Theory and Methodology

An exact method for the two-echelon, single-source, capacitated facility location problem

Suda Tragantalerngsak a, John Holt a, Mikael Rönnqvist b,*

a Mathematics Department, The University of Queensland, Brisbane, Qld., Australia
b Department of Engineering Science, The University of Auckland, Auckland, New Zealand

Received 30 September 1997; accepted 3 February 1999

Abstract

Facility location problems form an important class of integer programming problems, with applications in the telecommunication, distribution and transportation industries. In this paper we are concerned with a particular type of facility location problem in which there exist two echelons of facilities. Each facility in the second echelon has limited capacity and can be supplied by only one facility in the first echelon. Each customer is serviced by only one facility in the second echelon. The number and location of facilities in both echelons together with the allocation of customers to the second-echelon facilities are to be determined simultaneously. We propose a Lagrangian relaxation-based branch and bound algorithm for its solution. We present numerical results for a large suite of test problems of realistic and practical size. These indicate that the method is efficient. It provides smaller branch and bound trees and requires less CPU time as compared to LP-based branch and bound obtained from a 0–1 integer package. © 2000 Elsevier Science B.V. All rights reserved.

Keywords: Branch and bound; Facilities; Integer programming; Optimization

1. Introduction

Facility location problems are concerned with choosing the best location for facilities from a given set of potential sites so as to minimize the total cost while satisfying customer demand. This total cost is the sum of the fixed costs of opening facilities and the variable costs of assigning customers to the facilities. The literature in this area contains a wide range of variants and extensions, see for example Aikens [1], Brandeau and Chiu [6] for an overview.

In the uncapacitated facility location problem, each facility is assumed to have no limit on its capacity. In this case, each customer receives all its demand from exactly one facility. An extension of the uncapacitated facility location problem, in which two echelons of facilities are involved, is
called the two-echelon uncapacitated facility location problem. In this problem, the deliveries are made from the first-echelon facilities (such as plants or depots) to customers via the second-echelon facilities (such as warehouses). The objective is to determine the number and location of facilities in each echelon, the flow of products between the facilities in different echelons and the assigning of the customers to facilities in the second echelon.

In the capacitated facility location problem, each facility has a limit capacity. A special case of this problem, in which each customer receives supply from exactly one facility, is called the single-source capacitated facility location problem. It has been studied by several authors, including Barcelo and Casanovas [3], Sridharan [28], Klincewicz and Luss [16], Pirkul [22], Beasley [5], Holmberg et al. [14] and Rönnqvist et al. [23].

In this paper, we consider a particular model of facility location problems, called the two-echelon, single-source, capacitated facility location problem (TSCFLP). This is an extension of the single-source, capacitated facility location problem in that there exist two echelons of facilities. Second-echelon facility has a limited capacity and can be supplied by only one second-echelon facility, and each customer is serviced by only one facility. Applications of this model arise in several areas.

Telecommunication is one of the fastest growing industries in the world. Within this area, location problems occur in many different circumstances. One important class of applications is when equipment of different kinds is to be installed, it could be concentrators, see Pirkul [22], or other end equipment for optical fibers. The units to locate can be of very different sizes, from small electronic equipment to switching cabinets and exchange buildings. The units (i.e. facilities) to be installed are almost always capacitated, and the single sourcing property is also quite usual, i.e. the demand unit (i.e. customer) is connected to only one facility. A problem which involves decision about concentrators, cabinets and customers in a hierarchical structure is studied by Williams [31]. This is a project done for a major telecommunication company and is equivalent to the two-echelon problem considered in this paper.

Another area is distribution systems in which there are potentially multiple warehouses or depots from which vehicles in a fleet can operate. The model simultaneously determines how many depots and vehicles are needed, the location of the open depots, which vehicles should operate from which open depots and which customers each vehicle should service. A typical example is mail/box deliveries where there are several depots for the mail service company and where each depot has several delivery vans.

Lagrangian relaxation, combined with subgradient optimization, is one of the most widely used approaches in determining the solution to hard and large combinatorial optimization problems. Details and applications of Lagrangian relaxation can be found in e.g. Fisher [9] and Geoffrion [13]. It has been a widely used technique in solving several variants of the facility location problem, including the uncapacitated facility location, see Galvão and Raggi [11], the capacitated facility location problem, see Cornuejols et al. [8], Christofides and Beasley [7] and Beasley [5], and the single-source, capacitated facility location problem, see Klincewicz and Luss [16], Pirkul [22], Sridharan [28] Beasley [5] and Holmberg et al. [14].

Kaufman et al. [15] and Ro and Tcha [25] have proposed branch and bound (B&B) based algorithms to locate plants and warehouses in two-level distribution systems. Gao and Robinson [12] modified the uncapacitated facility location problem to the two-echelon uncapacitated facility location problem and implemented a dual-based B&B method. In these models, there are no capacity constraints and each facility and each customer can receive their supply/demand from multiple sources.

Tragantalerngsak et al. [29] consider six different Lagrangian heuristics for solving the TSCFLP. All heuristics are implemented and the results show that the Lagrangian-based heuristics provide lower bounds that are much better than those obtained from a Linear Programming (LP) relaxation. Generation of feasible solutions to obtain upper bounds can also be done efficiently. This provides an indication that a B&B algorithm based on Lagrangian relaxation could constitute an efficient approach. This is what could be expected
from such an approach, see Holmberg et al. [14] for such an example.

In this paper, we present a B&B algorithm for the TSCFLP based on the most efficient Lagrangian heuristic found in [29]. The proposed algorithm consists of three stages based on the model characteristics inherent in the mathematical model. The outline of the paper is as follows. In Section 2 we formulate the mathematical model for TSCFLP. The Lagrangian relaxation used, generation of lower and upper bounds and subgradient optimization is briefly described in Section 3. In Section 4 we discuss general B&B procedures for optimization is briefly described in Section 3. In Section 4 we discuss general B&B procedures for the problem, some reduction tests used and the three-staged B&B procedure that is used. We describe our test problems and discuss the numerical results in Section 5. A discussion about aspects of the implementation is carried out in Section 6 and finally in Section 7 we make some conclusions.

