R. 104
Sig 1207 Bar

1400008700
M-REPORT/213

A BRACH AND BOUND APPROACH FOR THE CAPACITATED PLANT LOCATION PROBLEM USING CUTS FROM THE PARTIAL CONVEX HULL

Jaime BARCELÓ

RR84/06

BRANCH AND BOUND PARA PROBLEMAS DE LOCALIZACION DE PLANTAS CON CAPACIDADES UTILIZANDO CARAS DE LA ENVOLVENTE CONVEXA PARCIAL

Resumen

La obtención de soluciones para problemas de localización de plantas con restricciones de capacidad de una forma eficiente, requiere habtualmente el desarrollo de un software específico, sobre todo cuando los problemas son enteros puros, dado que los códigos estandar para programación matemática son muy poco eficientes con este tipo de problemas. La formulación ordinaria de eta clase de problemas incluye algunas restricciones que son de tipo knapsack. Este hecho puede utilizarse aprovechando los procedimientos desarrollados por Balas, Padberg y otros, para calcular caras de los correspondientes politopos de kanpsack. Este trabajo recoge la experiencia de cálculo obtenido utilizando tales procedimientos en combinación con un código estandar para programación matemática.

A BRANCH AND BOUND APPROACH FOR THE CAPACITATED PLANT LOCATION PROBLEM USING CUTS FROM THE PARTIAL CONVEX HULL

J. Barceló (*)
Departament d'Investigació Operativa i Estadística
Facultat d'Informàtica
Universitat Politècnica de Catalunya
Jordi Girona Salgado, 31
Barcelona-34, SPAIN

Abstract

The efficient solution of capacitated plant location problems usually requires the development of specific software mainly when the problems are pure integer ones, standard mathematical programming codes being highly inefficient. The normal formulation of these problems includes different kinds of constraints which are of knapsack type. This fact can be used in connection with the procedures developped by Balas, Padberg and others, of computing facets of the kanapsack polytopes. Procedures which can be used with a standard mathematical programming code are described and computational experience is reported.

(*) This research was carried out between June and August, 1983, at the Department of Mathematics of the Technological Institute of Linköping, Sweden, and was supported by a Grant of the Margit and Folke Pehrzon Foundation. A first version was presented at the 11th IFIP. Conference, held in Copenhagen, July 25-29, 1983.

A BRANCH AND BOUND APPROACH FOR THE CAPACITATED PLANT LOCATION PROBLEM USING CUTS FROM THE PARTIAL CONVEX HULL.

1. INTRODUCTION

Many different algorithms for capacitated plant locationn problems have been developed lately, and for some of them the computational results reported look very promissing (Christofides and Beasley, /5/, Geoffrion and McBride, /11/, Guignard and Spielberg, /12/, Nauss, /13/, Van Roy, /17/, Barceló and Casanovas, /3/, etc.). Anyway, from the point of view of the possible user, that is to say: the decission maker responsible of the opening of the new warehauses or plants, all approaches used in implementing the above referenced algorithms have a common disadvantage: all of them require the development of a specific software with the consequent penalties in cost and time.

Usually the companies and the departments in the Public Administration which are faced such location problems, have their own computational resources including some commercial mathematical programming codes or have access to some computer facilities that posess such a kind of package. So the most natural tendence is trying to solve the location problem exploting the features of the mathematical programming code at hand.

The experience uses to be discouraging, mainly when the proposed location problem is a pure integer one. The classical branch and bound appraches, using the ordinary linear programming relaxation, are usually highly inefficient even when they include sophisticated techniques of computing pseudocosts or powerful branching procedures based on the special ordered sets.

From such empiric results arises the question of whether or not exists a complementary algorithmic tool which could be used in connection to a standard mathematical programming code at a low computational cost, in order to improve its performance in solving this class of mathematical programming problem, in such a way that they could be solved directly.

Our research work has been oriented to trying to answer this question. The first step in looking for a solution was trying to understand why the classical branch and bound approach was so inefficient when applied to this class of mathematical programming problems.

From an empirical point of view one can verify easily that a typical capacitated location problem shows a large duality gap, that very often reaches values about 10% of the optimal one, and even larger amounts in case of pure integer capacitated location problems, for which he duality gap could lie, on an average in the range between 20% and 30%.

This big duality gap could give a first explanation of the branch and bound inefficiency. Obviously such a big duality gap would require a big branching effort, before being reduced to size allowing the relaxed subproblems give solutions meaningful enough. That is, solutions whose value is close enough to that of the not relaxed subproblem.

Once verified this fact it seems reasonable to think that reduction in duality gap could produce the desired improvement in the branch and bound efficiency.

In a former work Spielberg, /16/, recommended, also from an empirical experience, formulating the problem in a desagregated way, including the redundant constraints:

$$x_{ij} - m_{ij} y_j <= 0$$

where index i identifies the plant or warehouse, (being I the index set of potential locations for them), index j identifies the demand center to be supplied from the selected plant, (being J the index set of demand centers), the decision variables x_{ij} and y_i represent the amount of demand of the center j satisfied from plant i and the decision of opening the i-th plant ($y_i = 1$), or not ($y_i = 0$); and the coefficient $m_{ij} = \min\{d_j, b_i\}$, where d_j is the amount of demand of center j, and b_i the total capacity of plant i.

