European Journal of Operational Research 33 (1988) 314-325
North-Holland

An algorithm for solving large capacitated warehouse location problems

J.E. BEASLEY
Department of Management Science, Imperial College, London SW7 2BX, England

Abstract: In this paper we present a lower bound for the capacitated warehouse location problem based upon lagrangean relaxation of a mixed-integer formulation of the problem. Feasible solution exclusion constraints are used together with problem reduction tests derived from both the original problem and the lagrangean relaxation.

By incorporating the lower bound and the reduction tests into a tree search procedure we are able to solve problems involving up to 500 potential warehouse locations and 1000 customers.

Keyword: Location

1. Introduction

The capacitated warehouse location problem (henceforth denoted by CWLP) is the problem of locating a number of warehouses which have to service a set of customers, at minimum cost, where each customer has an associated demand and there are constraints on the total demand that can be met from a warehouse.

In this paper we present an algorithm based upon lagrangean relaxation of a mixed-integer formulation of the problem. In a previous paper [10] we presented an algorithm which was also based upon lagrangean relaxation of a mixed-integer formulation of the problem. The main differences between this paper and our previous paper [10] are:

(a) a different mixed-integer formulation of the problem

(b) the use of feasible solution exclusion constraints

(c) computational results for very much larger problems than have been reported as solved by any other author.

Received May 1986; revised November 1986

1.1. Literature survey

In our previous paper [10] we briefly reviewed the literature relating to CWLP and so, for reasons of space, we shall only deal here with papers additional to those discussed in [10].

Bartezzaghi, Colorni and Palermo [7] presented a tree search algorithm based upon a lower bound derived from a transportation problem and gave computational results for problems involving up to 12 potential warehouse locations and 40 customers.

Bitran, Chandru, Sempolinski and Shapiro [8] applied a technique called inverse optimisation to the problem and presented computational results for problems involving up to 10 potential warehouse locations and 20 customers.

Baker [3] presented a theoretical analysis related to strengthening the linear programming relaxation of a mixed-integer formulation of the problem. He also presented [4] an algorithm based upon a partial dual of an LP formulation of the problem and gave computational results for problems involving up to 40 potential warehouse locations and 80 customers.

Jacobsen [19] presented a number of heuristics for the problem based on generalisations of heuristics for the uncapacitated warehouse location problem. Domschke and Drexl [11] extended his work to deal with the situation where warehouses have different capacities.

Barcelo and Casanovas [5] also presented a number of heuristics, based upon lagrangean relaxation, for the version of the problem in which each customer can only be supplied from one warehouse (this version of the problem being a special case of the generalised assignment problem [21]).

Guignard-Spielberg and Kim [16] presented a lower bound for the problem based upon lagrangean relaxation. Computational results are reported for problems involving up to 20 potential warehouse locations and 35 customers.

Barcelo [6] presented an algorithm based upon the automatic generation of cutting planes in an attempt to move the solution of the linear programming relaxation of the problem closer to the optimal mixed-integer solution. Computational results are reported for problems involving up to 30 potential warehouse locations and 50 customers.

Van Roy [24] presented an algorithm based upon cross decomposition and gave computational results for problems involving up to 100 potential warehouse locations and 200 customers.

For a complete survey of work relating to CWLP see Aikens [1], Domschke and Drexl [12] and Sridharan [22]. We would also like to mention here the work of Suhl [23] relating to solving large-scale mixed-integer programs with fixed charge variables which can be applied to solving CWLP.

2. Problem formulation

We formulate CWLP as a mixed-integer program. Let:

m	be the number of potential warehouse locations,
n	be the number of customers,
q_j	be the demand of customer j,
c_{ij}	be the cost of supplying all of the demand of customer j from warehouse i with $c_{ij} \geqslant 0$,
Q_i	be the capacity of warehouse i (i.e. the limit on the total demand that can be supplied from warehouse i),
L_i	be the lower limit on the demand that must be supplied from warehouse i (if it is open),
P_L	be a lower limit on the number of open warehouses,
P_U	be an upper limit on the number of open warehouses,
F_i	be the fixed cost associated with opening warehouse i,
x_{ij}	be the fraction of the demand of customer j supplied from warehouse i (so that $0 \leqslant x_{ij} \leqslant 1$),
y_i	$= 1$ if warehouse i is open, $= 0$ otherwise.

We also introduce into the problem an artificial customer (customer 0) with $c_{i0} = 0$, $i = 1, \ldots, m$, where q_0 represents a 'mopping-up' of the difference between the total capacity of the open warehouses and the total customer demand (i.e. $0 \leqslant q_0 \leqslant \sum_{i=1}^m Q_i y_i - \sum_{j=1}^n q_j$). Whilst initially $q_0 = 0$ we will see later that we can update q_0 during the course of the solution procedure. Hence our formulation of the problem is:

$$\min \quad \sum_{i=1}^{m} \sum_{j=0}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} F_i y_i, \tag{1}$$

$$\text{s.t.} \quad \sum_{i=1}^{m} x_{ij} \geqslant 1, \qquad j = 0, 1, \ldots, n, \tag{2}$$

$$\sum_{j=0}^{n} q_j x_{ij} \leqslant Q_i y_i, \qquad i = 1, \ldots, m, \tag{3}$$

$$\sum_{j=0}^{n} q_j x_{ij} \geqslant L_i y_i, \qquad i = 1, \ldots, m, \tag{4}$$

