Mathematical Programming 18 (1980) 138–145.
North-Holland Publishing Company

A NOTE ON SOME COMPUTATIONALLY DIFFICULT SET COVERING PROBLEMS

David AVIS*

McGill University, Montréal, Québec, Canada

Received 3 October 1978
Revised manuscript received 25 April 1979

Fulkerson et al. have given two examples of set covering problems that are empirically difficult to solve. They arise from Steiner triple systems and the larger problem, which has a constraint matrix of size 330×45 has only recently been solved. In this note, we show that the Steiner triple systems do indeed give rise to a series of problems that are probably hard to solve by implicit enumeration. The main result is that for an n variable problem, branch and bound algorithms using a linear programming relaxation, and/or elimination by dominance require the examination of a super-polynomial number of partial solutions

Key words: Set-covering Problem, Branch and Bound, Lower Bounds, Steiner Triple Systems.

1. Introduction

Fulkerson et al. [5] have given two empirically difficult to solve set covering problems arising from Steiner triple systems. In this note, we use a standard construction to generate a series of Steiner triple systems that cannot be solved efficiently by branch and bound methods that employ linear programming and/or dominance as a fathoming device. Such procedures are the basis of most successful approaches to this problem (e.g., Fulkerson et al. [5], Geoffrion [7], Lemke et al. [9]). More precisely, we exhibit a family of Steiner triple systems with $n = 3^k$ $(k = 1, 2, ...)$ variables, for which algorithms of this type require the examination of $2^{\sqrt{2n/3}}$ partial solutions. Results along these lines are still somewhat rare, although it is reasonable to suspect that similar results can be found for most NP-hard problems. This note was motivated by a result in the same vein for the Knapsack problem, recently obtained by Chvátal [1], where a very large class of computationally difficult problems is exhibited. Very deep results have been obtained for other NP-complete problems by Chvátal [3], Cook and Reckhow [4] and McDiarmid [10].

Since the set covering problem is NP-complete, the reader is hardly likely to be surprised that there exist problems that cannot be solved by branch and bound in a polynomial time bound. However, as pointed out in Fulkerson et al.

* This paper was written while the author was a CORE Fellow at the Université de Louvain, Louvain-la-Neuve, Belgium.

[5], there are many claims in the literature for algorithms that solve problems with a very large number of variables. The Steiner triple problems provide a set of compact and very easily generated problems that should provide a challenge for new ideas and techniques in integer programming.

2. The problems

We consider problems of the following form

$$\begin{aligned} w(A) = \min\ & e \cdot x, \\ & Ax \geq e, \\ & x = 0, 1 \end{aligned} \tag{1}$$

where $A = (a_{ij})$ is an $m \times n$ zero–one matrix, x is a vector of length n and e is the vector of ones of appropriate length. It is convenient to consider the columns of A as representing elements and the rows of A as representing certain subsets of elements. Such matrices are called *incidence matrices*. The problem is to find a minimum set of elements that represent all of the subsets.

The incidence matrices A that we consider arise from Steiner triple systems and have precisely 3 ones in each row. These matrices are characterized as follows: for every pair of columns j_1 and j_2, there exists exactly one row i such that $a_{ij_1} = a_{ij_2} = 1$. We say that $\{i, j, k\}$ is a *triple* of A if there exists a row r of A such that $a_{ri} = a_{rj} = a_{rk} = 1$. It is well-known that such matrices exist if and only if $n \geq 3$ and $n \equiv 1, 3 \pmod 6$, in which case $m = \frac{1}{6}n(n-1)$ and each column of A contains $\frac{1}{2}(n-1)$ non-zero entries. The interested reader is referred to Hall [8] for a wealth of information on this type of structure, which is an instance of a balanced incomplete block design.

