P. Michaud

Exact Implicit Enumeration Method for Solving the Set Partitioning Problem

Abstract: The partitioning problem may be stated as follows: Minimize $\sum_{j=1}^{n} c_j x_j$ subject to $\sum_{j=1}^{n} a_{ij} x_j = 1, \quad i = 1, \cdots, m$, where a_{ij} and x_j are binary numbers.

The proposed method consists in solving a new problem equivalent to the original one, the solution of this new problem being obtained by implicit enumeration. Computational experience indicates that this approach yields good results; for example, problems involving several hundred variables were solved in a few seconds.

Introduction

The set partitioning problem, a special case of the set covering problem with equality constraints, may be stated as follows: Minimize $\sum_{j=1}^{n} c_j x_j$ subject to the constraints

$$\sum_{j=1}^{n} a_{ij} x_j = b_i = 1, \qquad i = 1, \cdots, m, \tag{1}$$

in which

$a_{ij} = 0$ or 1, $\quad i = 1, \cdots, m$ and $j = 1, \cdots, n$;

$x_j = 0$ or 1, $\quad j = 1, \cdots, n$,

where the c_j's are arbitrary numbers.

Using matrix notation the problem is restated as:

Minimize cx subject to

$Ax = b$

$x_j = 0$ or 1 $\quad j = 1, \cdots, n$,

where A is an $m \times n$ binary matrix, c and x n-vectors and b an m-vector of ones.

The term "set partitioning" may be interpreted by considering a set S of m elements. Any column A_j $j = 1, \cdots, n$, of A represents a subset S_j of S where $a_{ij} = 1$ if $i \in S$; and 0 if $i \notin S_j \cdot c_j$ is the cost of S_j.

The problem is to select among the S_j's ($x_j = 1$ if S_j is selected, $x_j = 0$ if S_j is not selected) a partition of S with a minimal cost.

The literature devoted to methods for solving this problem is very extensive. Some of it is surveyed in [1] and [2].

Of special interest are the methods used for solving integer programming problems. Major approaches include implicit enumeration, cutting plane, and group theoretic techniques.

The implicit enumeration principle, first utilized by Land and Doig [3], became largely used after the Balas algorithm was published [4]. Basically any implicit enumeration method enumerates all the potential solutions but actually considers only a subset of them. The remaining solutions are said to be implicitly enumerated. Various algorithms using this approach have been proposed for the resolution of partitioning problems; for example, Garfinkel and Nemhauser [5] and Pierce [6] and also Rubin [7] for the crew scheduling problem.

The cutting plane and the group theoretic techniques, both originally proposed by Gomory [8,9], start with the solution of the continuous problem associated with the original one:

1. In the first case new constraints are added to the continuous problem and the augmented problem is solved by linear programming. This process is repeated until the solution of the augmented problem is a feasible solution of the original integer programming problem. Such a method has been used by Martin [10] for partitioning problems.
2. In the second case an asymptotic problem is defined. This new problem is itself converted into a group optimization problem. Unfortunately these two problems are not equivalent to the original one. This weakness may be avoided by adding constraints in the

group optimization problem. See, for instance, Shapiro's algorithm [11]. Numerical experience concerning crew scheduling (covering and partitioning) problems using the group theoretic approach may be found in the thesis of Thiriez [12].

The proposed method is a composite of these two methods. First the continuous problem is solved. This yields an equivalent problem with disjunctive constraints. Then instead of using the cutting plane or the group theoretic techniques as above, the new problem is solved by using an implicit enumeration method.

Proposed method

The present procedure has the following properties:

1. Although presented for solving the set partitioning problem, the method gives the exact solution for any problem where

$$a_{ij} \geq 0,\ b_i \geq 0,\ x_j = 0 \text{ or } 1,$$
$$j = 1, \cdots, n \text{ and } i = 1, \cdots, m.$$

2. Problem-solving time may be considerably shortened if a feasible solution or an upper bound of the optimal value of the problem is available at the beginning of the solving process.

3. In the case of a feasible problem, a (good) feasible solution is usually found prior to the ultimate completion of the problem-solving process.

We now describe how the initial problem (1) is replaced by an equivalent one and how this new problem is solved by implicit enumeration.

