Chapter 7

A GLOBAL OPTIMIZATION PROCEDURE FOR THE CAPACITATED EUCLIDEAN AND l_p DISTANCE MULTIFACILITY LOCATION-ALLOCATION PROBLEMS

7.1 Introduction

This Chapter is concerned with designing a procedure for determining global minima for the capacitated Euclidean and l_p distance location-allocation problems. Given the fixed location of m existing facilities or customers on a continuous plane and their associated demands, this problem seeks the location of n new facilities or sources having known capacities, as well as the allocation of their supplies, in order to satisfy the demand requirements of customers at a minimum total cost. Without loss of generality, we assume that the total supply is equal to the total demand. The decisions to be made are where to locate the n sources and how much shipment to send from each source to each customer. In this context, the costs are assumed to be directly proportional to the quantities shipped and the distance over which this shipment occurs. When a straight line distance measure is used, we obtain the Euclidean distance location-allocation problem (EDLAP) which can be mathematically stated as follows.

$$\text{EDLAP: Minimize } f(x,w) = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} w_{ij} \{(x_i - a_j)^2 + (y_i - b_j)^2\}^{1/2} \tag{7.1a}$$

$$\text{subject to } \sum_{j=1}^{m} w_{ij} = s_i \qquad i=1,\ldots,n \tag{7.1b}$$

$$\sum_{i=1}^{n} w_{ij} = d_j \qquad j=1,\ldots,m \tag{7.1c}$$

$$w_{ij} \geq 0 \qquad i=1,\ldots,n,\ j=1,\ldots,m, \qquad (7.1d)$$

where m = the number of customers,

n = the number of supply centers,

(a_j, b_j)location of the customer j,

s_i = (annual) capacity of supply center i,

d_j = (annual) demand of customer j,

c_{ij} = cost (> 0) of unit flow per unit distance from supply center i to customer j.

and where the decision variables are:

(x_i, y_i): location of source i,

w_{ij}: amount shipped from supply center i to customer j.

We assume feasibility $(\sum_i s_i = \sum_j d_j)$, and for convenience, we denote

$$W = \{w \equiv w_{ij} : w \text{ satisfies the (transportation) constraints of problem EDLAP}\}. \qquad (7.2)$$

Note that for a fixed set of allocations $w \equiv (w_{ij},\ i=1,\ldots,n;\ j=1,\ldots,m)$, Problem EDLAP reduces to a pure location problem (see Francis et al., 1991), whereas for a fixed set of locations (x_i, y_i), $i=1,\ldots,n$, the problem reduces to the ordinary transportation/allocation problem (see Bazaraa et al., 1990). We also consider in this Study the l_p distance location-allocation problem where the separation penalty in the objective function is based on the l_p distance $\left\{\left|x_i - a_j\right|^p + \left|y_i - b_j\right|^p\right\}^{1/p}$ between source i and customer j, $\forall\ (i, j)$. As shown empirically by Love and Juel (1982), a value of $1 < p < 2$ (i.e., between the rectilinear and the Euclidean distance measures) more accurately represents actual travel distances over road networks.

The remainder of this Chapter is organized as follows. We begin in Section 7.2 by presenting a basic branch-and-bound framework for solving EDLAP, using a simple projected location space lower bounding scheme. Section 7.3 discusses the use of various valid inequalities and develops a special application of the Reformulation-Linearization Technique (RLT) of Sherali and Tuncbilek (1992b) in order to derive an enhanced lower bounding linear program that approximates the convex envelope of the objective function over the feasible region. A best-first continuous partitioning branch-and-bound procedure is described in Section 7.4. These algorithms are extended in Section 7.5 to the case of l_p distances. Some computational experience is provided in Section 7.6 to demonstrate the efficacy of a hybrid approach that combines the two lower bounding schemes, and Section 7.7 concludes the Chapter with a summary and recommendations for future research.

7.2. A Branch-and-Bound Algorithm Using a Projected Location Space Bounding Scheme

In this section, we present a (finite) branch-and-bound algorithm that implicitly enumerates the vertices of the feasible region of the transportation constraints of Problem EDLAP in order to obtain a global optimum for this nonconvex problem. Since each of the flow variables w_{ij} for $w \in W$ is either positive or zero at an optimal vertex of W, we employ the binary partitioning technique introduced by Sherali and Tuncbilek (1992a) in their approach for solving the squared Euclidean distance location-allocation problem. In this approach, the arcs associated with the positive flows form a forest subgraph of the transportation graph. As this algorithm progresses, a partial solution is represented at each node by classifying the arcs into three disjoint sets J^0, J^+, and J^F, where J^0 represents the "*zero-fixed*" variables w_{ik}, i.e., $J^0 = \{(i, k): w_{ik} = 0\}$, J^+ represents the "*positive-fixed*" variables w_{ik}, i.e., $J^+ = \{(i, k): w_{ik} \geq 1\}$, and

where J^F represents the "*free-variables*" w_{ik}, i.e., $J^F = \{(i, k): (i, k) \notin J^0 \cup J^+\}$. Note that the "positive-fixed" variables are not fixed in value, but are designated to belong to the forest subgraph that represents the positive valued variables at an extreme point solution. In order to assure this property for the positive-fixed variables, the algorithm includes an active check that maintains the arcs in J^+ cycle-free. Hence, if a cycle is formed in the current forest by introducing any new arc (i, k) from J^F, w_{ik} is fixed to zero, i.e., we transfer w_{ik} into J^0 from J^F. If the cardinality of J^F is zero at any point in this process, then a corresponding basic feasible solution of W is at hand. We adopt these main ideas for the skeletal framework of our branch-and-bound algorithm. However, we develop different specializations for our problem for the bounding step, the partitioning scheme, and a strategy for obtaining good quality incumbent solutions to aid in the fathoming process. Before presenting our overall branch-and-bound procedure, we first discuss some of its principal components.

7.2.1 Logical Tests

Given any current bound intervals $[l_{ij}, u_{ij}]$ for the flow variables w_{ij}, $i = 1,\ldots, n$, $j = 1,\ldots, m$, we can implement suitable logical tests as prescribed by Sherali and Tuncbilek (1992a) in order to determine tighter bounds based on the structure of the transportation constraints. These bounds are initialized as $l_{ij} = 0$, $u_{ij} = \min\{s_i, d_j\}$, $\forall i = 1,\ldots, n$, $j = 1,\ldots, m$, and are updated in a sequential loop based on retaining nonnegative values for the maximum slacks in the supply and demand constraints (written as equivalent inequalities). Whenever a maximum slack value is detected to be negative, this implies infeasibility and the branch-and-bound algorithm fathoms the current node. In the implementation, an active list of all arcs that are eligible for applying this logical test is maintained for the sake of efficiency. These logical tests also detect the

situation in which the current set of bounds (including zero restrictions on variables) yield a unique completion to the supply and demand constraints, and in this case, the method will compute the corresponding basic solution.

7.2.2 Projected Location Space Lower Bounding Scheme (PLSB)

Given the current set of bounding intervals on the flow variables, since $w_{ij} \geq l_{ij} \; \forall \, (i,j)$, we compute a lower bound v_{LB1} on the node subproblem via the following Euclidean location problem.

$$v_{LB1} = \underset{(x,y)}{\text{minimum}} \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} l_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \, . \tag{7.3}$$

Note that (7.3) can be solved separably for each (x_i, y_i), $i = 1,\ldots, n$, using any procedure for the single facility Euclidean location problem (see Eyster et al., 1973, or Sherali and Al-Loughani, 1997).

7.2.3 Upper Bounding Procedure

Any effective branch-and-bound procedure must actively search for good quality feasible solutions or upper bounds in order to enhance its fathoming power. Toward this end, we employ the alternating procedure of Cooper (1972) starting from judicious location solutions. Note that the *current* bound restrictions on the flow variables are imposed in this process when solving the transportation subproblems. Also, the alternating procedure is terminated whenever the improvement in objective values falls below 10^{-3}.

For the initial node in the branch-and-bound enumeration tree, we determine a starting location as follows. The customer locations are first enclosed in a tightest bounding rectangle oriented along the *x* and *y* axes. The rectangle is partitioned into *n* equally spaced slices, and the aggregate demands $D_1, ..., D_n$ are determined within these slices 1,…,*n*. The lists $\{D_1, ..., D_n\}$

and $\{s_1, ..., s_n\}$ are both rearranged in nondecreasing order and are then matched one-to-one. Let $s_{i(1)}, ..., s_{i(n)}$ be respectively matched with $D_1, ..., D_n$. Accordingly, permute the new facilities in a list $S = \{i(1),\ldots,.i(n)\}$.

Again, proceeding from left to right, the customer facilities are partitioned into sets $G_1, ..., G_n$, splitting these facilities along with their corresponding split demands among consecutive sets as required, so that the total demand in set G_k equals $s_{i(k)}$, for $k = 1,\ldots, n$. For each new facility k, treating the demand in G_k as that to be served by this facility, the single facility squared-Euclidean problem is solved (see Francis et al., 1990, for a closed-form solution). This yields a set of locations for the facilities, which can be evaluated by solving the associated transportation problem. The foregoing process is repeated by proceeding similarly along the y-axis using horizontal slices. The better of the two resulting locations is selected as a starting solution for the alternating procedure.

For intermediate nodes in the branch-and-bound tree, having solved the lower bounding problem (7.3), we use the corresponding locations obtained to initialize the alternating procedure. Naturally, we keep a track of the best incumbent solution to be used for fathoming purposes throughout the tree.

7.2.4 Partitioning Scheme

Given a current node of the branch-and-bound tree and its associated subproblem, the algorithm selects a variable $w_{pq} \in J^F$ in order to partition the problem into two subproblems associated with designating w_{pq} either to be a positive integer, i.e., $J^+ \leftarrow J^+ \cup \{(p, q)\}$ or to be zero, i.e., $J^0 \leftarrow J^0 \cup \{(p, q)\}$. For the subproblem associated with $w_{pq} > 0$, l_{pq} is initialized as

1 and then by the logical tests, the lower and the upper bonds are tightened (for all affected variables). Note that by fixing w_{pq} to be positive, a new link is introduced or added to the current forest graph. This results in connecting two components of the forest, and therefore, in order to keep the current forest cycle-free, the arcs of the cut-set corresponding to these two components (other than (p,q)) are zero-fixed.