We are interested in a class of Steiner systems for which $n = 3^k$ $(k = 1, 2, 3, \ldots)$ that are obtained recursively by a standard technique (see Hall [8]). A_3 is simply 1×3 matrix of all ones. We obtain A_{3n} from A_n as follows. The columns of A_{3n} are indexed $\{(i, j)\colon 1 \leq i \leq n,\ 1 \leq j \leq 3\}$. The set $\{(i, r), (j, s), (k, t)\}$ is a triple of A_{3n} if and only if one of the following conditions hold:

(i) $r = s = t$ and $\{i, j, k\}$ is a triple of A_n, or

(ii) $i = j = k$ and $\{r, s, t\} = \{1, 2, 3\}$, or

(iii) $\{i, j, k\}$ is a triple of A_n and $\{r, s, t\} = \{1, 2, 3\}$.

To help clarify the construction we give the following informal description. The rows of A_{3n} can be divided into three parts corresponding to conditions (i)–(iii). The first two parts look like

$$\begin{bmatrix} A_n & 0 & 0 \\ 0 & A_n & 0 \\ 0 & 0 & A_n \\ I & I & I \end{bmatrix}$$

where I is the identity matrix of suitable dimension. The third part consists of $3! = 6$ blocks corresponding to the permutations of $\{1, 2, 3\}$; for each permutation π, the triples of the corresponding block are

$$\{(i, \pi(1)), (j, \pi(2)), (k, \pi(3))\}$$

where (i, j, k) runs through all triples of A_n.

It is easily verified that A_{3n} is again the incidence matrix of a Steiner triple system. An important fact that we will need later is that three disjoint copies of A_n appear in A_{3n}. Fig. 1 contains A_9, and the three disjoint copies of A_3 are indicated. A_{27} has size 117×27

$A_9 =$

{1,1}	{2,1}	{3,1}	{1,2}	{2,2}	{3,2}	{1,3}	{2,3}	{3,3}
1	1	1	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	1	1	1
1	0	0	1	0	0	1	0	0
0	1	0	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1
1	0	0	0	0	1	0	1	0
0	1	0	1	0	0	0	0	1
0	1	0	0	0	1	1	0	0
0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	1	0	0

Fig. 1.

and is the first of the two problems described in Fulkerson et al. [5]. They have solved (1) with constraint matrices A_9 and A_{27} obtaining $w(A_9) = 5$ and $w(A_{27}) = 18$. Their larger problem, which has size 330×45, has only recently been solved by H. Ratliffe (personal communication, 1979), who shows that $w(A_{45}) = 30$. The solution of this problem involved over $2\frac{1}{2}$ hours of computation on the large Amdahl V7.

We conclude this section by grouping the combinatorial properties of these incidence matrices that will be used in the proofs in Sections 4 and 5.

(P1) Every row of A_n contains exactly 3 ones.

(P2) Every column of A_n contains exactly $\frac{1}{2}(n-1)$ ones.

(P3) For every pair of rows of A_n, there is exactly one column in which they both contain a one.

(P4) For every triple $\{(i, r), (j, s), (k, t)\}$ of A_{3n} either $r = s = t$ or $\{r, s, t\} = \{1, 2, 3\}$.

3. Recursive algorithms

For the class of algorithms that we consider, we borrow the following model from Chvátal [1]. A *recursive algorithm* to solve (1) considers *partial vectors* of

the form $x_J = \{x_j: j \in J\}$ such that $J \subseteq \{1, 2, \ldots, n\}$ and each x_j is zero or one. At each stage we have a list of such vectors along with some feasible solution $x^* = (x_1^*, \ldots, x_n^*)$ of (1). Some partial vector extends to an optimal solution, or else x^* is optimal; this property is preserved throughout. We begin with $J = \emptyset$ and some feasible solution ($x^* = e$ will always work). For each partial vector x_J we define the set $U(x_J)$ of *uncovered rows* of A by

$$U(x_J) = \left\{i: \sum_{j \in J} a_{ij} x_j = 0\right\}.$$

An iteration of the algorithm consists of one of the following alternatives:

