SCHOOL OF OPERATIONS RESEARCH
AND INDUSTRIAL ENGINEERING
COLLEGE OF ENGINEERING
CORNELL UNIVERSITY
ITHACA, NEW YORK 14853-3801

TECHNICAL REPORT NO. 1008

December 1991

CARDINALITY-RESTRICTED CHAINS AND ANTICHAINS IN PARTIALLY ORDERED SETS[1]

by

Henry Shum[2] and L.E. Trotter, Jr.[3]

[1]Research supported by NSF grants ECS-8504077, DMS-8920550 and INT-9024300.

[2]Operations Analysis and Engineering Department, International Paper Company, 11Skyline Drive, Hawthorne, NY 10532, U.S.A.

[3]School of OR&IE, Cornell University, Ithaca, NY 14853, U.S.A. and Department de Mathematiques, E.P.F.L., 1015 Lausanne-Ecublens, Switzerland.

CARDINALITY-RESTRICTED CHAINS AND ANTICHAINS IN PARTIALLY ORDERED SETS [1]

Henry Shum[2] and L.E. Trotter, Jr.[3]

December 1991

Abstract

For a given poset and constant κ, four problems are considered. *Covering:* Determine a minimum cardinality cover of the poset elements by chains(antichains), each of length(width) at most κ. *Optimization:* Given also weights on the poset elements, find a chain (antichain) of maximum total weight among those of length(width) at most κ. It is shown that the chain covering problem is NP-complete, while chain optimization is polynomial-time solvable. Several classes of facets are derived for the polytope generated by incidence vectors of antichains of width at most κ. These facets provide the basis for a polyhedral combinatorial algorithm for the antichain optimization problem. Computational results are given for the algorithm on randomly generated posets with up to 1005 nodes and $4 \leq \kappa \leq 30$.

[1]Research supported by N.S.F. grants ECS-8504077, DMS-8920550 and INT-9024300.

[2]Operations Analysis and Engineering Department, International Paper Company, 11 Skyline Drive, Hawthorne, NY 10532, U.S.A.

[3]School of OR&IE, Cornell University, Ithaca, NY 14853, U.S.A. and Departement de Mathematiques, E.P.F.L., 1015 Lausanne-Ecublens, Switzerland.

1. Introduction

A classical result of Dilworth states that in any partially ordered set (poset) the minimum cardinality of a partition into chains is equal to the maximum width of an antichain. Here we investigate the following two related, but more general, problems, where κ is a fixed positive integer:

(1) determine a minimum cardinality partition of a partially ordered set into chains, each of length no greater than κ;

(2) determine a minimum cardinality partition of a partially ordered set into antichains, each of width no greater than κ.

We shall refer to problem 1 as the **M**inimum **P**artition into **L**imited–length **C**hains (MPLC) Problem, and to problem 2 as the **M**inimum **P**artition into **L**imited–width **A**ntichains (MPLA) Problem.

By associating to each element in the partially ordered set a positive integral *weight*, two closely related combinatorial problems arise:

(3) determine a chain of maximum total weight among all chains of length at most κ;

(4) determine an antichain of maximum total weight among all antichains of width at most κ.

Problems 3 and 4 will be referred to as the **M**aximum **W**eight **L**imited–length **C**hain (MWLC) Problem and the **M**aximum **W**eight **L**imited–width **A**ntichain (MWLA) Problem, respectively.

The problem MPLA is also motivated by the problem of scheduling unit–execution time, precedence–related jobs on κ identical machines, each capable of executing any of the jobs. A classical scheduling problem is to find an assignment of the jobs onto the machines so that all work is completed at the earliest possible time without violating the precedence requirements. Note that in any feasible schedule (i.e., one respecting

precedence), the jobs processed at any time instant form an antichain of cardinality at most κ in the partial order stipulated by the precedence relations. Hence any feasible schedule partitions the jobs into antichains, each of cardinality at most κ, and the minimum cardinality of such a decomposition provides a lower bound for the completion time of any scheduling.

The remainder of this paper is organized as follows. We show in Section 2 that MPLC is NP–hard and give simple dynamic programming recursions which provide polynomial–time algorithms for MWLC. The complexity status of MPLA and MWLA remains unresolved. In Section 3, we use an integer programming formulation for MWLA as the basis for a polyhedral combinatorial approach to this problem. For a given partially ordered set and width parameter κ, we direct attention to the polytope whose extremal elements are incidence vectors of antichains of width at most κ. We derive several classes of facets for such polytopes. (Note that MWLA is linear–objective optimization over such polytopes.) Implementational details of a strong cutting plane algorithm for MWLA are given in Section 4. Computational results are provided for random problem instances with up to 1005 nodes for values of κ in the range [4, 30].

2. Computational Complexity of MPLC and MWLC

We first consider MPLC in the following decision form: Given positive integers p, κ and a poset S, determine whether S can be decomposed into no more than p chains, each with at most κ elements. We show that this decision problem is NP–complete through a polynomial–time reduction from the NP–complete problem Exact Cover by 3–sets (X3C), defined as follows: Given a set S' containing 3p' elements, where p' is a positive

integer, and a collection $C = \{c_1, c_2, \ldots, c_{q'}\}$ of 3–element subsets of S', where q' is a positive integer, determine whether S' can be partitioned using only subsets contained in C. The reader is referred to Garey and Johnson [1979] for background material on computational complexity.

Theorem 2.1 MPLC is NP–complete.

Proof: Clearly MPLC $\in$ NP, as any partition of S into p chains, each of length at most κ, can be easily validated in polynomial-time.

