SIAM J. COMPUT.
Vol. 9, No. 3, August 1980

0097-5397/80/0903-0009 $01.00/0

GENERATING ALL MAXIMAL INDEPENDENT SETS: NP-HARDNESS AND POLYNOMIAL-TIME ALGORITHMS*

E. L. LAWLER†, J. K. LENSTRA‡ AND A. H. G. RINNOOY KAN¶

Abstract. Suppose that an independence system $(E, \mathcal{I})$ is characterized by a subroutine which indicates in unit time whether or not a given subset of E is independent. It is shown that there is no algorithm for generating all the K maximal independent sets of such an independence system in time polynomial in $|E|$ and K, unless $\mathcal{P} = \mathcal{NP}$. However, it is possible to apply ideas of Paull and Unger and of Tsukiyama et al. to obtain polynomial-time algorithms for a number of special cases, e.g. the efficient generation of all maximal feasible solutions to a knapsack problem. The algorithmic techniques bear an interesting relationship with those of Read for the enumeration of graphs and other combinatorial configurations.

Key words. independence system, satisfiability, maximality test, lexicography test, set packing, clique, complete k-partite subgraph, knapsack problem, on-time set of jobs, inequality system, facet generation, matroid intersection

1. Introduction. Let E be a finite set of elements and let $\mathcal{I}$ be a nonempty family of subsets of E satisfying a single axiom: if $I \in \mathcal{I}$ and $I' \subseteq I$, then $I' \in \mathcal{I}$. Under these conditions, $(E, \mathcal{I})$ is said to be an *independence system* and $\mathcal{I}$ is its family of *independent sets*. An independent set I is said to be *maximal* if there is no $I' \in \mathcal{I}$ such that $I' \supset I$. The subsets of E that are not contained in $\mathcal{I}$ are *dependent sets*. A dependent set J is called *minimal* if $J' \in \mathcal{I}$ for each $J' \subset J$.

Suppose that $|E| = n$ and that $(E, \mathcal{I})$ is characterized by a computer subroutine which indicates in unit time whether or not a given subset of E is an independent set. All independent sets can be generated in $O(n|\mathcal{I}|)$ time: given an independent set, $O(n)$ applications of the subroutine suffice to determine the next independent set in a lexicographic listing. But suppose that one is interested only in all the maximal independent sets, of which there are K, $K \leqq |\mathcal{I}|$. These can be found in time polynomial in n and K only in the unlikely event that $\mathcal{P} = \mathcal{NP}$, as we show in § 2.

There are, however, a number of special types of independence systems for which it is possible to generate all the maximal independent sets efficiently. In § 3, an analysis of a procedure due to Paull and Unger [5] reveals that there is a polynomial-time algorithm for this purpose, provided that a certain subproblem can be solved in polynomial time. Improvements in running time and storage requirements suggested by Tsukiyama et al. [8] are discussed as well. In § 4, we investigate some of these independence systems. Typical of these special cases is the problem of generating all the maximal feasible solutions to a knapsack problem. In § 5, we examine the relationship between our approach and a technique for the enumeration of graphs and other combinatorial configurations, recently proposed by Read [6].

2. Complexity. We shall show that the problem of generating all the K maximal independent sets of an arbitrary independence system is *NP*-hard, i.e., if there is an algorithm for the problem which runs in time polynomial in n and K, then there is a polynomial-time algorithm for solving the satisfiability problem [2].

---

* Received by the editors May 16, 1978. This research was partially supported by the National Science Foundation under Grant MC.S 76-17605, and by NATO under Special Research Grant 9.2.02 (SRG. 7).

† Computer Science Division, University of California, Berkeley, California 94720.
‡ Mathematisch Centrum, Amsterdam, The Netherlands.
¶ Erasmus University, Rotterdam, The Netherlands.

Let $F(X_1, \cdots, X_N)$ be a Boolean expression in conjunctive normal form. Let $E = \{T_1, F_1, \cdots, T_N, F_N\}$, and for any $j \in \{1, \cdots, N\}$ and any $J \subseteq E$, define

$$x_j(J) = \begin{cases} \textit{true} & \text{if } T_j \in J, F_j \notin J, \\ \textit{false} & \text{if } F_j \in J, T_j \notin J, \\ \textit{undefined} & \text{otherwise.} \end{cases}$$

Let $I \in \mathcal{I}$ if either

(i) there exists a $j \in \{1, \cdots, N\}$ such that both $T_j \notin I, F_j \notin I$, or

(ii) each clause of F contains a letter X_j or $\bar{X}_j$ whose defined value is *true*, i.e., $F(x_1(I), \cdots, x_N(I)) = \textit{true}$.