(1)(a) (Branching). Replace some partial vector x_J on the list by partial vectors y_K and z_K such that $K = J \cup \{k\}$ for some $k \notin J$, $y_k = 1$, $z_k = 0$ and $y_j = z_j = x_j$ whenever $j \in J$. (b) (Augmenting). Extend z_K to $z_{K \cup M}$, where $z_M = e$ and

$$M = \left\{l \notin K: \sum_{j \notin K} a_{ij} = a_{il} = 1 \text{ for some } i \in U(z_K)\right\}.$$

(2) (Bounding). Consider a partial vector x_J on the list and the linear programming relaxation

$$\begin{aligned} \bar{w}(x_J) = \min & \sum_{j \notin J} x_j + e \cdot x_J, \\ & \sum_{j \notin J} a_{ij} x_j \geq 1 \quad (i \in U(x_J)), \qquad (2) \\ & 0 \leq x_j \leq 1 \quad (j \notin J). \end{aligned}$$

If (2) has no solution or if its optimal solution is at least $e \cdot x^*$, then x_J is *bounded* and is deleted from the list.

(3) (Dominating). Choose partial vectors x_J and y_J. If $e \cdot x_J \geq e \cdot y_J$ and $U(y_J) \subseteq U(x_J)$, then y_J *dominates* x_J and x_J is removed from the list.

(4) (Improving the current solution). If the list contains a feasible solution x of (1) such that $e \cdot x < e \cdot x^*$, then replace x^* by x and delete x from the list.

The algorithm terminates when the list becomes empty, at which point x^* is an optimal solution of (1). One often describes the operation of such algorithms in terms of searching a binary tree. The root of such a tree is just the empty partial vector. Step 1(a) corresponds to taking a node of the tree and adjoining two descendants, each corresponding to a partial vector. The *depth* of a partial vector x_J is the distance to the root from the corresponding node in the search tree, or alternatively the number of times Step 1(a) was executed in arriving at x_J. A general description of implicit enumeration schemes as well as many detailed examples is contained in Garfinkel and Nemhauser [6].

We have used the structure of the set covering problem to allow the augmenting phase of Step 1, which arises from the following simple observation. We can consider the columns of A corresponding to zeroes in the partial solution vector

x_J as being deleted. If there remain uncovered rows in the reduced matrix with exactly one non-zero entry, then the variables corresponding to these columns must be set to one in any feasible extension of x_J. We assume that such variables are set to one as part of Step 1, that is, that Step 1(b) always follows Step 1(a). This is referred to as a reduction in Garfinkel and Nemhauser [6, p. 302]. It implies that for the matrices described in Section 2, the LP relaxation in Step 2 will always be feasible, and fathoming only occurs if the relaxation yields a higher objective value than the current feasible integer solution. The inclusion of Step 1(b) simplifies the analysis of these algorithms, although Theorem 1 remains true for algorithms that do not use the augmentation feature.

4. The main result

In this section we prove the following theorem.

Theorem 1. *Every recursive algorithm for solving* (1) *with constraint matrix* A_n, $n \geq 27$, *requires the creation of at least* $2^{\sqrt{2n/3}}$ *partial vectors.*

Proof. We will show that at level $q = [\sqrt{2n/3}]$ in the search tree, there exist 2^q nodes ($[x]$ is the smallest integer greater or equal to x). That is, neither Step 2 nor Step 3 will remove any partial vectors from the tree until a depth greater than q is reached. Let x_J be any partial vector in the tree. We make the following definitions:

j_0 = number of variables x_j ($j \in J$) fixed at zero,
j_1 = number of variables x_j ($j \in J$) fixed at one by branching,
j_2 = number of variables x_j ($j \in J$) fixed at one by augmenting.

It is easily seen that the following relationships hold:

$$|J| = j_0 + j_1 + j_2, \quad q = j_0 + j_1, \quad j_2 \leq \tfrac{1}{2} j_0(j_0 - 1).$$

The inequality follows from property (P3): since no two rows of A_n can have ones in the same two columns, every pair of variables that is set to zero forces at most one additional variable to be set to one. We will now obtain an upper bound on the value $\bar{w}(x_J)$ of the LP relaxation with fixed variables x_J.