In order to select the particular arc (p,q) for branching as above, we adopt the following strategy. Having solved the lower bounding problem, and having applied the alternating heuristic by starting with the resulting solution as described in Section 7.2.3, let $(\tilde{x}, \tilde{y}, \tilde{w})$ be the solution thus obtained of objective value $\tilde{v}$. Note that $\tilde{w}$ satisfies the current bounding restrictions. The branching variable w_{pq} is then determined as

$$(p,q) \in \arg \operatorname{lex} \max_{(i,j) \in J^F} \{\tilde{w}_{ij}, c_{ij}(u_{ij} - l_{ij})\}. \tag{7.4}$$

7.2.5 Other Features of the Branch-and-Bound Algorithm

Search Strategy

A depth first (LIFO) discrete partitioning strategy DP is adopted in the binary search in which a partial solution list, *PS*, records the history of branching that has been performed in the active branch-and-bound tree, using the framework of Geoffrion (1967). A $+(i,k)$ in *PS* indicates that w_{ik} is positive-fixed, a $-(i,k)$ in *PS* indicates that $w_{ik} = 0$, and an underlined $\underline{\pm(i,k)}$ indicates that the descendants of the corresponding complementary branch have been fathomed. Note that if at any node, the logical tests determine that $l_{pq} > 0$ for $(p, q) \in J^F$, then we transfer (p,q) from J^F to J^+, add it to the current forest, and augment *PS* by $\underline{+(p, q)}$. Similarly, if we

determine that $u_{pq} = 0$ for some $(p, q) \in J^F$, then we transfer (p,q) from J^F to J^0 and augment *PS* by $\underline{-(p, q)}$.

Optimality Criterion

In order to avoid excessive computations involved in sifting through alternative solutions or close to global optimal solutions, the following fathoming criterion is adopted:

$$v_{LB} \geq (1 - \varepsilon')v^*, \tag{7.5}$$

where $0 < \varepsilon' < 1$ (we used $\varepsilon' = 0.001$), v_{LB} is a lower bound on the objective function value computed at the current node of the branch-and-bound tree, and v^* is the current incumbent solution value. Accordingly, at termination of the algorithm, the global minimum of EDLAP is within $\varepsilon' \bullet 100\%$ of the current best solution.

7.2.6 Summary of the Branch-and-Bound Algorithm

Step (0) Initialization. Set $PS = \varnothing$, $J^+ = \varnothing$, $J^0 = \varnothing$, $J^F = \{(i, j) : i = 1,\ldots, n, j = 1,\ldots, m\}$. Set $l_{ij} = 0$, $u_{ij} = \min \{s_i, d_j\}$, $\forall (i, j) \in J^F$, and apply the logical tests of Section 7.2.1 to update these bounds. Accordingly, update J^+, J^0, J^F, and *PS* as necessary, (see the foregoing discussion), and transfer to Step 3.

Step (1) Cycle Prevention. Let (p,q) be the arc last added to J^+, connecting components C_p and C_q in the current forest. Let CC_{pq} be the cut-set arcs between C_p and C_q, other than (p,q). Set $u_{ij} = 0 \ \forall \ (i, j) \in CC_{pq} - J^0$ and accordingly, update J^0, J^F, and *PS*. Proceed to Step 2.

Step (2) Logical Tests. Perform the logical tests of Section 7.2.1.

(a) If infeasibility is detected in this process, go to Step 5.

(b) If this process sets $l_{ik} > 0$ for any $(i, k) \in J^F$, update J^+, J^F and *PS*, as

$J^+ \leftarrow J^+ \cup \{(i,k)\}$, $J^F \leftarrow J^F - \{(i, k)\}$, and $PS \leftarrow PS \cup \{+(\underline{i,k})\}$. Return to Step 1.

(c) If this process sets $l_{ik} = u_{ik} = 0$ for any $(i, k) \in J^F$, update J^0, J^F, and *PS*, as

$J^0 \leftarrow J^0 \cup \{(i,k)\}$, $J^F \leftarrow J^F - \{(i, k)\}$, and $PS \leftarrow PS \cup \{-(\underline{i,k})\}$. Continue with the logical tests on the other arcs in a sequential loop.

Once no scan eligible arc exists for performing logical tests, if $|PS|$ is less than the total number of variables, then proceed to Step 3. Otherwise, we have $w_{ik} = l_{ik} = u_{ik} \; \forall \; (i, k)$. Solve the corresponding location subproblem, update the incumbent if necessary, and go to Step 5.

Step (3) Bounding Step. Compute a lower bound v_{LB} (e.g., $v_{LB} = v_{LB1}$ of Equation (7.3)) and compute an upper bound as in Section 7.2.3 to update the incumbent solution and its value v^*. If $v_{LB} \geq (1 - \varepsilon')v^*$ (Equation (5)), then transfer to Step 5. Otherwise, proceed to Step 4.

Step (4) Branching Step. Select the branching variable w_{pq}, where $(p, q) \in J^F$, using the rule (4). Update J^+, J^F and *PS*, as $J^+ \leftarrow J^+ \cup \{(p,q)\}$, $J^F \leftarrow J^F - \{(p, q)\}$, and $PS \leftarrow PS \cup \{(p,q)\}$. Accordingly, set $l_{pq} = 1$, and return to Step 1.

Step (5) Fathoming Step. Let (*p*,*q*) be the right-most non-underlined entry in *PS* such that the lower bound computed when incrementing *PS* by (*p*,*q*) is less than $(1 - \varepsilon')v^*$. If such an entry does not exist, then stop; the incumbent solution is within 100 ε' % of optimality. Replace (*p*,*q*)

by $\underline{-(p, q)}$ in *PS*, and delete all elements to its right. Update J^+, J^0, and J^F according to the revised partial selection list *PS*. Since at least one positive-fixed variable has changed its status, the current forest is no longer valid. Hence, reconstruct the affected portion of the forest. Set $l_{ij} = 1 \; \forall \; (i, j) \in J^+$, $l_{ij} = 0, \; \forall (i, j) \in J^0 \cup J^F$, and $u_{ij} = \min\{s_i, d_j\} \; \forall \; (i, j)$, and return to Step 2 in order to revise these bounds via the logical tests.

7.3. A Reformulation-Linearization Technique for Computing Enhanced Lower Bounds

In the foregoing section, we have presented a general branch-and-bound algorithmic framework for solving EDLAP in which any suitable lower bounding scheme can be inserted in lieu of that described in Section 7.2.2. As an alternative to (7.3), we now describe an enhanced lower bounding procedure that is motivated by the Reformulation-Linearization Technique (RLT) of Sherali and Tuncbilek (1992b) for solving polynomial programming problems. Our purpose here is to study the tradeoff between a quicker versus a more involved, but stronger, lower bounding procedure with respect to the overall effort for solving this problem.

Toward this end, let us begin by restating Problem EDLAP as follows.

$$\text{Minimize} \left\{ \sum_i \sum_j c_{ij} w_{ij} \alpha_{ij} : \alpha_{ij} = \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \; \forall \; (i, j), w \in W \right\} \tag{7.6}$$

where W defined by (7.2) is henceforth assumed to contain the (*current*) variable bounds $l_{ij} \le w_{ij} \le u_{ij} \; \forall \; (i, j)$. We will now construct a lower bounding linear programming relaxation for the corresponding node subproblem (7.6) by generating appropriate supports for the nonlinear constraints, and by using RLT to derive a suitable approximation for the convex envelope of the bilinear objective function over a superset of the feasible region.

7.3.1 Relaxed Representation for the Nonlinear Constraints

Wendell and Hurter (1973) have shown that a set of optimal locations for the new facilities lies in the convex hull *Z* of the set of customer locations (a_j, b_j), $j = 1,...,m$. The set *Z* can be computed efficiently in O (*m* log *m*) time using Graham's scanning algorithm (see Manber, 1989, for example) which commences by detecting a vertex of *Z*, ordering the other customer locations with respect to angles subtended with a reference axis, and then determining facets of *Z* while sweeping through these locations in this order, adding and deleting vertices sequentially in linear time. Hence, let vert (*Z*) be the set of (known) vertices of *Z* and let us impose the valid constraints

$z_i = (x_i, y_i) \in Z \quad \forall\ i = 1,...,n$

where *Z* is defined by a set of *K* facetial inequalities of the form

$$\Psi_{1k} x + \Psi_{2k} y \le \Psi_{0k} \text{ for } k = 1,2,...,K. \tag{7.7}$$

Now, noting that the maximum of the convex distance function α_{ij} defined in (7.6) over *Z* occurs at a vertex of *Z*, and by virtue of the relationships between the Euclidean and the rectilinear norms, we derive the following sets of constraints implied by (7.6).

$$0 \le \alpha_{ij} \le u_{\alpha_{ij}}, \text{ where } u_{\alpha ij} = \underset{z \in \text{vert}(Z)}{\text{maximum}} \left\| z - (a_j, b_j) \right\| \forall\ i, \text{ for each facility } j \tag{7.8a}$$

$$\alpha_{ij} \ge \max\{|x_i - a_j|, |y_i - b_j|\} \forall i, j \tag{7.8b}$$

$$\sqrt{2}\ \alpha_{ij} \ge \left|x_i - a_j\right| + \left|y_i - b_j\right| \ \forall\ i, j. \tag{7.8c}$$

In addition, we impose the following valid inequality, where υ_1 is obtained by solving the stated location problem having unit weights.

$$\sum_i \sum_j \alpha_{ij} \geq \upsilon_1, \text{ where } \upsilon_1 = \min \left\{ \sum_i \sum_j \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \right\}. \tag{7.9}$$

Note that each of (7.8b) and (7.8c) can be replaced by four equivalent linear inequalities. Moreover, the minimum value of α_{ij} feasible to these constraints matches the exact value given by (7.6) along the *x* and *y*-axes as well as along the 45°-bisectors of the four quadrants of the coordinate system centered at $(a_j, b_j)\ \forall\, j$.