It is easily seen that $(E, \mathcal{I})$ is an independence system. Moreover, F is not satisfiable if and only if the only maximal independent sets are $E - \{T_j, F_j\}$ for $j = 1, \cdots, N$.

Assume there exists a general procedure for generating all the maximal independent sets of an arbitrary independence system with running time $\phi(n, K)$, where ϕ is a polynomial function of n and K. Apply this procedure to the independence system defined above and allow it to run for time $\phi(2N, N)$. Then F is satisfiable if and only if either

(i) $F(x_1(I), \cdots, x_N(I)) = \textit{true}$ for some generated I, or

(ii) the procedure fails to halt within the allotted time, establishing that there are more than N maximal independent sets.

For any given $J \subseteq E$, the conjunctive normal form can be evaluated in time proportional to its length. Appropriate modification of the unit-time assumption for independence testing thus establishes that the procedure solves the satisfiability problem in polynomial time. Since the latter problem is *NP*-complete, it can be solved in polynomial time if and only if $\mathcal{P} = \mathcal{NP}$ [2]. Hence, we have the following theorem.

THEOREM 1. *If there exists an algorithm for generating all the maximal independent sets of an arbitrary independence system in time polynomial in n and K, then* $\mathcal{P} = \mathcal{NP}$.

To obtain a reduction to, rather than from, the satisfiability problem, we now consider the problem of generating all maximal independent sets and all minimal dependent sets of an independence system. Let there be L such sets. We shall show that if there is a polynomial-time algorithm for the satisfiability problem, then there is an algorithm for generating all these sets in time polynomial in n and L. Each step of the latter algorithm yields a new set on the list.

Suppose then, that at a certain point sets $I_1, \cdots, I_l$ have been generated. Let $\mathcal{L} \subseteq \{1, \cdots, l\}$ indicate the generated sets which are maximal independent and $\bar{\mathcal{L}} = \{1, \cdots, l\}$-$\mathcal{L}$ those which are minimal dependent. Any new set I must satisfy $I \not\subseteq I_i$ for all $i \in \mathcal{L}$ and $I_i \not\subseteq I$ for all $i \in \bar{\mathcal{L}}$. Form the Boolean expression

$$(\wedge_{i \in \mathcal{L}} \vee_{j \notin I_i} X_j) \wedge (\wedge_{i \in \bar{\mathcal{L}}} \vee_{j \in I_i} \bar{X}_j).$$

The length of this expression is $O(nl)$ and by our assumption one can determine if it is satisfiable in $\psi(nl)$ time, for some polynomial function ψ. If the expression is not satisfiable, then $l = L$ and the algorithm terminates. Otherwise, construct a truth assignment in polynomial time, by successively fixing the value of each variable and determining if the reduced expression is satisfiable. Next define $I = \{j | X_j = \textit{true}\}$ and test I for independence in unit time. If I is independent, augment it until a maximal independent set results; if I is dependent, remove elements until a minimal dependent set is found. Either procedure requires $O(n)$ time. Since clearly $I \neq I_i$ for $i = 1, \cdots, l$, I is the new set on the list. We thus have the following theorem.

THEOREM 2. *If $\mathcal{P} = \mathcal{NP}$, then there exists an algorithm for generating all the maximal independent sets and all the minimal dependent sets of an arbitrary independence system in time polynomial in n and L.*

3. An algorithm.

3.1. A generalized Paull–Unger procedure. We now assume that $E = \{1, \cdots, n\}$ and that independence testing requires time c. Let $\mathcal{I}_j$ be the family of all independent sets that are maximal within $\{1, \cdots, j\}$. By definition, $\mathcal{I}_o = \{\varnothing\}$. We seek to construct $\mathcal{I}_j$ from $\mathcal{I}_{j-1}$ in order to obtain $\mathcal{I}_n$, the family of all K independent sets that are maximal within E.