Let us define the continuous problem derived from problem (1) as follows. Minimize cx subject to

$$Ax = b,$$
$$x \geq 0. \qquad (2)$$

The constraints $x_j = 0$ or 1, $j = 1, \cdots, n$ of problem (1) are replaced by $x_j \geq 0$, $j = 1, \cdots, n$ (the constraints $x_j \leq 1$, $j = 1, \cdots, n$ are not introduced since they are implied by $Ax = b$).

To the optimal solution obtained by solving problem (2) by linear programming corresponds the following decomposition: $x = (x_D, x_I)$, $A = (A_D, A_I)$ and $c = (c_D, c_I)$, where D is the set of indices of the dependent (basic) variables and I the set of indices of the independent (nonbasic) ones. The elimination of the basic variables in problem (1) yields the following formulation: Minimize $\bar{c}_I x_I + (K)$ subject to the constraints

$$x_D = A_D^{-1}(b - A_I x_I)$$
$$x_i = 0 \text{ or } 1, \qquad i \in D \text{ and}$$
$$x_i = 0 \text{ or } 1, \qquad i \in I, \qquad (3)$$

with $\bar{c}_I x_I + K = c_I x_I + c_D A_D^{-1}(b - A_I x_I) = cx$,

where $K = c_D A_D^{-1} b$ and $\bar{c}_I = c_I - c_D A_D^{-1} A_I$.

Problem (3), equivalent to problem (1), has $n - m$ variables and the cost coefficients $\bar{c}_i$, $i \in I$, are positive since A_D is an optimal linear programming (LP) basis of problem (2).

In this new problem the components of the m-dimensional x_D vector are no longer variables but express the fact that each of the relations in (3) takes the value zero or one. In fact problem (3) may be interpreted as an integer programming problem with disjunctive constraints of a very special type.

Computational experience with the cutting plane and group theoretic algorithms tends to demonstrate that the optimal solution of problem (3) [and (1)] is closely related to the optimal solution of (2) [2,10,12]. In both cases the optimal solution of (3) is derived rather easily from the optimal solution of (2). This situation is not true, however, for general integer programming problems.

This fact is one of the motivations of the implicit enumeration method we now describe for solving problem (3).

To begin let us call P the set of all the solutions (feasible or infeasible) of problem (3). We partition this set into $(n - m) + 1$ subsets P_j, $j = 0, 1, \cdots, n - m$ where the jth subset consists of all the solutions x_I of (3) with exactly j components equal to one (the remaining components being equal to zero). More precisely the P_j's are defined as

$$P_j = \left\{ x_I | x_I \in P, \quad \sum_{i \in I} x_i = j \right\},$$

where $|P_j| = \binom{n-m}{j}$.

To enumerate the solution of P we shall proceed as follows. We enumerate the solutions of P_0 then of P_1,P_2 and so on until P_{n-m}.

If we define the distance between the vectors in P by $d(x_I, y_I) = \sum_{i \in I} |x_i - y_i|$ the enumerative scheme may be interpreted as follows. Start with $x_I = 0$, the optimal solution of (2), then enumerate all the solutions within a distance of 1 (from the 0 vector), then all the solutions within a distance of 2, 3 and so on until $n - m$. The obtainment of a feasible solution as close as possible to the 0 vector is one of the essential points of the method.

We now give a few definitions.

- *Solution*

A solution will be defined by successively specifying the values of its $n - m$ components to zero or one. The components will be considered one after the other in any order.

• *Partial solution*

At the stage k, we have specified the value of k components $x_{\varphi_1}, \cdots, x_{\varphi_k}$. We shall define a partial solution (of order k) to be the set $S_j(x_{\varphi_1}, \cdots, x_{\varphi_k})$ of all the solutions of P_j that can be deduced by specifying the values of the $(n-m)-k$ remaining components.

• *Associated partial solution*

To a partial solution $S_j(x_{\varphi_1}, \cdots, x_{\varphi_k})$ or order k in P_j corresponds an associated partial solution $S_1(x_{\varphi'_1}, \cdots, x_{\varphi'_k})$ of order k in P_1, $1 = 0, 1, \cdots, n-m$. This associated partial solution in P_1, if existing, is obtained by specifying the components found in P_j to the same value. The components need not be fixed in the same order.