Remark 7.1. Observe that we could also impose the convex constraints

$$u_{\alpha_{ij}} \alpha_{ij} \geq (x_i - a_j)^2 + (y_i - b_j)^2 \ \ \forall\, i, j \tag{7.10}$$

based on (7.8a). This would additionally represent the exact relationship (7.6) along the boundary of the circle of radius $u_{\alpha_{ij}}$ centered at $(a_j, b_j)\ \forall\, j$. However, in our experience, we had difficulties with respect to the robustness of both the commercial software packages MINOS 5.4 and GRG2 in solving the resulting convex relaxations. Hence, we omit (7.10) in the development below and work with only linear relaxations. On the other hand, linear tangential supports to the convex function α_{ij} can be constructed at judiciously selected points, such as at incumbent location decisions when these locations do not already lie on the skeletal grid formed by the axes and the quadrant bisectors centered at the customer locations as referred to above. □

Linearizing (7.8a) and (7.8b), and including (7.7), (7.8a) and (7.9), along with $w \in W$, we obtain the following relaxation of (7.6).

$$\text{Minimize} \sum_i \sum_j c_{ij} w_{ij} \alpha_{ij} \tag{7.11a}$$

subject to

$$\alpha_{ij} \geq x_i - a_j, \alpha_{ij} \geq a_j - x_i, \alpha_{ij} \geq y_i - b_j, \alpha_{ij} \geq b_j - y_i \ \forall\, i, j \qquad (7.11b)$$

$$\sqrt{2}\ \alpha_{ij} \geq (x_i - a_j + y_i - b_j), \sqrt{2}\ \alpha_{ij} \geq x_i - a_j + b_j - y_i$$

$$\sqrt{2}\ \alpha_{ij} \geq a_j - x_i + y_i - b_j, \sqrt{2}\ \alpha_{ij} \geq a_j - x_i + b_j - y_i \ \forall\, i, j \qquad (7.11c)$$

$$\Psi_{1k} x_i + \Psi_{2k} y_i \leq \Psi_{0k} \ \forall\, i, k \qquad (7.11d)$$

$$\sum_i \sum_j \alpha_{ij} \geq \upsilon_1 \qquad (7.11e)$$

$$0 \leq \alpha_{ij} \leq u_{\alpha_{ij}} \ \forall\, i, j,\ w \in W\,. \qquad (7.11f)$$

7.3.2 Relaxed Representation for the Bilinear Objective Function

In a spirit similar to (7.9), let us begin by introducing two types of location-based valid inequalities that are updated for each node subproblem in the branch-and-bound process. The first of these is based on the incumbent solution w^* of objective value υ^* for Problem EDLAP and is given as follows

$$\sum_i \sum_j c_{ij} w^*_{ij} \alpha_{ij} \geq \upsilon^*\,. \qquad (7.12)$$

The second inequality is similar in concept to (7.12), but is based on a particular allocation $\bar{w}$ and its objective value $\bar{\upsilon}$ (determined by solving n separable Euclidean location problems), being given by

$$\sum_i \sum_j c_{ij} \bar{w}_{ij} \alpha_{ij} \geq \bar{\upsilon}, \text{ where,} \qquad (7.13a)$$

$$l_{ij} \leq \bar{w}_{ij} \leq u_{ij} \ \forall\, i, j \ \text{ and } \bar{\upsilon} = \min \left\{ \sum_i \sum_j c_{ij} \bar{w}_{ij} \sqrt{(x_i - a_j)^2 + (y_i - b_j)^2} \right\}. \qquad (7.13b)$$

Remark 7.2. The particular choice of $\bar{w}$ not only plays a role in tightening the relaxation, but as we shall see later in Proposition 7.1, is critical in ensuring the convergence of lower and upper

bounds at leaf nodes of the enumeration tree. Accordingly, we select $\bar{w}$ as follows. At the initial node, let $\bar{w}$ correspond to the incumbent solution obtained via the heuristic procedure described in Section 7.2.3. Hence, (7.13) would coincide with (7.12) for this case. At any intermediate node obtained by incrementing a partial solution, $\bar{w}$ and $\bar{v}$ are taken as the allocation solution and its objective value obtained by fixing $(x,y) = (\tilde{x},\tilde{y})$, and solving the resulting transportation problem over the *current* set *W*, where $(\tilde{x},\tilde{y})$ corresponds to the heuristic upper bounding location solution obtained at the parent node as described in Section 7.2.4. On the other hand, if this intermediate node is obtained via a backtracking process following a fathoming step, then we perform the foregoing step with $(\tilde{x},\tilde{y})$ fixed at the current incumbent solution's location (since the parent node's heuristic solution is not presently at hand). □

Next we generate a set of constraints using the Reformulation-Linearization Technique (RLT) concept as follows.

Reformulation Step. The following quadratic valid constraints are generated based on the products of the stated pairs of inequalities (written in the form $\{\bullet\} \geq 0$), or based on products of equations with variables.

(a) Multiply each *bound-factor* inequality in $0 \leq \alpha_{ij} \leq u_{\alpha_{ij}}$ with each *bound-factor* inequality in $l_{ij} \leq w_{ij} \leq u_{ij}, \ \forall\, i,j$. (For example, the *bound-factor* inequalities corresponding to the latter restrictions are given by $(w_{ij} - l_{ij}) \geq 0$ and $(u_{ij} - w_{ij}) \geq 0, \ \forall\, i, j$.)

(b) Multiply each inequality in (7.11b), (7.11c), and (7.11d) with each correspondingly indexed bound-factor inequality $l_{ij} \leq w_{ij} \leq u_{ij}$ for each i, j.

(c) Using the constraints defining *W*, generate the *equality* product constraints $\left(\sum_j w_{ij} - s_i\right) x_i = 0$ and $\left(\sum_j w_{ij} - s_i\right) y_i = 0 \ \forall\, i$.

Linearization Step. Linearize the resulting product constraints by substituting

$$\gamma_{ij} = w_{ij}\alpha_{ij},\ \theta_{ij} = w_{ij}x_i, \text{ and } \phi_{ij} = w_{ij}y_i\ \forall\ i,j\ . \tag{7.14}$$

This produces the following linear programming lower bounding problem, as a further relaxation of (7.11), and will be referred to as RLT(Ω) based on the hyperrectangle $\Omega = \{w\colon\ l_{ij} \le w_{ij} \le u_{ij},\ \forall\ (i,j)\}$. Here, the reformulation steps (a), (b), and (c) respectively generate the constraints (7.15b), (7.15c,d,e), and (7.15f), upon using the substitution (7.14). It is easy to verify that the constraints (7.11b,c, and d) are themselves implied by these RLT constraints. The remaining constraints below include (7.11e), (7.11f), (7.12), and (7.13a).

$$\text{RLT}(\Omega)\colon \text{Minimize } \sum_i \sum_j c_{ij}\gamma_{ij} \tag{7.15a}$$

subject to

$l_{ij}\alpha_{ij} \le \gamma_{ij} \le u_{ij}\alpha_{ij}$, and

$$u_{\alpha ij}w_{ij} + \alpha_{ij}u_{ij} - u_{\alpha ij}u_{ij} \le \gamma_{ij} \le u_{\alpha ij}w_{ij} + \alpha_{ij}l_{ij} - u_{\alpha ij}l_{ij}\ \forall\ i,j \tag{7.15b}$$

$$\theta_{ij} - w_{ij}a_j + l_{ij}(\alpha_{ij} - x_i + a_j) \le \gamma_{ij} \le \theta_{ij} - w_{ij}a_j + u_{ij}(\alpha_{ij} - x_i + a_j),$$

$$w_{ij}a_j - \theta_{ij} + l_{ij}(\alpha_{ij} + x_i - a_j) \le \gamma_{ij} \le w_{ij}a_j - \theta_{ij} + u_{ij}(\alpha_{ij} + x_i - a_j),$$

$$\phi_{ij} - w_{ij}b_j + l_{ij}(\alpha_{ij} - y_i + b_j) \le \gamma_{ij} \le \phi_{ij} - w_{ij}b_j + u_{ij}(\alpha_{ij} - y_i + b_j),$$

$$w_{ij}b_j - \phi_{ij} + l_{ij}(\alpha_{ij} + y_i - b_j) \le \gamma_{ij} \le w_{ij}b_j - \phi_{ij} + u_{ij}(\alpha_{ij} + y_i - b_j)$$

$$\forall\ (i,j) \tag{7.15c}$$

$$\theta_{ij} + \phi_{ij} - w_{ij}(a_j + b_j) + l_{ij}\left(\sqrt{2}\ \alpha_{ij} - x_i + a_j - y_i + b_j\right) \le$$

$$\sqrt{2}\ \gamma_{ij} \le \theta_{ij} + \phi_{ij} - w_{ij}(a_j + b_j) + u_{ij}\left(\sqrt{2}\ \alpha_{ij} - x_i + a_j - y_i + b_j\right)$$

$$\theta_{ij} - \phi_{ij} + w_{ij}(-a_j + b_j) + l_{ij}\left(\sqrt{2}\ \alpha_{ij} - x_i + a_j + y_i - b_j\right) \leq$$

$$\sqrt{2}\ \gamma_{ij} \leq \theta_{ij} - \phi_{ij} + w_{ij}(-a_j + b_j) + u_{ij}\left(\sqrt{2}\ \alpha_{ij} - x_i + a_j + y_i - b_j\right),$$

$$\phi_{ij} - \theta_{ij} + w_{ij}(a_j - b_j) + l_{ij}\left(\sqrt{2}\ \alpha_{ij} + x_i - a_j - y_i + b_j\right) \leq$$

$$\sqrt{2}\ \gamma_{ij} \leq \phi_{ij} - \theta_{ij} + w_{ij}(a_j - b_j) + u_{ij}\left(\sqrt{2}\ \alpha_{ij} + x_i - a_j - y_i + b_j\right),$$

$$-\phi_{ij} - \theta_{ij} + w_{ij}(a_j + b_j) + l_{ij}\left(\sqrt{2}\ \alpha_{ij} + x_i - a_j + y_i - b_j\right) \leq$$

$$\sqrt{2}\ \gamma_{ij} \leq -\phi_{ij} - \theta_{ij} + w_{ij}(a_j + b_j) + u_{ij}\left(\sqrt{2}\ \alpha_{ij} + x_i - a_j + y_i - b_j\right),$$

$$\forall\ (i, j) \qquad (7.15d)$$

$$w_{ij}\Psi_{0k} - u_{ij}(\Psi_{0k} - \Psi_{1k}x_i - \Psi_{2k}y_i) \leq \Psi_{1k}\theta_{ij} + \Psi_{2k}\phi_{ij} \leq w_{ij}\Psi_{0k} - l_{ij}(\Psi_{0k} - \Psi_{1k}x_i - \Psi_{2k}y_i)$$