Let S' with $|S'| = 3p'$ and the collection $C = \{c_1, c_2, \ldots, c_{q'}\}$ of 3-element subsets of S' be an instance of X3C. We now define S, p and κ, such that there exists an exact cover of S' by the 3–sets from C if and only if there exists a partition of the poset S into at most p chains, each of size no greater than κ. The poset S will be described as digraph G = (V, E) with vertices corresponding to poset elements and the directed edge $(i, j) \in E$ if and only if i is related to $j \neq i$ in the partial order, i.e., when i *dominates* j. For simplicity, edges implied by transitivity are omitted. To $c_i \in C$, say $c_i = \{x_i, y_i, z_i\}$, we associate a subgraph of 12 nodes and 11 edges in the graph G; this subgraph is denoted (V_i, E_i) -- see Figure 2.1. We then define $V = V_1 \cup V_2 \cup \ldots \cup V_{q'}$ and $E = E_1 \cup E_2 \cup \ldots \cup E_{q'}$, so that $|V| = 3p' + 9q' = 3(p' + 3q')$.

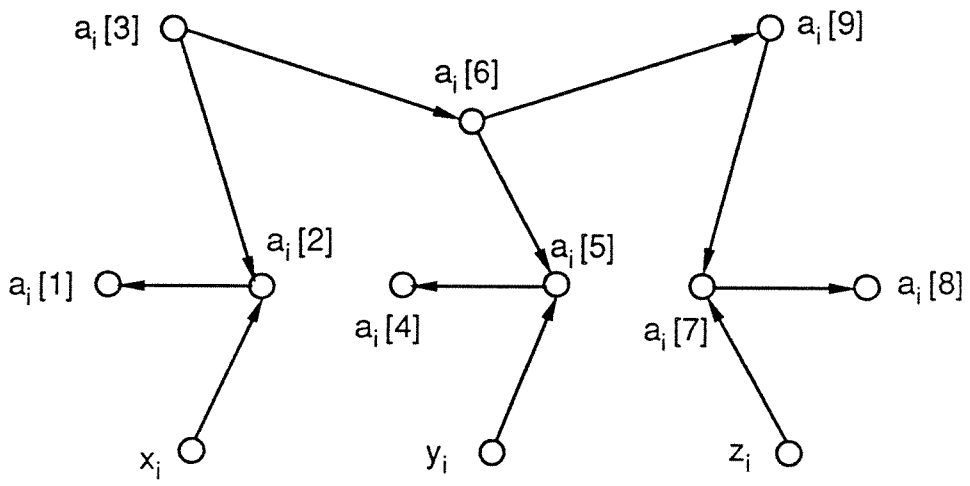


Figure 2.1: The subgraph (V_i, E_i) for $c_i = \{x_i, y_i, z_i\}$.

Finally, we take $S = V$, $\kappa = 3$ and $p = (p' + 3q')$ as an instance of MPLC. It is easily observed that if a feasible partition for the chain problem exists, then the partially ordered set must be decomposed into exactly p chains, each having exactly 3 elements. Moreover, each of these chains can only consist of elements within a single configuration as in Figure 2.1. These two conditions together imply that each subgraph corresponding to a c_i must be partitioned into chains of size three in one of the two ways illustrated in Figures 2.2 and 2.3. Note that only the nodes denoted by x_i, y_i and z_i can be shared between two different configurations, and any chain must be contained within a single configuration.

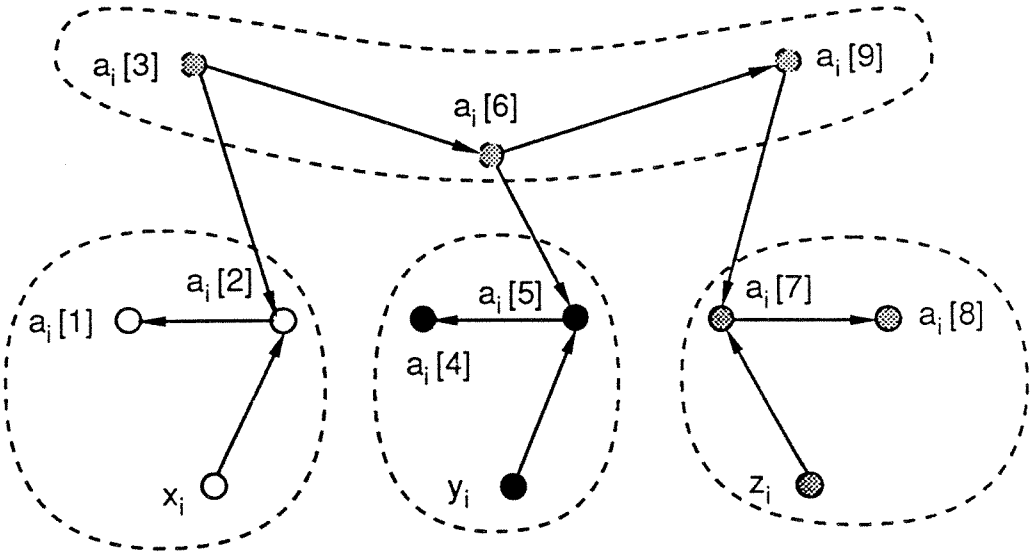


Figure 2.2 : One partition of c_i into 3–chains.

Now, if there exists an exact cover for X3C, then a chain partition can be derived in the following manner. If c_i is a subset of S' included in the exact cover, the chain partition for the configuration in G corresponding to c_i will be (see Figure 2.2) $\{a_i[3],a_i[6],a_i[9]\}$, $\{x_i,a_i[2],a_i[1]\}$, $\{y_i,a_i[5],a_i[4]\}$, $\{z_i,a_i[7],a_i[8]\}$; otherwise the partition will be (Figure 2.3) $\{a_i[3],a_i[2],a_i[1]\}$, $\{a_i[6],a_i[5],a_i[4]\}$, $\{a_i[9],a_i[7],a_i[8]\}$. Thus V is partitioned into $p = 4p' + 3(q' - p') = p' + 3q'$

chains, each having exactly 3 elements.