Suppose that $I \in \mathcal{I}_{j-1}$. If $I \cup \{j\} \in \mathcal{I}$, then clearly $I \cup \{j\} \in \mathcal{I}_j$. If $I \cup \{j\} \notin \mathcal{I}$, then $I \in \mathcal{I}_j$. It follows that

$$|\mathcal{I}_0| \leqq |\mathcal{I}_1| \leqq \cdots \leqq |\mathcal{I}_n| = K.$$

Observing that the elements of E can be numbered arbitrarily, we obtain the following result.

THEOREM 3. *For any $J \subseteq E$, the number of independent sets maximal within J does not exceed K.*

Suppose that $I' \in \mathcal{I}_j$ and $j \in I'$. Since $I' - \{j\}$ is independent and included in $\{1, \cdots, j-1\}$, there must be some $I \in \mathcal{I}_{j-1}$ such that $I' - \{j\} \subseteq I$. Moreover, I' is an independent set that is maximal within $I \cup \{j\}$. This observation suggests the following procedure to obtain $\mathcal{I}_j$ from $\mathcal{I}_{j-1}$, which is a generalization of an algorithm due to Paull and Unger [5].

Step 1. For each $I \in \mathcal{I}_{j-1}$, find all independent sets I' that are maximal within $I \cup \{j\}$.

Step 2. For each such I', test I' for maximality within $\{1, \cdots, j\}$. Each set I' that is maximal within $\{1, \cdots, j\}$ is a member of $\mathcal{I}_j$, and we have seen that each member of $\mathcal{I}_j$ can be found in this way. However, a given $I' \in \mathcal{I}_j$ may be obtained from more than one $I \in \mathcal{I}_{j-1}$. In order to eliminate duplications, we need one further step.

Step 3. Reject each I' that passes the maximality test if it appears among the sets already found to be in $\mathcal{I}_j$. Suppose that in Step 1, for each $I \in \mathcal{I}_{j-1}$, at most K' sets I' are found in time c'; by Theorem 3, we have $K' \leqq K$. For each I', the maximality test in Step 2 requires $O(nc)$ time, and the duplication test in Step 3 can be accomplished with $O(K)$ pairwise set comparisons, each of which requires $O(n)$ time. It follows that, for fixed j, $O(c'K)$ time suffices for the first step, $O(ncKK')$ time for the second step, and $O(nK^2K')$ time for the third step. Thus, the overall running time to obtain $\mathcal{I}_n$ is $O(nc'K + n^2cKK' + n^2K^2K')$. This yields the following theorem.

THEOREM 4. *All the maximal independent sets of an independence system can be generated in time polynomial in n, c and K, if it is possible to list in polynomial time all independent sets that are maximal within $I \cup \{j\}$, for arbitrary $I \in \mathcal{I}_{j-1}$, $j = 1, \cdots, n$.*

In § 4, we investigate several cases in which the subproblem referred to in Theorem 4 (the "$I \cup \{j\}$ problem") can be solved in polynomial time.

3.2. Improvements of Tsukiyama et al. A technique suggested by Tsukiyama et al. [8] enables one to eliminate duplications more efficiently. It yields significant improvements in both running time and storage requirements of the Paull–Unger procedure.

Instead of comparing a set I' with all members of $\mathcal{I}_j$ found previously, one retains I' only if it is obtained from the *lexicographically smallest* $I \in \mathcal{I}_{j-1}$ from which it can be produced. Hence Step 3 is modified in the following way.

Step 3′. For each I' obtained from $I \in \mathscr{I}_{j-1}$ that is maximal within $\{1, \cdots, j\}$, test for each $i < j$, $i \notin I$, the set $(I' - \{j\}) \cup (I \cap \{1, \cdots, i-1\}) \cup \{i\}$ for independence. Reject I' if any of these tests yields an affirmative answer.

If, indeed, any affirmative answer is obtained, then $I' - \{j\}$ is included in an independent set that is lexicographically smaller than I, and hence in a lexicographically smaller maximal independent set from $\mathscr{I}_{j-1}$.

For each I', the lexicography test in Step 3′ requires $O(nc)$ time, which is the same as required by the maximality test in Step 2. Hence, the overall running time of the revised procedure is $O(nc'K + n^2cKK')$.