$$\forall\ (i, j, k) \qquad (7.15e)$$

$$\sum_j\ \theta_{ij} - s_i x_i = 0 \text{ and } \sum_j\ \phi_{ij} - s_i y_i = 0\ \ \forall\ i \qquad (7.15f)$$

$$\sum_i\ \sum_j\ \alpha_{ij} \geq \nu_1 \qquad (7.15g)$$

$$\sum_i\ \sum_j\ c_{ij} w^*_{ij} \alpha_{ij} \geq \nu^* \qquad (7.15h)$$

$$\sum_i\ \sum_j\ c_{ij} \bar{w}_{ij} \alpha_{ij} \geq \bar{\nu} \qquad (7.15i)$$

$$w \in W,\ 0 \leq \alpha \leq u_\alpha. \qquad (7.15j)$$

Remark 7.3. Note that while we have stated the general relaxation RLT(Ω) above for any Ω corresponding to a node subproblem, several constraints might be null/implied depending on the current bounds on *w*, and should therefore be omitted. Specifically, given the sets J^+, J^0, and J^F for the current node, define the set $J' = \{(i, j): (i, j) \in J^+ \text{ and } l_{ij} = u_{ij} = w_{ij}\}$. If $J^0 \cup J' \neq \emptyset$, then for every $(i, j) \in J^0 \cup J'$, we have $u_{ij} - w_{ij} = 0$ and $w_{ij} - l_{ij} = 0$. Hence, the constraints

(15b,c,d,e) of Problem RLT(Ω) that are obtained by the multiplication of the corresponding bound-factors $(u_{ij} - w_{ij})$ and $(w_{ij} - l_{ij})$ with other inequalities will be null, and should therefore be dropped from the bounding problem. In addition, for each $(i, j) \in J'$, noting (14), we set $\gamma_{ij} \equiv \alpha_{ij}\, l_{ij}$ in the bounding problem RLT(Ω). □

Furthermore, for this restricted node problem, we include another class of valid constraints, developed by Sherali and Tuncbilek (1992a), which are implied by the requirement that the current forest represented by the arcs of J^+ be cycle-free. In other words, for every pair of components C_p and C_q of the current forest, at most one arc in the cut-set

$$CC_{pq} = [\{(i, j): i \in C_p, j \in C_q\} \cup \{(i, j): i \in C_q, j \in C_p\}] \cap J^F \qquad (7.16a)$$

can be positive-fixed. If the cardinality of CC_{pq} is at least two, we can impose the following constraint for each such pair of components.

$$\sum_{(i,j) \in CC_{pq}} w_{ij} \le u(CC_{pq}), \text{where } u(CC_{pq}) = \max\{u_{ij}: (i, j) \in CC_{pq}\}. \qquad (7.16b)$$

Remark 7.4. The overall branch-and-bound algorithm described in Section 7.2.6 remains the same except that at the bounding step (Step 3), in lieu of using $v_{LB} = v_{LB1}$, we use

$$v_{LB} = \max\{v_{LB1}, v_{LB2}\}, \text{ where } v_{LB2} \equiv \text{optimal value of RLT}(\Omega). \qquad (7.17)$$

(See Proposition 7.2 below for a relationship between v_{LB1} and v_{LB2}.) Moreover, when computing upper bounds at intermediate nodes at Step 3, in lieu of the strategy described in Section 7.2.3, we commence the alternating heuristic with the allocation produced via the solution of Problem RLT(Ω). Furthermore, we use the following branching strategy, where $\hat{w}$ is the optimal allocation obtained via the solution of RLT(Ω).

$$(p, q) \in \arg\max_{(i,j) \in J^F} \{(u_{ij} - l_{ij}) \min\{\hat{w}_{ij} - l_{ij}, u_{ij} - \hat{w}_{ij}\}\}. \qquad (7.18a)$$

As alternatives to (18a), in addition to (4), we can also use variants of the above strategy as stated below.

$$(p,q) \in \arg\max_{(i,j)\in J^F} \{c_{ij} \min\{\hat{w}_{ij} - l_{ij}, u_{ij} - \hat{w}_{ij}\}\}. \tag{7.18b}$$

$$(p,q) \in \arg\max_{(i,j)\in J^F} \{\min\{\hat{w}_{ij} - l_{ij}, u_{ij} - \hat{w}_{ij}\}\}. \tag{7.18c}$$

We provide some comparative results for (7.18a) versus (7.4), (7.18b) and (7.18c) in Section 5.

Remark 7.5. In order to provide for a robust and effective bounding mechanism via RLT(Ω), it is expeditious to scale the original problem data. Letting $x_i = sx_i'$ and $y_i = sy_i' \; \forall \, i$, for some $s > 0$, Problem EDLAP becomes

Minimize $\sum_i \sum_j c_{ij} w_{ij} \sqrt{\left(x_i' - \frac{a_j}{s}\right)^2} + \sqrt{\left(y_i' - \frac{b_j}{s}\right)^2}$

which is an equivalent location-allocation problem having customers at locations $(a_j / s,\ b_j / s) \; \forall \, j$. If the original coordinates $(a_j, b_j) \; \forall \, j$ lie in the tightest enclosing box $0 \le x \le \bar{a},\ 0 \le y \le \bar{b}$, that is oriented along the coordinate axes (possibly after a shift of the origin), we select $s = \sqrt{\bar{a}^2 + \bar{b}^2}$ and scale the customer location coordinates by replacing (a_j, b_j) by $(a_j / s,\ b_j / s) \; \forall \, j$. □

Now, consider the following result which motivates the role of (7.15i) and validates the use of RLT(Ω) as a lower bounding mechanism in the proposed branch-and-bound algorithm. In particular, it also asserts that at feasible leaf nodes of the total enumeration tree at which the logical tests assure that $l_{ij} = u_{ij} \; \forall \, i, j$, the lower bound produced by RLT(Ω) would equal the upper bounding value of the corresponding feasible solution to EDLAP, hence inducing a potentially useful fathoming process.

Proposition 7.1. Suppose that Problem RLT(Ω), where $\Omega = \{w\colon l_{ij} \le w_{ij} \le u_{ij}\ \forall\ i,j\}$, is formulated with a constraint (7.15i) for some $\overline{w} \in W$ such that $\overline{w}_{ij} = l_{ij}$ or $\overline{w}_{ij} = u_{ij}\ \forall\ i,j$. Furthermore, suppose that $w = \overline{w}$ in an optimal solution to Problem RLT(Ω). Then $\overline{w}$, along with the corresponding locations $(\overline{x},\overline{y})$ that solve for $\overline{v}$ in (7.13b), yield an optimal solution for the node location-allocation subproblem, with the lower bound (optimal value of RLT(Ω)) equaling the upper bound for this node problem. In particular, this situation occurs when $l_{ij} \equiv u_{ij}\ \forall\ i,j$.

Proof. Clearly, $\overline{v}$ is an upper bound for the node subproblem corresponding to the feasible solution $\overline{w} \in W$. Hence, given that $\overline{w}$ satisfies the bound-value condition and that it solves RLT(Ω), we need to show that

$$\sum_i \sum_j c_{ij}\bar{\gamma}_{ij} = \overline{v} \tag{7.19}$$

holds true, where $\bar{\gamma}$ is the optimal value of γ obtained for RLT(Ω). Toward this end, consider (7.15b) and let $\alpha = \overline{\alpha}$ at the given optimum to RLT(Ω). Note that these constraints state that

$$l_{ij}\ \overline{\alpha}_{ij} \le \bar{\gamma}_{ij} \le u_{ij}\ \overline{\alpha}_{ij}, \text{ and} \tag{7.20a}$$

$$u_{ij}\ \overline{\alpha}_{ij} - u_{\alpha_{ij}}(u_{ij} - \overline{w}_{ij}) \le \bar{\gamma}_{ij} \le (\overline{w}_{ij} - l_{ij})u_{\alpha_{ij}} + l_{ij}\ \overline{\alpha}_{ij}\ \forall\ i,j. \tag{7.20b}$$

Hence, if $\overline{w}_{ij} = l_{ij}$, then (7.20) yields $l_{ij}\ \overline{\alpha}_{ij} \le \bar{\gamma}_{ij} \le l_{ij}\ \overline{\alpha}_{ij}$, and if $\overline{w}_{ij} = u_{ij}$, then (7.20) yields $u_{ij}\ \overline{\alpha}_{ij} \le \bar{\gamma}_{ij} \le u_{ij}\ \overline{\alpha}_{ij}$. In either case, we get $\bar{\gamma}_{ij} \equiv \overline{w}_{ij}\ \overline{\alpha}_{ij}$. Therefore, by (7.15i), we obtain $\sum_i \sum_j c_{ij}\bar{\gamma}_{ij} = \sum_i \sum_j c_{ij}\overline{w}_{ij}\overline{\alpha}_{ij} \ge \overline{v}$ while $\sum_i \sum_j c_{ij}\bar{\gamma}_{ij} \le \overline{v}$ since $\overline{v}$ is an upper bound on the node subproblem and the optimal value of RLT(Ω) is a lower bound on this problem. This implies that (7.19) holds true. Finally, since $\overline{w} \in W$ by the selection prescribed in Remark 7.2, the

stated condition is trivially satisfied in the special case when $l_{ij} = u_{ij} \forall i, j$. This completes the proof. □

Proposition 7.1 motivates the dynamic derivation of (7.15i) in the formulation of RLT(Ω) where an estimate $\overline{w}$ is derived for an optimum solution to the node subproblem and is used to formulate this constraint. To relate the bounds υ_{LB1} and υ_{LB2} produced by (7.3) and RLT(Ω), respectively, consider the following result.

Proposition 7.2. Suppose that Problem RLT(Ω) is formulated with $\overline{w}_{ij} = l_{ij}$ in (15i), where $\overline{v}$ is determined accordingly via (7.13b). Then

$$\upsilon_{LB2} \equiv v[RLT(\Omega)] \geq \upsilon_{LB1} \equiv \overline{v} \tag{7.21}$$

where $v[\bullet]$ denotes the optimal value of any problem $[\bullet]$, and υ_{LB1} is given by (7.3).

