Operations Research Letters 8 (1989) 67–71
North-Holland

A PROBABILISTIC HEURISTIC FOR A COMPUTATIONALLY DIFFICULT SET COVERING PROBLEM *

Thomas A. FEO
The University of Texas, Austin, TX 78712, USA

Mauricio G.C. RESENDE
University of California, Berkeley, CA 94720, USA

Received May 1988
Revised October 1988

An efficient probabilistic set covering heuristic is presented. The heuristic is evaluated on empirically difficult to solve set covering problems that arise from Steiner triple systems. The optimal solution to only a few of these instances is known. The heuristic provides these solutions as well as the best known solutions to all other instances attempted.

algorithms * heuristic * tests * integer programming * set covering

1. Introduction

Given n finite sets $P_1, P_2, \ldots, P_n$ we denote sets $I = \bigcup(P_j\colon 1 \leqslant j \leqslant n) = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$. A subset J^* of J is called a *cover* if $\bigcup(P_j\colon j \in J^*) = I$. The *set covering problem* is to find a cover of minimum cardinality. Define the $m \times n$ (0, 1)-matrix A such that $a_{ij} = 1$ if and only if $i \in P_j$ An integer programming formulation for the set covering problem is

$$\text{minimize} \quad e_n x \tag{1}$$

$$\text{subject to} \quad Ax \geqslant e_m, \tag{2}$$

$$x = 0, 1, \tag{3}$$

where e_k is a vector of ones of length k, and x is a (0, 1)-vector of length n with $x_j = 1$ if and only if $j \in J^*$. Set covering is a well known NP-complete problem [8].

Fulkerson, Nemhauser and Trotter [7] describe a class of computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems. They suggest that these are good problems for testing new algorithms for integer programming and set covering. This is because they have far fewer variables than numerous solved problems in the literature, yet experience shows that they are hard to compute and verify. The β-width of a (0, 1)-matrix A is the minimum number of columns that can be selected from A such that all row sums of the resulting submatrix of A are at least β. The incidence matrices A that arise from Steiner triple systems have precisely 3 ones per row. Furthermore, for every pair of columns j and k there is exactly one row i for which $a_{ij} = a_{ik} = 1$. (i, j, k) are said to be a triple of A if there exists a row q such that $a_{qi} = a_{qj} = q_{qk} = 1$. Hall [10] discusses this structure in detail and shows a standard technique for recursively generating Steiner systems for which $n = 3^k$ ($k = 1, 2, 3, \ldots$). A_3 is the 1×3 matrix of ones. A_{3n} is obtained from A_n as follows: The columns of A_{3n} are indexed $\{(i, j), 1 \leqslant i \leqslant n, 1 \leqslant j \leqslant 3\}$. The set $\{(i, r), (j, s), (k, t)\}$ is a triple of A_{3n} if and only if one of the following holds:

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

* This research was partially funded by Project QUEST at the University of Texas at Austin.

0167-6377/89/$3.50

We refer to instances of set covering problems that arise from Steiner triple systems by their incidence matrices. Two examples for which $n \neq 3^k$ are given in [7]: A_{15} and A_{45}.

Fulkerson, Nemhauser and Trotter [7] discuss computational experience with A_9, A_{15}, A_{27} and A_{45}. They are able to solve A_9 with a cutting plane code after generating 44 cuts, but this approach fails with the three other problems. Using an implicit enumeration algorithm similar to the one described in [9] they are able to solve A_{15} and A_{27} but not A_{45}. Avis [1] reports that A_{45} was solved in 1979 by H. Ratliff, requiring over two and a half hours on an Amdahl V7 computer. Avis also suggests why these problems may be so difficult to solve by showing that any branch and bound algorithm that uses a linear programming relaxation, and/or elimination by dominance requires the examination of $2^{\sqrt{2n/3}}$ partial solutions, where n is the number of variables of the integer program.