Possibly of even greater interest for some applications is the fact that storage requirements can be greatly reduced by organizing the computation as a depth-first search of a tree. Nodes at level j correspond to members of $\mathscr{I}_j$, with the tree rooted at $\varnothing$, the unique member of $\mathscr{I}_0$. Since for each $I \in \mathscr{I}_{j-1}$, either $I \cup \{j\} \in \mathscr{I}_j$ or $I \in \mathscr{I}_j$, each node has at least one and at most K' children. Whenever in the depth-first search a member of $\mathscr{I}_n$ is encountered, it is outputted. The maximum number of subproblems that must be maintained in stack to allow backtracking is $O(nK')$. A further decrease in storage requirements can be obtained at the expense of an increase in running time.

4. Applications. In this section we investigate various independence systems for which all maximal independent sets can be generated in polynomial time.

4.1. Set packing. Let S be a finite set with $|S| = m$ and let $\mathscr{S} = \{S_1, \cdots, S_n\}$ be a family of (not necessarily distinct) subsets of S. A subfamily $I \subseteq \mathscr{S}$ is a *packing* in S if the sets in I are pairwise disjoint. The packings correspond to the independent sets of an independence system with $E = \mathscr{S}$. All maximal packings can be generated in polynomial time, as shown below.

First consider the "$I \cup \{j\}$ problem". Let $A_j \subseteq \mathscr{S}$ consists of the sets S_i for which $S_i \cap S_j \neq \varnothing$. Given $I \in \mathscr{I}_{j-1}$, the only sets which can possibly be maximal within $I \cup \{S_j\}$ are I itself and $(I - A_j) \cup \{S_j\}$. Thus $K' \leqq 2$. It follows that, given A_j, the $I \cup \{j\}$ problem can be solved in $O(n)$ time.

Assuming the sets S_i are specified by ordered lists of indices, one can find the sets $A_1, \cdots, A_n$ in $O(mn^2)$ time. It follows that Step 1 requires $O(mn^2 + n^2K)$ time.

The maximality test for I' is equivalent to verifying that $I' \cap A_i \neq \varnothing$ for all $i < j$, $S_i \notin I$. Since each such test can be carried out in $O(n^2)$ time, Step 2 requires $O(n^3K)$ time.

The lexicography test is easily seen to be equivalent to verifying that $[I - (A_j \cap \{S_{i+1}, \cdots, S_{j-1}\})] \cap A_i \neq \varnothing$ for all $i < j$, $S_i \notin I$. Thus, Step 3′ requires $O(n^3K)$ time as well.

It follows that the overall running time of the procedure is $O(mn^2 + n^3K)$. Since it is possible to implement the search tree in $O(n)$ space, $O(mn)$ space is sufficient overall.

Suppose $\mathscr{S}$ is induced by an undirected m-edge n-vertex graph G with edge set S. S_j denotes the set of edges incident to vertex j and A_j denotes the set of vertices adjacent to vertex j. Then each packing $I \subseteq \mathscr{S}$ is an *independent* or *stable set* of vertices of G, or, equivalently, a *clique* of the complementary graph $\bar{G}$. It was in this context that the Paull–Unger procedure and the improvements of Tsukiyama et al. were originally proposed.

For the graph problem, it is natural for the sets A_j to be given as input in the form of ordered lists. Under this assumption, and noting that $\sum_{j=1}^{n} |A_j| = 2m$, one can reduce the time bound to $O(mnK)$ and the space bound to $O(m+n)$, as shown in [8].

4.2. Complete k-partite subgraphs. Let G be an undirected graph with vertex set $V = \{v_1, \cdots, v_n\}$ and edge set S with $|S| = m$. A *complete k-partite subgraph* of G is

defined by a collection $\{V_1, \cdots, V_k\}$ of pairwise disjoint subsets of V such that $\{v_i, v_j\} \in S$ for $v_i \in V_g$, $v_j \in V_h$, if and only if $g \neq h$. Note that an independent set of vertices defines a complete 1-partite subgraph and that a complete k'-partite subgraph is also a complete k-partite subgraph for $k = k'+1, \cdots, n$.