$$P_L \leqslant \sum_{i=1}^{m} y_i \leqslant P_U, \tag{5}$$

$$x_{ij} \leqslant \min(1, Q_i/q_j) y_i, \quad i = 1, \ldots, m, \quad j = 0, 1, \ldots, n, \tag{6}$$

$$0 \leqslant x_{ij} \leqslant 1, \quad i = 1, \ldots, m, \quad j = 0, 1, \ldots, n, \tag{7}$$

$$y_i \in (0, 1), \qquad i = 1, \ldots, m. \tag{8}$$

Equation (2) ensures that the demand of every customer is satisfied. Equations (3) and (4) ensure that no customer can be supplied from a closed warehouse and that the total demand supplied from a warehouse lies within the limits imposed upon it. Equation (5) ensures that the number of open warehouses lies between P_L and P_U. Equation (6) relates the amount supplied to an individual customer to the effective capacity of a warehouse. Equation (7) provides bounds on the allocation variables (x_{ij}) whilst equation (8) is the integrality constraint. It is clear that this formulation is a large mixed-integer program.

Equation (4) and equation (5) are not usually found in a formulation of CWLP but are included here since we can (as discussed in [10] and also later) update L_i $(i = 1, \ldots, m)$, P_L and P_U during the course of the solution procedure, thereby tightening our formulation of CWLP.

With reference to equation (6) we would expect that for most problems q_j will be less than Q_i and in such cases the constraint $x_{ij} \leqslant \min(1, Q_i/q_j) y_i$ reduces to $x_{ij} \leqslant y_i$ (the usual constraint). However, since it is not *necessarily* true that q_j will be less than Q_i (nor that q_0 (as updated during the course of the solution procedure) will be less than Q_i) we, for completeness, use equation (6) instead of the usual constraint $x_{ij} \leqslant y_i$.

Whilst the formulation of the problem given above is adequate we can enlarge it in a way that should prove computationally effective. In addition, as is explained below, this approach (called *feasible solution exclusion*) is generally applicable to any mixed-integer (or pure integer) program.

3. Feasible solution exclusion

It is well-known that for any mixed-integer program it is sufficient, in specifying feasible solutions to the program, merely to give values for the integer variables, since with these fixed the mixed-integer program reduces to a linear program which can be solved to generate values for the continuous (non-integer) variables.

Hence, in the context of CWLP, suppose that we have (somehow) found W feasible solutions with the wth feasible solution being denoted by E_w where

$$E_w = \{ i \mid y_i = 1,\ i = 1, \ldots, m \} \tag{9}$$

i.e. E_w consists of the set of warehouses which are open in the wth feasible solution.

Now to eliminate feasible solution E_w from the problem we add the constraint

$$\sum_{i \notin E_w} y_i + \sum_{i \in E_w} (1 - y_i) \geqslant 1. \tag{10}$$

This constraints excludes feasible solution E_w since that solution violates the constraint but all other feasible solutions satisfy it, since all other feasible solutions must differ from E_w either by some y_i $(i \notin E_w)$ being one and/or by some y_i $(i \in E_w)$ being zero.

Rearranging equation (10) and generalising to exclude all W feasible solutions we have the set of constraints

$$\sum_{i \in E_w} y_i - \sum_{i \notin E_w} y_i \leqslant |E_w| - 1, \quad w = 1, \ldots, W. \tag{11}$$

It is plain that such *feasible solution exclusion* constraints can be written for any mixed-integer, or pure integer, program. (Although we have dealt here only with zero-one integer variables it is well-known that any general integer variable can be written as the weighted sum of zero–one variables.)

Recall now that, in general, mixed-integer (or pure integer) programs are solved by tree search procedures. Often, as for example in the algorithm we develop later for CWLP, a number of feasible solutions are available before the start of the tree search (e.g. from heuristics for the problem). If feasible solution exclusion constraints are applied throughout the tree search then it would seem reasonable to suppose that the extent of that search will be reduced compared to what it would otherwise have been. The algorithm we develop below illustrates the use of feasible solution exclusion constraints.

4. The lower bound

In order to solve CWLP we develop a lower bound, for use in a tree search procedure, from a Lagrangean relaxation of the problem.

We are going to relax equations (2), (3), (4) and (11) in a lagrangean fashion. We found it computationally advantageous to scale these equations so that they are all $\geqslant$ equations and their right-hand sides have a maximum (absolute) value of one.

Hence with multipliers s_j $(\geqslant 0,\ j = 0, 1, \ldots, n)$, t_i $(\geqslant 0,\ i = 1, \ldots, m)$, u_i $(\geqslant 0,\ i = 1, \ldots, m)$ and v_w $(\geqslant 0,\ w = 1, \ldots, W)$ for (scaled) equations (2), (3), (4) and (11) respectively we have the lagrangean dual program

$$\min \quad \sum_{i=1}^{m} \sum_{j=0}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} F_i y_i$$
$$+ \sum_{j=0}^{n} s_j \left[1 - \sum_{i=1}^{m} x_{ij} \right]$$

$$+ \sum_{i=1}^{m} t_i \left[-y_i + \sum_{j=0}^{n} (q_j/Q_i) x_{ij} \right]$$
$$+ \sum_{i=1}^{m} u_i \left[y_i - \sum_{j=0}^{n} (q_j/L_i) x_{ij} \right]$$
$$+ \sum_{w=1}^{W} v_w \left[-1 + \left(\sum_{i \in E_w} y_i - \sum_{i \notin E_w} y_i \right) \Big/ (|E_w| - 1) \right], \qquad (12)$$

s.t. (5), (6), (7) and (8).