2. Mathematical model

To formulate the TSCFLP problem, we introduce the following notation. Let

- \( I = \{1, \ldots, m\} \), the set of potential facilities,
- \( J = \{1, \ldots, n\} \), the set of customers,
- \( K = \{1, \ldots, o\} \), the set of potential depots,
- \( a_j \) = demand of customer \( j \), \( \forall j \in J \),
- \( b_i \) = capacity of facility \( i \), \( \forall i \in I \),
- \( f_{ik} \) = cost of assigning facility \( i \) to depot \( k \),
- \( \forall i \in I, \ k \in K \),
- \( c_{ijk} \) = cost of facility \( i \) from depot \( k \) servicing customer \( j \), \( \forall i \in I, \ j \in J, \ k \in K \),
- \( g_k \) = cost of setting a depot at location \( k \),
- \( \forall k \in K \),
- \( y_{ik} = \begin{cases} 1 & \text{if facility } i \text{ is open and is served from depot } k, \forall i \in I, \ k \in K, \\ 0 & \text{otherwise}, \end{cases} \)
- \( z_k = \begin{cases} 1 & \text{if a depot is set at location } k, \forall k \in K, \\ 0 & \text{otherwise}. \end{cases} \)

We can then state the problem as

\[
[P]\min \sum_{i \in I} \sum_{k \in K} \sum_{j \in J} c_{ijk} x_{ijk} + \sum_{i \in I} \sum_{k \in K} f_{ik} y_{ik} + \sum_{k \in K} g_k z_k
\]

subject to

\[
\sum_{j \in J} a_j x_{ijk} \leq b_i \quad \forall i \in I, k \in K, \quad (2)
\]

\[
\sum_{i \in I} \sum_{k \in K} x_{ijk} = 1 \quad \forall j \in J, \quad (3)
\]

\[
\sum_{k \in K} y_{ik} \leq 1 \quad \forall i \in I, \quad (4)
\]

\[
x_{ijk} \leq y_{ik} \quad \forall i \in I, j \in J, k \in K, \quad (5)
\]

\[
y_{ik} \leq z_k \quad \forall i \in I, k \in K, \quad (6)
\]

\[
x_{ijk}, y_{ik}, z_k \in \{0, 1\}. \quad (7)
\]

The objective function (1) is the total cost consisting of the cost of assigning customers to facilities, the cost of establishing facilities and the cost of opening depots. Constraint set (2) ensures that the customer demand serviced by a certain facility cannot exceed its capacity. Constraint set (3) ensures that each customer is assigned to exactly one facility. Constraint set (4) ensures that each facility can be serviced by only one depot, and (5) ensures that the assignments are made only to open facilities. The corresponding requirement of assigning facilities to open depots is taken care of by (6). Finally, all variables used in the models are required to be binary. The standard capacitated facility location problem is NP-hard, see e.g. Mirchandani and Francis [19]. As this is a special case of our problem it follows that [P] also is NP-hard.
3. Lagrangian heuristics

A Lagrangian relaxation is created by removing (relaxing) a set of constraints, weighting them with Lagrangian multipliers and then placing them in the objective function. The purpose is to obtain a relaxed problem, called Lagrangian subproblem, which is easier to solve than the original problem. The objective value from the Lagrangian relaxation problem, for any given set of multipliers, provides a lower bound (in the case of minimization) for the optimal solution to the original problem. The best lower bound can be derived by solving the Lagrangian dual. Since the dual function most often is non-differentiable there is a need to use a special method for this class of problems. A frequently used and efficient method is subgradient optimization. Information obtained from the Lagrangian relaxation is then often used by application-dependent heuristics to construct feasible solutions and hence upper bounds to the original problem.

In Tragantalerngsak et al. [29] six different relaxations are investigated and tested. It turns out that a strategy where the constraint sets (6) and (3) are relaxed provides high quality lower bounds. An even better relaxation is obtained by introducing constraints that are redundant in the original formulation but that are active in the relaxed version. This technique will generate so-called strengthened subproblems. One way to establish such redundant constraints is to consider practical and logical restrictions. In our model we note that at least one depot must be built and that the total capacity found among the open facilities must exceed to total demand from the customers. It was found that this version provides lower bounds for some problem sets that were about 95% of the optimal objective value as compared to 85% for the LP relaxation. We have used this relaxation and we describe some important characteristics which we need for our B&B algorithm. For a more detailed description we refer to Tragantalerngsak et al. [29].

3.1. Lagrangian relaxation

Constraint set (6) in [P] connects the \( y \) and \( z \) variables. If these constraints are relaxed, [P] separates into two subproblems: one problem involving \( x \) and \( y \) and the other the \( z \) variables. Furthermore, if constraint set (3) which forces each customer to be assigned to exactly one facility is relaxed, we can make use of the structure of the knapsack problem in constraint set (2). This relaxation can be improved by including the following two redundant constraint sets that may tighten the relaxed formulation. The first is a constraint forcing at least one depot to be open,

\[
\sum_{k \in K} z_k \geq 1. \tag{8}
\]

The second is the constraint forcing open facilities to have sufficient capacity for the total customer demand,

\[
\sum_{i \in I} \sum_{k \in K} b_i y_{ik} \geq \sum_{j \in J} a_j. \tag{9}
\]

Adding the above constraints and relaxing constraint sets (3) and (6) with multiplier vectors \( \lambda \) and \( \omega \), respectively, gives the following Lagrangian relaxation:

\[
\text{[LR]}
\]

\[
\min \sum_{i \in I} \sum_{k \in K} \sum_{j \in J} (c_{ijk} - \lambda_j) x_{ijk} + \sum_{i \in I} \sum_{k \in K} (f_{ik} + \omega_i k) y_{ik} + \sum_{k \in K} \left( g_k - \sum_{i \in I} \omega_i k \right) z_k + \sum_{j \in J} \lambda_j
\]

subject to (2), (4), (5), (7), (8) and (9).

This model separates into two subproblems, namely

\[
\text{[LR_2]}
\]

\[
\min \sum_{k \in K} \left( g_k - \sum_{i \in I} \omega_i k \right) z_k
\]

subject to

\[
\sum_{k \in K} z_k \geq 1,
\]

\[
z_k \in \{0, 1\} \quad \forall k \in K
\]
and
\[
[LR_{xy}]
\]
\[
\min \sum_{i \in I} \sum_{k \in K} \sum_{j \in J} (c_{ijk} - \lambda_j)x_{ijk}
\]
\[
+ \sum_{i \in I} \sum_{k \in K} (f_{ik} + \omega_{ik})y_{ik} + \sum_{j \in J} \lambda_j
\]
subject to
\[
\sum_{j \in J} a_{jx_{ijk}} \leq b_i \quad \forall i \in I, \ k \in K,
\]
\[
\sum_{k \in K} y_{ik} \leq 1 \quad \forall i \in I,
\]
\[
\sum_{i \in I} \sum_{k \in K} b_{jy_{ik}} \geq \sum_{j \in J} a_{j},
\]
\[
x_{ijk} \leq y_{ik} \quad \forall i \in I, \ j \in J, \ k \in K,
\]
\[
x_{ijk}, y_{ik} \in \{0, 1\} \quad \forall i \in I, \ j \in J, \ k \in K.
\]

The first problem is a knapsack problem and the second can be reformulated into a number of knapsack problems.

The lower bound on the optimal objective function value for \([P]\) is given by \(v(LR_{xy}) + v(LR_z)\). (We use the notation \(v(X)\) to denote the optimal value from solving problem \([X]\).) Problem \([LR_z]\) can easily be solved by inspection. To solve problem \([LR_{xy}]\), we can reformulate the problem into a series of knapsack problems as described in [29].