The complete k-partite subgraphs of G correspond to the independent sets of the following independence system. Let $E = V$ and let $I \in \mathcal{I}$ if there exists a partition $P(I) = \{V_1, \cdots, V_k\}$ of I (i.e., $\bigcup_{h=1}^{k} V_h = I$ and $V_g \cap V_h = \emptyset$ for $1 \leqq g < h \leqq k$) that defines a complete k-partite graph on I. We will show how to generate all maximal complete k-partite subgraphs of G in polynomial time.

Again consider the "$I \cup \{j\}$ problem". Let $P(I) = \{V_1, \cdots, V_{k'}\}$ with $V_h \neq \emptyset$ $(h = 1, \cdots, k')$ and $k' \leqq k$.

First, suppose that $\{v_i, v_j\} \in S$ for all $v_i \in I$. If $k' < k$, then the single independent set I' that is maximal within $I \cup \{v_j\}$ is $I \cup \{v_j\}$ itself, with $P(I \cup \{v_j\}) = P(I) \cup \{v_j\}$. If $k' = k$, then there are $k+1$ sets I', for which $P(I')$ is obtained by deleting any one of the members of $P(I) \cup \{v_j\}$.

Suppose now that $\{v_i, v_j\} \in S$ only for all $v_i \in V_h' \subseteq V_h$ $(h = 1, \cdots, k')$, where $V_h' = \emptyset$ for $h = 1, \cdots, a$, $\emptyset \subset V_h' \subset V_h$ for $h = a+1, \cdots, b$ and $V_h' = V_h$ for $h = b+1, \cdots, k'$, with $0 \leqq a \leqq b \leqq k'$ and $b > 0$. In this case, $b+1$ independent sets I' that are maximal within $I \cup \{v_j\}$ are defined by $P(I') = P(I)$ and $P(I') = \{V_1', \cdots, V_{h-1}', (V_h - V_h') \cup \{v_j\}, V_{h+1}', \cdots, V_b', V_{b+1}, \cdots, V_{k'}\}$ for $h = 1, \cdots, b$. In the special case that $a = 0$, even more sets I' may exist. If $k' < k$, then the single additional set I' is defined by $P(I') = \{V_1', \cdots, V_b' V_{b+1}, \cdots, V_{k'}, \{v_j\}\}$. If $k' = k$, then there are $k - b$ additional set I', for which $P(I')$ is obtained by deleting any one of the sets $V_{b+1}, \cdots, V_{k'}$ from $\{V_1', \cdots, V_b', V_{b+1}, \cdots, V_{k'}, \{v_j\}\}$. (Note that these sets are not maximal in the case that $a > 0$.)

Since $K' = O(k)$ and independence testing requires $O(m)$ time, the overall running time of the procedure is $O(n^2 mkK)$.

4.3. Knapsack problems. Next consider the *knapsack inequality* $\sum_{j=1}^{n} a_j x_j \leqq b$, $x_j \in \{0, 1\}$ $(j = 1, \cdots, n)$, where $a_1 \geqq a_2 \geqq \cdots \geqq a_n > 0$. The feasible solutions to this inequality correspond in a natural way to the independent sets of an independence system with $E = \{1, \cdots, n\}$ and $I \in \mathcal{I}$ if $\sum_{j \in I} a_j \leqq b$. We are interested in generating all maximal feasible solutions.

Consider the $I \cup \{j\}$ problem and assume that $I \cup \{j\} \notin \mathcal{I}_j$. Feasibility is restored by removing any element h from $I \cup \{j\}$. Thus $K' \leqq j$, and the $I \cup \{j\}$ problem can be solved in $O(n)$ time.

For a given $I \in \mathcal{I}_{j-1}$, define $m(h) = \max\{i \mid i < h, i \notin I\}$; let $a_{\max\emptyset} = \infty$. A set $I' = (I - \{h\}) \cup \{j\}$ $(h \in I)$ passes the maximality test if and only if $\sum_{i \in I'} a_i + a_{m(j)} > b$, and it passes the lexicography test if and only if $\sum_{i \in I} a_i - a_h + a_{m(h)} > b$. Moreover, for *all* I' arising from $I \cup \{j\}$, these tests can be carried out in $O(n)$ time altogether. It follows that the overall running time of the procedure is $O(n^2 K)$.