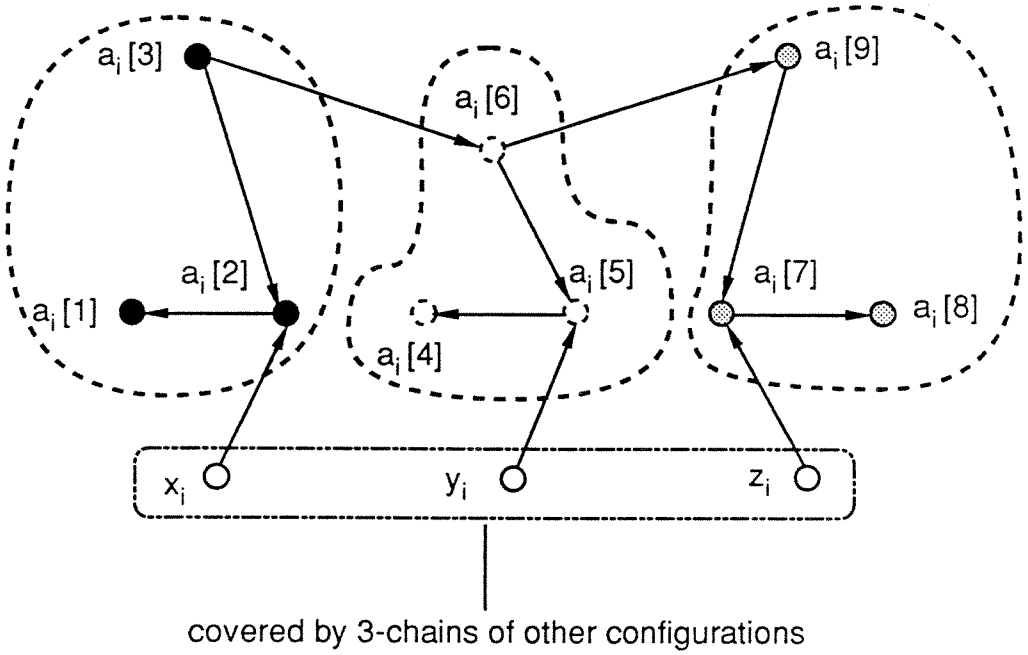


Figure 2.3 : A second partition of c_i into 3–chains.

Conversely, suppose there is a partition of V into p 3–chains. Then c_i is included in the exact cover for S' if its corresponding configuration decomposes as $\{a_i[3],a_i[6],a_i[9]\}$, $\{x_i,a_i[2],a_i[1]\}$, $\{y_i,a_i[5],a_i[4]\}$, $\{z_i,a_i[7],a_i[8]\}$; otherwise c_i is not in the exact cover. ■

MPLC is polynomial–time solvable if $\kappa \leq 2$ or if $\kappa \geq \omega$, where ω denotes the size of a longest chain in S. When $\kappa \geq \omega$, Dilworth's Theorem applies, and MPLC can be solved via a transformation to the max–flow problem (see Ford and Fulkerson [1962]), for which efficient algorithms are known. When $\kappa = 1$, the problem is trivial. For the case $\kappa = 2$, we recall that a *matching* of a graph is a set of edges with distinct endpoints. A matching M^* is a *maximum matching* provided $|M^*| \geq |M|$ for every matching M in the graph. In a matching in the directed acyclic graph representing the poset S, the endpoints of each edge in the matching form a 2–chain; hence, each

matching M gives rise to a partition P of S into $|M|$ 2–chains and $(|S| - 2|M|)$ 1–chains. Note $|P| = |M| + (|S| - 2|M|)$, so that $|P| + |M| = |S|$. Similarly, in a partition P of the poset S into chains of length no greater than 2, the 2–chains in the partition induce a matching M of size equal the the number of 2–chains, and again $|P| + |M| = |S|$. Since $|P| + |M| = |S|$, it is clear that minimizing $|P|$ is equivalent to maximizing $|M|$. Thus one may solve MPLC when $\kappa = 2$ by solving the corresponding maximum matching problem. Polynomial–time algorithms for the maximum matching problem are well–known (see Lawler [1976]).

From the proof of Theorem 2.1, it is clear that MPLC is NP–complete even for κ fixed at the value 3. Furthermore, since $\kappa \leq \omega \leq |S|$ for any nontrivial instance, the problem is *strongly NP–complete*, even for $\kappa = 3$.

The Bellman–Ford shortest path algorithm (see Lawler [1976]) can be used to solve MWLC in polynomial–time. Suppose we are given a poset represented by the acyclic digraph $G = (V,E)$. We create a new graph $G' = (V', E')$ by adding an artificial (source) node v_0 and $|V| = n$ artificial edges. Let $V' = V \cup \{v_0\}$, and $E' = E_1 \cup E_2$, where $E_1 = \{(v_0, v) : v \in V\}$. For $v \in V$ and $(u, v) \in E'$, we define the *length* of (u, v) as $-w(v)$, where $w(v)$ denotes the (nonnegative) weight of node v in G. Note that G' also contains no directed cycles. We can thus apply the Bellman–Ford algorithm to determine a shortest path from v_0 to v, for each $v \in V$, limiting the number of intermediate edges to κ. The shortest among these paths determines a maximum weight chain of size no greater than κ. A straightforward analysis shows that this algorithm for MWLC runs in $O(n^2\kappa)$ time.