Note here that we require $Q_i > 0$ (trivially true), $L_i > 0$ (always true, see equation (40)) and $|E_w| \geqslant 2$, (true if we arbitrarily exclude from $\{E_w \mid w = 1,\dots,W\}$ any feasible solutions with $|E_w| = 1$). Define

$$A_i = \{w \mid y_i \in E_w,\ w = 1,\dots,W\}, \qquad (13)$$
$$C_{ij} = c_{ij} - s_j + t_i(q_j/Q_i) - u_i(q_j/L_i), \qquad (14)$$
$$f_i = F_i - t_i + u_i + \sum_{w \in A_i} v_w(|E_w| - 1)^{-1} - \sum_{w \notin A_i} v_w(|E_w| - 1)^{-1} \qquad (15)$$

whereupon the lagrangean dual program becomes

$$\min \sum_{i=1}^{m} \sum_{j=0}^{n} C_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i + \sum_{j=0}^{n} s_j - \sum_{w=1}^{W} v_w, \qquad (16)$$

s.t. (5), (6), (7) and (8).

This lagrangean dual problem is easily solved. Consider the effect of setting $y_i = 1$ (i.e. opening warehouse i), then the contribution from warehouse i to the lagrangean dual objective function (equation (16)) is given by

$$a_i = f_i + \sum_{j=0}^{n} \min[0,\ C_{ij} \min(1,\ Q_i/q_j)]. \qquad (17)$$

Hence the lagrangean dual program reduces to

$$\min \sum_{i=1}^{m} a_i y_i + \sum_{j=0}^{n} s_j - \sum_{w=1}^{W} v_w, \qquad (18)$$

$$\text{s.t.} \quad P_L \leqslant \sum_{i=1}^{m} y_i \leqslant P_U, \qquad (19)$$

$$y_i \in (0, 1), \quad i = 1,\dots,m. \qquad (20)$$

This zero-one program can be solved by inspection as follows:

(a) form a list of the warehouses arranged in ascending a_i order,

(b) set $y_i = 1$ for the first P_L warehouses in this list,

(c) if $P_L = P_U$ go to step (e),

(d) proceed along the list from the $(P_L + 1)$th warehouse setting $y_i = 1$ until either (i) have opened P_U warehouses or (ii) the a_i values become positive,

(e) let the set of values for the (y_i) as decided above be denoted by (Y_i) then the corresponding values for the (x_{ij}) are given by $X_{ij} = \min(1,\ Q_i/q_j)$ if $C_{ij} \leqslant 0$ and $Y_i = 1$ ($X_{ij} = 0$ otherwise).

Hence the optimal value of the lagrangean dual program (Z_D) is given by

$$Z_D = \sum_{i=1}^{m} a_i Y_i + \sum_{j=0}^{n} s_j - \sum_{w=1}^{W} v_w \qquad (21)$$

and this value is a lower bound on the optimal solution of the original mixed-integer formulation (equations (1)–(8), (11)) of CWLP.

The lagrangean dual program (equations (18)–(20)) has the integrality property (see Geoffrion [15]) which means that the maximum lower bound that can be achieved by that program is equal to the value of the linear programming relaxation of the original formulation.

The lagrangean dual program developed in our previous paper [10] also had this property, and therefore since the formulation of CWLP given in this paper is stronger than the formulation of CWLP given in that paper the bound developed in this paper dominates (theoretically) the bound developed in [10].

The computational effort required to solve the lagrangean dual program given in [10] is much greater than the computational effort required to solve the lagrangean dual program given above. Hence we would expect the algorithm developed in this paper to be both theoretically and computationally superior to the algorithm given in [10].

5. Problem reduction

There are a number of tests that can be used to reduce the size of the problem (typically in terms of the number of potential warehouse locations) that we need to consider. In this section we will outline those that we found most effective. Note here that a number of these tests were presented in [10] but are given here for completeness.

Let Z_{UB} be an upper bound on the optimal solution of CWLP and define

$$K = \{i \mid i = 1, \ldots, m\}, \tag{22}$$

$$K_1 = \{i \mid y_i = 1,\ i = 1, \ldots, m\}, \tag{23}$$

$$K_2 = \{i \mid y_i = 0,\ i = 1, \ldots, m\}, \tag{24}$$

$$M = \{i \mid Y_i = 1,\ i = 1, \ldots, m\}, \tag{25}$$

i.e. K is the set of warehouses, K_1/K_2 the set of warehouses that have been opened/closed at the current stage in the procedure and M the set of warehouses that are open in the current lagrangean dual solution. Note here that modifying the solution to the lagrangean dual program (equations (18)–(20)) to deal with K_1 and K_2 is trivial and will not be given here. Then the problem reduction tests are:

(1) *Opening warehouses*

The lower bound that results if warehouse i $(i \in K - M - K_2)$ is opened is given by

$$Z_{\mathrm{D}} + a_i - \max(a_j \mid j \in M - K_1) \quad \text{if } |M| = P_{\mathrm{U}}, \tag{26}$$

$$Z_{\mathrm{D}} + a_i - \max(0, \max(a_j \mid j \in M - K_1)) \quad \text{if } |M| \neq P_{\mathrm{U}}. \tag{27}$$

If this lower bound exceeds Z_{UB} then it is clear that warehouse i cannot be opened.