An important feature of the Lagrangian relaxation approach is the quality of the feasible solution providing the upper bound. Here, we employ a heuristic based on solving a generalized assignment problem (GAP) in which customers are assigned to a pre-selected subset of facilities which are pre-assigned to a subset of depots. This heuristic is adapted from a heuristic proposed by Pirkul [22] for determining the feasible solution for the single-source, capacitated facility location problem. A feasible (and initial feasible solutions) solution can be determined by algorithms described in [29].

Subgradient optimization is an effective method to find the optimal Lagrangian multiplier and solve the dual problem. Given an initial vector, \(\lambda^0\), a sequence of \(\lambda'\) is generated by taking a step size \(\theta_t\) in a search direction (based on the subgradient \(\gamma\)). The step size can be defined by many rules, a commonly used rule in practical application is
\[
\theta_t = d_i(\hat{Z} - v(LR_{\gamma'})/\|\gamma\|^2),
\]
where \(\hat{Z}\) is an upper bound on the optimal objective value. Most often, the best known feasible solution of \([P]\) is used, \(||\cdot||\) denotes the Euclidian norm and \(d_i\) is initially set to \(0 < d_i < 2\). \(d_i\) will be halved every time the procedure goes through a fixed number of consecutive iterations with no improvement in the lower bound. Beasley [4,5] suggests some improvements, which we have used, which are computationally useful.

4. Branch and bound procedures

B&B-based procedures based on Lagrangian relaxation and subgradient optimization have been applied successfully by several authors to solve facility location problems. One reason is that the lower bound from the Lagrangian relaxation will be at least comparable to, and most often much better than the bound obtained from the LP relaxation. This has the effect of lowering the number of nodes required in the B&B tree.

Nauss [20] proposes a B&B method based on Lagrangian relaxation to solve the capacitated facility location problem. Special penalties are combined into the subgradient optimization scheme to attempt to fix the free facilities to be open or closed. At each node, the free facility variable with maximum penalty will be chosen as the branching variable. LIFO (Last In First Out) is used as the candidate problem selection criterion.

Christofides and Beasley [7] present a similar procedure for the capacitated facility location problem with a lower and an upper limit on the number of the facilities to be open. Several reduction tests have been proposed and combined in the subgradient optimization. At the node in which all facilities are fixed to be open or closed, the problem is reduced to a network flow problem.
associated with the set of open facilities. Beasley [5] modifies this method by introducing feasible solution exclusion constraints in the model to reduce the extent of the tree search.

For the single-source case, Neebe and Rao [21] formulate the problem as a set partitioning problem. An LP-relaxation-based B&B of this formulation is developed to find the optimal solution. Pirkul [22] proposes a two-stage Lagrangian relaxation-based B&B method for the same problem. In the first stage, the B&B tree is formulated by fixing facility variables. At the root node, i.e., before any branching, penalties are employed to attempt to reduce the size of the problem and to select the branching variables. The order of branching variable is determined by using decreasing order of the corresponding penalties obtained from this problem reduction. The LIFO is used as the criterion for candidate problem selection. The second stage of the B&B is to continue from the unfathomed terminal nodes. At these terminal nodes, all facilities have been fixed to be either open or closed. The problem is then reduced to a generalized assignment problem. The solution algorithm proposed by Ross and Soland [26] is employed in solving this problem. Holmberg et al. [14] also suggest a similar B&B procedure as Pirkul but use a more powerful primal heuristic to generate upper bounds and develop some efficient reduction tests.

4.1. Problem reduction

Problem reduction tests is a popular technique in B&B algorithms to increase their efficiency. It can be implemented through the use of conditional bound and penalty. Reduction tests can be incorporated in B&B using the fact that a candidate problem whose lower bound is higher than the incumbent (best feasible solution found) will be fathomed and dismissed from further consideration. The estimate of the increase in the lower bound that is caused by requiring a variable to be zero or one, called the penalty, is a tool used in problem reduction techniques. If the value of the penalty plus the lower bound is greater than the incumbent, the condition which results in the penalty cannot occur in the optimal solution of that candidate problem or in any of its descendants.

For the Lagrangian relaxation-based approach, the penalty can be determined as the increase in lower bound that is caused by requiring, for example, $x_i = 1 - \bar{x}_i$, where $\bar{x}$ is the solution from the Lagrangian relaxation problem. If the penalty is greater than the difference between the lower bound and the incumbent, then $x_i$ should be set equal to $\bar{x}_i$ in all successors of that node, see for example Pirkul [22].

These penalties are often found to be very efficient in Lagrangian dual schemes. The reason is that the relaxed problem has already separated into a number of simpler problems. To force a variable to a certain value (0 or 1) and find the penalty there is a need to re-solve the problem. If this can be done for easier subproblems, then the required computational time will be much lower than re-solving the full problem. This property has been proposed in a B&B procedure for the capacitated single-source facility location problem by Holmberg et al. [14].

4.2. Branch and bound algorithm for TSCFLP

Problem [P] contains three sets of integer variables, i.e., $x$, $y$ and $z$. The decision not to open a depot, i.e., $z_k = 0$ has a strong effect on many other variables. We also make the observation that the correct choice for the number and locations of the depots would, in practice, probably have the biggest impact on the cost of the operation, followed by decisions about facilities and then individual customer assignments. With these thoughts in mind, we propose a three stage branch and bound algorithm for solving the problem.

In the first stage, the B&B tree is created using branching strategies involving only the $z_k$ variables. Nodes in the tree at which all $z_k$ variables have been fixed are called terminal nodes. In general, not all candidate problems at these nodes will be fathomed. The second stage of the process is continued from these unfathomed terminal nodes using branching strategies involving the $y_k$ variables. Finally, a third stage is carried out starting
from the unfathomed terminal nodes from the second stage. This third stage problem is reduced to a generalized assignment problem, for which efficient solution algorithms exist.

Lagrangian relaxation incorporating subgradient optimization is used to compute lower bounds. The initial multipliers for the subgradient iteration are set equal to the multipliers associated with the lower bound at the predecessor node. Problem reduction is used at the root nodes of both the first and second stages of the process before starting the tree search of that stage.

As noted by Christofides and Beasley [7], the facility with the smallest reduced cost is most likely to be open in the optimal solution of the problem at that node. Therefore, the minimum reduced cost criteria combined with 1-branch and LIFO criteria for candidate problem selection seems to provide the strongest bounds. A node can be fathomed when the lower bound at that node is greater than or equal to the best feasible solution found, or the relative gap (the difference between upper bound and lower bound divided by the lower bound) at that node is less than a small fixed value (ε). The procedure terminates if all nodes are fathomed. The details of each stage are described in the following three sections.