Note that the above procedure works with the entire acyclic digraph corresponding to the poset. We now indicate an alternative algorithm, also with an $O(n^2\kappa)$ worst–case time–bound, but which is more efficient in that it requires only the *Hasse diagram* of the poset as input; i.e., we now denote

H = (V, F) as the subgraph of G obtained by omitting those edges implied by transitivity. For $v \in V$, we define $P_v = \{v\} \cup \{ u \in V: (u, v) \in E\}$. When $P_v = \{v\}$, v is termed a *source node*. Finally, for $v \in V$, denote $W_v[m]$ as the maximum weight of a chain having at most m elements, each belonging to P_v. (Note that we do not stipulate that v itself be in the chain.)

A maximum weight chain in P_v with at most m elements either contains v or does not. If it does contain v, its weight is w(v) plus that of a chain of maximum weight having at most m–1 elements in some P_u, for $(u, v) \in F$; otherwise, its weight is equal to that of a maximum weight chain having at most m elements in some P_u, where again $(u, v) \in F$. This observation leads to the following dynamic programming recursion for MWLC. We assume that the nodes in G are indexed 1, 2,..., |V| so that $(i, j) \in F$ implies $i < j$. For $m=1,...,\kappa$, initialize $W_j[m] = w(j)$, if j is a source node; else $W_j[m] = 0$. Then for each non-source node j, in order of increasing index, compute:

$$W_j[1] = \max \{ \max_{i:(i,j)\in F} W_i[1] , w(j) \} ,$$

$$W_j[m] = \max \{ \max_{i:(i,j)\in F} W_i[m] , \max_{i:(i,j)\in F} \{ W_i[m-1] + w(j) \} \} , \quad m > 1 .$$

Upon termination, the optimum value for MWLC is $\max\{ W_v[\kappa] : v \in V \}$.

3. Antichain Polytopes

As shown in the previous section, the covering and weighted optimization problems for limited-length chains are NP–complete and polynomial–time solvable, respectively. With no cardinality limitation, the corresponding antichain problems are polynomial-time solvable. A minimum cover by antichains is determined simply by recursive removal of the set of minimal elements (source nodes) in the poset (see Fulkerson [1972]). On the other hand, it is well–known that an antichain of maximum weight can be

determined via transformation to a bipartite max–flow problem (see Cameron and Edmonds [1979], Shum [1989]). Throughout the present section, we assume we are given a poset specified by an acyclic digraph $G = (V, E)$ with $V = \{1, 2,...,n\}$, with antichain width limit κ and weight $c_j \geq 0$ associated with $j \in V$. When G is bipartite, MPLA can be solved in polynomial-time (see Hansen, Hertz and Kuplinsky [1990]) and when the complement of G is also the graph of a partial order, the recursion of Section 2 solves MWLA. But, in general, the computational complexity of MPLA and MWLA remains unresolved.

We now develop polyhedral theory related to MWLA which is used in the following section in a linear programming–based algorithm for this problem, i.e., a polyhedral combinatorial algorithm. We consider the following integer programming formulation of MWLA:

$$\begin{aligned} & \max \ c \cdot x \\ \text{subject to} \quad & \sum_{j \in C} x_j \leq 1 \quad \text{for each maximal chain } C \\ & \sum_{j=1}^{n} x_j \leq \kappa \qquad\qquad (IP_{AC}) \\ & x_j = 0 \text{ or } 1, \qquad j \in V. \end{aligned}$$

We denote by P the convex hull of feasible solutions for (IP_{AC}). It is clear that the extreme points of P are precisely the incidence vectors of antichains of size at most κ in the poset. We thus refer to P as the *antichain polytope*. P is full–dimensional and hence uniquely determined (up to positive scalar multiplication) by its facet inequalities (see Schrijver [1986]). We now give an explicit description of several classes of facets for P.

First note that each nonnegativity stipulation $x_j \geq 0$ is satisfied at equality by the n affinely independent points given by the origin and the unit vectors e^i for $i \neq j$, and hence provides a (*trivial*) facet of the antichain polytope.

Chain facets. The simplest nontrivial facets result from maximal chains.

Theorem 3.1 (Fulkerson[1972]) Let C be a maximal chain of the poset and let $\kappa \geq 2$. Then relation (3.1) defines a facet of the antichain polytope.

$$\sum_{j \in C} x_j \leq 1 \qquad (3.1)$$

Proof: Since any chain and antichain have at most one common element, inequality (3.1) is clearly *valid* for P; i.e., all points of P satisfy (3.1). Given the maximal chain C, we note that for each $i \in C$, the unit vector e^i satisfies (3.1) at equality. Furthermore, since C is maximal, to each $i \in V \backslash C$, we can associate an element $j \in C$ such that $\{i, j\}$ is an antichain and $e^i + e^j$ $(\in P)$ satisfies (3.1) at equality. The $|V|$ affinely independent vectors so determined prove that (3.1) gives a facet of P. ■

$(\kappa + 1)$-antichain facets. We next show that any antichain of size $\kappa + 1$ gives rise to a facet of P.

Theorem 3.2 Suppose the entire poset is an antichain; i.e., $E = \emptyset$. If $1 \leq \kappa = |V| - 1$, then relation (3.2) defines a facet of the antichain polytope.

$$\sum_{j=1}^{n} x_j \leq \kappa \qquad (3.2)$$

Proof: The validity of the inequality (3.2) is obvious because κ is the cardinality limit. For each $j \in V$, denote $u^j = \mathbf{1} - e^j$, where $\mathbf{1}$ is the vector of $|V|$ ones and e^j is the unit vector for node j. It is easy to see that the vectors u^j are linearly independent . Thus $\{u^j : j \in V\}$ consists of $|V|$ independent antichain incidence vectors, each satisfying (3.2) at equality, and (3.2) gives a facet for the antichain polytope. ■

Our use of Theorem 3.2 is not restricted to the case in which the entire poset is an antichain. For antichain $A \subseteq V$ with $|A| = \kappa + 1$, replacement of V by A in (3.2) yields a valid inequality for P which can be "lifted" to a facet (see Nemhauser and Wolsey [1988]) through appropriate definition of coefficients for variables $\{x_j : j \in V \backslash A\}$. In Section 4 below we discuss heuristic means for accomplishing the lifting procedure.