Later on, Guignard and Spielberg, /12/, insist again on the importance of such desagregated formulation (which in the uncapacitated location case leads to the so called "strong linear relaxation", Cornuejols et al., /6/, Erlenkotter, /8/), but Geoffrion and Mc Bride, /11/, where those who explained why this apparently redundant constraints were so important, by showing that they characterize a part of the convex hull of the capacitated plant location problem and thus by including them in the problem formulation the duality gap was reduced and consequently the relaxed subproblems provide better bounds. However this approach has the disadvantage of increasing enormously the size of the LP problems to be solved, and it looks computationaly efficent only when the problem is solved by means of other procedures than the simplex, as for instance the decomposition procedures (Van Roy, /17/), or the lagrangean techniques (Geoffrion and Mc Bride, /11/).

Remembering that our purpose is trying to solve capacitated plant location problems using only ordinary mathematical programming tools, the question at the beginning can be restated in the following terms: starting from an agregated formulation of the problem, (in order to keep the size as small as possible), are we able to characterize efficiently and cheaply some facets of the convex hull without significantly increasing the size of the problem?

2. PROBLEM FORMULATION

To start with, we have choosen the problem formulation as a pure integer capacitated location problem, alternative which had been studied before by ourselves from a heuristic lagrangean approach in a former work (Barceló and Casanovas, /8/). With this formulation the problem is stated as:

$$[\text{MIN}] \quad \sum_{i\in I} \sum_{j\in J} c_{ij}\, x_{ij} + \sum_{i\in I} f_i\, y_i \qquad (1)$$

$$\sum_{i\in I} x_{ij} = 1, \quad \forall j \in J \qquad (2)$$

$$(P) \qquad \sum_{i\in I} y_i \leq K \qquad (3)$$

$$\sum_{j\in J} d_j\, x_{ij} \leq b_i\, y_i, \quad \forall i \in I \qquad (4)$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I, \forall j \in J \qquad (5)$$

$$y_i \in \{0,1\} \quad \forall i \in I \qquad (6)$$

where as usually c_{ij} are the supplying costs from plant i to center j, f_i are the fixed costs of opening a plant at the potential site i, the constraints set (2) assures that the demand of all centers will be satisfied from only a plant, constraint (3) limits the number of plants to be opened to a maximum of K and the constraint set (4) prevents trying to supply a center from a not opened plant ($y_i = 0 \Rightarrow x_{ij} = 0$, $j \in J$), and does not allow the supply of more than the total capacity of the plant when it is opened.

From this formulation a first possibility of describing partially the convex hull is given by the capacity constraints (4), assuming the decision of opening the i-th plant, $y_i = 1$, the corresponding constraint is

$$\sum_{j\in J} d_j\, x_{ij} \leq b_i \qquad (7)$$

which is a knapsack constraint.

Balas, /1/, Balas and Zemel, /2/, Padberg, /14/, and others, have characterized many facets of the knapsack polytope, some of them not difficult to compute. On the other hand if P is the polytope of problem (P), and P_i is the knapsack polytope corresponding to the i-th knapsack constraint, then the polytope P is such that

$$P \subset \bigcap_{i \in I} P_i$$

and it follows that any characterization of the facets of the polytopes P_i contributes to a partial facetial description of P.

Consequently it could be thought that any formulation of problem (P) including some facets of the knapsack polytopes of (4), would be a stronger formulation, with a smaller duality gap, producing subproblems which give better bounds.

The first computational research that we have carried out has been to study the way of computing such facets and verify their influence in the branch and bound procedure.

3. COMPUTING FACETS OF THE POLYTOPE FROM THE CAPACITY CONSTRAINTS

Before dealing with the family of constraints defined by (7) let us introduce the following notation, induced from Balas, /1/.

Assume J and all of its subsets ordered so that

$$d_j \geqslant d_{j+1} \quad , \quad j = 1,2,\dots, n-1$$

let us define for every $S \subseteq J$, the extension of S to J as

$$E(S) = S \cup S'$$

where

$$S' = \{ j \in J-S \mid d_j >= d_k \ , \forall k \in S \}$$

and the elements

$$d_{k_1} = \underset{k \in S}{\mathrm{MAX}}\ d_k \quad , \quad d_{i_1} = \underset{j \in J-E(S)}{\mathrm{MAX}}\ d_j \quad , \quad d_{k_2} = \underset{k \in S-\{k_1\}}{\mathrm{MAX}}\ d_k$$

and amounts

$$\Delta = \sum_{j \in S} d_j - b_i \quad , \Delta_1 = d_{k_1} - d_{i_1} \quad , \Delta_2 = d_{k_1} + d_{k_2} - d_1$$

Note that with the ordering in J, the elements d_{k_1} and d_{k_2} are respectively the first and second elements in S; d_{i_1} is the first element in J-E(S), and d_1 is the first (and biggest) element in J.

With the help of this notation we can stablish the following propositions.

PROPOSITION 3.1

a) If $\Delta > 0$ then S is a minimal cover
b) If $\Delta <= \Delta_1$ then S is a strong minimal cover
c) If $\Delta <= \Delta_2$ then $\sum_{j\in E(S)} x_{ij} \leq |S| - 1$ is a facet of the polytope

$$P_i = \text{conv}\,\{x_{ij} \in \{0,1\}^n \mid \sum_{j\in J} d_j\, x_{ij} \leq b_i\}$$

corresponding to the i-th capacity constraint (7).