In this paper, we pursue a non-deterministic method for solving these difficult set covering problems. The procedure is based on Chvátal's iterative cost to benefit greedy approach [5]. In accordance with the terminology defined by Hart and Shogan [11] our method can be classified as either a percentage-based or cardinality-based semi-greedy heuristic. In order to improve upon Chvátal's heuristic we introduce randomization. Our intent is similar in nature to the deterministic work of Balas and Ho [2]. In their paper, several variations of the greedy cost to benefit objective are tried, such as taking the ratio of the cost to the logarithm of the benefit.

Past empirical experience with our probabilistic method has been very good. Bard and Feo have incorporated various implementations of this heuristic into the solution methods of practical problems involving corporate acquisition of flexible manufacturing equipment [3], computer aided process planning [4], and maintenance scheduling for major airlines [6]. In all these studies the solutions provided by the probabilistic approach dominate those generated by Chvátal's method. Given our heuristic's success on these real world set covering problems, it is interesting to note its performance on a theoretically based problem deemed in the literature to be very difficult.

In Section 2, we describe the probabilistic heuristic and discuss its relation to the deterministic method fo Chvátal [5]. We define efficient data structures and derive worst case time and storage requirements for our procedure. In Section 3, we describe the computational experiment. We conclude the paper with an intuitive discussion of why the probabilistic approach works well.

2. The heuristic

Chvátal [5] describes a greedy heuristic for the set covering problem and establishes a tight bound on its worst-case behaviour. Chvátal's method is shown in Pseudo-Code 1. It takes as input the n discrete sets $P_1, \ldots, P_n$ defined earlier and returns a cover J^*. In line 1 of the pseudo-code the cover is initially set empty. The loop in lines 2–6 is repeated until all n sets $P_1, \ldots, P_n$ are empty, i.e. until a cover is constructed. In line 3 the subscript k maximizing $\{|P_1|, \ldots, |P_n|\}$ is selected, where $|P|$ denotes the cardinality of set P. This subscript is added to J^* in line 4, and in line 5, P_k is subtracted from sets $P_1, \ldots, P_n$.

The heuristic presented in this paper (Pseudo-Code 2) is a non-deterministic variation of Chvátal's greedy approach. Our method departs from Chvátal's in line 3 of Pseudo-Code 1, where the index k is chosen. Instead of selecting the index k corresponding to the set P_j with the maximum cardinality, we select *at random* from the sets that have cardinality at least $\alpha \times \max\{|P_j|: 1 \leqslant j \leqslant n\}$, where $0 \leqslant \alpha \leqslant 1$. Furthermore, we remove any superfluous elements from the partial cover $J^0 \cup \{k\}$. Chvátal's method is executed once, whereas the probabilistic version is repeated N times in the loop going from line 2 to 14. Our heuristic takes as input sets $P_1, \ldots, P_n$, the parameter α and the number of repetitions N, and returns a cover J^*. In line 1 of the pseudo-code, Γ, the cardinality of the best cover found so far is

Pseudo-Code 1
Chvátal's greedy heuristic

```
procedure GREEDY (n, P_1,..., P_n, J*)
1 Set J* := ∅;
2 do while P_j ≠ ∅, ∀j = 1,..., n →
3     k := argmax{ |P_j| : 1 ≤ j ≤ n };
4     J* := J* ∪ {k};
5     do j = 1,..., n → P_j := P_j − P_k od;
6 od;
end GREEDY;
```

Pseudo-Code 2
Probabilistic heuristic

```
procedure PROBABILISTIC (n, P_1,..., P_n, α, N, J*)
 1 Set Γ := n;
 2 do i = 1,..., N →
 3     do j = 1,..., n → P_j^0 := P_j od;
 4     Set J^0 = ∅;
 5     do while P_j^0 ≠ ∅, ∀j = 1,..., n →
 6         Γ̄ := max{ |P_j^0| : 1 ≤ j ≤ n };
 7         𝒫 := { j : |P_j^0| ≥ α × Γ̄, 1 ≤ j ≤ n };
 8         Select k at random from 𝒫;
 9         J^0 := J^0 ∪ {k};
10         do j = 1,..., n → P_j^0 := P_j^0 + P_k^0 od;
11         Remove superfluous j from J^0;
12     od;
13     if |J^0| < Γ → Γ := |J^0|; J* := J^0 fi;
14 od;
end PROBABILISTIC;
```