(2) *Closing warehouses*

The lower bound that results if warehouse i $(i \in M - K_1)$ is closed is given by

$$Z_{\mathrm{D}} - a_i + \min(a_j \mid j \in K - M - K_2) \quad \text{if } |M| = P_{\mathrm{L}}, \tag{28}$$

$$Z_{\mathrm{D}} - a_i + \min(0, \min(a_j \mid j \in K - M - K_2)) \quad \text{if } |M| \neq P_{\mathrm{L}}. \tag{29}$$

If this lower bound exceeds Z_{UB} then it is clear that warehouse i cannot be closed.

(3) *Number of open warehouses*

The lower bound that results if there are exactly p open warehouses is given by

$$Z_{\mathrm{D}} - \sum_{i \in M - K_1} a_i + \text{the } p - |K_1| \text{ smallest } a_j \quad (j \in K - K_1 - K_2). \tag{30}$$

If this value exceeds Z_{UB} when $p = P_{\mathrm{L}}$ $(p = P_{\mathrm{U}})$ then we increase P_{L} (decrease P_{U}) by one.

(4) *Artificial customer demand*

We can update the artificial customer demand (q_0) in the following way. Resolve the lagrangean dual program with $v_w = 0$ $(w = 1, \ldots, W)$ and then consider the following zero–one integer program:

$$\min \quad \sum_{i=1}^{m} Q_i y_i - \sum_{j=1}^{n} q_j, \tag{31}$$

$$\text{s.t.} \quad P_{\mathrm{L}} \leqslant \sum_{i=1}^{m} y_i \leqslant P_{\mathrm{U}}, \tag{32}$$

$$\sum_{i=1}^{m} Q_i y_i \geqslant \sum_{j=0}^{n} q_j, \tag{33}$$

$$\sum_{i=1}^{m} a_i y_i + \sum_{j=0}^{n} s_j \leqslant Z_{\mathrm{UB}}, \tag{34}$$

$$\sum_{i \in E_w} y_i - \sum_{i \notin E_w} y_i \leqslant |E_w| - 1, \quad w = 1, \ldots, W, \tag{35}$$

$$y_i = 1, \quad \forall i \in K_1, \tag{36}$$

$$y_i = 0, \quad \forall i \in K_2, \tag{37}$$

$$y_i \in (0, 1), \quad i = 1, \ldots, m. \tag{38}$$

Equations (32) and (35) are as before (equations (5) and (11) respectively). Equation (33) ensures that the total capacity of the open warehouses exceeds the total customer demand (including the artificial customer) whilst equation (34) ensures that the lower bound associated with the open warehouses is less than the current upper bound. Equations (36) and (37) ensure that warehouses are fixed open/closed as appropriate whilst equation (38) is the integrality constraint.

It is clear that the optimal value of this zero–one integer program can be used to update q_0 (recall here that q_0 represents a 'mopping-up' of the difference between the total capacity of the open warehouses and the total customer demand).

In practice we (for computational reasons) solved the linear programming relaxation of the above program and used the optimal value of that relaxation to update q_0. Note here that increasing q_0 may increase the value of the linear programming relaxation of our original formulation of the problem (equations (1)–(8), (11)) and hence increase the maximum lower bound attainable from the lagrangean dual problem (equations (18)–(20)).

(5) *Warehouse capacity lower limit*

In order to update L_i, the minimum total demand that is supplied from warehouse i (if it is open), we define

$$B_i = \{ j \mid c_{ij} < c_{kj}, \forall k \in K - K_2 - \{i\}, \quad j = 1, \ldots, n \} \tag{39}$$

i.e. B_i represents the set of customers whom it is cheaper to supply from warehouse i than from any other warehouse. Hence we can update L_i using

$$L_i = \max\left(1, \min\left(Q_i, \sum_{j \in B_i} q_j\right)\right) \tag{40}$$

where the max(1,...) term arises since we assume that if warehouse i is open then at least one unit of demand is supplied from it.

(6) *Allocation saving*

Define d_{ij} as the minimum cost of allocating all of the demand of customer j to a warehouse other than i. Then

$$d_{ij} = \min(c_{kj} \mid k \in K - K_2 - \{i\}) \tag{41}$$

Consider the following linear program

$$\max \quad \sum_{j \in B_i} (d_{ij} - c_{ij}) x_{ij} - F_i, \tag{42}$$

$$\text{s.t.} \quad \sum_{j \in B_i} q_j x_{ij} \leqslant Q_i, \tag{43}$$

$$0 \leqslant x_{ij} \leqslant 1, \quad \forall j \in B_i, \tag{44}$$

where $(d_{ij} - c_{ij})$ represents the additional cost of supplying a customer j ($\in B_i$) from other than its cheapest warehouse i. If the optimal value (G_i) of this linear program exceeds zero then warehouse i should be opened. The above program is the linear programming relaxation of a knapsack problem and can be easily solved.

Note here that, as discussed in [10], this reduction test *cannot* be used if the optimal solution to CWLP is changed by the removal of the constraint on the minimum amount supplied from a warehouse (equation (4)) or by the removal of the constraint upon the number of open warehouses (equation (5)). We shall assume for the rest of this paper that the removal of these two constraints does not affect the optimal solution to CWLP.

6. The subgradient procedure

Subgradient optimisation was used in an attempt to maximise the lower bound obtained from the lagrangean dual program. The procedure we adopted was as follows:

(1) Set initial values for the multipliers. We used $t_i = u_i = 0$, $i = 1, \ldots, m$, and $v_w = 0$, $w = 1, \ldots, W$, with the values of the s_j being given by

$$s_j = \min(d_{ij} \mid i \in K - K_2), \quad j = 0, 1, \ldots, n. \tag{45}$$

(2) Solve the lagrangean dual program with the current set of multipliers and, as before, let the solution be Z_D, (X_{ij}), (Y_i).