Condition (a) follows directly from the definition of minimal cover, Balas, /1/,. To show (b) it suffices to verify that

$$\Delta \leq \Delta_1 \Rightarrow \sum_{j\in S} d_j - b_i \leq d_{k_1} - d_{i_1} \Rightarrow$$

$$\sum_{j\in S} d_j - d_{k_1} + d_{i_1} \leq b_i$$

and then S satisfies the definition of strong minimal cover, Balas, /1/, if in addition $\Delta \leq \Delta_2$ one also has

$$\sum_{j\in S} d_j - b_i \leq d_{k_1} + d_{k_2} - d_1 \Rightarrow$$

$$\sum_{j\in S} d_j - (d_{k_1} + d_{k_2}) + d_1 \leq b_i$$

then S satisfies the conditions of Balas theorem (Balas, /1/, pg. 151), and thus

$$\sum_{j \in E(S)} x_{ij} \leqslant |S| - 1$$

is a facet of P .

PROPOSITION 3.2

Let S_h be the subset of the h first elements of $S \subseteq J$, and define the subset J as:

$$J_h = \{i \in J \mid \sum_{j \in S_h} d_j \leqslant d_i < \sum_{j \in S_{h+1}} d_j\},\ h = 1,\dots,|S| - 1$$

Let dk_h be the first element in J with the order defined in J. Then if S defines a valid cut, (Balas, /1/), this is a facet of the polytope P_i if

$$\sum_{j \in S - S_{h+1}} d_j + d_{k_h} \leqslant b_i \quad , \quad h = 1,\dots,|S| - 1$$

The proof follows directly from the fact that with the ordering in J, $d_{k_h} >= d_k$,$\forall k \in J_h$, thus

$$\sum_{j \in S - S_{h\ 1}} d_j + d_{k_h} \leqslant b_i \Rightarrow \sum_{j \in S - S_{h\ 1}} d_j + d_k \leqslant b_i \quad , \forall k \in J_h$$

and then the conditions of Balas' theorem (Balas, /1/, pg. 156) hold.

Proposition 3.1 is the basis of a search algorithm that generates in a systematic way strong minimal covers of the polytope P_i, which are the input of a procedure designed by Balas, /1/, of generating valid cuts. A complementary step based on proposition 2 cheks whether the generated valid cut is a facet or not.

ALGORITHM 3.1

Input: The set of demands dj, $j \in J$, of the centers and the set b_i, $i \in I$, of the capacities.

Output: The first part gives a strong minimal cover of polytope P_i, (which could be eventually a facet), when performed over the i-th capacity constraint. If the computed strong minimal cover is not a facet then the second part computes a valid cut and cheks whether or not is a facet.

```
begin

        Set LIST : = I
        Order the set of demands $d_j$ in increasing
        order
        (dj >= dj+1, j= 1,2,..., |J| - 1);
        while LIST $\neq$ Ø do
        begin let i be the first plant in LIST;
              search for two elements in J, $d_{k_1}$ and
              $d_{k_2}$ such that
                      $d_{k_1} + d_{k_2} \geq d_1$;
        again: Set $\Delta_2 := d_{k_1} + d_{k2} - d_1$
               set S:= {$K_1$,$K_2$};
        (Comment: at this point the procedure tries to
                  generate a strong minimal cover for
                      the i-th capacity constraint
                 starting from $d_{k2}$ and $d_{k_2}$)
               while $\sum_{j \in S} d_j \leq b_i$ do
                      begin
                        search for an element dj
                       such that $j>k_1$ and $j>k_2$;

                       set S:= S ∪ {j};
```

```
                    set $\Delta := \sum_{j \in S} d_j - b_i$
                end

            set: $S' := \{j \in J \mid j < k_1\}$
            set $E(S) := S \cup S'$;
            Let di1 be the first element in
            J-E(S)
            set $\Delta_1 := d_{k_1} - d_{i_1}$;
            If $\Delta \leqslant \Delta_1$ then S is a strong minimal
            cover           goto              check

                else
            other begin
                    search for two different $d_{k_1}$
                    and dk2 in J, such that $d_{k_1}$ +
                    + $d_{k_2}$ in J >= $d_1$;
                    goto again
                end
check: If $\Delta \leqslant \Delta_2$ then  $\sum_{j \in E(S)} x_{ij} \leqslant |S| - 1$ is a facet of
                $P_i$ by proposition 1
            (If another facet of P is desired
            goto other, otherwise repeat with a
            new capacity constraint);

            else goto validcut

validcut: (comment: the first part of the procedure is
          an adhoc version of the Balas algorithm,
          (Balas, /1/), for generating validcuts; the
          second part checks whether the valid cut is
          a facet or not).

          begin
```

```
            for h:= 1,2,..., $|S|-1$ do set $S_h$ := { set of
            the first h elements in S } ;
            set $J_o := J - E(S)$;
            for h:= 1,2,..., $|S| - 1$ do
                 begin
                 set $J_h := \{ j \in J \mid \sum_{k \in S_k} d_k \leqslant d_j < \sum_{k \in S_{h+1}} d_k \}$
                 set $J\,U := J\,U \cup \{J_h\}$
                 end
        set  $J := E(S) - J\,U$  ;
        for h:= 1,2,..., $|S| - 1$ do
                 while   $j \in J_h$ do $\pi_j = h$
        (Comment: then
                 $\sum_{j \in J} \pi_j x_{ij} \leqslant |S| - 1$

                   is a valid cut for Pi)
        end
    begin
        for h:= 1,2,..., $|S| - 1$ do
             set $d_{k_h}$ := { first element in $J_h$ }
        If $\sum_{j \in S - S_{h+1}} d_j + d_{k_h} \leqslant b_i$, h:= 1,2,...,$|S|$-1
            then by proposition 3.2
            the valid cut is a facet of $P_i$, go to other
            if another facet of $P_i$ is desired, otherwise
            repeat with a new capacity constraint.

        end

    end
```