Proof. Given that $\overline{w}_{ij} = l_{ij} \ \forall \ i, j$ and since $c_{ij} \geq 0 \ \forall \ i, j$, we have by (7.15b) and (7.15i) that $\sum_i \sum_j c_{ij}\gamma_{ij} \geq \sum_i \sum_j c_{ij} l_{ij} \alpha_{ij} \geq \overline{\upsilon} \equiv \upsilon_{LB1}$. Hence, $\upsilon_{LB2} \equiv \upsilon[\text{RLT}(\Omega)] \geq \upsilon_{LB1}$, and this completes the proof. □

Proposition 7.2 asserts that if we select $\overline{w}_{ij} \equiv l_{ij} \ \forall \ i, j$ in (7.15i), the lower bound obtained via RLT(Ω) is at least as good as that obtained via (7.3). However, Proposition 7.1 motivates a preferable selection for $\overline{w}$ as in Remark 7.2 rather than taking $\overline{w} \equiv l$. In our computational results, we will comment on the relative values of υ_{LB1} versus υ_{LB2}.

7.4. Alternative Branching Scheme for the Euclidean Distance Location-Allocation Problem

The lower bounding problem RLT(Ω) can be embedded in any suitable branch-and-bound procedure to solve Problem EDLAP globally to a specified percentage tolerance of optimality. In the previous section, we have discussed a finite discrete partitioning procedure DP that implicitly

enumerates the vertices of *W*. In this section, we consider a finitely convergent continuous partitioning procedure CP that uses the logical tests of Section 3 for tightening any given set of flow intervals, and that employs, a best-first tree enumeration strategy, similar to the infinitely convergent process described in Sherali and Tuncbilek [1992b]. Here again, each branch-and-bound node principally differs in the specification of the hyperrectangle Ω. The hyperrectangle associated with node *t* of the branch-and-bound tree at the main iteration or stage *S* of the procedure is denoted by $\Omega^{S,t} = \{w: l^{S,t} \le w \le u^{S,t}\}$. In our implementation of the branch-and-bound procedure, we successively partition the hyperrectangle defined by the initial bounds $\Omega^{1,1} \equiv \Omega$ on the flow variables into smaller and smaller hyperrectangles. At any stage *S* of the branch-and-bound algorithm, we have at hand a set of active or nonfathomed nodes denoted as T_S. We then select an active node *t** in T_S that has the least lower bound, breaking ties arbitrarily, and we partition the hyperrectangle associated with this node according to a suitable branching variable selection strategy that ensures convergence of the overall procedure to a global optimum for Problem EDLAP. (One such strategy is recommended below.) This process continues by solving the bounding problems for the resulting two node subproblems, and fathoming nodes as permissible based on this analysis. Whenever the set of active nodes is empty, the process terminates.

Branching Variable Selection Strategy: The prescribed choice of a branching variable arc (*r*, *s*) attempts to reduce the gap in the relaxation as motivated by Proposition 1 by selecting

$$(r, s) \in \arg\max_{(i,j)} \{c_{ij}(u_{ij} - l_{ij})\}. \qquad (7.22)$$

We then partition the interval $[l_{rs}, u_{rs}]$ into two sub-intervals as $[u_{rs}, \lfloor (l_{rs}+u_{rs})/2 \rfloor]$ and $[\lceil (l_{rs}+u_{rs})/2 \rceil, u_{rs}]$, noting that its is sufficient to consider integral values of flow bounds. (Note

that if $(l_{rs}+u_{rs})/2$ is integral, we take the value $\lfloor (l_{rs}+u_{rs})/2 \rfloor$ to be the next lower integral value.) As an alternative to (7.22), we could use (18a) to select a branching variable arc. However, it was found that such a strategy was useful only in the initial stages of branching, because the flow values tend to be driven close to either their lower or upper bounding interval end-points as the algorithm progresses, thereby making the process stall using this strategy. This would be case even for the DP scheme using (7.18a). Hence, we need a mechanism that allows us to switch to (7.22) whenever the situation described above occurs. Toward this end, we use (7.18a) whenever the flow value $\hat{w}_{pq}$ is sufficiently interior to its bounding interval as ascertained via

$$\min\{\ \hat{w}_{pq} - l_{pq}, u_{pq} - \hat{w}_{pq}\} \geq \mu (u_{pq} - l_{pq}), \text{ where } \mu \in (0, 0.5). \tag{7.23}$$

A value of 0.1 was chosen for μ based on computational tests, although any value for μ in the range (0, 0.5) would guarantee convergence of the overall procedure. Note that a choice of μ ≥ 0.5 corresponds to simply using (7.22) to select the branching variable arc. While this modification did not result in a significant improvement in results for the DP scheme, it proved to be relatively more beneficial to the CP scheme and we provide some computational experience towards this in Section 7.6.

Branch-and-Bound Algorithm

Step 0. Initialization Step

Select an optimality percentage tolerance 100ε% (0<ε<1). Set the stage counter S = 1, and let the set of active nodes be T_1={1}. Determine a set of initial flow bounds Ω using the logical tests and cycle prevention tests described in Section 3. If the allocation Ω is a singleton, then the solution to the location problem solves Problem EDLAP. Otherwise, use the alternating heuristic

procedure for the initial node to obtain an initial incumbent/global upper bound GUB. Let $\overline{w}$ represent the flow allocation vector corresponding to the incumbent solution. Denote the initial hyperrectangle as $\Omega^{1,1} \equiv \Omega$. Set $t^* = 1$ and solve the node-zero relaxation problem RLT($\Omega^{1,1}$). Initialize the global lower bound GLB_1 to the optimal value $LB_{1,1}$, say, thus obtained. Furthermore, by fixing the resulting locations (x_i, y_i) obtained by solving the lower bounding problem, apply the alternating heuristic. Also, starting with the allocations obtained via the lower bounding solution, re-apply the alternating heuristic. Update the global upper bound GUB and the incumbent solution using the resulting solutions. If $GLB_1 \geq (1-\varepsilon)$GUB stop; the GUB value is within 100(1-ε) % of optimality. Otherwise, determine the branching variable index (r, s) using (7.18a,7. 22, 7.23).

Step 1. Partitioning Step

Having identified the active node (S, t^*) to be partitioned, and given the choice (r, s) of the branching variable, partition this node into two sub-nodes associated with the two hyperrectangles Ω^{S, t_1} and Ω^{S, t_2} that are identical to Ω^{S, t^*} except that the two respective interval restrictions on w_{rs} are modified according to the prescribed branching strategy (18a or 22). Beginning with the sub-node t_1, apply the logical tests of Section 2. If the current partition induces cycles in the forest, or if the maximum slack is negative, then fathom this sub-node because the current partition is either invalid or infeasible. Otherwise, solve RLT(Ω^{S, t_1}), with the allocation vector $\overline{w}$ being obtained as prescribed by Remark 2, and denote its objective value by LB_{S, t_1}. If the condition of Proposition 1 holds true, then fathom the corresponding node (after updating the incumbent solution if necessary) as this solution solves the corresponding node subproblem. Use the lower bounding location and flow solution to obtain an upper bound UB_{S, t_1} via the alternating heuristic. If $UB_{S,}$

t_1<GUB, update GUB←UB_S, t_1 along with the associated incumbent solution. Also, determine and store a branching variable index (r, s) if this node is to be later selected for further partitioning. Repeat this process for the node subproblem corresponding to t_2, similar to t_1. Update the set of active branch-and-bound nodes for stage S+1 to $T_{S+1} = (T_S - \{t^*\}) \cup \{t_1, t_2\} - \{t : LB_{S,t} \geq GUB(1-\varepsilon)\}$. (Note that $LB_{S+1,t}$ now refers to the respective lower bounds found for the various active nodes $t \in S$+1.) Increment S by 1.

Step 2. Node Selection Step

If $T_S = \phi$ then stop; the incumbent solution is optimal (within the ε-tolerance). Else, select an active node (S, t^*) where $t^* \in \text{argmin}\{LB_{S,t} : t \in T_S\}$. Set $GLB_S = LB_{S,t^*}$. Note that this is the least lower bound over the active nodes at stage S. Return to Step 1.

7.5. Extension to the Case of l_p Distances

Consider the l_p distance location-allocation problem given by

$$\boldsymbol{l_p}\textbf{-LAP}: \text{Minimize} \left\{ \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} w_{ij} \left\{ \left| x_i - a_j \right|^p + \left| y_i - b_j \right|^p \right\}^{1/p} : w \in W \right\}, \qquad (7.24)$$

when $p \geq 1$. As shown by Love and Juel (1982), typical values of p that are of practical interest lie in the interval [1,2], and can be empirically estimated for different geographical regions. The branch-and-bound algorithm of Section 7.2 using the projected location space bounding scheme (PLSB) can be applied identically to (7.24), with the Euclidean norm being replaced by the l_p norm throughout. The related l_p distance location problems can be solved using the procedure analyzed by Brimberg and Love (1993).

Likewise, the RLT based procedure is also applicable as derived in Section 7.3, with α_{ij} now representing the l_p distance between (x_i, y_i) and $(a_j, b_j)\ \forall\ (i, j)$, and with the factor $\sqrt{2}$ in constraints (7.15d) being replaced by $2^{(p-1)/p}$. The latter claim is supported by the following result, which generalizes (7.8c) and establishes a similar relationship as that between (7.6) and (7.8) as discussed in Section 7.3.1.

Proposition 7.3. For any $p > 1$, the following constraints are valid inequalities

$$2^{(p-1)/p}\alpha_{ij} \geq \left|x_i - a_j\right| + \left|y_i - b_j\right| \ \forall\ i, j. \tag{7.25}$$

Moreover, equality holds true in (7.25) whenever $\left|x_i - a_j\right| = \left|y_i - b_j\right|$.