4.3. The first stage

The root node of the B&B tree is defined as problem [P]. If the solution found cannot be proved to be optimal, i.e. a duality gap exists, then the following B&B procedure is carried out. Let \( z' \) be the solution obtained from solving [LR]. A problem reduction test can be done by separately setting each \( z_k \) equal to \( 1 - z'_k \) and re-solve the Lagrangian subproblem. If this condition makes the lower bound higher than the upper bound, then \( z_k \) will be fixed at \( z'_k \) in all successor nodes. For the Lagrangian relaxation presented, fixing a \( z_k \) variable affects only the subproblem [LR\( g_k \)]. Therefore, the problem reduction test can be done efficiently by inspection independently for each variable.

Let LB and UB be the lower and upper bound, respectively. Let \( \bar{g}_k = g_k - \sum_l \omega_{lk} \) be the objective function coefficient of the Lagrangian subproblem [LR\( g_k \)] which is associated with the solution providing the best lower bound. The reduction test is then: If \( LB + p_k > UB \), \( z_k \) should be fixed to \( z'_k \) for all successor nodes. It is straightforward to compute the penalty \( p_k \). If there is more than one \( \bar{g}_k \) that is less than or equal to zero, changing \( z'_k \) from 0 to 1 or from 1 to 0 will result in the lower bound increasing by \( |\bar{g}_k| \). Therefore, the penalty \( p_k \) equals \( |\bar{g}_k| \). In the case where only one value \( \bar{g}_k \) is less than or equal to zero we need to separate \( z'_k = 0 \) and \( z'_k = 1 \). If \( z'_k = 0 \), it means that \( \bar{g}_k > 0 \). When we change \( z'_k \) to 1, the lower bound will increase \( g_k \), hence \( p_k = \bar{g}_k \). If \( z'_k = 1 \), it means that \( \bar{g}_k \leq 0 \). When we change \( z'_k \) to 0, we must open another depot which means that we get \( p_k = \min \{ \bar{g}_l, \ l \neq k \} - \bar{g}_k \).

If all values \( \bar{g}_k \) are greater than zero, the depot with minimum \( \bar{g}_k \) is opened. If \( z'_k = 1 \), it means that \( \bar{g}_k = \min \{ \bar{g}_l, \ l \in K \} \). When we change \( z'_k \) to 0, we should open a new depot with the second minimum value of \( \bar{g} \). Therefore, \( p_k = \min \{ \bar{g}_l, \ l \neq k \} - \bar{g}_k \). Finally, if \( z'_k = 0 \) and we change it to 1, we should close the previous open depot. Therefore, \( p_k = \bar{g}_k - \min \{ \bar{g}_l, \ l \in K \} \).

The main features of the B&B algorithm used are based on those used by Galvao and Raggi [11] for solving the uncapacitated facility location problem. The principal steps in our algorithm are discussed below. Anyone who is interested in a more detailed description is referred to [30].

We need to define three fixation sets corresponding to depot variables (i.e. \( z \)) that are fixed to be open, fixed to be closed and free (not fixed). We use \( v^* \) as the incumbent value. There is, of course, other information also needed to keep track of the B&B tree but this is outside the scope of this description. The principal steps of the process are given below.

At the root node, we check if the solution of the Lagrangian dual problem has provided the optimal solution. If not, we update the incumbent value \( v^* \) with the upper bound found in this node. We then perform the reduction tests and update the fixation sets accordingly. We then select a branching variable, say \( z_k \), and select the 1-branch, i.e. \( z_k = 1 \). The multipliers are initialized from the multipliers found at the parent node. The upper bound used is the initial upper bound. The
Lagrangian dual is solved and the incumbent is updated if possible. We can also test if the node can be fathomed at this stage by using the lower bound found.

If there are no more depot variables to branch on we store the information which will be needed for the second stage of the B&B process. In the backtracking, we investigate the zero branch using the same approach as for the 1-branch. Once we have backtracked to the root node we have completed the first stage. At the end of the first stage, if there are remaining terminal nodes which are not fathomed, the second stage of branch and bound will be performed. The terminal node with the smallest lower bound is selected first for further branching.

4.4. The second stage

Here we continue from each unfathomed terminal node in the first stage. The problem of assigning facilities and customers to the predetermined depots is identified and solved. Let \( K' = \{ k \mid z_k = 1 \text{ at that terminal node} \} \), then

\[
\text{[P2]} \quad \min \sum_{i \in I} \sum_{k' \in K'} \sum_{j \in J} c_{ijk'} x_{ijk'} + \sum_{i \in I} \sum_{k' \in K'} f_{ik'} y_{ik'} \tag{10}
\]

subject to

\[
\sum_{j \in J} a_{ijk} x_{ijk'} \leq b_i \quad \forall i \in I, k' \in K', \tag{11}
\]

\[
\sum_{k' \in K'} x_{ijk'} = 1 \quad \forall j \in J, \tag{12}
\]

\[
\sum_{k \in K'} y_{ik} \leq 1 \quad \forall i \in I, \tag{13}
\]

\[
\sum_{k \in K'} b_j y_{ik} \geq \sum_{j \in J} a_j \tag{14}
\]

\[
x_{ijk'} \leq y_{ik} \quad \forall i \in I, j \in J, k' \in K', \tag{15}
\]

\[
x_{ijk'}, y_{ik} \in \{0, 1\}. \tag{16}
\]

Lagrangian relaxation, in which constraints (12) are relaxed with multiplier \( \theta \), together with subgradient optimization is used to determine the lower bound of [P2]. The Lagrangian relaxation is

\[
\text{[LR2 prev]} \quad \min \sum_{i \in I} \sum_{j \in J} \sum_{k' \in K'} (c_{ijk'} - \theta) x_{ijk'} + \sum_{i \in I} \sum_{k' \in K'} f_{ik'} y_{ik'} + \sum_{j \in J} \theta_j
\]

subject to

\[
\sum_{j \in J} a_{ijk} x_{ijk'} \leq b_i \quad \forall i \in I, k' \in K', \tag{11}
\]

\[
\sum_{k' \in K'} x_{ijk'} = 1 \quad \forall j \in J, \tag{12}
\]

\[
\sum_{k \in K'} y_{ik} \leq 1 \quad \forall i \in I, \tag{13}
\]

\[
\sum_{k \in K'} b_j y_{ik} \geq \sum_{j \in J} a_j \tag{14}
\]

\[
x_{ijk'} \leq y_{ik} \quad \forall i \in I, j \in J, k' \in K', \tag{15}
\]

\[
x_{ijk'}, y_{ik} \in \{0, 1\}. \tag{16}
\]

Following similar ideas as used earlier for [LR prev] in Section 3, this model can be rewritten and solved by a series of simpler problems. The lower bound is determined by \( v(\text{LR2 prev}) + \sum_{k' \in K'} g_{ik'} \). The upper bound is obtained using the same method as in [31]. If the upper bound is less than the incumbent value \( v^* \), update \( v^* \). In case the lower bound is greater than or equal to the incumbent value or the optimal solution to [P2] is obtained (i.e. the relative gap \(< \varepsilon \), the node is fathomed. Otherwise it will be the root node of a new tree, and the second stage of the B&B process on the \( y \) variables will be carried out.