A feasible solution to this program may be obtained by setting $x_j = \frac{1}{2} (j \notin J)$, provided we perform Step 1(b) after Step 1(a). Therefore

$$\begin{aligned} \bar{w}(x_J) &\leq \tfrac{1}{2}(n - |J|) + j_1 + j_2 \leq \tfrac{1}{2} n + \tfrac{1}{2} j_1 + \tfrac{1}{4} j_0^2 - \tfrac{3}{4} j_0 \\ &= \tfrac{1}{2} n + \tfrac{1}{4}(j_0 + j_1)^2 - \tfrac{1}{2}(j_0 + j_1) - \tfrac{1}{2} j_0 j_1 - \tfrac{1}{4} j_1^2 + j_1 - \tfrac{1}{4} j_0 \\ &\leq \tfrac{1}{2} n + \tfrac{1}{4} q^2 - \tfrac{1}{2} q - \tfrac{1}{4} \leq \tfrac{1}{2} n + \tfrac{1}{4}(q-1)^2 < \tfrac{2}{3} n \quad \text{for } n \geq 27. \end{aligned}$$

To show that x_J is not fathomed, we need to show that $w(A_n) \geq \frac{2}{3} n$. This follows

from the fact that $w(A_{27}) = 18$ (see Fulkerson et al. [1]) and the fact that A_{3n} contains three disjoint copies of A_n, therefore $w(A_{3n}) \geq 3w(A_n)$.

We will now show that elimination by dominance does not remove any partial vector from the list. Let x_J and y_J be defined as in Step 3 and let i be some index such that $x_i = 1$ and $y_i = 0$. If such an index does not exist then $x_J = y_J$ which is impossible. By (P2), column i of A contains $(n-1)/2$ non-zero entries. The corresponding rows of A must be covered by y_J, because these rows are covered by x_J and $U(y_J) \subseteq U(x_J)$. Let I denote the index set of these rows. Since all these rows have a non-zero entry in column i, by (P3) there exists no other column in which any pair of these rows both contain a one. Therefore y_J contains at least $(n-1)/2$ non-zero elements, one to cover each row of I. At a depth q in the search tree the number of non-zeroes in any partial vector is

$$j_1 + j_2 \leq j_1 + \tfrac{1}{2}j_0(j_0 - 1) \leq \tfrac{1}{2}q(q-1) < \tfrac{1}{2}(n-1) \quad \text{for } n \geq 27.$$

Therefore elimination by dominance can remove no partial vector from the list at a depth of q or less.

5. An upper bound

In this section we derive an upper bound on $w(A_n)$. We begin with a definition. Let J be a set of columns of A_n and let I be defined by

$$I = \left\{ i : \sum_{j \in J} a_{ij} = 3 \right\}. \tag{3}$$

We say that a subset K of J is a *cover* (of J) if

$$\sum_{j \in K} a_{ij} \geq 1 \quad (i \in I).$$

The importance of a cover derives from the fact that any partial vector x_J can be extended into a feasible solution of (1) if $x_K = e$. In particular, if $J = \{1, 2, \ldots, n\}$, then $|K|$ is an upper bound on $w(A_n)$. In what follows, let $c = 1/\log_2 3$.

Theorem 2. *Every set J of columns of A_n admits a cover of size at most $|J| - |J|^c$.*

Proof. The theorem is easily verified for A_3. We assume inductively that the result is true for A_n. We label the columns of A_{3n} by the ordered pair (i, j) as described in Section 2. Let J be any set of columns of A_{3n} and set $q = |J|$. Let q_j $(j = 1, 2, 3)$ be the number of columns of J which are labelled with last co-ordinate j. We may assume, by relabeling if necessary, that $q_1 \leq q_2 \leq q_3$. The cover of J is constructed as follows.