Proof. The result is trivially true if $\left|x_i - a_j\right| = 0$ or $\left|y_i - b_j\right| = 0$. Hence suppose that both these quantities are positive. Since $f(z) = z^p$ is convex in z over the region $z > 0$ for any $p > 1$, we get for any $z_1 > 0$, $z_2 > 0$ that $f(z_1/2 + z_2/2) \leq [f(z_1) + f(z_2)]/2$. Using $z_1 \equiv \left|x_i - a_j\right|$ and $z_2 = \left|y_i - b_j\right|$, this gives

$$\left[\frac{\left|x_i - a_j\right| + \left|y_i - b_j\right|}{2}\right]^p \leq \frac{\left|x_i - a_j\right|^p + \left|y_i - b_j\right|^p}{2}$$

which establishes (7.25) upon simplification. Moreover, noting that equality holds in the convexity relationship whenever $z_1 = z_2$, this completes the proof. □

The remainder of the procedure is identical to that for EDLAP, with the obvious replacement of Euclidean with l_p distance location subproblems throughout, including in the constraints (7.9), (7.12), and (7.13) that are used to formulate (7.15g, h, i), respectively. In the

following section, we provide some computational experience for this generalized problem l_p-LAP using various values of $p > 1$.

7.6. Computational Experience

In this section, we begin by presenting computational results on a collection of twenty documented test problems, as well as five randomly generated problems for the Euclidean distance case. The former set include the twelve instances (Problems 1-12) for which the data is given in Al-Loughani (1997), and the two instances (Problems 13 and 14) constructed by Selim (1979). The remaining six test cases of this set (Problems 15-20) each have 10 customer locations whose demands and locations are the same as for the first ten customers in Problem 12, the c_{ij} values for $i = 1,...,n,\ j = 1,...,m \equiv 10$ are taken as the first $10n$ c_{ij}-values given for Problem 12, while the supply values are specified in Table 7.1. The five randomly generated instances of EDLAP were used to study the algorithmic performance on cases where only a pair of new facilities need to be located, similar to the study conducted in Chen, et al. (1998) for the uncapacitated version of this problem. Accordingly, we generate five problems having $n = 2$ and m ranging from 10 - 50. The data for these problem instances were generated as follows. The customer locations $(a_j,\ b_j)$ were distributed uniformly within a rectangle whose bottom-left corner coincided with the origin. The values of c_{ij} and d_j were generated using an exponential distribution. (It was observed that uniformly distributed values for these parameters, or values generated via distributions having low variance, resulted in a high level of degeneracy in the problem.) The source capacities for the two source nodes were generated as

$u\sum_j D_j$ and $(1-u)\sum_j D_j$, where u is a uniformly distributed random variable between 0 and 1. Based on the parameters (m, n), we classify test problems that have $m + n \leq 12$ to be small-sized,

test problems having $12 < m + n \leq 15$ to be medium-sized, and the other test problems to be large-sized.

Table 7.2 presents computational results on a mix of small to large-sized problems for the projected location space bounding procedure (PLSB) described in Section 7.2 that uses the lower bound given by (7.3) along with branching strategy (7.4), as well as for the enhanced RLT-based lower bounding scheme given by (7.17) along with branching strategy (7.18a), as described in Section 7.3. An optimality tolerance of $\varepsilon' = 0.001$ in Equation (7.5) was used for these runs. A branch-and-bound node limit of 5000 was imposed for all runs using the RLT based bounds, and a node limit of 10,000 was set when using the relatively quicker, but weaker, PLSB bounding procedure.

For the other (larger) test cases that could not be solved within the set computational limits, a modified branching scheme that uses a heuristic backtracking step was used to obtain good quality solutions. This heuristic procedure is similar in all respects to the exact procedure except that the backtracking step is performed in a slightly different manner. Here, upon fathoming of a node, instead of choosing the rightmost nonunderlined entry $(p, q) \in J^P$ in the partial solution list that lies within $(1-\varepsilon)$ of the best upper bound for further branching, we impose an additional condition that the chosen entry in the list should correspond to setting a positive restriction $w_{pq} > 0$, and should have no additional allocations fixed (positive or zero) by logical tests or cycle prevention tests when this occurs. Such a restriction helps in heuristically limiting the length of the tree by focussing on certain key periodic branching steps, thereby enabling a relatively greater breadth of the branch-and-bound tree to be explored as compared with a premature termination of the exact procedure. The results for this procedure are presented in Table 7.3.

The efficiency of the alternative branching strategies (7.18b), (7.18c) and (7.4) are explored using the larger test cases in concert with the heuristic backtracking process described above, and these results are presented in Table 7.4. Note that a similar relative performance was observed for these branching strategies when using the exact branch-and-bound procedure as well.

Table 7.5 provides computational results for the exact solution of the l_p distance based location-allocation problem using a subset of the test problems run under the same computational limit as above, and Table 7.6 provides results obtained using the aforementioned modified heuristic backtracking procedure on the other (larger) sized problems. The branching rule (7.18a) was used for the RLT runs. Also, a value of $p = 1.647$ was used, which happens to be the average of the empirical values of p determined for various cities in the world by Love and Walker (1993). Furthermore, Table 7.7 presents comparative computational results for the l_p distance case using various values of p in the range [1, 2] for some selected problems from the test set.

Finally, Table 7.8 presents results for all the Euclidean distance cases of the test problems using the alternative continuous partitioning scheme CP described in Section 7.4, and Table 7.9 provides some computational experience on some selected problems using the RLT based allocation partitioning procedure that is further enhanced by adding tangential supporting hyperplanes for the distance function.

All computations were performed on a SUN ULTRA 1 workstation having 256 Megabytes of RAM. The procedure of Brimberg and Chen (1998) was used to solve the location subproblems since it was observed to provide more accurate results when run with a fairly tight termination tolerance as compared with the Hyperboloid approximation procedure

(see Eyster et al. 1973). The package CPLEX 6.0 was used to solve the linear programming relaxations RLT(Ω). In each of the tables, the CPU time required to the solve the problem within the desired optimality tolerance, the number of branch-and-bound nodes enumerated to solve the problem, and the initial and final objective values are recorded. In addition, the initial lower bounds (achieved at the initial node of the branch-and-bound tree) using the PLSB and RLT procedures are recorded in Table 7.2, while Table 7.4 records the node number at which the final solution was obtained. Table 7.8 records the final lower bound obtained after the procedure was terminated when either a solution lying within the optimality tolerance of $\varepsilon' = 0.001$ is found, or a node limit of 5000 is reached. In addition, Tables 7.9 and 7.10 record the final optimality gap for each test problem.

Examining Table 7.2, the PLSB-based bounding (exact) procedure was able to obtain optimal solutions to small problems of size (n, m) ranging up to (3, 9) - (2, 10), while the medium-sized problems remained unsolved. On the other hand, the RLT-based bounding (exact) procedure was able to obtain optimal solutions for all small and medium sized problems ranging in size (n, m) = (4, 8) - (2, 20) within the set limits. In addition, the computational effort for the RLT approach was comparable to that of the PLSB approach for the smaller problems, and relatively lesser for the medium sized problems ($12 < m + n \leq 15$). Furthermore, note that the optimal solutions found by the RLT based approach for the only two available test problems in the literature (due to Selim) yielded solutions having objective values of 2.0136 and 3.0549, in contrast with the respective values of 2.1 and 7.8 reported by Selim. In this case, the RLT approach was able to obtain optimal solutions with a relatively lesser computational effort when compared with the PLSB-based procedure.

When the heuristic backtracking rule was implemented (Table 7.3), the PLSB bounding procedure was able to obtain good solutions to problems of size (n, m) ranging up to (4, 8) - (5, 10). On the other hand, the RLT procedure was able to obtain better solutions when compared with the PLSB approach for problems ranging in size (n, m) = (5, 20) - (10, 10) within the set limits. Because of the relatively stronger bounds, the RLT approach was able to explore more regions of the enumeration tree, terminating within the set limits for all the problems in Tables 7.2 and 3 except for Problem #12 having a size of (n, m) = (5, 30). In general, both the PLSB and RLT-based procedures considerably improved upon the solution obtained by the alternating heuristic applied at the initial node. A reduction in the objective value of the alternating heuristic solution of more than 80% was observed in some cases. On the average, the PLSB-based exact and heuristic procedures were able to reduce the initial objective value by 30% and 31%, respectively. On the other hand, the RLT-based exact and heuristic procedures were able to reduce the same by 34% and 35%, respectively, within the prescribed node-limit. As might be expected, even when the heuristic backtracking procedure terminated within this node limit, the gap between the final global lower bound and the best found solution value remained significant for the large test cases because of the unexplored nodes that were skipped in the branch-and-bound tree. For example, for Problem # 20, the final global lower bound value was equal to 2200, while the incumbent solution value was 7764.

Test cases 21-25 (see Tables 7.2 and 7.3) are instances of Problem EDLAP that are concerned with locating a pair of facilities to serve a relatively large number of customers. The results indicate that problems of this type having up to m = 30 customers can be solved to optimality using the RLT approach. In most of the cases, the alternating heuristic and/or the upper bounding heuristic applied after solving the lower bounding problem at the initial node

yielded the optimal solution. Note that much larger instances of the simpler uncapacitated cases of these problems can be solved relatively more quickly (Chen et al., 1998).

The RLT-based algorithmic heuristic and exact procedures were also implemented using branching strategies (7.18b), (7.18c), and (7.4), as alternatives to (7.18a). However, as evident from Table 7.4, while (7.18c) yielded a better performance than (7.18b) and (7.4), these alternative branching strategies were less effective when compared with (7.18a). (While only the results for the heuristic backtracking procedure are shown, the results are relatively similar for the exact procedure.) The computational efficiency appeared to depend on how early the best solution is found by the particular branching strategy. In all cases tested using strategy (7.18a), the branch-and-bound node that yielded the best solution was discovered within thirty CPU minutes. Hence, among the DP schemes, this strategy is recommended for use in the branching procedure, either when solving a problem to optimality or when running the method as a heuristic procedure.

Tables 7.5 and 7.6 present results for the l_p distance cases for a value of $p = 1.647$ using the exact and heuristic backtracking procedures, respectively. The generalized l_p distance PLSB and RLT procedures were used to solve these problems. (For the heuristic backtracking approach, both the procedures are run with a node limit of 5,000.) Branching strategy (18a) is again used for the RLT procedure. The relative effectiveness of these procedures with respect to the quality of bounds and solutions, and the number of nodes enumerated, are similar to that observed for the Euclidean distance case. However, the overall CPU expense for the l_p distance case is usually somewhat more, as expected. Also, with regard to the correlation between computational effort and the value of p used, it can seen from Table 7.7 that there was no significant difference in the number of nodes enumerated for different values of $p \in [1, 2]$.