• *Representation of a solution by a partial solution*

When a partial solution of order k has j components equal to one, the only solution of P_j that it is possible to deduce will be the one where the $(n-m)-k$ remaining components are set to zero. In that case we shall identify the partial solution with the solution. During the process of enumeration the best feasible solution enumerated so far will be referred to as $\bar{x}_I$ while $\bar{s}$ will represent $\bar{c}_I \bar{x}_I$.

We now define the enumeration scheme in P_j as follows:

Take P_j as a starting partial solution, i.e., no components are assigned a value.

1. Is the partial solution just defined equivalent to a solution?
   - Yes, go to 3.
   - No, go to 2.
2. Select a new component and assign it a value.
   - Go to 1.
3. Has the partial solution, obtained by giving the last component fixed its complementary value (0 instead of 1 or 1 instead of 0), already been considered?
   - Yes, go to 5.
   - No, go to 4.
4. Give this new value to the last component specified. Go to 2.
5. Rule out (free) this last component. Is the partial solution just defined equal to P_j (backtracking)?
   - Yes, go to 6.
   - No, go to 3.
6. All the solutions have been enumerated; END.

The enumeration procedure described above generates once and only once the $\binom{n-m}{j}$ solutions of P_j. The proof for similar enumeration schemes (the enumeration being in P instead of P_j) can be found in Glover [13] or Geoffrion [14].

As is true in every implicit enumeration method only a subset of all the possible solutions need actually be considered. This will be achieved by ruling out partial solutions for nonoptimality or nonfeasibility considerations and by fixing the components in a special way.

The elimination of the partial solutions is obtained thanks to the following rules:

1. *Infeasibility*

   If a partial solution $S_j(x_{\varphi_1}, \cdots, x_{\varphi_k})$ satisfies

   $$\sum_{j=1}^{k} A_{\varphi_j} x_{\varphi_j} \not\leq b, \quad \text{then} \quad A_I x_I \not\leq b \ \forall\, x_I \in S_j(x_{\varphi_1}, \cdots, x_{\varphi_k}).$$

   (This results from the fact that, in the original problem (1), $A_I x_I + A_D x_D = b$ implies $A_I x_I \leq b$); therefore a feasible solution of (3) must satisfy $A_I x_I \leq b$). Thus this partial solution of P_j is eliminated.

   Practically we shall test, at stage k, whether

   $$A_{\varphi_k} x_{\varphi_k} \leq b - \sum_{j=1}^{k-1} A_{\varphi_j} x_{\varphi_j}.$$

   This will only be necessary if x_{φ_k} is set to one. In that case the test is reduced to $A_{\varphi_k} \leq b - \sum_{i \in I} A_i$, where L is the set of the indices of variables already set to one.

2. *Nonoptimality*

   Similarly, if a partial solution $S_j(x_{\varphi_1}, \cdots, x_{\varphi_k})$ satisfies

   $$\sum_{j=1}^{k} \bar{c}_{\varphi_j} x_{\varphi_j} > \bar{s}, \text{ then } \bar{c}_I x_I > \bar{s} \ \forall\, x_I \in S_j(x_{\varphi_1}, \cdots, x_{\varphi_k})$$

   and this partial solution of P_j is also eliminated.

   As above, the corresponding test, at stage k, is

   $$\bar{c}_{\varphi_k} x_{\varphi_k} > \bar{s} - \sum_{j=1}^{k-1} \bar{c}_{\varphi_j} x_{\varphi_j}$$

   and again this test will be necessary only if x_{φ_k} is fixed to one, the test being reduced to $\bar{c}_{\varphi_k} > \bar{s} - \sum_{i \in L} \bar{c}_i$, with L as defined above.

   In both cases if a partial solution of P_j is eliminated, so are the associated partial solutions of P_1 for $1 \geq j$. When a partial solution is ruled out one must go directly to stage 4 in the enumeration scheme.

During the enumeration process we wish to have a ratio of exclusion as large as possible. For this reason it is desirable to eliminate the partial solutions at an early stage k. This will be achieved by adequately selecting, at stage k, the new variable.