4. COMPUTATIONAL EXPERIENCE I

We have carried out two parallel series of computational experiences. The objective of the first series was to study the percent decreasing in the duality gap, with reference to the desagregated formulation and lineal relaxation used, while the second series tried to determine the efficiency of a standard branch and bound commercial code in solving the corresponding formulation.

Table 1 shows all the sets of constraints to be included in the different desagregated formulations. In the first step we have adopted as basic formulation that defined by sets (1), (2), (3), (4), and (5) in Table 1, and as alternative desagregated formulations those derived from the basic one substituing (5) by (6) and/or (7). The constraint set (8) is the set of facets of knapsack polytopes of capacity constraints (4), one facet for each capacity constraints, computed by algorithm 1.

As test problems we used the same set already employed in Barceló and Casanovas, /3/,. From this set we have selected three cases of different sizes that could be considered as representatives of the average behaviour. Tables 2,3 and 4 show the results obtained for these three cases.

The computational exprience was carried out in a DEC-2060 computer using LAMPS (Linear and Mathematical Programming System) of CAP Scientific Limited, as standard mathematical programming code. Results included in the branch and bound reference of each table are the best ones obtained by using pseudocosts and special ordered sets, following the recommendations form the studies of Forrest et al., /9/, Gauthier and Ribiere, /10/, and Benichou et al., /4/.

The experiences performed gave us an insigth into the difficulties in solving this class of problems, the main difficulty lies more in the relationship between the coefficients d_j and b_i than in the problem size. Consequently the ratio

$$h = \sum_{j \in J} d_j \Big/ \sum_{i \in I^*} b_i$$

where $I^* \subseteq I$ is the set of plants opened in the optimal solution, could be interpreted as a measure of how hard is a problem.

Table 2 shows the results for an average problem of intermediate difficulty. From the first set of experiences it could be pointed out the small influence of constraints (7) in the decreasing of the duality gap, when compared with constraints (6). As rows (2.2) and (2.3) show, including constraints (7) in the desagregated formulation decreases very few the duality gap but increases a lot the computing cost (number of simplex iterations, CPU time, etc.) Row (2.4) shows the effect of including facets of capacity constrains polytopes (4) through constraints (8), and row (2.5) shows what happens when constraints (7) are included in the formulation together with constraints (8), again constraints (7) show a high degree of inefficiency in reducing the size of the duality gap in terms of computational cost.

Rows (2.6) and (2.7) are the start of the results of the second set of experiences, in which a standard branch and bound procedure is employed. The results show clearly that if constraints (7) present a small advantage when compared to (6) in what is refered to the searching effort, (the number of explored nodes in (2.7) is smaller than in (2.6)), this advantage does n ot compensate the increase of CPU time required. That follows from the fact that including (7) increases a lot the size of the linear program to be solved at each step. Anyway the results showed in row (8) confirm empirically the hypothesis that constraints (8), computed at low computational cost by means of Algorithm 3.1, are an efficient resource in helping to solve this class of problems using only standard computing facilities.

T A B L E I

[MIN] $\sum_{i\in I} \sum_{j\in J} c_{ij} x_{ij} + \sum_{j\in I} f_i y_i$ (1)

(2) $\sum_{i\in I} x_{ij} = 1 \quad \forall j \in J$

(3) $\sum_{i\in I} y_i \leq K$ $\quad \sum_{i\in I} b_i y_i \geq D$ (3a) where $D = \sum_{j\in J} d_j$

(4) $\sum_{j\in J} d_j x_{ij} \leq b_i y_i \quad \forall i \in I$

(5) $x_{ij} \geq 0, \; y_i \geq 0, \quad \forall i \in I$ and $\forall j \in J$

(6) $x_{ij} \geq 0, \; y_i \leq 1, \quad \forall i \in I$ and $\forall j \in J$

(7) $x_{ij} \leq y_i, y_i \leq 1, \quad \forall i \in I$ and $\forall j \in J$

(8) $\sum_{j\in E(S_i)} x_{ij} \leq (|S_i| - 1) \, y_i, \; \forall i \in I$

(9) $\sum_{j\in E(S_i)} x_{ij} \leq (|S_i| - 1) \, y_i, \; \forall i \in I^*$

(10) $\sum_{i\in E(S)} y_i \geq |S| - 1$

(11) $\sum_{i\in I} a_i y_i \geq a_o$

(12) $x_{ij} \in \{0,1\}, \; y_i \in \{0,1\} \; \forall i \in I$ and $\forall j \in J$

TABLE 2

NUMBER OF PLANTS m = 6

NUMBER OF CENTERS n = 10

OPTIMAL VALUE z = 2057

ratio (hard) = 0.85

(66 integer variables)