To attempt to reduce the size of the problem before starting the second stage of the branch and bound, penalties are once again calculated. Let \( y' \) be the solution from [LR2 prev]. The penalties \( p_{ik} \) are defined by setting \( y_{ik} = 1 - y'_{ik} \) and re-solving problem [LR2 prev]. If the new lower bound is greater than or equal to the upper bound at the root node, \( y_{ik} \) will be fixed to \( y'_{ik} \) in every solution. If \( y_{ik} \) is fixed to 1, \( y_{ik}, k \neq k' \) will be fixed to 0. If all \( y_{ik} \) can be fixed, then the third stage of branch and bound is performed. Otherwise the B&B procedure for the second stage is continued.

Note that, in performing problem reduction, we use the upper bound at the node and not the incumbent \( (v^*) \). This is due to the fact that the set of open depots at the node to be considered is not necessarily the same as the set of open depots in the solution which corresponds to the incumbent.
The B&B algorithm in this stage can be extended from the algorithm in the first stage as described below.

We select the unfathomed terminal node from the first stage with minimum lower bound and whose lower bound is less than the incumbent \( v \). If there are none, the second stage of branch and bound is completed, and the optimal solution has already been found. Otherwise the Lagrangian dual problem at the root node of the second stage is solved. Again, the incumbent is updated if a better upper bound is found. If the node is fathomed, we stop the second stage for this node. Otherwise calculate the penalties, carry out the reduction tests and update fixation sets related to the \( y_{ik} \) variables.

We then select a branching variable, say \( y_{ik} \), and select the 1-branch, i.e. \( y_{ik} = 1 \). If the possible capacity of the facilities fixed to one plus the free facilities is less than total demand, the node is fathomed. Otherwise we initialize the multipliers from the multipliers found at the parent node. The upper bound used is the initial upper bound. The Lagrangian dual is then solved. We update the incumbent if possible and test if the node can be fathomed using the lower bound found. If there are no more facility variables to branch on we store the information which will be needed for the third stage of the B&B process. In the backtracking, we investigate the zero branch using the same approach as for the 1-branch.

4.5. The third stage

At a terminal node in the second stage, all depots and facilities have been fixed. If this node is not fathomed, the problem of assigning customers to the established facilities is identified. This problem is a special case of the generalized assignment problem (GAP), see below. Let

\[
\hat{I} = \{ (i, k) \mid y_{ik} \text{ is fixed to } 1 \text{ at that terminal node} \},
\]

\[
\hat{x}_{ij} = \{ x_{ij} \mid (i, k) \in \hat{I} \},
\]

\[
\hat{c}_{ij} = \{ c_{ijk} \mid (i, k) \in \hat{I} \},
\]

\[
I^* = \{ i \mid (i, k) \in \hat{I} \},
\]

\[
\min \sum_{i \in I^*} \sum_{j \in J} \hat{c}_{ij} \hat{x}_{ij} \tag{17}
\]

subject to

\[
\sum_{j \in J} a_{ij} \hat{x}_{ij} \leq b_i \quad \forall i \in I^*, \tag{18}
\]

\[
\sum_{i \in I^*} \hat{x}_{ij} = 1 \quad \forall j \in J, \tag{19}
\]

\[
\hat{x}_{ij} \in \{0, 1\} \quad \forall i \in I^*, j \in J. \tag{20}
\]

There are several algorithms which can be used to solve this problem. This includes those of Ross and Soland [26] and Martello and Toth [17]. We have used the code MTG developed by Martello and Toth [18]. The MTG code implements a depth-first B&B method. At the root node, two lower bounds proposed by Martello and Toth [17] and by Fisher et al. [10] are determined. The maximum of these two values is selected to be the lower bound at the root node. At other nodes, the lower bounds are derived from the maximum of the lower bounds proposed by Ross and Soland [26] and by Martello and Toth [17]. The branching scheme, the problem reduction and the method for finding a feasible solution are based on those proposed by Martello and Toth [17].

As noted by Pirkul [22] and Martello and Toth [18] (see computational experience), the solution of the GAP, especially for large problems, may be relatively time consuming. Hence, we incorporate the idea suggested by Pirkul of terminating the B&B process for the GAP if the computing time exceeds some predetermined value. If this happens, the best lower bound is kept and is used to determine a final lower bound for \( [P] \).

5. Computational experience

We have implemented the proposed algorithm and tested its performance on six sets of test problems. The procedures were coded in FORTRAN and run on a DEC2000 300AXP Workstation. The quality of the solutions and the computational times are compared with a 0–1 integer programming package called ZIP, developed by Ryan [24]. The ZIP package uses an LP-based
branch and bound approach. This package consists of a number of subroutines where the user provides routines to control the LP solution, problem data as well as the search of the B&B tree. This makes it possible to develop an overall algorithm that can be tailor made for specific applications.

5.1. Test problems

Six sets of test problems were constructed to test the performance of the proposed algorithm. Five of these sets are described in detail in [29] and the extra (set 3 in this paper) is just a larger version of sets 1 and 2. The details of these test problems are summarized in Table 1. Column “size” shows the number of potential depots, potential facilities and customers. The second column displays the number of constraints in the model. The fourth column shows the number of variables in the models. Columns “location” and “demand” give information about the uniform distributions used to generate the problems. The capacity of facilities is presented in the column “capacity”. The last column of the table shows the ratio between the total demand and total capacity of all facilities. It is reasonable to assume that problems corresponding to higher ratios are more difficult to solve than problems with lower ratios. This is because there is a need to have more facilities open which makes the problem harder. For each of the four choices for the $b_i$’s, four specific problem instances were generated in which the depot fixed costs ($g_k$) were varied from 100 to 500 in increments of 100. Therefore, in each problem set, there will be twenty problems.

Table 1
The characteristic of the test problem set

<table>
<thead>
<tr>
<th>Problem set</th>
<th>Size $p \times m \times n$</th>
<th>Num. cons.</th>
<th>Num. var.</th>
<th>Location</th>
<th>Demand</th>
<th>Capacity</th>
<th>$\sum_j a_j / \sum_i b_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$4 \times 10 \times 20$</td>
<td>910</td>
<td>844</td>
<td>U[0, 100]</td>
<td>U[20, 50]</td>
<td>150</td>
<td>0.47</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>300</td>
<td>0.23</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[150, 300]</td>
<td>0.33</td>
</tr>
<tr>
<td>2</td>
<td>$4 \times 10 \times 20$</td>
<td>910</td>
<td>844</td>
<td>U[10, 150]</td>
<td>U[20, 60]</td>
<td>150</td>
<td>0.54</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td>0.41</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>300</td>
<td>0.27</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[100, 500]</td>
<td>0.26</td>
</tr>
<tr>
<td>3</td>
<td>$4 \times 10 \times 50$</td>
<td>2140</td>
<td>2044</td>
<td>U[0, 200]</td>
<td>U[10, 30]</td>
<td>150</td>
<td>0.65</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>200</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>300</td>
<td>0.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[200, 500]</td>
<td>0.28</td>
</tr>
<tr>
<td>4</td>
<td>$5 \times 10 \times 100$</td>
<td>5210</td>
<td>5055</td>
<td>Random</td>
<td>Random</td>
<td>250</td>
<td>0.58</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>300</td>
<td>0.48</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[100, 500]</td>
<td>0.52</td>
</tr>
<tr>
<td>5</td>
<td>$5 \times 10 \times 100$</td>
<td>5210</td>
<td>5055</td>
<td>Random–clustered</td>
<td>Random–clustered</td>
<td>300</td>
<td>0.57</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
<td>0.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>450</td>
<td>0.34</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[200, 600]</td>
<td>0.47</td>
</tr>
<tr>
<td>6</td>
<td>$5 \times 10 \times 100$</td>
<td>5210</td>
<td>5055</td>
<td>Clustered–clustered</td>
<td>Clustered–clustered</td>
<td>300</td>
<td>0.60</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>400</td>
<td>0.45</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>450</td>
<td>0.36</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>U[200, 600]</td>
<td>0.49</td>
</tr>
</tbody>
</table>
5.2. Numerical results