Suppose we have generated a partial solution of order $k-1$ and we are attacking a stage k. For elimination considerations, the partial solution of order k with $x_{\varphi_k} = 1$ will always be considered first (at least implicitly).

The choice of the variable x_{φ_k} will be obtained by ordering the variables x_i. This ordering will vary according to the subsets P_j and the enumeration status (i.e., whether a feasible solution has already been found or not).

For example, in case of P_1, the ordering of the variables will be the one obtained at the end of the LP process of problem (2). Since no solution is a priori eliminated for infeasibility reasons the (partial) solutions actually considered are $S_1(1)$, $S_1(0,1)$, $\cdots$, $S_1(0,0,\cdots,1)$ etc, until a feasible solution x_I is found.

In this case, the generation of the partial solutions will continue by setting directly to zero all the variables for which $\bar{c}_i > \bar{s}$.

Every time a new best solution is found, the value of $\bar{s}$ is updated. Moreover if an upper bound of the optimal value of problem (2) is known at the beginning of the process (take this value instead of $\bar{s}$) still fewer solutions need be enumerated.

We now consider the case of the subset P_j, $j > 1$. If no feasible solution has already been found and if no upper bound of the optimal value of problem (3) is known, a simple way of ordering the indices of the variables is as follows:

Define m blocks (subsets) $S_1, S_2, \cdots, S_m$, where the block S_1 is composed of all the indices of the vectors A_i for which $a_{1i} = 1$; the set S_2 is composed of all the indices of the vectors A_i not in S_1 for which $a_{2i} = 1$; and in general the jth subset S_j is composed of all the indices of the vectors A_i not in $S_1, S_2, \cdots, S_{j-1}$ for which $a_{ji} = 1$. Within a block the indices are ordered arbitrarily.

One method of refinement is to order the indices within a block according to the increasing values of the $\bar{c}_i$ or the $\bar{c}_i/m_i$, where m_i denotes the number of ones of the vector A_i. Although it may seem attractive, this ordering within the sets S_j may well be time consuming, especially in the case of a block including a large number of indices.

At a stage k we shall choose the first variable not already fixed. Suppose φ_k is in block j. If the value $x_{\varphi_k} = 1$ is not rejected by the infeasibility test, then all the indices not yet considered of block j and all the indices of the blocks 1 for which $a_{\varphi_k 1} = 1$ are directly introduced in the next partial solution and the corresponding components are immediately set to zero.

If an upper bound is available or if a (best) feasible solution has been enumerated so far, this scheme may still be used. However, if we want to make the nonoptimality test more powerful, other ways of ordering may be utilized.

One can think of ordering the whole set of the components according to the increasing values of the $\bar{c}_i$. The result is for a given subset P_j the enumeration of the solutions in a lexicographical order. But, as we have already said this ordering process is time consuming for large problems. Hence we suggest the following approach.

Partition the indices into four blocks B_0, B_1, B_2, B_3, where

$$B_0 = \{i | \bar{c}_i = 0\};$$

$$B_1 = \{i | i \notin B_0, \bar{c}_i < \bar{s}/j\};$$

$$B_2 = \{i | i \notin B_0, i \notin B_1, \bar{c}_i < \bar{s}\};$$

$$B_3 = \{i | i \notin B_0, i \notin B_1, i \notin B_2, \bar{c}_i \geq \bar{s}\}.$$

(It is possible to consider $B_0 \cup B_1$ as a single block B_1.)

Incidentally, these blocks will be used later on to derive optimality conditions. Within the blocks the indices are ordered arbitrarily. The indices are selected first in B_0, then in B_1, B_2 and finally in B_3. From such a strategy, good (feasible) solutions are expected and, thanks to the nonoptimality test, it is possible to exclude a large number of solutions (all the components with their indices in B_3 will be automatically set to zero).

The B_i's are dependent on P_j and $\bar{s}$. Therefore during the process of enumeration some of the indices move from one set to another (from B_1 to B_2 or B_2 to B_3). As a result B_3 tends to grow larger and larger while the set $B_0 \cup B_1 \cup B_3$ becomes smaller and smaller. If this set is small enough the bookkeeping task concerning the position of the indices can be simplified if we order the indices in $B_0 \cup B_1 \cup B_3$ (actually only $B_1 \cup B_2$ need be considered) according to the increasing values of the $\bar{c}_i$'s.