Although this experiment was done using the heuristic backtracking procedure for convenience, it is an indication of the sensitivity of the fathoming efficiency of the bounds with respect to the value of *p*, and a similar effort was observed on some trial cases using the exact RLT procedure.

Table 7.8 presents results for all the test problem instances of EDLAP using the alternative continuous partitioning (CP) scheme along with the branching rule (7.22). Since the DP approaches generated lower bounds close to the initial lower bounds in most of the larger test cases, the quality of the final solution is not guaranteed in such cases. A noteworthy contribution of the CP procedure is to obtain proven (near) global optimal solutions to such test problems. The results indicate that the CP scheme obtains optimal solutions to the smaller and medium sized problems, but at a relatively higher computational cost when compared with any of the DP schemes. However, the CP scheme obtains relatively better solutions for the larger problems when compared with the DP schemes, and in fact, discovered good quality solutions to two larger sized problems that lie within 10% of global optimality (Problems 16 and 20). On the average, the CP scheme reduces the initial objective value by 35%, and reduces the optimality gap in several cases to within 10% of optimality under a node limit of 5000, hence affording an improved proof of optimality than the previous partitioning scheme. Also, a large proportion of the computational effort for this procedure occurred during the tail-end of the convergence process.

Another improvement that empirically appears to benefit the CP scheme relatively more than the DP scheme is the tightening of the lower bounding relaxation. For example, the addition of tangential supports for the distance variables α_{ij} can help tighten the lower bound. These supporting hyperplanes can be generated as stated below for any demand location *j*, based on the fact that the Euclidean distance function is convex and lies above its first-order approximation.

$$\alpha_{ij} \geq g_j(\hat{x},\hat{y}) + \nabla g_j(\hat{x},\hat{y}) \bullet (x_i - \hat{x}, y_i - \hat{y}) \; \forall j = 1,...,m, (\hat{x},\hat{y}) \neq (a_j, b_j), \text{ for each } i = 1,...,n, \tag{7.26}$$

where $g_j(x,y) = \sqrt{(x-a_j)^2 + (y-b_j)^2} \forall j = 1,...,m.$ For each α_{ij}, the points of support $(\hat{x},\hat{y})$ can be taken as the coordinates of the new facility locations corresponding to the current incumbent solution, after perturbing these coordinates if any of these points coincide with the coordinates of the demand location under consideration. A total of n^2m such constraints can be generated (n inequalities for each α_{ij}). Alternatively, the points of support used to generate the constraints can be taken as the coordinates of the other demand locations. In this case, a total of $nm(m\text{-}1)$ inequalities can be added to the lower bounding formulation. While the number of additional constraints added are more than the former case, the latter procedure seems to yield better lower bounds in practice and was used in our lower bounding scheme.

The introduction of such tangential supports results in tighter lower bounds, albeit a relatively larger sized relaxation. However, as seen in Table 7.9, empirical tests show that this enhancement results in up to a 5% reduction in the number of branch-and-bound nodes enumerated for problems that were solved to optimality in Table 7.8. Using this strategy, an additional larger sized problem (#17) was solved to within 10% optimality.

Table 7.10 presents results for the CP approach using the alternative branching rule (7.18a-7.22, 7.23), along with the additional supporting constraintsh (7.26) being used to enhance the lower bound. It can be seen from these results that the use of this branching rule for the CP approach yields significantly better lower bounds when compared with the use of simply (7.22). Several larger sized problems are now solved to within 10.2% of optimality (Problems 9, 16, 17, and 18). In fact, Problem #20 is now solved to within 3.4% of global optimality. Hence,

the branching rule (7.18a-7.22, 7.23) is a preferable strategy for Problem EDLAP under the CP scheme.

7. 7. Summary and Conclusions

In this paper, we have presented a global optimization approach for solving capacitated Euclidean and l_p distance location-allocation problems. This class of problems is notoriously difficult because of the nonconvexity and the nondifferentiability of the objective function. In fact, there exists only one previous attempt at devising a non-exhaustive exact solution method for the case of Euclidean distances (Selim, 1979), but as illustrated in the foregoing section, this algorithm had actually failed to solve the two test problems having five customers and five new facilities as presented therein.

We have developed the framework of a finitely convergent branch-and-bound procedure that conducts an enumeration in the allocation space, and that incorporates specialized logical tests and cut-set based valid inequalities in order to exploit the special structures of the underlying transportation constraint set. This framework permits the use of any lower bounding scheme coordinated with a branching variable selection strategy. We have described two such bounding procedures, one based on solving a special projected location space subproblem (PLSB), and the other based on a linear programming relaxation generated by applying a special variant of the Reformulation-Linearization Technique (RLT), and including several additional objective function based valid inequalities derived in the location space. Some alternative partitioning strategies have also been explored. The results indicate that the proposed methods offer a promising and viable approach for this challenging class of problems. The PLSB-based procedure, when terminated within a 10000 node limit or the PLSB-based heuristic backtracking

procedure when terminated within a 5000 node limit, is capable of obtaining optimal solutions to small-sized problems, and reasonably good approximate solutions to problems having the number of facilities (n) and customers (m) ranging up to (10, 10) and (2, 50) within 60 CPU minutes on a SUN Ultra 1 workstation. On the other hand, for obtaining more accurate solutions, the RLT-based approach can be used to solve small and medium-sized problems to optimality having (n,m) ranging up to (5, 10) and (2, 30) within the same limitations. For larger problems, the RLT-based heuristic procedure (with a node limit of 5000) can be used to obtain good solutions. On the average, a 34% improvement in the reduction of the objective value of the initial solution was obtained by the RLT-based approaches, as compared with a 30% improvement obtained using the PLSB-based approaches. The continuous partitioning approach CP is a viable alternative branching procedure. While this procedure is somewhat slower than the foregoing approaches, on the average, it is able to reduce the initial objective value by 35% (36% with branching rule (7.18a-7.22, 7.23) and using (7.26)), while also producing significantly tighter global lower bounds at termination in most cases, thereby providing an improved verification of near optimality. Hence, it is suggested that this approach be used when solving Problem EDLAP and l_p-LAP to provable (near) optimality. If the particular application requires only quick, near-optimal solutions to these problems, then the RLT-based and PLSB-based heuristic backtracking schemes are recommended for use. Note, however, that the CP procedure can also be effective in such situations by running the algorithm with larger optimality tolerances (say, 10% or 20%).

A further enhancement might be possible by using an efficient Lagrangian relaxation procedure to derive quick approximate dual solutions to RLT(Ω) for deriving lower bounds. Alternative lower bounding mechanisms could also be explored using other types of convex

relaxations (see Remark 1), and global optimization methods (see Horst and Tuy, 1993, for a general exposition on this subject). In addition, we also recommend the development of more effective methods for solving the pure location problem, and heuristic procedures for solving the location-allocation problem as topics for future research. The test-bed of twenty instances described in this study will hopefully serve to be useful for the purpose of evaluating both exact and heuristic methods for this class of problems.

Table 7.1. Supply Values for Test Problems 15-20.

Problem	n	Supplies $s_i,\ i = 1,\ldots,n$
15	5	180, 2, 20, 42, 100
16	6	18, 40, 22, 152, 2, 110
17	7	8, 40, 12, 162, 2, 90, 30
18	8	102, 90, 50, 40, 30, 22, 2, 8
19	9	8, 10, 22, 112, 2, 90, 20, 50, 30
20	10	8, 10, 22, 197, 2, 10, 2, 60, 30, 3

Table 7.2. Computational Results for Problem EDLAP using the Exact Branch-and-Bound Procedure.

Problem	(n,m)	Initial υ^*	PLSB Approach				RLT Approach Branching Rule (7.18a)			
			Initial LB	Final υ^*	# Nodes Enum	CPU Time	Initial LB	Final υ^*	# Nodes Enum	CPU Time
1	(2,2)	0	0	0	1	0.04 sec	0	0	1	0.04 sec
2	(2,4)	310.04	116.53	247.28	5	0.10 sec	216.03	247.28	7	0.20 sec
3	(2,4)	375.58	0	214.37	18	0.50 sec	109.07	214.37	19	0.90 sec
4	(3,5)	24.0	0	24.0	111	3.20 sec	6.72	24.0	22	2.30 sec
5	(3,5)	462.48	0	73.96	116	4.10 sec	7.61	73.96	13	2.00 sec
6	(3,9)	240.19	0	221.40	1244	55.60 sec	0	221.40	313	66.4 sec
7	(3,9)	1017.78	0	871.62	1076	57.30 sec	46.60	871.62	173	42.2 sec
8	(4,8)	802.49	0	609.23	10000+	14 min	0	609.23	601	6 min
13	(5,5)	14.89	0	2.01	564	45.56 sec	0	2.01	19	9.14 sec
14	(5,5)	20.85	0	3.05	312	25.53 sec	0.84	3.05	18	10.77 sec
15	(5,10)	5893.38	0	3206.20	10000+	13 min	157.20	2595.47	302	8 min
21	(2, 10)	5125.33	0	5099.27	231	16.1 sec	2927	5099.27	68	19.2 sec
22	(2, 20)	12694	0	12694.29	10000+	24 min	2658	12694.29	356	9 min
23	(2, 30)	2558029.0	0	2558029	10000+	21 min	785911.0	2558029.0	706	35 min

Legend: υ^* = incumbent objective value

Table 7.3. Computational Results for Problem EDLAP using the Heuristic Branching Procedure.

Problem	(n,m)	Initial υ^*	PLSB based Heuristic Procedure				RLT based Heuristic Procedure Branching Rule (7.18a)			
			Initial LB	Final υ^*	# Nodes Enum	CPU Time	Initial LB	Final υ^*	# Nodes Enum	CPU Time
9	(5,15)	11228.85	0	8215.55	10000+	23 min	1366.34	8169.79	302	23 min
10	(5,20)	25948.27	0	12935.65	10000+	13 min	2462.75	12846.87	1110	134 min
11	(5,20)	1790.18	0	1582.62	10000+	29 min	0	1107.18	1339	73 min
12	(5,30)	31785.26	0	25292.8	10000+	31 min	0	23990.04	5000+	316 min
16	(6,10)	8837.29	0	8045.11	10000+	21 min	0	7797.21	258	9 min
17	(7,10)	11549.59	0	7142.96	10000+	21 min	0	6974.77	268	14 min
18	(8,10)	3800.75	0	2179.44	10000+	31 min	0	1576.83	2272	120 min
19	(9,10)	3911.34	0	3403.93	10000+	21 min	0	3250.68	1367	12 min
20	(10,10)	8923.45	0	8040.60	10000+	30 min	2000.70	7764.05	231	19 min
24	(2, 40)	1936290	0	1936290	10000+	26 min	1256080	1936290	1969	172 min
25	(2, 50)	2510535	0	2510535	10000+	29 min	1685820	2510535	663	115 min

Table 7.4. Comparative Computational Results for Various Branching Strategies using the RLT-based Heuristic Approach.