initialized to n. In line 3, working sets $P_1^0, \ldots, P_n^0$ are initialized and in line 4 the cover for repetition i, J^0, is initialized empty. In the while loop going from line 5 to 11, the i-th cover is constructed. In line 6 the maximum cardinality, $\bar{\Gamma}$, of working sets $P_1^0, \ldots, P_n^0$ is computed. $\mathscr{P}$, in line 7, is the set of indices corresponding to the sets whose inclusion in cover J^0 will cover at least $\alpha \times \bar{\Gamma}$ elements of set $J = \{1, \ldots, m\}$. In line 8 index k is selected at random from the set $\mathscr{P}$ of candidate indices, and this index is added to J^0 in line 9. In line 10 sets $P_1^0, \ldots, P_n^0$ are updated by set P_k^0 and in line 11 any superfluous element is removed from the partial cover. Finally, in line 13, if a better cover is found in iteration i, this cover, J^*, is recorded. Chvátal's greedy heuristic is a special case of this probabilistic procedure, where $\alpha = 1.0$ and $N = 1$.

The computationally burdensome operations in our method involve: (1) ranking the set of unchosen elements; (2) updating the benefits of the remaining elements after one is selected; and (3) removing superfluous elements. To efficiently implement these operations, we employ a triple pointer data structure. These pointers keep track of the rank of the candidates, identify what elements belong to each set, and denote what sets contain each element. Within an iteration of the *while* loop, the ranking and updating procedures require $O(n)$ time. Note that it is not beneficial to use a heap data structure for these operations because each column of A possesses $\frac{1}{2}(n-1)$ entries. Removing a maximum number of superfluous elements from a partial cover (line 11) is, in itself, a set covering problem. That is, find the smallest cardinality subset of the selected set of elements that covers the currently satisfied set of constraints. We employ a simplified version of Chvátal's heuristic in conjunction with the data structures mentioned. This operation requires $O(n)$ time.

Since there can be up to n elements in a cover and, thus, n iterations of the *while* loop, our method requires $O(n^2)$ time. Note that the number of nonzero elements in the matrix A is $O(n^2)$. Therefore, our heuristic's running time per constructed cover is linear with respect to the input size of the problem. The space requirement of our method is $O(n^2)$.

3. Computational experience

The computational experiment tested the probabilistic heuristic on six set covering problems: A_9, A_{15}, A_{27}, A_{45}, A_{81} and A_{243}.

Problems A_9, A_{15}, A_{27} and A_{45} are taken from [7] and problems A_{81} and A_{243} were generated by Ramakrishnan [13] using the recursive method of Hall [10]. Of these problems, optimal solutions are known only for the first four.

The algorithm was implemented in FORTRAN and the tests were carried out on an IBM 3090 running VM/SP CMS. The FORTVS compiler was used to compile the code using options OPT(3), NOSYM and NOSDUMP. All times are measured with the utility routine DATETM.

Table 1 identifies the test problems, the sizes of the best known covers, and the results obtained by both Chvátal's greedy heuristic and our probabilistic method. For each problem our code was

Table 1
Problem set and test results

Problem	Integer program variables/ constraints	Best known cover	Size of cover Chvátal's heuristic	Size of cover probabilistic heuristic
A_9	9/12	5 [a]	5	5
A_{15}	15/35	9 [a]	9	9
A_{27}	27/116	18 [a]	19	18
A_{45}	45/330	30 [a]	31	30
A_{81}	81/1080	61	64	61
A_{243}	243/9801	204	208	204

[a] Optimal cover.