To avoid unnecessary work in solving the Lagrangian subproblem at each node of the B&B tree, Fisher [9] suggests a strategy in which the initial multipliers are set equal to those associated with the lower bound at the parent node, and the maximum number of subgradient iterations as well as the initial parameter \( d_t \) (used in the step length formula) depend on the type of node being treated. Higher values of these parameters are used at the root node of the tree, and smaller values are used at successor nodes.

Based on this strategy, a number of different values for the initial step length, for the threshold value for halving the step length and for the maximum of subgradient iteration were tested. For the numerical results reported here, the following values of parameters were selected. For the first stage, at the root node the initial \( d_t \) is set to 1 and then halved whenever the lower bound does not improve in 15 consecutive iterations. The maximum number of iterations is 800. At other nodes, the initial \( d_t \) is set to 0.7 and halved if the process goes through 10 consecutive iterations without improvement in the lower bound. The maximum number of iterations is 200. For the second stage, at root nodes the initial \( d_t \) is set to 0.7 and halved if the lower bound does not improve in 15 consecutive iterations. The maximum number of iterations is 400. At other nodes, the initial \( d_t \) is set to 0.50 and halved if the process goes through 10 consecutive iterations with no improvement in the lower bound. The maximum number of iterations is 200.

The ZIP code is used to obtain the LP-relaxation lower bound and the optimal solution to the problems in problem set 1–3. We compute the optimal solution only for the small and medium-sized problems (problem set 1–3). The reason is that for the large-size problems, too large amounts of CPU time and memory are required to find the optimal solution.

The computational results for each problem in problem sets 1–3 are presented in Tables 2–4. The set of columns in these tables give the following information:

- The lower bound, duality gap and CPU time in second from the Lagrangian relaxation heuristic before starting the B&B process.
- The characteristics of the Lagrangian relaxation-based B&B, namely the number of nodes considered and the number of unfathomed terminal nodes (ufn) in the first stage, the number of trees and the number of nodes considered in the second stage, the number of unfathomed terminal nodes from the second stage in which the third stage will be continued, and the total time in seconds required to solve the problem.
- The results obtained from ZIP code, namely the lower bound from LP-relaxation, CPU time in calculating the LP problem, the optimal value, total time and the number of nodes used.

Comparison between the lower bounds from Lagrangian relaxation heuristic at the root node (in the first stage) and the lower bounds from the LP-relaxation show that the lower bounds from the Lagrangian relaxation heuristics are significantly higher than those from the LP-relaxation. Furthermore, the computational time required by the Lagrangian relaxation heuristic are less than the time required to solve the LP problem.

From column “ufn”, it can be seen that only a few unfathomed terminal nodes are left from the first stage of the B&B. Furthermore, some of these nodes are fathomed without further consideration in the second stage because the lower bound at these nodes is found to be higher than the current incumbent value at the time they are considered. Also, some are fathomed at root node of the second stage because the optimal solution of the problem at that root node is found or the lower bound at that root node is higher than the current incumbent value. Here we do not count the unfathomed nodes which lead only to a root node without any branching as trees. For these reasons, the number of trees in the second stage may be less than the number of unfathomed nodes in column “ufn”. In the small and medium-sized problems, it was never necessary to perform the third stage of branch and bound. In all cases, we can obtain the optimal or near optimal solution in a reasonable amount of computing time.

Comparison between the results of the Lagrangian relaxation-based B&B algorithm and those from the LP-based ZIP code shows that the
Table 2
The results obtained from data set 1

<table>
<thead>
<tr>
<th>$g_k$ value</th>
<th>Lagrangian relaxation</th>
<th>ZIP code</th>
<th>LP-relaxation</th>
<th>Branch and bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Root node</td>
<td>First-stage</td>
<td>Second-stage</td>
<td>Third-stage Total</td>
</tr>
<tr>
<td></td>
<td>LB</td>
<td>Gap%</td>
<td>Time</td>
<td>Node</td>
</tr>
<tr>
<td>$b_i = 150$</td>
<td>100</td>
<td>3583.94</td>
<td>3.7</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3740.46</td>
<td>5.9</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>3851.84</td>
<td>11.7</td>
<td>3.9</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>3952.28</td>
<td>11.4</td>
<td>3.8</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>4052.09</td>
<td>14.0</td>
<td>3.9</td>
</tr>
<tr>
<td>$b_i = 200$</td>
<td>100</td>
<td>3336.48</td>
<td>5.6</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3442.22</td>
<td>13.2</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>3542.40</td>
<td>11.1</td>
<td>3.1</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>3641.99</td>
<td>10.8</td>
<td>3.0</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>3742.16</td>
<td>11.9</td>
<td>3.0</td>
</tr>
<tr>
<td>$b_i = 300$</td>
<td>100</td>
<td>3109.43</td>
<td>15.3</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3210.57</td>
<td>12.1</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>3309.78</td>
<td>14.3</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>3408.76</td>
<td>11.5</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>3509.19</td>
<td>11.1</td>
<td>1.9</td>
</tr>
<tr>
<td>$b_i = \text{random}$</td>
<td>100</td>
<td>3218.18</td>
<td>4.3</td>
<td>2.8</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>3325.79</td>
<td>11.5</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>3425.46</td>
<td>13.2</td>
<td>2.4</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>3525.27</td>
<td>13.2</td>
<td>2.6</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>3625.57</td>
<td>12.8</td>
<td>2.6</td>
</tr>
</tbody>
</table>
Table 3
The results obtained from data set 2