Problem	Branching Rule (7.18a)		Branching Rule (7.18b)		Branching Rule (7.18c)		Branching Rule (7.4)	
	# Nodes Enum	Optimal Node	# Nodes Enum	Optimal Node	# Nodes Enum	Optimal Node	# Nodes Enum	Optimal Node
1	1	1	1	1	1	1	1	1
2	2	1	2	1	3	1	3	1
3	13	1	14	1	11	1	13	1
4	6	1	4	1	6	1	17	1
5	10	1	7	1	15	1	18	1
6	15	5	49	31	39	14	52	5
7	32	6	80	26	12	4	86	17
8	118	8	301	65	171	20	489	277
9	302	4	231	9	150	25	2392	235
10	1110	255	5000+	2672	2145	510	5000+	4777
11	1339	206	5000+	3678	1708	290	5000+	1056
12	5000+	357	5000+	640	5000+	521	5000+	613
13	19	3	29	14	48	14	180	157
14	18	5	31	16	24	16	46	26
15	105	74	249	116	102	78	646	440
16	258	105	509	1338	325	148	1928	378
17	268	5	945	115	242	14	3945	23
18	2272	166	5000+	118	4914	3756	5000+	894
19	1367	782	5000+	3765	1776	313	5000+	656
20	231	39	519	196	1091	75	5000+	2437

Table 7.5. Computational Results for Problem l_p-LAP using the Exact Branch-and-Bound Procedure.

Problem	(n,m)	Initial υ^*	PLSB Approach				RLT Approach Branching Rule (18a)			
			Initial LB	Final υ^*	# Nodes Enum	CPU Time	Initial LB	Final υ^*	# Nodes Enum	CPU Time
1	(2,2)	0	0	0	1	0.03 sec	0	0	1	0.04 sec
2	(2,4)	324.83	122.35	260.30	7	0.20 sec	256.18	260.30	7	0.50 sec
3	(2,4)	394.15	0	220.26	20	0.70 sec	113.98	220.26	19	1.20 sec
4	(3,5)	24.00	0	24.00	35	2.20 sec	6.86	24.00	31	3.90 sec
5	(3,5)	494.69	0	78.32	9	0.88 sec	7.91	78.32	17	3.30 sec
6	(3,9)	232.09	0	229.88	513	50.1 sec	0	229.88	210	92 sec
7	(3,9)	1076.09	0	892.81	1003	126 sec	49.10	892.81	212	71 sec
8	(4,8)	850.04	0	642.09	6733	28 min	0	642.09	660	8 min
13	(5,5)	15.39	0	2.09	97	24.0 sec	0	2.09	26	14.70 sec
14	(5,5)	13.77	0	3.27	345	69.7 sec	0.86	3.27	42	23.50 sec
15	(5,10)	5953.01	0	2801.51	10000+	40 min	158.20	2739.29	359	10 min

Table 7.6. Computational Results for Problem l_p-LAP using the Heuristic Branching Procedure.

Problem	(*n*,*m*)	Initial υ^*	PLSB Approach				RLT Approach Branching Rule (7.18a)			
			Initial LB	Final υ^*	# Nodes Enum	CPU Time	Initial LB	Final υ^*	# Nodes Enum	cpu (sec)
9	(5,15)	11987.67	0	8486.50	5000+	10 min	1380.60	8445.77	106	10 min
10	(5,20)	27530.00	0	13519.44	5000+	15 min	2604.68	13415.23	1052	75 min
11	(5,20)	1421.28	0	1357.81	5000+	29 min	0	1152.90	1267	120 min
12	(5,30)	32783.88	0	25271.83	5000+	33 min	0	24544.57	5000+	412 min
16	(6,10)	8911.14	0	8391.68	5000+	21min	0	8131.43	158	7 min
17	(7,10)	12075.10	0	7297.67	5000+	25 min	0	7297.67	242	15 min
18	(8,10)	3843.88	0	2218.54	5000+	30 min	0	1703.63	2515	61 min
19	(9,10)	4132.44	0	3443.52	5000+	51 min	0	3323.30	1389	30 min
20	(10,10)	9294.79	0	8483.18	5000+	43 min	2114.02	8378.11	255	24 min

Table 7.7. Comparative Computational Results for Various Values of *p* using the RLT-based Heuristic Procedure.

Problem #	*p*	Final υ^*	CPU Time (sec)	Nodes
8	1.00	793.00	107.30	150
	1.25	710.20	114.49	123
	1.50	661.90	126.71	136
	1.75	630.72	141.94	165
	2.00	609.23	91.31	118
9	1.00	9619.00	419.13	90
	1.25	8998.93	998.44	154
	1.50	8646.61	588.50	97
	1.75	8350.95	806.43	133
	2.00	8169.79	1396.74	302
13	1.00	2.55	16.53	29
	1.25	2.29	30.14	50
	1.50	2.15	13.49	21
	1.75	2.07	12.23	19
	2.00	2.01	11.27	22
14	1.00	4.26	17.80	18
	1.25	3.72	22.38	17
	1.50	3.40	10.97	17
	1.75	3.20	13.56	18
	2.00	3.05	11.61	17
15	1.00	3427.00	158.00	84
	1.25	3046.07	254.37	109
	1.50	2827.55	294.39	112
	1.75	2689.12	215.42	106
	2.00	2595.47	184.76	105

Table 7.8. Computational Results for the Euclidean Distance Case using the RLT-based CP Approach with Branching Rule (7.22).

Problem	(*n*,*m*)	Initial υ^*	Final LB	Final υ^*	Optimality Gap %	# Nodes Enum	CPU Time
1	(2,2)	0	0	0	< 0.1	1	0.06 sec
2	(2,4)	310.04	247.28	247.28	< 0.1	4	0.40 sec
3	(2,4)	375.58	214.37	214.37	< 0.1	24	2.30 sec
4	(3,5)	24.0	24.00	24.00	< 0.1	35	5.80 sec
5	(3,5)	462.48	73.80	73.96	< 0.1	31	6.10 sec
6	(3,9)	240.19	221.40	221.40	< 0.1	555	5 min
7	(3,9)	1017.78	870.64	871.63	< 0.1	467	5 min
8	(4,8)	802.49	608.10	609.23	< 0.1	2350	30 min
9	(5,15)	11228.85	7039.97	8169.79	13.8	5000+	407 min
10	(5,20)	25948.27	9871.30	12857.45	23.2	5000+	732 min
11	(5,20)	1790.18	762.20	1107.18	31.2	5000+	210 min
12	(5,30)	31785.26	3443.47	23990.04	85.6	5000+	1443 min
13	(5,5)	14.89	2.01	2.01	< 0.1	94	42.80 sec
14	(5,5)	20.85	3.10	3.10	< 0.1	89	50.30 sec
15	(5,10)	5893.38	2580.18	2595.47	0.6	5000+	128 min
16	(6,10)	8837.29	6943.02	7797.21	11.0	5000+	188 min
17	(7,10)	11549.59	863.87	1627.67	46.9	5000+	315 min
18	(8,10)	3800.75	6277.26	6967.90	9.9	5000+	200 min
19	(9,10)	3911.34	1550.67	3250.68	52.3	5000+	365 min
20	(10,10)	8923.45	7110.99	7719.00	7.9	5000+	462 min

Table 7.9. Computational Results on selected problems for the Euclidean distance case using the RLT-based CP Approach with Branching Rule (7.22) and enhanced by Additional Supporting Constraints (7.26).

Problem	Final LB	Final υ^*	Optimality Gap %	# Nodes Enum	CPU Time
3	214.37	214.37	< 0.1	24	2.6 sec
4	24.00	24.00	< 0.1	35	6.6 sec
5	73.80	73.80	< 0.1	23	5.9 sec
6	221.34	221.40	< 0.1	455	7 min
7	870.79	871.63	< 0.1	315	5 min
8	608.60	609.23	< 0.1	2325	40 min
9	7288.88	8169.79	10.8	5000+	672 min
10	10127.29	12882.32	21.4	5000+	1630 min
11	772.56	1107.18	30.2	5000+	1673 min
13	2593.64	2595.47	< 0.1	239	11 min
14	2.01	2.01	< 0.1	94	56 sec
15	3.09	3.09	< 0.1	51	33 sec
16	7043.34	7797.21	9.7	5000+	263 min
17	6387.14	6974.78	8.4	5000+	334 min
18	865.65	1627.67	46.8	5000+	423 min
19	2231.02	3250.68	31.4	5000+	480 min
20	7246.0	7719.79	6.1	5000+	564 min

Table 7.10. Computational results on selected problems for the Euclidean distance case using the RLT-based CP Approach with Branching Rule (7.18a-7.22, 7.23) and Enhanced by Additional Supporting Constraints (7.26).

Problem	Final LB	Final υ^*	Optimality Gap %	# Nodes Enum	CPU Time
3	214.37	214.37	< 0.1	12	1.2 sec
4	24.00	24.00	< 0.1	15	2.7 sec
5	73.80	73.80	< 0.1	21	5.2 sec
6	221.40	221.40	< 0.1	127	2 min
7	871.63	871.63	< 0.1	130	2 min
8	609.23	609.23	< 0.1	1011	18 min
9	7337.34	8169.79	10.2	5000+	718 min
13	2595.47	2595.47	< 0.1	71	3 min
14	2.01	2.01	< 0.1	69	36 sec
15	3.09	3.09	< 0.1	17	8.3 sec
16	7111.26	7797.21	8.8	5000+	271 min
17	6437.84	6967.90	7.6	5000+	349 min
18	1031.29	1564.46	34.1	5000+	468 min
19	2264.42	3250.68	30.3	5000+	479 min
20	7448.67	7719.79	3.5	5000+	567 min