Other facets. We now present several more complex classes of facets. Consider the Hasse diagram (all edges are assumed directed "downwards") of Figure 3.1, in which $|S_1| = m$, $|S_1'| = m - 1$ and $|S_2| = |S_2'| = \kappa + 1 - m$, with $m \geq 2$. The nodes in $S_2 \cup S_1' \cup S_2'$ constitute an antichain, as do those in S_1. Each node of S_2 is related to a single node of S_1, and similarly for the nodes of S_2', though the relative in S_1 is distinct from that for S_2. Finally, each node of S_1' is related to precisely two nodes of S_1, as indicated.

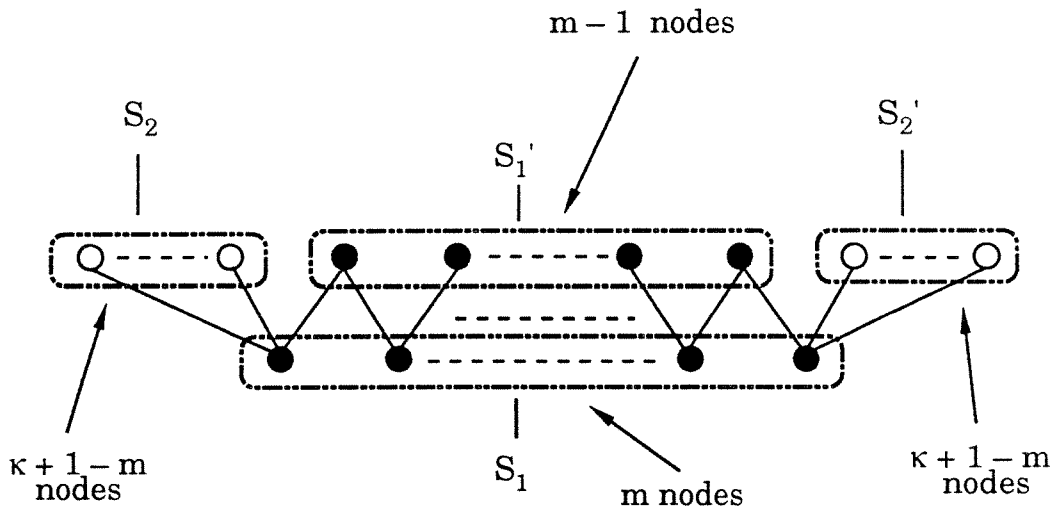


Figure 3.1 : The facet class described in Theorem 3.3.

Theorem 3.3 For the poset indicated in Figure 3.1, with $\beta = \kappa + 1 - m \geq 2$, relation (3.3) defines a facet of the antichain polytope.

$$\sum_{j \in S_1 \cup S_1'} \beta x_j + \sum_{j \in S_2 \cup S_2'} x_j \leq m\beta \qquad (3.3)$$

Proof: To show the validity of (3.3), one observes that if an antichain contains only elements from $S_1 \cup S_1'$, then there can be at most m elements in this antichain, and so its incidence vector clearly satisfies (3.3). If an antichain has elements from $S_2 \cup S_2'$, then at least one element in S_1 cannot be included. Moreover, each element in S_1' is related to exactly two elements in S_1 and *vice versa* (except for the two nodes at the ends of S_1). As a result, the "heaviest" (with respect to the coefficients in (3.3)) antichain containing at least one node in $S_2 \cup S_2'$ is one that contains $m - 1$ nodes from $S_1 \cup S_1'$ and $\kappa - (m - 1) = \kappa + 1 - m$ nodes from $S_2 \cup S_2'$. This antichain has weight $m\beta$ and size κ. Therefore (3.3) is a valid inequality.

Since (3.3) is valid for the antichain polytope P, the vectors in P for which (3.3) holds at equality define a face of P, which is contained in a maximal proper face (i.e., a facet – see Schrijver [1986]) of P. Suppose this facet is given by the elements of P which satisfy relation (3.4) at equality.

$$\sum_{j \in V} a_j x_j \leq b \qquad (3.4)$$

We now show that (3.3) defines a facet of P by demonstrating that the coefficients in (3.3) are the same (up to positive scaling) as those in (3.4). Let i and k be distinct elements in $S_2 \cup S_2'$, and $B \subseteq S_2 \cup S_2'$, $|B| = \beta - 1$, $i, k \notin B$. Consider the antichains $A_i = S_1' \cup B \cup \{i\}$ and $A_k = S_1' \cup B \cup \{k\}$. Note that $|A_i| = |A_k| = (m - 1) + (\beta - 1) + 1 = \kappa$. Moreover, the incidence vectors of A_i and A_k satisfy (3.3), and hence (3.4), at equality; i.e.,

$$\sum_{j \in S_1'} a_j + \sum_{j \in B} a_j + a_i = b, \qquad \sum_{j \in S_1'} a_j + \sum_{j \in B} a_j + a_k = b.$$

The difference of the above two equalities yields $a_i = a_k$. Since i and k are selected arbitrarily from $S_2 \cup S_2'$, all variables corresponding to the elements in $S_2 \cup S_2'$ must have identical coefficients in (3.4).