<table>
<thead>
<tr>
<th>(g_k) value</th>
<th>(b_i) value</th>
<th>Root node</th>
<th>First-stage</th>
<th>Second-stage</th>
<th>Third-stage</th>
<th>Total time</th>
<th>ZIP code</th>
<th>LP-relaxation</th>
<th>Branch and bound</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>LB</td>
<td>Gap%</td>
<td>Time</td>
<td>Node</td>
<td>ufn</td>
<td>Tree</td>
<td>Node</td>
<td>Opt.</td>
</tr>
<tr>
<td></td>
<td>150</td>
<td>4304.72</td>
<td>12.8</td>
<td>3.9</td>
<td>21</td>
<td>6</td>
<td>2</td>
<td>92</td>
<td>41.5</td>
</tr>
<tr>
<td></td>
<td>200</td>
<td>4464.78</td>
<td>23.6</td>
<td>4.5</td>
<td>21</td>
<td>6</td>
<td>2</td>
<td>68</td>
<td>34.9</td>
</tr>
<tr>
<td></td>
<td>300</td>
<td>4613.99</td>
<td>13.2</td>
<td>4.4</td>
<td>21</td>
<td>3</td>
<td>2</td>
<td>74</td>
<td>31.3</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>4727.53</td>
<td>28.6</td>
<td>4.8</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td>72</td>
<td>33.8</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td>4827.48</td>
<td>22.8</td>
<td>4.9</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td>70</td>
<td>29.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4018.81</td>
<td>20.0</td>
<td>4.3</td>
<td>23</td>
<td>8</td>
<td>4</td>
<td>64</td>
<td>35.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4137.18</td>
<td>22.2</td>
<td>4.2</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>16.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4237.23</td>
<td>23.6</td>
<td>4.1</td>
<td>29</td>
<td>3</td>
<td>1</td>
<td>19</td>
<td>27.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4337.01</td>
<td>24.5</td>
<td>4.3</td>
<td>21</td>
<td>2</td>
<td>1</td>
<td>7</td>
<td>28.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4436.78</td>
<td>25.4</td>
<td>4.3</td>
<td>29</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>26.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3824.37</td>
<td>13.9</td>
<td>2.2</td>
<td>9</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>5.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3924.37</td>
<td>13.5</td>
<td>1.8</td>
<td>19</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>8.2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4024.51</td>
<td>13.1</td>
<td>2.4</td>
<td>17</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>7.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4124.75</td>
<td>12.8</td>
<td>2.3</td>
<td>22</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>11.7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4224.78</td>
<td>12.5</td>
<td>2.2</td>
<td>20</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>11.0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3838.89</td>
<td>6.6</td>
<td>2.1</td>
<td>17</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>7.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3939.28</td>
<td>16.5</td>
<td>2.3</td>
<td>13</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>7.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4039.35</td>
<td>16.1</td>
<td>2.3</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>8.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4139.07</td>
<td>15.7</td>
<td>2.3</td>
<td>21</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>10.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>4239.21</td>
<td>15.4</td>
<td>2.4</td>
<td>25</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>11.8</td>
</tr>
<tr>
<td>$g_k$ Value</td>
<td>Lagrangian Relaxation</td>
<td>ZIP Code</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------------------------</td>
<td>----------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Root node</td>
<td>First-stage</td>
<td>Second-stage</td>
<td>Third-stage</td>
<td>Total time</td>
<td>LP-relaxation</td>
<td>Branch and bound</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>LB</td>
<td>Gap%</td>
<td>Time</td>
<td>Node</td>
<td>ufn</td>
<td>Tree</td>
<td>Node</td>
<td>LP</td>
<td>Time</td>
</tr>
<tr>
<td>$b_i = 150$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>8107.80</td>
<td>6.1</td>
<td>7.1</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td>30</td>
<td>–</td>
<td>50.1</td>
</tr>
<tr>
<td>200</td>
<td>8289.24</td>
<td>15.0</td>
<td>7.8</td>
<td>15</td>
<td>1</td>
<td>1</td>
<td>25</td>
<td>–</td>
<td>37.5</td>
</tr>
<tr>
<td>300</td>
<td>8446.12</td>
<td>12.6</td>
<td>7.4</td>
<td>23</td>
<td>2</td>
<td>2</td>
<td>40</td>
<td>–</td>
<td>56.7</td>
</tr>
<tr>
<td>400</td>
<td>8582.18</td>
<td>22.3</td>
<td>6.7</td>
<td>19</td>
<td>2</td>
<td>1</td>
<td>20</td>
<td>–</td>
<td>38.7</td>
</tr>
<tr>
<td>500</td>
<td>8713.23</td>
<td>20.8</td>
<td>6.8</td>
<td>27</td>
<td>2</td>
<td>1</td>
<td>21</td>
<td>–</td>
<td>51.8</td>
</tr>
<tr>
<td>$b_i = 200$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>8317.26</td>
<td>7.3</td>
<td>5.7</td>
<td>17</td>
<td>2</td>
<td>2</td>
<td>38</td>
<td>–</td>
<td>34.7</td>
</tr>
<tr>
<td>200</td>
<td>7903.75</td>
<td>10.6</td>
<td>6.5</td>
<td>17</td>
<td>1</td>
<td>1</td>
<td>29</td>
<td>–</td>
<td>30.4</td>
</tr>
<tr>
<td>300</td>
<td>8035.58</td>
<td>10.5</td>
<td>6.9</td>
<td>23</td>
<td>4</td>
<td>4</td>
<td>98</td>
<td>–</td>
<td>74.6</td>
</tr>
<tr>
<td>400</td>
<td>8145.89</td>
<td>14.3</td>
<td>7.0</td>
<td>26</td>
<td>5</td>
<td>4</td>
<td>104</td>
<td>–</td>
<td>86.8</td>
</tr>
<tr>
<td>500</td>
<td>8253.20</td>
<td>20.7</td>
<td>6.8</td>
<td>29</td>
<td>3</td>
<td>3</td>
<td>77</td>
<td>–</td>
<td>76.2</td>
</tr>
<tr>
<td>$b_i = 300$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>7398.44</td>
<td>3.7</td>
<td>4.9</td>
<td>13</td>
<td>0</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>14.1</td>
</tr>
<tr>
<td>200</td>
<td>7534.21</td>
<td>4.6</td>
<td>5.1</td>
<td>15</td>
<td>3</td>
<td>2</td>
<td>23</td>
<td>–</td>
<td>28.6</td>
</tr>
<tr>
<td>300</td>
<td>7651.46</td>
<td>9.9</td>
<td>4.4</td>
<td>17</td>
<td>2</td>
<td>2</td>
<td>38</td>
<td>–</td>
<td>30.7</td>
</tr>
<tr>
<td>400</td>
<td>7750.48</td>
<td>9.8</td>
<td>4.9</td>
<td>19</td>
<td>2</td>
<td>2</td>
<td>50</td>
<td>–</td>
<td>36.3</td>
</tr>
<tr>
<td>500</td>
<td>7842.39</td>
<td>9.9</td>
<td>5.0</td>
<td>21</td>
<td>2</td>
<td>2</td>
<td>48</td>
<td>–</td>
<td>35.5</td>
</tr>
<tr>
<td>$b_i = \text{random}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>7393.49</td>
<td>7.5</td>
<td>4.4</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>10.1</td>
</tr>
<tr>
<td>200</td>
<td>7544.76</td>
<td>1.3</td>
<td>4.4</td>
<td>9</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>10.0</td>
</tr>
<tr>
<td>300</td>
<td>7661.56</td>
<td>14.5</td>
<td>4.3</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>12.0</td>
</tr>
<tr>
<td>400</td>
<td>7765.30</td>
<td>15.0</td>
<td>4.3</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>10.5</td>
</tr>
<tr>
<td>500</td>
<td>7868.14</td>
<td>14.8</td>
<td>4.4</td>
<td>11</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>–</td>
<td>11.0</td>
</tr>
</tbody>
</table>
Lagrangian relaxation approach produces significantly smaller B&B trees and consumes much less computing time. The efficiency of the B&B method is critically dependent on the quality of the bounds produced at each node of the tree. The Lagrangian relaxation provides much stronger lower bounds and uses less computational time than the lower bounds obtained from the ZIP code. This means that the nodes can be fathomed rapidly by bound and the size of the B&B tree as well as the computation time for the Lagrangian relaxation-based approach is less than the LP-based approach.