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN LP RELAXATION	OBJECTIVE FUNCTION VALUES	LPU TIME (seconds) DEC-2060	DUALITY GAP IN %	NUMBER OF SIMPLEX ITERATIONS
2.1	(1),(2),(3),(4),(5)	1412.83	4.83	31.31	28
2.2	(1),(2),(3),(4),(6)	1719.73	4.83	16.39	28
2.3	(1),(2),(3),(4),(6),(7)	1721.10	6.76	16.32	51
2.4	(1),(2),(3),(4),(6),(8)	1838.8	5.78	10.60	60
2.5	(1),(2),(3),(4),(6),(7),(8)	1841.6	7.65	10.47	69

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN B - B	NUMBER OF NODES EXPLORED	CPU TIME (seconds) DEC-2060
2.6	(1),(2),(3),(4),(6),(12)	253	47.83
2.7	(1),(2),(3),(4),(6),(7),(12)	220	1:30.83
2.8	(1),(2),(3),(4),(6),(8),(12)	16	10.04
2.9	(1),(2),(3),(4), (7),(8),(12)	22	13.32

TABLE 3

NUMBER OF PLANTS m = 10

NUMBER OF CENTERS n = 20

OPTIMAL VALUE 3668

ratio (hard) = 0.97

(210 integer variables)

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN LP RELAXATION	OBJECTIVE FUNCTION VALUES	CPU TIME (seconds) DEC-2060	DUALITY GAP IN %	NUMBER OF SIMPLEX ITERATIONS
3.1	(1),(2),(3),(4),(5)	3028.12	6.90	1 44	44
3.2	(1),(2),(3),(4),(6),	3491.38	7.47	4.81	66
3.3	(1),(2),(3),(4),(6),(8)	3493.159	9.38	4.76	86
3.4	(1),(2),(3),(4),(6),(8)	3497.54	11.12	4.64	104
3.5	(1),(2),(3a),(4),(6),(11)	3632.66	7.84	0.96	74
3.6	(1),(2),(3a),(4),(6),(11),(9)	3636.97	8.53	0.84	94
3.7	(1),(2),(3a),(4),(6),(11),(9),(11	3640.12	11.24	0.76	111

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN B - B	SOLUTION FOUND	CPU TIME seconds DEC-2060	NODES EXPLORED
3.8	(1),(2),(3),(4),(6),(8),(12)	NO	10:00.00	-
3.9	(1),(2),(3a),(4),(6),(11),(12)	NO	10:00.00	-
3.10	(1),(2),(3a),(4),(6),(11),(9),(12)	3670	4:32.08	481
3.11	(1),(2),(3a),(4),(6), (11),(9),(11),(12)	3668	1:58.23	131

TABLE 4

NUMBER OF PLANTS $m = 10$

NUMBER OF CENTERS $n = 20$

OPTIMAL VALUE 3714

ratio (hard) = 0.95

(210 integer variables)

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN LP RELAXATION	OBJECTIVE FUNCTION VALUES	CPU TIME (seconds) DEC-2060	DUALITY GAP IN %	NUMBER OF SIMPLEX ITERATIONS
4.1	(1),(2),(3),(4),(5),	2676.48	7.19	27.93	56
4.2	(1),(2),(3),(4),(6)	3286.04	7.75	11.52	64
4.3	(1),(2),(3),(4),(6),(8)	3287.16	9.60	11.49	91
4.4	(1),(2),(3),(4),(7),(8)	3287.61	7.59	11.48	90
4.5	(1),(2),(3a),(4),(6),(11)	3641.54	7.55	1.95	61
4.6	(1),(2),(3a),(4),(6),(11),(9)	3669.04	7.71	1.21	63

EXPERIENCE NUMBER	SETS OF CONSTRAINTS IN B - B	SOLUTIONS FOUND	CPU TIME DEC-2060	NODE
4.7	(1),(2),(3),(4),(6),(8),(12)	3749 3740	4:01.36 7:15.28	444 734
4.8	(1),(2),(3a),(4),(6),(11),(12)	3753 3740	48.63 1:54.08	72 221
4.9	(1),(2),(3a),(4),(6),(11),(12),(9)	3740 3714	32.27	46 385 SEARCH COMPLETED AFTER 435 NODES IN 2:59.4

Results of the type shown in Table 2 appear again when this procedure is applied to problems with a ratio h of hardness of this level. One is tempted to draw optimistic conclusions which are denied by results included in Tables 3 and 4.

Problems in Tables 3 and 4 are two test problems specially hard, with very high values of h. The first set of experiences (rows (3.1) to (3.4), and (4.1) to (4.4) respectively) shows clearly that in such difficult cases although constraints (6) are better than (7), because they still produce a similar decrease in the duality gap at a much lower computational cost, constraints (8), however, no longer offer a meaningful advantage, even from the point of view of branch and bound, since the second set of experiences (rows (3.8) and (4.7)) shows, that either it is not able to solve the problem within some given CPU time limit (10 minuts in all the experiments done), or it only can reach some feasible solutions, although acceptably good, within such limits.

From this experiences two conclusions can be drawn. The first one is the low interest of using constraints (7) in formulating and solving pure integer capacitated location problems. The second conclusion is that even being recommendable the use of desagregated formulations including constraints of type (8), its efficiency depends heavily on the level of difficulty of the problem.