If at the same time it is desired to exclude more solutions for infeasibility, it is possible to use the schemes defined for the no-feasible-solution case within each subset B_j. The bookkeeping task will always be kept in mind since in that case a subset of the form S_i, $i = 1, \cdots, m$, may appear in the four blocks B_j.

When the enumeration process terminates we are in one of the following situations:

1. No feasible solution has been found and the problem is infeasible.
2. The best feasible solution enumerated is obviously the optimal solution of the problem.

This result may be achieved prior to the complete exhaustion of the partial solutions thanks to the following rules.

1. *Infeasibility*

   As already pointed out, any feasible vector x_I of P must satisfy $\sum_{i \in L} A_i \leq b$, where again L denotes the set of the indices of the variables (of the x_I vector) set to one.

   Therefore any solution of a subset P_j is infeasible for $j > m$. Suppose that more than m vectors A_i are present. At least two of them have a same component equal to one and the vector inequality cannot be satisfied. As a result none of the P_j's $j > m$ need be considered in the enumeration process.

2. *Nonoptimality*

   Similarly, a solution x_I of P with a cost less than $\bar{s}$ satisfies $\sum_{I \in L} \bar{c}_i < \bar{s}$.

Applied to P_j this yields $\bar{c}_i < \bar{s}\ \forall\ i \in L$ and $\bar{c}_i < \bar{s}/j$ for at least one component equal to one.

Let us partition the indices of the variables of problem (3) into three subsets defined as follows:

$$B_1 = \{i | \bar{c}_i < \bar{s}/j\}\ ;$$

$$B_2 = \{i | i \notin B_1,\ \bar{c}_i < \bar{s}\}\ ;$$

$$B_3 = \{i | i \notin B_1,\ i \notin B_2,\ \bar{c}_i \geq \bar{s}\}\ .$$

If $|B_1| = 0$ or $|B_1| + |B_2| < j$ in P_j, then no better value than $\bar{s}$ can be obtained in P_j. Since the above relations are also true for $1 > j$ we conclude that $\bar{x}_I$ is an optimal solution of problem (3).

Finally, in the case where the cost components of problem (1) are integers (any practical problem can be formulated in this way, using an adequate scale), another useful relation implying the optimality of $\bar{x}_I$ is given by $\bar{s} \leq \{K\} - K$, where K is the optimal value of problem (1) [defined after Eq. (3)] and $\{K\}$ the smallest integer greater than or equal to K.

Extensions and conclusions

As we have already said, the method is also valid in the case of an original problem (1) with $a_{ij} \geq 0$ and $b_i \geq 0$, $j = 1, \cdots, n$ and $i = 1, \cdots, m$. This is obvious since nonnegativity is the only assumption used in the exclusion rules. However, in this case some of the results are slightly modified. For instance there is no reason for the solutions of the P_j's to be infeasible a priori for $j > m$. (In some cases a weaker condition may be drawn from the values of the b_i's.) Also, in the definition of the S_i's the condition $a_{ij} = 1$ must be replaced by $a_{ij} > 0$, in which case the application of the infeasibility rules is not so straightforward. And, of course, in this general case the optimal solution of problem (3) may be very far from the solution of (2), making the enumeration process much longer.

We have written an experimental FORTRAN code based on the method we have just described; the performances of this code are largely dependent of its implementation. This implementation will not be described here, but let us say that, for instance, the LP procedure must be very efficient. In many cases solving the LP part with an inefficient LP code will require more time than to solve the entire problem with a code like the one of Pierce [6]; see also Ref. 15.

To conclude, we would like to give some of the computational experiences we have obtained so far. Not very large problems with 10 to 20 constraints and 5 to 250 variables were run on an IBM 1130 in approximately from 5 to 30 seconds. Larger problems are reviewed in Table 1.

Most of these problems are to be found, with their data, in Marilley's thesis [15]. All problems were run on an IBM 360/75. In the rightmost column are the timings obtained by Marilley on an IBM 360/65 using Pierce's method.

Table 1. Computational experience.