Table 5 summarizes the performance of the Lagrangian relaxation-based B&B algorithm for all sets of the test problems. For each problem set (each with 20 problems), the following information is reported:

- The average number of nodes considered per problem. The numbers in brackets show the total number of nodes for all problems considered in the first stage, in the second stage and the number of unfathomed terminal nodes left from the second stage which will be solved using the GAP code in the third stage of the branch and bound tree.
- The average CPU time per problem in seconds.
- The number of problems which terminated in the first stage, in the second stage and in the third stage of the branch and bound.
- The average of the final relative gap (%) if there are cases in which the procedure terminates before obtaining the optimal solution. The details of this case will be discussed later.

Only one test problem from a total of 120 had to continue to the third stage. For this problem, the B&B for the GAP problem is terminated due to the time limit before the process could finish. The final relative gap is however less than 0.3%.

Problem set 6 (clustered–clustered problem) with random facility capacity is the hardest. Problems in this set have the characteristic that the ratio of the assignment costs $c_{ijk}$ to the customer demands $a_i$, $v_i$, $k$ are grouped and close to each other. The 0–1 knapsack problems arising in the Lagrangian relaxation subproblem also have this property which makes them hard to solve, see Balas and Zemel [2]. Furthermore, the randomness in the capacity makes the problem harder in the B&B process in solving the knapsack problem. Therefore, in solving these problems, we set a maximum time at each node in the tree. If the problem cannot terminate due to the termination rules of the Lagrangian relaxation within this maximum time (60 seconds), the process will switch to another node.

### Table 5

<table>
<thead>
<tr>
<th>Problem set</th>
<th>Avg. nodes</th>
<th>Avg. time</th>
<th>Terminate</th>
<th>Gap (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem set 1</td>
<td>34.3 (321, 366, 0)</td>
<td>12.3</td>
<td>6, 14, 0</td>
<td>0</td>
</tr>
<tr>
<td>Problem set 2</td>
<td>44.1 (400, 481, 0)</td>
<td>18.9</td>
<td>3, 17, 0</td>
<td>0</td>
</tr>
<tr>
<td>Problem set 3</td>
<td>49.9 (353, 645, 0)</td>
<td>36.8</td>
<td>1, 19, 0</td>
<td>0</td>
</tr>
<tr>
<td>Problem set 4</td>
<td>51.3 (623, 404, 1)</td>
<td>94.5</td>
<td>0, 19, 1</td>
<td>0.2</td>
</tr>
<tr>
<td>Problem set 5</td>
<td>32.7 (511, 144, 0)</td>
<td>82.3</td>
<td>4, 16, 0</td>
<td>0</td>
</tr>
<tr>
<td>Problem set 6</td>
<td>37.7 (440, 126, 0)</td>
<td>54.8</td>
<td>0, 15, 0</td>
<td>0</td>
</tr>
<tr>
<td>Random capacity</td>
<td>30.4 (127, 25, 0)</td>
<td>331.3</td>
<td>0, 5, 0</td>
<td>0</td>
</tr>
</tbody>
</table>

6. Discussion

In calculating step sizes in the subgradient optimization we also investigate the strategy in which the incumbent $v'$ is used in place of the upper bound value at each node. From our experiments, we found that the use of this strategy, in general, improves the performance in terms of both the size of tree generated and the CPU time. However, a detailed analysis of each stage shows that the first stage performs better using the upper bound
criterion, in terms of the number of nodes considered and the number of unfathomed terminal nodes left. However, the reverse tends to occur for the second stage. The details are presented in Table 6.

As noted by Shapiro [27], the use of incumbent in place of the upper bound in computing the step size works quite well in the situation when the optimal objective value of the Lagrangian dual problem (in that node) is higher than the incumbent. The subgradient method has then a good chance of finding multipliers such that the lower bound of that node is higher than the incumbent. However, in situations where the incumbent is higher than the optimal objective value of the Lagrangian dual problem, the considering node will never be fathomed by bound. The lower bounds produced from this strategy tend to be oscillating. Note that the optimal objective value of the Lagrangian dual problem is not known except when we can find the optimal solution at that node.

Generally in a B&B method, the incumbent will be improved as the tree search is further explored. Therefore, there is a higher possibility that the situation in which the incumbent is less than the optimal objective value of the Lagrangian dual problem occurs in the second stage as compared to the first stage. Therefore, the use of the incumbent in the subgradient optimization would work better in the second stage.

From these results, a hybrid approach, in which the use of upper bound for determining step size is employed when the gap between the lower bound and the incumbent is still high (as in the first stage) and the use of the incumbent is employed when the gap is small (as in the second stage) would improve the efficiency of the algorithms.

We also considered another B&B algorithm using a strategy adapted from the algorithm proposed by Pirkul [22] for solving the single-source capacitated facility location problem. The order of branching variables in the first and the second stages is determined according to the decreasing order of penalties at the root node of that stage. We branch first on the 0-branch or the 1-branch for the selected branching variable depending on the value of this variable from the Lagrangian relaxation problem corresponding to the lower bound at the root node. The LIFO strategy is used for the candidate problem selection. From our numerical experiment, we found that our proposed algorithm performs better in terms of average number of nodes in the tree and average CPU time.

7. Conclusion

There are some important applications that can be formulated as a two-echelon, single-source, capacitated facility location problem. As such models become large even for moderate-sized problems it is important to have access to efficient methods that can produce optimal or near optimal solution. We have proposed a Lagrangian relaxation-based B&B algorithm where the B&B is performed due to model-based characteristics in three different stages. Special reduction tests have been included. Numerical results for this algorithm on a suit of test problems show that the proposed algorithm is efficient. The size of the B&B tree and CPU time required by the algorithm are significantly smaller than those from a standard LP-based 0–1 integer programming package.
References