The first conclusion is surprising in that it contradicts the asserts of Geoffrion and Mc Bride, /11/, and Spielberg, /16/ for the mixed integer capacitated location problem, given that constraints (7) are a particular case of constraints $x_{ij} - m_{ij} y_i <= 0$, and, as was mentiiones adove, Geoffrion and Mc Bride show that such constraints are facets of the convex hull of the problem. But in the pure integer case the empirical conclusion drawn can be explained theoretically because when $mij = 1$ and $x_{ij} \in \{0,1\}$, then usually conditions of Proposition 2 in (Balas, /1/, pg. 149) hold (because usually $d_j < b_i$), and $x_{ij} - y_i <= 0$ are trivial facets of the corresponding knapsack polytope. This fact expains why in this case constraints (7) have such a low contribution in the solution of the problem.

A second conclusion would indicate that a deeper analysis is needed of the facets of the convex hull, that means, identifying whether it is posible or not to generate more useful facets.

5. ADDITIONAL PROCEDURES OF GENERATING FACETS OF THE KNAPSACK POLYTOPE

A possibility of improving the former results would be not computing a priori facets of the knapsack polytopes P_i, associated with the capacity constraints (4), ignoring how good they are, but computing only those of guaranteed uselfulness strictly when we need them. A way to do this consists in solving the ordinary linear programming relaxation, and identifying which is for each capacity constraint (4), the minimal cover S_i, whose asociated inequality

$$\sum_{j \in S_i} x_{ij} \leq |S_i| - 1 \qquad (8)$$

would be more violated by the continuous solution $\bar{x}_{ij}$, whenever such an inequality exists.

The following theorem, (Crowder, Johnson and Padberg, /7/) answers this question:

THEOREM 5.1

There exists a minimal cover inequality (8) that cuts off $\bar{x}_{ij}$ if and only if the optimal objective function value of

$$\text{MIN } \{\sum_{j \in J} (1-\bar{x}_{ij}) z_j \mid \sum_{j \in J} d_j z_j > b_i, \; z_j \in \{0,1\}, \; \forall j \in J\} \quad (9)$$

is less than one.

However, whenever such an inequality (8) exists, it will be a facet of the polytope P_i defined by

$$P_{S_i} = P_i \cap \{x \in R^n \mid x_{ij} = 0, \; \forall i \notin S_i\} \quad (10)$$

and we were interested in computing a facet of P_i. The problem is then to know whether or not we are able to compute the desired facet of P_i from (8), than means knowing when an inequality which is a facet of a polytope, holds its property of defining a facet when its number of variables is increased.

The following theorem (Padberg, /15/) answers the question of lifting the facets of P_{S_i} into the n-space to yield a facet of P_i.

THEOREM 5.2

Let P_{S_i} be the polytope defined by (10). If the inequality

$$\sum_{j \in S_i} \alpha_j x_{ij} \leq \pi_o$$

defines a facet of P_{S_i}, then there exist non negative numbers $\beta_j \leq \pi_o$, such that

$$\sum_{j \in S_i} \alpha_i x_{ij} + \sum_{j \in J-S_i} \beta_j x_{ij} \leq \pi_o$$

is a facet of P_i.

The possibility of calculating the coefficients β_j in a recursive way is based on the following theorem (Padberg, /15/, Balas and Zemel, /2/).

THEOREM 5.3

Let S_i be a minimal cover for i-th constraint (4), and $J-S_i = \{j, \ldots, J_r\}$ arbitrarily ordered ($p = | J - S_i |$), and consider the sequence of knapsack problems K_{ji}, defined recursively as:

$$z_{ji} = [\text{MIN}] \sum_{j \in S_i} x_{ij} + \sum_{j=j_1}^{j_{i-1}} \beta_j x_{ij}$$

$$\sum_{j \in S_i} d_j x_{ij} + \sum_{j=j_1}^{j_{i-1}} d_j x_{ij} \leq b_i - d_{ji} \qquad (11)$$

$$x_{ij} \in \{0,1\}, \forall j \in S_i \cup \{j_1,\ldots,j_{i-1}\}$$

for $i = 1,\ldots,p$ (vhere summation over the empty set is 0), with coefficients β_j defined by

$$\beta_j = |S_i| - 1 - z_j \ , \ j = j_1,\ldots, j_{i-1}$$

Then the inequality

$$\sum_{j \in S_i} x_{ij} + \sum_{j \in J-S_i} \beta_j x_{ij} \leq |S_i| - 1$$

is a facet of P_i.

The coefficients β_j depend on the sequence in which they are calculated, each sequence given a different set of coefficients, but anyway the calculation of every sequentially lifted facet requires the solution of a sequence of knapsack problems K_{ji}, one for each coefficient.