(3) Update the maximum lower bound found ($Z_{\max}$) with Z_D.

(4) Stop if $Z_{UB} = Z_{\max}$ (where Z_{UB} an upper bound on the problem corresponding to a feasible solution) since then Z_{UB} is the optimal solution, else go to step (5).

(5) Calculate the subgradients

$$S_j = 1 - \sum_{i=1}^{m} X_{ij}, \quad j = 0, 1, \ldots, n, \tag{46}$$

$$T_i = -Y_i + \sum_{j=0}^{n} (q_j/Q_i) X_{ij}, \quad i = 1, \ldots, m, \tag{47}$$

$$U_i = Y_i - \sum_{j=0}^{n} (q_j/L_i) X_{ij}, \quad i = 1, \ldots, m, \tag{48}$$

$$V_w = -1 + \left(\sum_{i \in E_w} Y_i - \sum_{i \notin E_w} Y_i \right) / (|E_w| - 1), \quad w = 1, \ldots, W, \tag{49}$$

We found it computationally useful to adjust the subgradients, if they were not going to effectively contribute to the update of the multipliers, in the following manner:

$$S_j = 0 \quad \text{if } s_j = 0 \text{ and } S_j < 0, \quad j = 0, 1, \ldots, n, \tag{50}$$

$$T_i = 0 \quad \text{if } t_i = 0 \text{ and } T_i < 0, \quad i = 1, \ldots, m, \tag{51}$$

$$U_i = 0 \quad \text{if } u_i = 0 \text{ and } U_i < 0, \quad i = 1, \ldots, m, \tag{52}$$

$$V_w = 0 \quad \text{if } v_w = 0 \text{ and } V_w < 0, \quad w = 1, \ldots, W. \tag{53}$$

(6) Define a step size e by

$$e = f(Z_{\text{UB}} - Z_{\text{D}}) \Big/ \left(\sum_{j=0}^{n} (S_j)^2 + \sum_{i=1}^{m} (T_i)^2 + \sum_{i=1}^{m} (U_i)^2 + \sum_{w=1}^{W} (V_w)^2 \right) \tag{54}$$

where $0 \leqslant f \leqslant 2$ and update the multipliers by

$$s_j = \max(0, s_j + eS_j), \quad j = 0, 1, \ldots, n, \tag{55}$$

$$t_i = \max(0, t_i + eT_i), \quad i = 1, \ldots, m, \tag{56}$$

$$u_i = \max(0, u_i + eU_i), \quad i = 1, \ldots, m, \tag{57}$$

$$v_w = \max(0, v_w + eV_w), \quad w = 1, \ldots, W. \tag{58}$$

(7) Go to step (2) to resolve the lagrangean dual program with this new set of multipliers unless sufficient subgradient iterations have been performed, in which case stop.

In deciding a value for f (equation (54)) we followed the approach of Fisher [13,14] in setting $f = 2$ initially and halving f if the best lower bound ($Z_{\max}$) did not increase within 30 subgradient iterations. The subgradient procedure was stopped when f fell below 0.0005. We also found it useful to use a target value in equation (54) 5% above the upper bound Z_{UB}.

7. The tree search procedure

In order to solve CWLP we used a binary depth-first tree search procedure, computing a lower bound at each tree node via the lagrangean relaxation and the subgradient procedure. The details of the tree search procedure are given below.

7.1. Initial tree node

(1) *Reduction*

We first reduced the problem by carrying out the warehouse capacity lower limit and allocation saving tests given before.

(2) *Upper bound*

In order to generate an initial feasible solution Z_{UB} corresponding to an upper bound on the problem we note that once a set of open warehouses has been specified the problem (equations (1)–(3), (6)–(8)) reduces to a simple network flow (transportation) problem which can be easily solved.

Hence in order to generate Z_{UB} we first created a list of warehouses sorted into descending G_i (equations (42)–(44)) order and ran down this list:

(a) opening the next warehouse in the list,

(b) solving the associated network flow problem to update Z_{UB},

(c) repeating steps (a) and (b) until we find that the solution value from step (b) begins to increase.

Note here that all feasible solutions found are added to $\{E_w\}$ for use in feasible solution exclusion constraints.

On a computational note the network flow code that we used was a primal simplex procedure based upon the work of Jacobsen [17] and Branman [9].

(3) *Subgradient ascent*

The subgradient procedure was then carried out where, at each subgradient iteration, we carried out the reduction tests relating to opening/closing warehouses and the number of open warehouses. Each time f (equation (54)) was halved we recalled the multipliers associated with the best lower bound ($Z_{\max}$) found so far, resolved the lagrangean dual program with that set of multipliers and attempted to find a feasible solution based upon the set of warehouses M (equation (25)) open in that lagrangean solution. At the same time we performed the reduction test associated with artificial customer demand and, in the event that some warehouses had been closed since f was last halved, performed the reduction tests relating to warehouse capacity lower limit and allocation saving.

(4) *Reduction*

At the end of the subgradient procedure the set of multipliers associated with the best lower bound found were recalled, the lagrangean dual program resolved with that set of multipliers and all the reduction tests given before performed until no further reduction could be achieved.