Size ($m \times n$)	*Problem type*	*Time in seconds* *Proposed method*	*Pierce method*
42 × 249	crew scheduling	6.5	600[a]
67 × 536	crew scheduling	20	600[a]
12 × 538	delivery	5	41.3
18 × 273	stability	5	

[a]No optimal solution after 600 seconds.

On more than 25 practical problems, the computation time obtained was always good and the optimal solutions were always found in a level less than four (two was a very frequent value).

At first it would seem that the level of the optimal solution is closely related to the problem-solving time. However, matters are much more complicated and in many cases other factors are at least as important, e.g.,

- the level of the first feasible solution (if no value is assigned to $\bar{s}$ at the beginning of the enumeration process);
- the starting point (and the number of iterations) of the LP;
- the optimal basis used.

This last point is extremely important since, for degeneracy reasons, the optimal basis of the continuous problem is not unique, even in the case of a unique optimal continuous solution, and the levels of the first feasible solution and of the optimal solution are dependent on the optimal basis selected. This dependence is still more complicated in the case of more than one optimal solution, for the LP problem as well as for the partitioning problem.

For these reasons all the numerical experiments mentioned here have been made in the following way: No starting solution was used for the LP (a phase 1 was used), the nonbasic variables used for the enumeration were the ones naturally found by the LP algorithm, and no initial feasible solution or bound was assumed to start the enumeration.

References

1. M. L. Balinski, "Integer Programming: Methods, Uses, Computation" *Management Science* **12,** 253 (1965).
2. M. L. Balinski, and R. E. Quandt, "On an Integer Program for a Delivery Problem", *Operations Research* **12,** 300 (1964).
3. A. H. Land and A. G. Doig, "An Automatic Method of Solving Discrete Programming Problems", *Econometrica* **28,** 497 (1960).

4. E. Balas, "An Additive Algorithm for Solving Linear Programs with Zero-One Variables", *Operations Research* **13,** 517 (1965).
5. R. S. Garfinkel and G. L. Nemhauser, "The Set Partitioning Problem: Set Covering with Equality Constraints", *Operations Research* **17,** 848 (1969).
6. J. F. Pierce, "Application of Combinatorial Programming to a Class of All-Zero-One Integer Programming Problems", *Management Science* **15,** 191 (1968).
7. J. Rubin, "A Technique for the Solution of Massive Set Covering Problems, with Application to Air Line Crew Scheduling" IBM Philadelphia Scientific Center Technical Report 320-3004 1971.
8. R. E. Gomory, "An Algorithm for Integer Solutions to Linear Programs", *Recent Advances in Mathematical Programming,* McGraw-Hill Book Co., Inc., New York, 1963, p. 269.
9. R. E. Gomory, "On the Relation between Integer and Non-Integer Solutions to Linear Programs", *Proc. National Academy of Sci.* **53,** 250 (1965).
10. G. Martin, "An Accelerated Euclidean Algorithm for Integer Linear Programming", *Recent Advances in Mathematical Programming* (Graves and Wolfe Ed.), McGraw-Hill Book Co. Inc., New York, 1963.
11. J. F. Shapiro, "Group Theoretic Algorithms for the Integer Programming Problem-II: Extension to a General Algorithm", *Operations Research* **16,** 928 (1968).
12. H. Thiriez, "Airline Crew Scheduling: a Group Theoric Approach", *FTL Report R-69-1,* MIT, 1969.
13. F. Glover, "A Multiphase-Dual Algorithm for the Zero-one Integer Programming Problem", *Operations Research* **13,** 879 (1965).
14. A. M. Geoffrion, "Integer Programming by Implicit Enumeration and Balas' Method", *SIAM Rev.* **9,** 178 (1967).
15. J. P. Marilley, "Contribution à l'Etude Pratique d'une Classe de Programmes Linéaires en Nombres Entiers les Problèmes de Recouvrement Exact", *CNAM Thesis,* Conservatoire des Arts et Métiers, Paris, 1970.
16. M. L. Balinski and K. Spielberg, "Method for Integer Programming: Algebraic, Combinatorial and Enumerative", *Progress in Operations Research,* Vol. III (Aronofsky, Ed.), Wiley, New York, 1969.

Received November 3, 1971

The author is located at the IBM World Trade Corporation Paris Scientific Center, 20, avenue Gambetta, 92 Courbevoie, France.