The first one of the knapsack problems can be solved trivially, as follows from the following proposition (Balas and Zemel, /2/):

PROPOSITION 5.1

For all minimal covers S_i and all $j \in J - S_i$, $\beta'_j = h$, where h is defined by

$$\sum_{k \in S_i - S_{h+1}} d_k \leq b_i - d_j \sum_{k \in S_i - S_h} d_k$$

(where S_h is the subset, of the h first elements in S_h). The optimal solution to the knapsack problem

$$\text{MAX}\ \{ \sum_{j\in S_i} x_{ij} \mid \sum_{j\in S_i} d_j\, x_{ij} \leq b_i - d_k,\ x_{ij} \in \{0,1\},\ j \in S \}$$

(being k the index of the first element in $N - S_i$) is given by

$$x_{ij} = \begin{cases} 1, & j \in S_i - S_{h+1} \\ 0, & j \in S_{h+1} \end{cases}$$

The remaining knapsack problems can not be sólved so trivially, however, as in the case of Proposition 3.2 above, there still exists the possibility of computing easily if not a lifted facet, at least a valid cut, which very often is also a facet. Balas and Zemell show in /2/ that if an inequality like

$$\sum_{j\in S_i} x_{ij} + \sum_{j\in J-S_i} \alpha_j\, x_j \leq |S_i| - 1 \qquad (12)$$

is a valid inequality for P_i, then $\alpha_j \leq \beta'$, $\forall j \in J-S_i$, and that there exists an important class of valid inequalities derived from the minimal covers, in particular if Si is the most violated cover identified by (9), and $E(S_i)$ is its extension, then the subprocedure validcut, in algorithm 3.1, computes a valid inequality belonging to the family of (12), which, if complementary conditions of Proposition 3.2 hold, is a facet of P_i.

Thus the sequence of knapsack problems which gives the facet can allways be replaced by subprocedure, validcut if only a valid cut, eventually facet, but computationally much cheaper is desired.

When the facet of Pi is desired, the sequence of knapsack problems can be systematically solved by means of the following.

RECURSIVE LIFTING PROCEDURE (Padberg)

```
    Set $a_j = 1, \forall j \in S_i,\quad a_o = |S_i| - 1$
    Iterative step
        Let $k \in J - S_i$ and determine

            $z_k = [MAX] \{\sum_{j \in S_i} a_j x_{ij} \mid \sum_{j \in S_i} d_j x_{ij} \leq b_i - d_k, x_{ij} \in \{0,1\}, \forall j \in S_i\}$

        Define $a_k = a_o - z_k$
    Redefine Si to be $S_i \cup \{K\}$ and repeat until $J-S_i$ is empty
```

The resulting inequality

$$\sum_{j \in J} a_j x_{ij} \leq a_o$$

is a facet of the knapsack polytope P_i associated with the i-th capacity constraint (4).

This recursive lifting procedure can be combined with a standard mathematical programming code, as in 4 § , to give the following global procedure

```
    Procedure new facet
    begin
    start:   Solve the ordinary linear programming
             relaxation of the pure integer capacitated
             location problem
             Identify partial solution $\bar{x}_{ij}$ (set of centers
             $J_i$ complete or parcially assigned to plant i)

             set LIST:= I
```

```
while LIST:$\neq \emptyset$ do
    begin
        Let i be the first plant in
        LIST, remove it form LIST
        Determine the most violated
        minimal cover by partial
        solution $\bar{x}_{ij}$, solving

        $[\mathrm{MIN}]\{\sum_{j\in J}(1-\bar{x}_{ij})z_j \mid \sum_{j\in J} d_j z_j > b_i, z_j \in \{0,1\}, \forall j \in J\}$

        Apply the RECURSIVE LIFTING
        PROCEDURE to the minimal cover
        $S_i$ identified. (Alternatively
        apply validcut from Algoritm
        3.1 if only a valid inequality
        is required)
        Add the new inequality

            $\sum_{j\in J} a_j x_{ij} \leq a_o$

        to the constraint set

        end
    If more facets are desired then go to
    start
        else end
```

Once procedure New Facet has been performed, proceed with the branch and bound searching.

6. COMPUTATIONAL EXPERIENCE II: CONCLUSIONS

The algorithm just described was applied again to the set of test problems which had not been solved by the first algorithm, that is to the hardest ones which exhibited a behaviour like that referenced in Tables 3 and 4. Procedure New Facet computed a set of constraints of type (9) referenced in Table 1. Results of this new series of experiences have not been included in the table because dissapointingly, although being theoretically stronger, the new inequalities were no more efficient than the previous ones. Computational behaviour of the new algorithm was almost the same as the first one.

This discouraging fact lead us to investigate further which was the contribution of the different facets of the convex hull to closing the duality gap. The analysis reveals that constraints like (8) jor (9) in Table 1 act only over the x_{ij} variables, that is the decission variables of the assignment of centers to open plants, and only constraint (3) acts over the selection of plants, but since usually, the f_i' s are bigger than the c_{ij}, the contribution of the y_i' s to closing the duality gap is obviously stronger than the contribution of the x_{ij}.

Consequently any further improvement would be possible only if we can act over the y_i' s. with the problem formulated as in Table 1, there is only a constraint acting on the yi, this is

$$\sum_{i\in I} y_i \leqslant K \qquad (3)$$

which does not offer any possibility of computing inequalities of type (8) or (9) in terms of variables yi. The only possibility of doing that comes from a new formulation of the problem. Indeed, substituing (3) by the limiting constraint:

$$\sum_{i\in I} b_i\, y_i \geqslant D \qquad (3a)$$

where $D = \sum_{j\in J} d_j$, already used by Nauss, /13/, and Guignard and Spielberg, /12/, formerly, and by Van Roy, /17/, in a recent paper. Constraint (3a) is again a knapsack type constraint, which affects only to the y_i variables, representing the decisions of opening or not one plant.