7.2. Other tree nodes

(1) *Lower bound*

At each tree node we started with an initial set of multipliers equal to the set associated with the best lower bound found at the predecessor tree node. Subgradient iterations were carried out in the same manner as at the initial tree node except that:

(a) we started with a value of f (equation (54)) equal to 0.125,

(b) 30 subgradient iterations were carried out, with f being halved every 10 iterations, at tree nodes associated with forward branching, but both these figures were doubled for tree nodes associated with backtracking,

(c) the reduction tests relating to the artificial customer demand and warehouse capacity lower limit were performed before any subgradient ascent,

(d) the allocation saving reduction test was not used, nor did we attempt to find a feasible solution from the lagrangean dual solution.

(2) *Branching*

In forward branching we adopted a mixed branching strategy, branching both by opening warehouses and adjusting P_L and P_U in the following manner:

(a) if $|K_1| = P_L$ and $P_U \neq P_L$ then branch by setting $P_U = P_L$,

(b) otherwise branch by opening warehouse i where

$$a_i = \min(a_j \mid j \in K - K_1 - K_2). \qquad (59)$$

The reasoning behind (a) above is that the forward branch will create a terminal node (a node at which all warehouses have been fixed open/closed) corresponding to a solution with P_L warehouses open much faster than such a node would be reached if we continued to branch using just (b) above.

(3) *Terminal nodes*

Early computational experience indicated that, especially for large problems, the resolution of the network flow problem associated with the set of open warehouses at a terminal node was relatively time-consuming. Hence to avoid unnecessary computation we, at each terminal node, before solving the network flow problem:

(a) checked $\{E_w \mid w = 1, \ldots, W\}$ to see if the set of open warehouses had been encountered (and solved) before; and, if not:

(b) performed a subgradient ascent (in the same manner as for a tree node associated with backtracking) in an attempt to show that the terminal node could not lead to an improved feasible solution.

(4) *Backtracking*

We can backtrack in the tree when $Z_{\max} > Z_{UB}$ or when the artificial customer demand reduction test indicates that there is no feasible completion of the current tree node.

8. Computational results

The algorithm presented in this paper was programmed in FORTRAN and run on a Cray-1S using the CFT compiler (with maximum optimisation) for a number of test problems drawn from the literature and some randomly generated problems.

The Cray-1S is an example of what is called a 'supercomputer' in that, for certain calculations, it is much faster than conventional computers. For example, adding together two one-dimensional vectors, each of length h, would involve $O(h)$ operations on a conventional computer but would involve only (essentially) $O(1)$ operations on the Cray-1S. This ability to substantially speed up certain vector calculations is known as the vector processing capability of the Cray-1S.

The primary use made of this vector processing ability in the algorithm presented in this paper was in relation to the calculation of the lower bound and subgradients and in our implementation of the network flow code of Jacobsen [17] and Branman [9].

Table 1 presents a summary of the test problems, with problem sets IV to XIII being taken from the literature (see [2,10]) and problem sets A to E being randomly generated.

The random problems were generated by locating the customers/warehouses within a 1000 by 1000 Euclidean square and taking the cost per unit of demand supplied as proportional to the Euclidean distance between the customer and the warehouse multiplied by a real random number from the range [1.00, 1.25]. For each customer the demand was a randomly generated integer in the

Table 1
Test problem details

Problem set	Number of potential warehouse locations (m)	Number of customers (n)	Warehouse capacity	Desired number of open warehouses (P_0)	Fixed cost per warehouse ($'000)
IV	16	50	5000	–	7.5/12.5/17.5/25.0
V	16	50	10000	–	17.5
VI	16	50	15000	–	7.5/12.5/17.5/25.0
VII	16	50	58268	–	7.5/12.5/17.5/25.0
VIII	25	50	5000	–	7.5/12.5/17.5/25.0
IX	25	50	15000	–	7.5/12.5/17.5/25.0
X	25	50	58268	–	7.5/12.5/17.5/25.0
XI	50	50	5000	–	7.5/12.5/17.5/25.0
XII	50	50	15000	–	7.5/12.5/17.5/25.0
XIII	50	50	58268	–	7.5/12.5/17.5/25.0
A	100	1000	8000/10000/12000/14000	5	Random
B	100	1000	5000/6000/7000/8000	10	Random
C	100	1000	5000/5750/6500/7250	15	Random
D	500	1000	7000/8000/9000/10000	5	Random
E	500	1000	5000/6000/7000/8000	10	Random

range [1, 100]. Each warehouse i had the same upper capacity limit (shown in Table 1) with $L_i = 0$. To generate a fixed cost F_i for warehouse i each problem had specified a number P_0 (shown in Table 1) equal to the desired number of open warehouses in the optimal solution. We then ran-

Table 2
Results for problems not requiring branching

Problem number	Number of subgradient iterations	Network flow problems		Number of open warehouses in optimal solution	Total time Cray-1S seconds
		Number solved	Total time Cray-1S seconds		
IV–1	506	9	0.6	13	1.2
–2	338	9	0.5	12	1.0
–4	333	8	0.4	12	0.9
V–1	924	10	0.2	8	1.2
VI–1	202	4	0.1	11	0.3
–2	221	8	0.1	9	0.5
–4	167	7	0.1	5	0.3
VII–1	144	4	0.0	11	0.2
–2	218	7	0.0	9	0.2
–3	308	10	0.0	5	0.2
–4	118	5	0.0	4	0.1
IX–1	268	8	0.3	15	0.8
–2	344	11	0.3	11	0.9
X–1	339	10	0.0	15	0.4
–2	380	11	0.0	11	0.4
–3	413	12	0.0	8	0.5
–4	145	8	0.0	4	0.2
XII–1	315	13	0.4	15	1.5
XIII–1	365	14	0.0	15	0.8
–2	395	9	0.0	11	0.6
–3	341	12	0.0	8	0.6
–4	415	12	0.0	4	0.7
A–4	313	8	17.9	4	28.1
D–3	350	11	39.6	6	105.4
–4	556	11	43.6	6	139.2

