the symmetrical single facility location problem. *Journal of Nonlinear and Convex Analysis*, 1(3):321–342, 2000.

[20] A. Rodríguez-Chía, S. Nickel, J. Puerto, and F.R. Fernández. A flexible approach to location problems. *Mathematical Methods of Operations Research*, 51:69–89, 2000.

Appendix A

The idea that sorting preserves isotonicity, i.e. that if p and q are two vectors with $p \geq q$ (componentwise) then the sorted vector p is also (componentwise) at least as big as the sorted vector q, is a powerful and useful idea in location problems generally, as other authors have noted. A result of this kind is given in Theorem 1 in [7], in which the function defined on the vector space to return the jth-largest element of its operand is shown to be isotone (non-decreasing). We make use of this idea in strengthening the integer linear programming formulations, via Proposition 4, and in deriving combinatorial lower bounds, via Propositions 5 and 6. Although these applications could be proved as corollaries of Theorem 1 in [7], the approach is unnecessarily complicated for our purposes; we provide short, direct proofs of the results we need.

Theorem 1 *Let $p = (p_1, \ldots, p_r) \in \mathbb{R}^r$ and $q = (q_1, \ldots, q_s) \in \mathbb{R}^s$ be two vectors of real numbers with $r \leq s$. Let σ^P and σ^Q denote a permutation of $1, \ldots, r$ and $1, \ldots, s$, respectively, such that*

$$p_{\sigma^P(1)} \leq \cdots \leq p_{\sigma^P(r)}, \tag{42}$$

and

$$q_{\sigma^Q(1)} \leq \cdots \leq q_{\sigma^Q(s)}. \tag{43}$$

Then, if

$$p_k \geq q_k, \quad \forall k = 1, \ldots, r, \tag{44}$$

it holds that

$$p_{\sigma^P(k)} \geq q_{\sigma^Q(k)}, \quad \forall k = 1, \ldots, r.$$

Proof.
Consider $k \in \{1, \ldots, r\}$. We use the fact that because σ^P and σ^Q are permutations, the set $\{\sigma^P(1), \ldots, \sigma^P(k)\}$ consists of k distinct elements and the set $\{\sigma^Q(1), \ldots, \sigma^Q(k-1)\}$ consists of $k-1$ distinct elements, where if $k = 1$ the the latter set is simply taken to be the empty set. Thus there must exist $m \in \{1, \ldots, k\}$ such that $\sigma^P(m) \notin \{\sigma^Q(1), \ldots, \sigma^Q(k-1)\}$. (If $k = 0$ simply take $m = 1$.) Now by (43), it must be that $q_{\sigma^Q(k)} \leq q_{\sigma^P(m)}$. Furthermore, by (44), we have $q_{\sigma^P(m)} \leq p_{\sigma^P(m)}$ and by (42) we have $p_{\sigma^P(m)} \leq p_{\sigma^P(k)}$. Hence $q_{\sigma^P(k)} \leq p_{\sigma^Q(k)}$, as required. □

The following corollary to Theorem 1 shows that if two sorted vectors are given, and a permutation can be found to ensure one vector is (componentwise) no greater than the other, then the former vector in its original sorted state must also be no greater (componentwise) than the latter. This result is used in Proposition 4, to validate the strengthenings we apply to the (LDOMP$_2$) formulation.

Lemma 6 *Suppose $w, \hat{w} \in \mathbb{R}^M$ satisfy*

$$w_1 \leq w_2 \leq \cdots \leq w_M, \tag{45}$$

$$\hat{w}_1 \leq \hat{w}_2 \leq \cdots \leq \hat{w}_M, \tag{46}$$

and

$$\hat{w}_{\sigma(i)} \leq w_i, \quad \forall i = 1, \ldots, M \tag{47}$$

for some σ a permutation of $\{1, \ldots, M\}$. Then

$$\hat{w}_i \leq w_i, \quad \forall i = 1, \ldots, M. \tag{48}$$

Proof.
It is a corollary of Theorem 1. This can be seen by setting $w'_i = \hat{w}_{\sigma(i)}$ for all $i = 1, \ldots, M$, and by taking $r = s = M$, $p = (w_1, \ldots, w_M)$, $q = (w'_1, \ldots, w'_M)$, σ^P to be the identity permutation and $\sigma^Q = \sigma^{-1}$, the inverse permutation of σ. Note that in this case $w_{\sigma^P(i)} = w_i$ for all $i = 1, \ldots, M$, and furthermore

$$w'_{\sigma^Q(i)} = w'_{\sigma^{-1}(i)} = \hat{w}_{\sigma(\sigma^{-1}(i))} = \hat{w}_i, \quad \forall i = 1, \ldots, M. \tag{49}$$

It is obvious from (45) that the elements of p form an increasing sequence under the permutation σ^P. It is also obvious from (46) and (49) that the elements of q form an increasing sequence under the permutation σ^Q. Thus the first two conditions of Theorem 1 are met. From (47) and the definition of w', we have that $w_i \geq w'_i$ for all $i = 1, \ldots, M$ and the final condition of Theorem 1 is met. From Theorem 1 we thus deduce that $w_{\sigma^P(i)} \geq w'_{\sigma^Q(i)}$ for all $i = 1, \ldots, M$. Now for all $i = 1, \ldots, M$ we have that $w_{\sigma^P(i)} = w_i$ and $w'_{\sigma^Q(i)} = \hat{w}_i$, so $\hat{w}_i \leq w_i$ as required. □

The following corollary to Theorem 1 shows that a vector of r real numbers that is componentwise no less than r elements chosen from a vector Q of s real numbers, $s \geq r$, is, when sorted, no less than (componentwise) the vector of the r smallest real numbers in Q, sorted. This result is used in Proposition 6, to validate the lower bound we use in our B& B method when cheapest self-service applies.

Lemma 7 *Let $T = (t_1, \ldots, t_s)$ be a vector of $s \geq 1$ real numbers with*

$$t_1 \leq \cdots \leq t_s.$$

Let $r \in \{1, \ldots, s\}$ and let $S \in \mathbb{R}^r$ be a vector with elements no less, componentwise, than r of the elements of T, say $S = (t'_1, \ldots, t'_r)$ where for all $j = 1, \ldots, r$ there exists a unique $i(j) \in \{1, \ldots, s\}$ such that $t'_j \geq t_{i(j)}$, with

$$t'_1 \leq \cdots \leq t'_r.$$

Then

$$t'_j \geq t_j, \qquad \forall j = 1, \ldots, r.$$

Proof.
The claim follows from Theorem 1, as follows. Take r and s as given. Take $q_j = t_{i(j)}$ for all $j = 1, \ldots, r$ and define $q_{r+1}, \ldots, q_s$ to be the components from the vector T which do not have index in $\{i(j) \ : j = 1, \ldots, r\}$. Note there is a one-to-one correspondence between the elements of T and q, i.e. q is a permutation of T. Take $p_j = t'_j$ for all $j = 1, \ldots, r$, so $p_j = t'_j \geq t_{i(j)} = q_j$ for all $j = 1, \ldots, r$. Also take σ^P to be the identity permutation, and note that σ^Q can be taken so that that $q_{\sigma^Q(i)} = t_i$ for all $i = 1, \ldots, s$, by the definition of T and since q is a permutation of T. Now by Theorem 1 (Appendix A) it must be that for all $j = 1, \ldots, r$, $p_{\sigma^P(j)} \geq q_{\sigma^Q(j)} = t_j$, and so $p_j = t'_j \geq t_j$, as required. □