Begin by taking the q_1 columns of J with last co-ordinate 1. Delete all the row indices from the corresponding set I, defined by (3), that correspond to rows that

have now been covered. It follows from property (P4) that the remaining elements in I correspond to rows of A_{3n} in which *all* the ones occur in columns with the *same* last index: either 2 or 3. Thus the original problem reduces to two subproblems, to each of which we can apply the inductive hypothesis. It follows that a cover of size at most $f(q_1, q_2, q_3)$ can be found, where

$$f(q_1, q_2, q_3) = q_1 + q_2 - q_2^c + q_3 - q_3^c.$$

To see how large $f(q_1, q_2, q_3)$ can get, we consider the program:

$$\begin{aligned} \text{maximize} \quad & f(q_1, q_2, q_3), \\ \text{subject to} \quad & q_1 + q_2 + q_3 = q, \\ & q_2 - q_3 \leq 0, \\ & q_1 - q_2 \geq 0, \\ & q_1 \geq 0. \end{aligned}$$

It is easily verified that f is a convex function. As is well-known, the maximum of such a function occurs at an extreme point of the feasible region. Here it can be seen by inspection that the maximum occurs simultaneously at the extreme points $(0, 0, q)$ and $(\frac{1}{3}q, \frac{1}{3}q, \frac{1}{3}q)$ with objective value $q - q^c$. Thus the theorem follows.

Corollary. $w(A_n) \leq n - 2^k$, *where* $n = 3^k$.

This result has an interesting relationship to a certain heuristic solution of the original problem (1). This heuristic, known as GREEDY, operates by selecting a column of A with the maximum number of ones. This column is deleted from A together with all rows that had a one in this column. The procedure is repeated until a cover is found. We make this procedure definite by adding the rule that ties will be broken in the column selection process by choosing the column of least index. Worst case bounds are given by Chvátal [2] for GREEDY, but in this case we can compute the solution exactly. It turns out to be the cover constructed in Theorem 2 with weight $n - 2^k$. As an example of Theorem 2 consider the submatrix of A_9 given in Fig. 2, with $|J| = 7$. The method used in the proof of the theorem constructs the cover consisting of columns $\{\{1, 1\}, \{2, 1\}, \{1, 3\}\}$ with cardinality 3, which agrees with the theorem.

$$\begin{array}{c} \{1,1\}\{2,1\}\{2,2\}\{3,2\}\{1,3\}\{2,3\}\{3,3\} \\ \begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix} \end{array}$$

Fig. 2.

Acknowledgment

The author gratefully acknowledges stimulating conversations with V. Chvátal and many useful comments from J. Tind, who read an earlier version of this paper. I am also indebted to the referees for suggesting ways of improving the narrative.

References

[1] V. Chvátal, "Hard knapsack problems", *Journal of Operations Research*, to appear.
[2] V. Chvátal, "A greedy heuristic for the set covering problem", Publication No. 284, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal (1978).
[3] V. Chvátal, "Determining the stability number of a graph", *SIAM Journal on Computing* 6 (1977) 643–662.
[4] S.A. Cook and R.A. Reckhow, "On the Lengths of Proofs in the Propositional Calculus", *Proceedings of 6th Annual ACM Symposium on Theory of Computing* (1974) 135–148.
[5] R. Fulkerson, G. Nemhauser and L. Trotter, "Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems", *Mathematical Programming Study 2* (1974) 72–81.
[6] R. Garfinkel and G. Nemhauser, *Integer programming* (Wiley, New York, 1969).
[7] A. Geoffrion, "An improved implicit enumeration approach for integer programming", *Journal of Operations Research* 17 (1969) 437–454.
[8] M. Hall Jr., *Combinatorial theory* (Blaisdell Company, Waltham, MA, 1967).
[9] C.E. Lemke, H.M. Salkin and K. Spielberg, "Set covering by single branch enumeration with linear programming subproblems", *Journal of Operations Research* 19 (1971) 998–1022.
[10] C. McDiarmid, "Determining the chromatic number of a graph", *SIAM Journal on Computing* 8 (1979) 1–14.