Let $S_1 = \{1, 2, \ldots, m\}$, $S_1' = \{1', 2', \ldots, (m - 1)'\}$, as shown in Figure 3.2. For

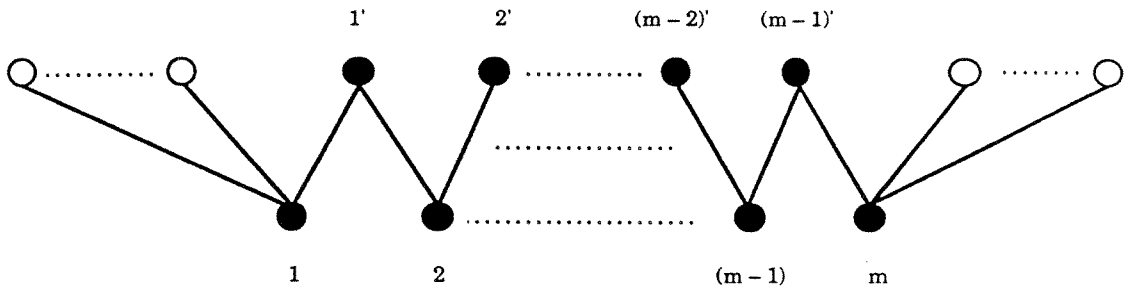


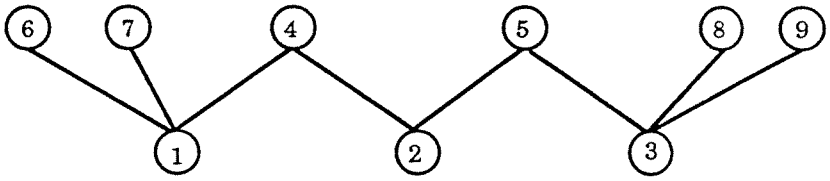
Figure 3.2 : Labeling of the nodes in S_1 and S_1'.

$1 \le k \le m - 1$, we consider antichains $D_k = \{1, 2,..., k, (k + 1)',..., (m - 1)'\} \cup S_2'$ and $D_{k-1} = \{1, 2,..., (k - 1), k', (k + 1)',..., (m - 1)'\} \cup S_2'$. Note that $|D_k| = |D_{k-1}| = \kappa$. The incidence vectors of D_k and D_{k-1} satisfy (3.3) at equality, as $\beta(m - 1) + 1(\beta) = m\beta$; consequently, they satisfy (3.4) at equality. The difference of these two equalities arising from (3.4) yields $a_k = a_{k'}$. Repeated application of this argument shows that $a_k = a_{k'}$ for $k = 1, 2,..., m-1$. Similarly, from antichains $E_k = \{1', 2',..., k', (k + 2), (k + 3),..., m\} \cup S_2$ and $E_{k-1} = \{1', 2',..., (k - 1)', (k + 1), (k + 2),..., m\} \cup S_2$, we obtain $a_{k'} = a_{k+1}$ for $k = 1, 2,..., m - 1$. Hence all variables corresponding to elements in $S_1 \cup S_1'$ have identical coefficients in (3.4).

Without loss of generality, we may thus assume that the coefficients in (3.4) of variables indexed by $S_2 \cup S_2'$ are ones. All that remains to show is that $a_i = \beta$, $i \in S_1 \cup S_1'$. Since we know that the coefficients of the variables corresponding to the nodes in $S_1 \cup S_1'$ are identical, it is sufficient to show that $a_m = \beta$. To this end, we consider antichains $G_1 = \{1, 2,..., m\}$ and $G_2 = \{1, 2,..., m - 1\} \cup S_2'$. Note that $|G_1| = m < \kappa$ and $|G_2| = m - 1 + \beta = \kappa$. Again the incidence vectors of these two antichains satisfy (3.3), and hence (3.4), at equality and the difference of the two equalities arising from (3.4) yields $a_m = \beta = |S_2'|$. Hence relations (3.3) and (3.4) are equivalent. ■

Figure 3.3 demonstrates the facet class of Theorem 3.3. The rows of the

matrix in Figure 3.3 give linearly independent antichain incidence vectors lying on the facet.



6	7	8	9	4	5	1	2	3
1	1	0	0	1	1	0	0	0
0	1	1	0	1	1	0	0	0
1	0	1	0	1	1	0	0	0
1	0	0	1	1	1	0	0	0
1	1	0	0	1	0	0	0	1
0	0	1	1	0	1	1	0	0
0	0	1	1	0	0	1	1	0
1	1	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1	1

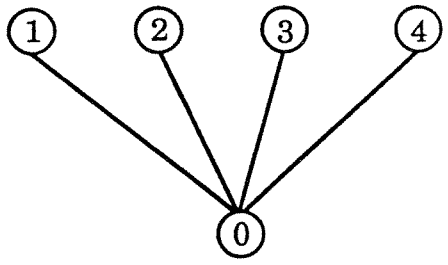
$$\text{Facet}: \sum_{j=1}^{5} 2x_j + \sum_{j=6}^{9} x_j \le 6$$

Figure 3.3 : An example of the facet class of Theorem 3.3 for $\kappa = 4$.

Note that for $m = 1$, $\kappa \ge 2$ and $\beta = \kappa$ with $|S_2 \cup S_2'| = \kappa + 1$, $S_1' = \varnothing$ and $S_1 = \{x_0\}$, we obtain a degenerate case of this facet ; namely,

$$\beta x_0 + \sum_{j \in S_2 \cup S_2'} x_j \le \beta .$$

A simple example of the degenerate case is shown in Figure 3.4. Although the present proof no longer is valid, it is not difficult to see that this inequality still determines a facet of the antichain polytope.