This constraint opens up the possibility of acting on the plants by means of procedure New Facet, substituing constraints (4) by (3a) in the procedure. Procedure New Facet will then compute constraints line (11) in Table 1. Thus Procedure New Facet will then adopt the form:

```
begin:

start:  Solve the ordinary linear programming
          relaxation of the pure capacitated location
        problem.
        Identify partial solution $\bar{y}$.
          Determine the most violated minimal cover of
        (3a) by partial solution $\bar{y}$, solving

        $[\mathrm{MIN}] \sum_{i\in I} (1-\bar{y}_i)\, z_i \mid \sum_{i\in I} b_i\, z_i > D,\ z_i\in\{0,1\}, \forall i \in I\}$

          Apply the RECURSIVE LIFTING PROCEDURE to the
            minimal cover S identified (Or procedure
        validcut when only a valid inequality is enough)
        Add the new inequality

            $\sum_{i\in I} a_i\, y_i \geqslant a_o$

         corresponding to the computed facet, to the
        constraint set
```

```
If more facets are desired then go to start
                           else end
```

Computational experience with this last algorithm is reported in rows (3.5) to (3.7) and (4.5) and (4.6) in Tables 3 and 4 respectively, in what refers to the first series of experiences about the decrease in the duality gap. (3.5) and (4.5) show undoubtfully that this is the most powerful alternative. From the point of view of the second series of computational experiences, the most important conclusion is that using constraints of types (11) and (9) together, can solve the problem, even in very hard cases, using only a standard software within acceptable time limits as rows (3.9) to (3.11) and (4.8) to (4.9) in Tables 3 and 4 show.

Constraints of types (3) and (11) were already suggested by Guignard and Spielberg in /12/, from an empirical point of view, without further justifying and without giving any systematic way of computing (11) from (3a). Our work analyses why such type of constraints is interesting and gives an easy procedure of computing them that can be used with any standar mathematical programming software.

The last version of the algorithm, the one acting only on the y' s, can be used also in mixed integer capacitated location problems. Changing the integrality constraints $x_{ij} \in \{0,1\}$ in (12), Table 1, by the relaxation $0 <= x_{ij} <=1$, the corresponding mixed integer problems in Tables 3 and 4 were solved to optimality exploring only 3 and 2 nodes respectively.

7. REFERENCES

/1/ E. Balas
Facets of the Knapsack Polytope
Mathematical Programming 8, (1975), 146-164

/2/ E. Balas and E. Zemel
Facets of the Knapsack Polytope from Minimal Covers
Siam Journal on Applied Mathematics, 34, (1978), 119-148

/3/ J. Barceló and J. Casanovas
A Heuristic Lagrangean Algorithm for the Capacitated Plant Location Problem
European Journal of Operational Research, 15, (1984), 212-226

/4/ M. Benichou, J.M. Gauthier, G. Hentges and G. Ribière
The Efficient Solution of Large-Scale Linear Programming Problems-Some Algorithmic Techniques and Computational Results
Mathematical Programming, 12, (1977), 280-322

/5/ N. Christofides and J.E. Beasley
Extensions to a Lagrangean Relaxation: Approach for the Capacitated Warehouse Location Problem
European Journal of Operational Research, 12, (1983), 19-28

/6/ G. Cornuejols, M.L. Fisher and G.L. Nemhauser
Location of Bank Accounts to Optimize Float: An Analytic Study of Exact and Approximate Algorithms
Management Science, 23, (1977), 789-810

/7/ H. Crowder, E.L. Johnson and M.W. Padberg
Solving Large-Scale Zero-One Linear Programming Problems
IBM Research Report RC8888, (1981)

/8/ D. Erlenkotter
A Dual Based Procedure for Uncapacitated Facility Location
Operations Research, 26, (1978). 992-1009

/9/ J.J.H. Forrest, J.P.H. Hirst and J.A. Tomlin
Practical Solution of Large Mixed Integer Programming Problems with Umpire
Management Science, 20, (1974), 736-773

/10/ J.M. Gauthier and G. Ribière
Experiments in Mixed Integer Programming Using Pseudocosts
Mathematical Programming, 12, (1977), 26-47

/11/ A.M. Geoffrion and R. Mc Bride
Lagrangean Relaxation Applied to Capacitated Facility Location Problems
AIIE Transactions, 10, (1978), 40-47

/12/ M. Guignard and K. Spielberg
A Direct Dual Method for the Mixed Plant Location Problem with some Side Constraints
Mathematical Programming, 17, (1979), 198-228

/13/ R.M. Nauss
An Improved Algorithm for the Capacitated Facility Location Problem
J. Opl. Res. Soc., 29, (1978), 1195-1201

/14/ M.W. Padberg
Covering, Packing and Knapsack Problems
Annals of Discrete Mathematics, 4, (1979), 265-287

/15/ M.W. Padberg
A Note on Zero-One Programming
Operations Research, 23, (1975), 833-837

/16/ K. Spielberg
Plant Location with Generalized Search Origin
Management Science, 16, (1969), 165-178

/17/ T.J. Van Roy
A Cross-Decomposition Algorithm for Capacitated Facility Location
Working Paper, Afdeling Industrieel Beleid, Katholieke Universiteit Leuven (To appear in Operations Research)