The *unbounded* knapsack inequality, in which the x_j are allowed to take on any nonnegative integer value, is reducible to the 0-1 case by introducing $2a_j, 4a_j, \cdots, 2^k a_j$ into the problem in addition to a_j, where k is the smallest integer such that $2^{k+1} a_j > b$. Then E contains $O(n \log b)$ elements, and the algorithm is still strictly polynomial.

4.4. On-time sets of jobs. Suppose there are n *jobs* to be processed, one at a time, by a single *machine* starting at time 0. Job j requires an uninterrupted *processing time* of p_j units and has a *deadline* d_j. Let $E = \{1, \cdots, n\}$ and let $I \in \mathcal{I}$ if all the jobs in I can be scheduled for completion by their deadlines. It is well known that such a schedule exists if and only if the jobs in I are all completed on time when sequenced in order of nondecreasing deadlines. Hereafter, assume $d_1 \leqq d_2 \leqq \cdots \leqq d_n$.

Again consider the $I \cup \{j\}$ problem and assume that $I \cup \{j\} \notin \mathcal{I}_j$. In this case, we have $\sum_{i \in I} p_i + p_j > d_j$. Independence is restored by removing job j from $I \cup \{j\}$ or by removing some jobs from I such that job j, which can be assumed to remain in the last position, is completed on time. It follows that solving the $I \cup \{j\}$ problem is equivalent to finding all maximal subsets $H \subseteq I$ such that $\sum_{i \in H} p_i \leqq d_j - p_j$, which can be accomplished by applying the knapsack procedure of § 4.3. By Theorem 3, the number of maximal subsets H does not exceed $K-1$. Hence the $I \cup \{j\}$ problem can be solved in $O(n^2K)$ time.

Since maximality and lexicography tests require $O(n)$ time, it follows that the overall running time of the procedure is $O(n^3K^2)$.

4.5. Inequality systems. The problems considered in §§ 4.1, 4.3 and 4.4 can all be viewed as special instances of the general problem of finding all maximal feasible solutions to an *inequality system* of the form $Ax \leqq b$, $x_j \in \{0, 1\} (j = 1, \cdots, n)$, where the $m \times n$-matrix $A = (a_{ij})$ and the m-vector $b = (b_i)$ have nonnegative components.

For example, given a set $S = \{1, \cdots, m\}$ and a family $\mathcal{S} = \{S_1, \cdots, S_n\}$ of subsets of S, define $a_{ij} = 1$ if $i \in S_j$, $a_{ij} = 0$ otherwise. In the case that $b_i = 1$ $(i = 1, \cdots, m)$, the maximal feasible solutions correspond to the maximal packings in S; they can be generated in polynomial time, as has been shown in § 4.1. In the case that $b_i = \sum_{j=1}^{n} a_{ij} - 1$ $(i = 1, \cdots, m)$, the maximal feasible solutions correspond to the complements of the minimal coverings of S. We have not been able to devise a polynomial-time algorithm for this problem. Nor have we been able to obtain an *NP*-hardness result similar to Theorem 1 for this case or even for a general inequality system, although we conjecture that no polynomial-time algorithm exists unless $\mathcal{P} = \mathcal{NP}$.

For the scheduling problem discussed in § 4.4, we have $m = n$, $a_{ij} = p_j$ if $i \geqq j$, $a_{ij} = 0$ otherwise, and $b_i = d_i$ (cf. [4]). The same technique as above can be applied to a slightly wider class of inequality systems, where b is an m-vector with nondecreasing components and A is a nonnegative $m \times n$-matrix such that

(i) $a_{ij} > 0$ implies $a_{ij'} > 0$ for all $j' < j$, and

(ii) the strictly positive entries in each column are nonincreasing.

In this case, the $I \cup \{j\}$ problem with $I \cup \{j\} \notin \mathcal{I}_j$ can be solved by applying the knapsack procedure of § 4.3 to the constraint of smallest index h such that $a_{hj} > 0$. Any maximal subset of $I \cup \{j\}$ that satisfies constraint h will then satisfy the remaining constraints as well.

The reader may be able to construct other examples in which a certain property of A permits one to restrict attention to a single constraint when independence has to be restored. In each such case, the knapsack procedure can be applied to solve the $I \cup \{j\}$ problem in polynomial time.