Facet : $3x_0 + x_1 + x_2 + x_3 + x_4 \leq 3$, $\kappa = 3$.

Figure 3.4: A degenerate example of the facet class of Theorem 3.3 .

We next consider the poset represented by the Hasse diagram of Figure 3.5, with node set $S = S_1 \cup S_2 \cup S_3 \cup S_4$, where $|S_1| = m$, $|S_2| = mr + 2$, $|S_3| = 1$, $|S_4| = \beta$, and parameters satisfying $m \geq 1$, $r \geq 2$, $\beta \geq 2$, $q \geq 1$. Each of S_1, S_2 and $S_3 \cup S_4$ is an antichain. Each node in S_1 is dominated by exactly $r + 1$ nodes in S_2. In S_2 the node v^* is independent of S_1 and the node u^* dominates all nodes in S_1. Each of the remaining nodes in S_2 dominates exactly one node in S_1. S_3 consists of only a single node which dominates (any) $mr + 2 - q$ nodes in S_2, including v^*, but not u^*. The node v^* in S_2 is dominated by all the nodes in S_4. The nodes in S_4 only dominate v^*.

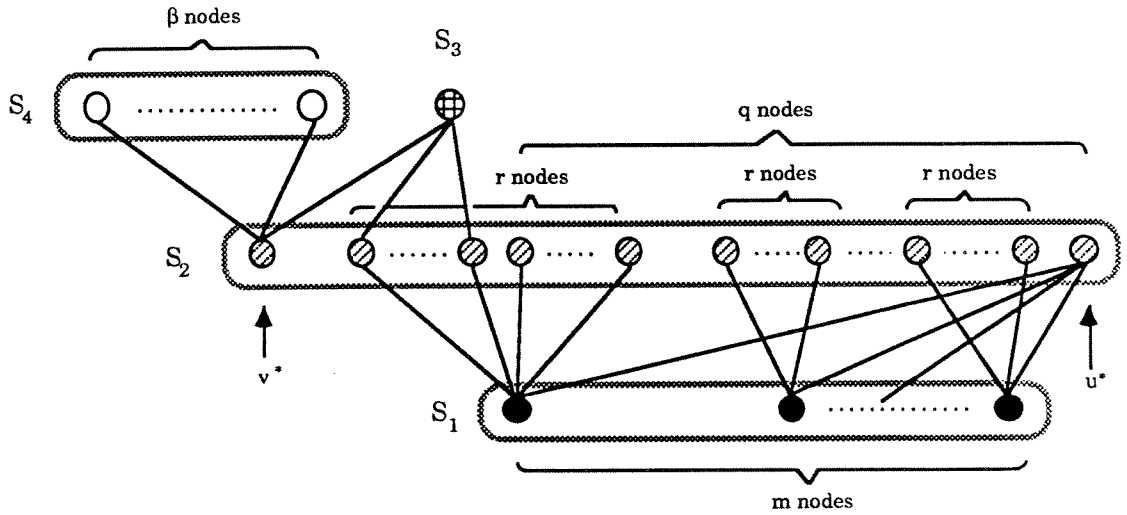


Figure 3.5 : The facet class described in Theorem 3.4 .

Theorem 3.4 Let S be the poset depicted in Figure 3.5 and let $\sigma = r\beta$, $\mu = (\kappa - q)(\beta - 1) + 1$. If (1) $\kappa = mr + 1 \geq m + \beta$ and (2) $1 \leq \kappa - q - 1 \leq \beta - 1$ are satisfied, then relation (3.5) determines a facet of the antichain polytope.

$$\sum_{j \in S_1} \sigma x_j + \sum_{j \in S_2} \beta x_j + \sum_{j \in S_3} \mu x_j + \sum_{j \in S_4} x_j \leq \kappa\beta \qquad (3.5)$$

Proof: We show the validity of inequality (3.5) by considering first an antichain A of size no greater than κ containing no elements from S_1. If $A \subseteq S_2 \cup S_4$, it is easy to see that the maximum (coefficient) weight A can attain is $\kappa\beta$, achieved by κ nodes from S_2. If $A \subseteq S_2 \cup S_3 \cup S_4$ and A contains the single node in S_3, then the maximum weight is attained when A contains the q nodes in S_2 unrelated to S_3, the node in S_3 and any additional $(\kappa - q - 1)$ nodes of S_4. The resulting weight is $q\beta + \mu + (\kappa - q - 1) = \kappa\beta$.

Alternatively, suppose A is an antichain of size at most κ containing at least one element in S_1. Note $S_1 \cap A \neq \emptyset$ implies $u^* \notin A$. Let $A \subseteq S_1 \cup S_2 \cup S_4$. Note that excluding any k_1 (< m) nodes from S_1 can at most result in including rk_1 nodes of S_2 in A and $\sigma = r\beta$. Since each antichain of size no greater than κ in $S_2 \cup S_4$ has weight at most $\kappa\beta$, A has weight at most $\kappa\beta$.

Finally, let $A \subseteq S_1 \cup S_2 \cup S_3 \cup S_4$ and suppose A contains nodes in both S_1 and S_3. Note that if k_2 (< m) nodes in S_1 are included in A, then $rk_2 + 1$ nodes among the q nodes in S_2 unrelated to S_3 cannot be included in A. Since there are only β nodes in S_4, the "heaviest" weight that A could attain is $k_2\sigma + (q - rk_2 - 1)\beta + \mu + \beta = q\beta + \mu < \kappa\beta$.