Table 3
Results for problems requiring branching

Problem number	Initial tree node							Number of tree nodes	Network flow problems		Number of open warehouses in optimal solution	Total time Cray-1S seconds
	Final P_L	Final P_U	Number fixed open	Number undecided	Duality gap (%)	Number of subgradient iterations	Total time Cray-1S seconds		Number solved	Total time Cray-1S seconds		
IV-3	12	13	8	8	0.021	965	1.3	13	7	0.4	12	1.9
VI-3	5	8	4	5	0.132	693	0.9	7	12	0.2	7	1.1
VIII-1	13	19	11	12	0.181	626	1.9	33	19	0.8	17	3.8
-2	14	16	13	4	0.062	696	1.5	5	12	0.5	14	1.8
-3	13	15	10	5	0.061	588	1.4	9	12	0.5	14	1.8
-4	12	16	3	21	0.202	757	2.0	41	20	0.7	13	4.0
IX-3	5	10	4	7	0.179	1148	1.6	23	20	0.3	8	2.4
-4	6	10	3	11	0.399	925	1.5	57	16	0.3	7	3.1
XI-1	16	17	15	2	0.034	637	2.0	3	15	0.7	17	2.4
-2	13	17	9	10	0.135	663	2.2	37	20	0.9	15	4.3
-3	13	16	6	14	0.261	1025	2.8	129	25	0.9	14	8.7
-4	12	16	4	26	0.261	774	2.6	151	22	0.7	13	8.8
XII-2	10	12	9	5	0.090	661	1.7	11	17	0.4	11	2.3
-3	6	13	3	20	0.123	569	1.7	49	27	0.5	9	3.8
-4	5	11	2	23	0.427	832	2.3	175	21	0.3	7	7.5
A-1	7	7	0	33	0.336	557	40.5	43	25	58.9	7	87.4
-2	6	6	0	33	0.264	846	53.2	39	20	45.0	6	79.9
-3	5	5	0	27	0.312	736	41.8	27	17	34.1	5	59.3
B-1	11	11	6	10	0.053	961	51.6	7	10	53.3	11	74.2
-2	9	10	0	93	0.731	695	68.9	309	68	200.6	9	321.6
-3	8	11	0	98	0.652	613	58.4	257	61	147.3	8	244.9
-4	7	9	0	41	0.558	1546	55.6	47	19	50.1	7	89.5
C-1	11	11	3	21	0.178	737	72.7	59	27	113.1	11	150.6
-2	9	20	0	96	0.533	669	74.3	237	64	199.7	10	294.7
-3	8	17	0	46	0.159	701	71.3	45	21	72.8	9	108.7
-4	8	14	4	18	0.051	795	76.1	7	15	63.5	9	87.7
D-1	8	8	0	88	0.100	607	118.2	41	21	50.9	8	158.8
-2	7	7	0	81	0.308	871	137.7	119	40	97.6	7	237.1
E-1	11	12	0	482	0.453	767	189.9	843	143	498.5	11	1024.5
-2	9	10	0	336	0.507	1209	191.2	279	87	239.0	9	541.7
-3	8	22	0	500	0.338	786	192.6	159	48	144.1	8	406.4
-4	7	20	0	467	0.123	954	212.0	37	19	68.5	8	271.1

domly generated a set containing $(P_0 + 1)$ warehouses and evaluated the cost of this set assuming each customer was allocated to its cheapest warehouse (giving a cost D_1 say). Taking two (random) warehouses away from this set we evaluated the cost of the remaining $(P_0 - 1)$ warehouses in a similar way (giving a cost D_2 say). Each fixed cost F_i was then set equal to $(D_2 - D_1)/2$ multiplied by a real random number from the range [0.75, 1.25]. The reasoning behind this was to attempt to ensure that the final optimal solution did contain (roughly) P_0 open warehouses. Note however that in deciding initial values for P_L and P_U for each problem we did not use P_0 at all, but set P_U equal to m and set P_L equal to the minimum number of warehouses needed to supply all the customers.

The linear programming relaxation associated with the zero-one integer program for artificial customer demand (equations (31)–(38)) was solved using the NAG library routine E04MBF.

Table 2 gives the results for the problems that terminated at the initial tree node and so did not require branching (as is common we use IV-1 to refer to the first problem in set IV (i.e. with a fixed cost of 7500 for all warehouses), IV-2 to refer to the second problem (fixed cost 12500) etc). Table 3 gives the results for the problems that did require branching where the duality gap (%) at the initial tree node is given by 100 (optimal value $- Z_{\max}$)/(optimal value).

Comparing Tables 2 and 3 with the corresponding tables in [10] (but see the appendix for a discussion of the validity of this comparison) the algorithm presented in this paper achieves duality gaps as small, or smaller, than those achieved by the algorithm given in [10] in all but 6 of the 37 test problems (problems IV-1 to XIII-4). The results for problem sets A to E indicate that the algorithm is capable of solving very large problems provided that the number of open warehouses is not too large (note how the time required to solve the network flow problems encountered during the course of the procedure becomes a significant component of the total computer time as the number of open warehouses increases).