4.6. Facet generation. Consider the convex hull P of all 0-1 vectors x satisfying the general inequality system $Ax \geqq b$, where $A \geqq 0$. Balas and Zemel [1] have established a correspondence between the *facets* of P and the *minimal covers* of A, i.e. the minimal feasible solutions to $Ax \geqq b$. Such covers are in one-one correspondence to the maximal feasible solutions to $Ax' \ngeq b'$, where $b_i' = \sum_{j=1}^{n} a_{ij} - b_i - 1$ $(i = 1, \cdots, m)$, under the assumption that all data are integers.

Thus, in order to generate the facets of P, it suffices to generate the K maximal feasible solutions to $Ax' \ngeq b'$. This inequality system can be considered as the *disjunction of m* knapsack inequalities $\sum_{j=1}^{n} a_{ij} x_j' \leqq b_i'$ $(i = 1, \cdots, m)$, the ith such inequality having K_i maximal feasible solutions. In the case that $m = 1$, the procedure of § 4.3 can be applied to yield all minimal covers in polynomial time. In the general case, the

following procedure may have some practical value, even though it is not polynomial in K.

A maximal feasible solution to the entire system has to be feasible and maximal with respect to at least one of the separate inequalities. The procedure of § 4.3 is now applied to each of these inequalities in turn. However, a maximal feasible solution to inequality i is accepted as a maximal feasible solution to $Ax' \not> b'$ only if it is

(i) infeasible for each of the inequalities $1, \cdots, i-1$, and

(ii) infeasible or maximal feasible for each of the inequalities $i+1, \cdots, m$.

It is not hard to see that this procedure generates all minimal covers without duplication.

For inequality i, application of the knapsack procedure requires $O(n^2 K_i)$ time, and conditions (i) and (ii) can be checked in $O(mn)$ time for any candidate solution, or in $O(mnK_i)$ time altogether. It follows that the overall running time of the procedure is $O((mn+n^2)\sum K_i)$. Unfortunately, there exist inequality systems for which $\sum K_i$ is exponentially related to K. For example, in the simple case that $m = n-1$, $a_{ij} = 1$, $b_i' = i$ $(i = 1, \cdots, m, j = 1, \cdots, n)$, we have $K_i = \binom{n}{i}$ $(i = 1, \cdots, m)$, $\sum K_i = 2^n - 1$, and $K = n$.

For some special cases, truly polynomial-time algorithms can still be obtained. For example, suppose A is such that the entries in each row are monotone nonincreasing. If $I \cup \{j\} \notin \mathcal{I}$, then removal of any element from $I \cup \{j\}$ restores feasibility, so that $K' \leqq n$.

In analogy to the above approach, one might view a general inequality system $Ax \leqq b$ as the *conjunction* of m knapsack inequalities. In this case, however, a maximal feasible solution to the entire system can be feasible but nonmaximal with respect to each of the separate inequalities. It seems hard to make any significant progress beyond the special cases discussed in § 4.5.

4.7. Matroid intersections. A matroid $M = (E, \mathcal{I})$ is an independence system such that for all $J \subseteq E$, all independent sets maximal within J have the same cardinality [3]. Given m matroids $M_i = (E, \mathcal{I}_i)$ $(i = 1, \cdots, m)$ with $E = \{1, \cdots, n\}$, their *intersection* $(E, \mathcal{I})$ is an independence system defined by $\mathcal{I} = \bigcap_{i=1}^{m} \mathcal{I}_i$. We are interested in generating all maximal independent sets in $(E, \mathcal{I})$, assuming that independence testing in M_i requires time c_i $(i = 1, \cdots, m)$.

Consider the $I \cup \{j\}$ problem. If $I \cup \{j\} \notin \mathcal{I}_j$, then addition of j must have destroyed independence in some of the m matroids, say, in $M_1, \cdots, M_l$. Each of these matroids M_i contains a unique minimal dependent set or *circuit* C_i, and independence in M_i is restored by removing any one element from C_i.

It follows that, in order to solve the $I \cup \{j\}$ problem, it is necessary to find all minimal subsets of $\bigcup_{i=1}^{l} C_i$ that contain at least one element from each circuit, i.e., all minimal coverings of $(C_1, \cdots, C_l)$. In view of our remark in § 4.5, we settle for a brute force approach: consider all n^m possible solutions. This yields an overall running time of $O(n^{m+2} K \sum c_i)$, which is, at least, polynomial for fixed m.