Table 2
Test results (probabilistic heuristic)

Problem	α	Size of cover found	Iteration cover found	Time to find cover (secs)
A_9	0.5	5	1	0.0003
	0.6	5	1	0.0003
	0.7	5	1	0.0003
	0.8	5	1	0.0003
	0.9	5	1	0.0003
A_{15}	0.5	9	1	0.0006
	0.6	9	1	0.0006
	0.7	9	1	0.0006
	0.8	9	1	0.0006
	0.9	9	1	0.0005
A_{27}	0.5	18	2	0.0037
	0.6	18	1	0.0018
	0.7	18	2	0.0034
	0.8	18	3	0.0050
	0.9	18	1	0.0017
A_{45}	0.5	30	908	4.734
	0.6	30	250	1.264
	0.7	30	573	2.750
	0.8	30	3096	13.81
	0.9	30	26517	121.2
A_{81}	0.5	61	385	7.668
	0.6	61	1801	34.22
	0.7	61	179	3.208
	0.8	61	274	4.529
	0.9	61	3043	51.82
A_{243}	0.5	205	769	251.6
	0.6	204	148	46.37
	0.7	204	3824	1091.0
	0.8	204	1173	281.3
	0.9	205	657	145.6

run five times, varying α. Table 2 summarizes these results, providing the size of the cover found, the iteration in which it was identified and the total time required.

4. Conclusion

For the first three Steiner triple instances our method immediately constructs the optimal solutions. It is interesting to note that for the third instance, A_{27}, Chvátal's heuristic yields a sub-optimal cover of size 19. As predicted by the literature problem A_{45} is considerably more difficult. For various values of α, we obtain an optimal cover of size 30 in less than 5 seconds of CPU time. The optimal solutions to the last two instances are currently unknown. A cover of size 61 has been obtained for A_{81} by a novel interior point approach for zero–one integer programming [12]. However, A_{243} is conjectured to be beyond today's methods. At present our probabilistic heuristic provides the best known covers for both these problems.

In conclusion, consider the following intuitive explanation of our method's performance. Chvátal's heuristic is analogous to an iterative steepest descent method. However, it possesses two drawbacks in practice. First, it does not guarantee a minimal solution. A constructed cover may possess superfluous elements. Second, this heuristic is deterministic, producing only a single cover. In comparison, our approach always yields minimal solutions and attempts to break free of local optima through randomization. Our method incorporates the power of Chvátal's greedy approach by searching many *good* neighborhoods for their best solutions.

References

[1] D. Avis, "A note on some computationally difficult set covering problems", *Mathematical Programming* **18**, 138–145 (1980).

[2] E. Balas and A. Ho, "Set covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study", *Mathematical Programming Study* **12**, 37–60 (1980).

[3] J.F. Bard and T.A. Feo, "Minimizing the acquisition cost of flexible manufacturing equipment", Technical Report, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, December 1987.

[4] J.F. Bard and T.A. Feo, "Operations sequencing in discrete parts manufacturing", Technical Report, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, August 1987.

[5] V. Chvátal, "A greedy heuristic for the set covering problem", *Mathematics of Operations Research* **4**, 233–235 (1979).

[6] T.A. Feo and J.F. Bard, "A network approach to flight scheduling and maintenance base planning", Technical Report, Operations Research Group, Department of Mechanical Engineering, The University of Texas at Austin, Austin, TX 78712, August 1987.

[7] D.R. Fulkerson, G.L. Nemhauser and L.E. Trotter, Jr., "Two computationally difficult set covering problems that arise in computing the 1-width of incidence matrices of Steiner triple systems", *Mathematical Programming Study* **2**, 72–81 (1974).

[8] M.R. Garey and D.S. Johnson, *Computers and intractability: A guide to the theory of NP-completeness*, Freeman and Company, New York, 1979.

[9] A.M. Geoffrion, "An improved implicit enumeration approach for integer programming", *Operations Research* **17**, 437–454 (1969).

[10] M. Hall, Jr., *Combinatorial theory*, Blaisdell Company, Waltham, MA, 1967.

[11] J.P. Hart and A.W. Shogan, "Semi-greedy heuristics: An empirical study", *Operations Research Letters* **6**, 107–114 (1987).

[12] N. Karmarkar, K.G. Ramakrishnan and M.G.C. Resende, *Private communication*, AT&T Bell Laboratories, Murray Hill, NJ, 1988.

[13] K.G. Ramakrishnan, *Private communication*, AT&T Bell Laboratories, Murray Hill, NJ, 1987.