Comparing the results given in this paper with those given in [10] and in Van Roy [24] it is clear that the algorithm developed by Van Roy is (computationally) the most effective algorithm known for small problems. Whether his algorithm retains its computational advantage for the very large problems considered in this paper is currently an open question.

Appendix

It is well-known that there have been small discrepancies between the computational results of different authors for the standard test problems (problem sets IV to X in Table 1) used in this paper. For example, Jacobsen [18] in a in a private communication highlighted a small difference between his results [19] and ours [10]. Recently Whitaker [25] noted that there are discrepancies in the original table of data [20] on which these test problems are based. This prompted us to examine that data and hence to identify the following discrepancies:

(a) Jacksonville to Portland given as 3184 and 3148, we use in this paper 3184.

(b) Memphis to Mobile given as 384 and 394, we use in this paper 394.

(c) Richmond to Oklahoma City given as 1390 and 3190, we use in this paper 1390.

In addition to the above we also detected some minor errors of our own making in the data used in our previous work [10]. This means that (strictly) the computational results presented in this paper are not comparable with those given in [10] since different data has been used. However, we believe that any differences that would arise are likely to be small and so in the text we have made comparison between this paper and [10].

References

[1] Aikens, C.H., "Facility location models for distribution planning", *European Journal of Operational Research* 22 (1985) 263–279.

[2] Akinc, U., and Khumawala, B.M., "An efficient branch and bound algorithm for the capacitated warehouse location problem", *Management Science* 23 (1977) 585–594.

[3] Baker, B.M., "Linear relaxations of the capacitated warehouse location problem", *Journal of the Operational Research Society* 33(5) (1982) 475–479.

[4] Baker, B.M., "A partial dual algorithm for the capacitated warehouse location problem", *European Journal of Operational Research* 23 (1986) 48–56.

[5] Barcelo, J., and Casanovas, J., "A heuristic lagrangean algorithm for the capacitated plant location problem", *European Journal of Operational Research* 15 (1984) 212–226.

[6] Barcelo, J., "Computational experiments with location problems using automatic constraint generation", Report RR85/06, Universitat Politecnica de Barcelona, Dept. d'Investigacio Operativa i Estadistica, Facultat d'Informatica, Jordi Girona Salgado, 31, 08034 Barcelona, Spain, 1985.

[7] Bartezzaghi, E., Colorni, A. and Palermo, P.C., "A tree search algorithm for plant location problems", *European Journal of Operational Research* 7 (1981) 371–379.

[8] Bitran, G.R., Chandru, V., Sempolinski, D.E. and Shapiro, J.F., "Inverse optimisation: an application to the capacitated plant location problem", *Management Science* 27(10) (1981) 1120–1141.

[9] Branman, J., "A note on pivoting in transportation codes", *European Journal of Operational Research* 2 (1978) 377–378.

[10] Christofides, N., and Beasley, J.E., "Extensions to a Lagrangean relaxation approach for the capacitated warehouse location problem", *European Journal of Operational Research* 12 (1983) 19–28.

[11] Domschke, W. and Drexl, A., "ADD-heuristics' starting procedures for capacitated plant location models", *European Journal of Operational Research* 21 (1985) 47–53.

[12] Domschke, W. and Drexl, A., *Location and Layout planning: An International Bibliography*", Volume 238 in: Lecture Notes in Economics and Mathematical Systems, Springer, Berlin, 1985.

[13] Fisher, M.L., "The lagrangean relaxation method for solving integer programming problems", Management Science 27 (1981) 1–18.

[14] Fisher, M.L., "An applications oriented guide to lagrangean relaxation", *Interfaces* 15(2) (1985) 10–21.

[15] Geoffr A.M., "Lagrangean relaxation and its uses in integer programming", *Mathematical Programming Study* 2 (1974) 82–114.

[16] Guignard-Spielberg M. and Kim, S., "A strong lagrangean relaxation for capacitated plant location problems", Department of Statistics Technical Report No. 56, The Wharton School, University of Pennsylvania, 1983.

[17] Jacobsen, S.K., "On the use of tree-indexing methods in transportation algorithms", *European Journal of Operational Research* 2 (1978) 54–65.

[18] Jacobsen, S.K., Private communication, 1982.

[19] Jacobsen, S.K., "Heuristics for the capacitated plant location model", *European Journal of Operational Research* 12 (1983) 253–261.

[20] Rand-McNally, *Rand-McNally Cosmopolitan World Atlas*, Rand-McNally and Company, Chicago, 1951.

[21] Ross, G.T. and Soland, S.M., "Modelling facility location problems as generalised assignment problems", *Management Science* 24 (1977) 345–357.

[22] Sridharan, R., "A survey of the capacitated plant location problem", Working paper, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh PA 15213, USA, 1984.

[23] Suhl, U.H., "Solving large scale mixed-integer programs with fixed charge variables", Research report RC 10266, IBM Thomas J. Watson Research Centre, Yorktown Heights, New York 10598, USA, 1983.

[24] Van Roy, T.J., "A cross decomposition algorithm for capacitated facility location", *Operations Research* 34 (1986) 145–163.

[25] Whitaker, R.A., "Some add-drop and drop-add interchange heuristics for non-linear warehouse location", *Journal of the Operational Research Society* 36(1) (1985) 61–70.