For certain special cases, e.g. the generation of all spanning trees [7], the special structure of the system can be exploited and significant improvements made.

5. An enumeration procedure of Read. We conclude by noting a relationship between our techniques and those proposed by Read [6] for the enumeration of graphs, digraphs, and other combinatorial configurations. We restate the essential features of Read's procedure in our terms, as follows.

The family $\mathcal{I}_j$ is to be obtained from the family $\mathcal{I}_{j-1}$ by applying an *augmentation operation* to each set in $\mathcal{I}_{j-1}$. These sets are processed in a *canonical linear order* "$<$" and the augmentation routine produces sets I' from each $I \in \mathcal{I}_{j-1}$ in this same order. For

each $I' \in \mathcal{I}_j$, let $f(I')$ denote the first set in $\mathcal{I}_{j-1}$ which produces I' when subjected to the augmentation operation. Suppose that the canonical order is *weakly monotonic* in the sense that for all I', $I'' \in \mathcal{I}_j$, $I' < I''$ implies $f(I') \leqq f(I'')$. Then it is simple to avoid duplications: when applying the augmentation operation, retain the next set produced only if it follows the member of $\mathcal{I}_j$ that has been obtained lastly.

Consider, for example, how this procedure is applied by Read to generate all the nonisomorphic digraphs on five vertices. The nondiagonal elements of the adjacency matrix are written as a string of 20 bits, which can be interpreted as a binary integer. A *canonical* digraph is one which has the largest such integer of all digraphs in its isomorphism class, and this integer is its *code*. Let $\mathcal{I}_{j-1}$ be the family of all canonical digraphs with $j-1$ arcs; their codes specify the canonical linear order. For each $I \in \mathcal{I}_{j-1}$, the augmentation operation produces digraphs I' with j arcs by systematically changing a 0 to a 1 in the 20-bit representation of I. Each such I' is tested for canonicity. Each I' that passes the canonicity test is added to the list $\mathcal{I}_j$ if and only if its code is strictly greater than that of the most recently obtained member of $\mathcal{I}_j$. It can be shown that the property of weak monotonicity is satisfied. Thus, all canonical digraphs with j arcs are generated in this way, without duplication.

We have been unable to devise a weakly monotonic ordering for the problems considered in this paper. The lexicography test of Tsukiyama et al. is, in effect, an alternative to Read's technique for eliminating duplications and amounts to an analysis of the *inverse* of the augmentation operation. That is, when I' is obtained from $I \in \mathcal{I}_{j-1}$, I' is retained only if $f(I') = I$, where $f(I')$ denotes the lexicographically smallest set in $\mathcal{I}_{j-1}$ which produces I' when subjected to the augmentation operation.

REFERENCES

[1] E. BALAS AND E. ZEMEL, *All the facets of zero-one programming polytopes with positive coefficients*, Management Sciences Research Report 374, Carnegie–Mellon University, Pittsburgh, 1975.

[2] S. A. COOK, *The complexity of theorem-proving procedures*, Proc. 3rd Annual ACM Symp. Theory Comput., (1971), pp. 151–158.

[3] E. L. LAWLER, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York, 1976.

[4] E. L. LAWLER AND J. M. MOORE, *A functional equation and its application to resource allocation and sequencing problems*, Management Sci., 16 (1969), pp. 77–84.

[5] M. C. PAULL AND S. H. UNGER, *Minimizing the number of states in incompletely specified sequential switching functions*, IRE Trans. Electron. Comput., EC-8 (1959), 356–367.

[6] R. C. READ, *Every one a winner, or how to avoid isomorphism search when cataloguing combinatorial configurations*, Ann. Discrete Math, 2 (1978), pp. 107–120.

[7] R. C. READ AND R. E. TARJAN, *Bounds on backtrack algorithms for listing cycles, paths, and spanning trees*, Networks, 5 (1975), pp. 237–252.

[8] S. TSUKIYAMA, M. IDE, M. ARIYOSHI AND I. SHIRAWAKA, *A new algorithm for generating all the maximal independent sets*, this Journal, 6 (1977), pp. 505–517.