Thus (3.5) is a valid inequality for the antichain polytope. To see that (3.5) defines a facet, let F_1 denote the face of P determined by (3.5); as in the proof of Theorem 3.3, there exists a facet F_2 represented by

$$\sum_{j \in V} a_j x_j \leq b \qquad (3.6)$$

such that $F_1 \subseteq F_2 = \{x \in P : x \text{ satisfies (3.6) at equality}\}$. Again we prove that

$F_1 = F_2$ by showing that (3.5) and (3.6) are equivalent inequalities.

We first consider the coefficients of variables corresponding to the nodes in S_4. Let A be the set of q nodes in S_2 independent of S_3. For $i, k \in S_4$, $i \neq k$, select $B_{i,k} \subseteq S_4$ such that $i, k \notin B_{i,k}$ and $|B_{i,k}| = \kappa - q - 2 \geq 0$ (by condition 2). Note that $B_{i,k}$ may be empty. Consider antichains $C_i = \{i\} \cup B_{i,k} \cup S_3 \cup A$, $C_k = \{k\} \cup B_{i,k} \cup S_3 \cup A$. Each contains exactly κ elements and has weight $\kappa - q - 1 + \mu + q\beta = \kappa\beta$. Evaluating (3.6) for C_i and C_k and then taking the difference of the resulting two relations yields $a_i = a_k$. Hence $a_i = a_k$, for all $i, k \in S_4$.

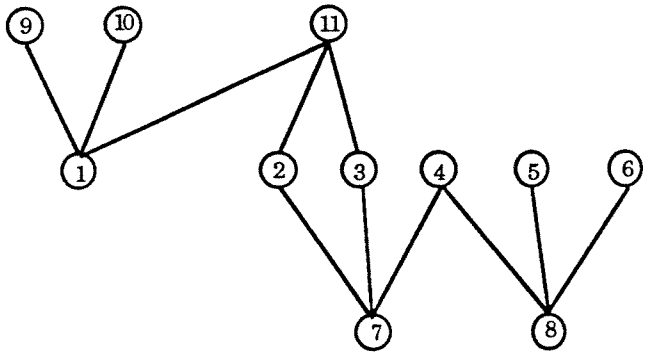
For $i, k \in S_2$ and $i \neq k$, let $D_i = S_2 \backslash \{i\}$ and $D_k = S_2 \backslash \{k\}$. Note that each of these antichains has width $mr + 1 = \kappa$ (by condition 1) and weight $\beta(mr + 1)$ in (3.5). Evaluating (3.6) for D_i and D_k shows that $a_i = a_k$, for all $i, k \in S_2$.

Without loss of generality, we now assume $a_j = 1$, for all $j \in S_4$. We next show that $a_{v^*} = \beta$ by considering antichains $E_1 = S_1 \cup S_4$ and $E_2 = S_1 \cup \{v^*\}$. E_1 has width $m + \beta \leq \kappa$ (by condition 1) and weight $\beta + m\sigma$ in (3.5), while E_2 has width $m + 1 < \kappa$ (by condition 1) and the same weight as E_1. By substituting the incidence vectors of E_1 and E_2 into (3.6) and subtracting, we obtain $a_{v^*} = \beta$, since $a_j = 1, j \in S_4$; hence, $a_j = \beta, j \in S_2$ in (3.6).

We now suppose $m \geq 2$ and show that the coefficients of x_j, $j \in S_1$, are identical in (3.6). For any $j \in S_1$, define $H_j = \{i : i \in S_2;$ i only dominates $j \in S_1\}$ and note that $|H_j| = r$. Consider the antichains $G_i = (S_1 \backslash \{i\}) \cup \{v^*\} \cup H_i$ and $G_k = (S_1 \backslash \{k\}) \cup \{v^*\} \cup H_k$, where $i \neq k$, $i, k \in S_1$. G_i has width $m + r < \kappa$ (by condition 1) with weight $(m - 1)\sigma + \beta + \beta r = m\sigma + \beta$ (by condition 1). Similarly, G_k has identical width and weight as G_i. Evaluation of G_i and G_k in (3.6) yields $a_i = a_k$, for all $i, k \in S_1$.

Since $a_j = \beta$, $j \in S_2$, consideration of antichains E_2 and D_i yields $a_j = r\beta$, for all $j \in S_1$. Finally, we determine the coefficient of the singleton in S_3 using antichains E_2 and C_i, along with the definition of μ. ■

An example of a facet of this type is given in Figure 3.6 along with an invertible (0,1)–matrix whose rows are incidence vectors of antichains lying on this facet. Here C(5,6) denotes the incidence matrix of 5-element subsets of {1,2,3,4,5,6}.



1 2 3 4 5 6 7 8 9 10 11

C (5,6)						0				
1	0	0	0	0	0	1	1	0		
1	1	1	0	0	0	0	1			
0	0	0	0	0	0	1	1	1	1	0
0			1	1	1	0	0	0	1	1
			1	1	1	0	0	1	0	1

$$\text{Facet}: \sum_{j=1}^{6} 2x_j + \sum_{j=7}^{8} 4x_j + \sum_{j=9}^{10} x_j + 3x_{11} \le 10$$

Figure 3.6: An example of the facet class of Theorem 3.4 for $\kappa = 5$.

Further discussion and additional, more complicated facet classes for the antichain polytope are presented in Shum [1989]. We conclude this section by

indicating one such class. Under appropriate technical stipulations, (3.7) defines a facet of the antichain polytope for the poset of Figure 3.7, where σ, β, μ are positve integers. Figures 3.8–3.10 demonstrate this type of facet.

$$\sum_{j \in S_1} \sigma x_j + \sum_{j \in S_2 \cup S_3 \cup S_4} \beta x_j + \sum_{j \in S_5 \cup S_7} x_j + \sum_{j \in S_6} \mu x_j \leq \kappa\beta \qquad (3.7)$$

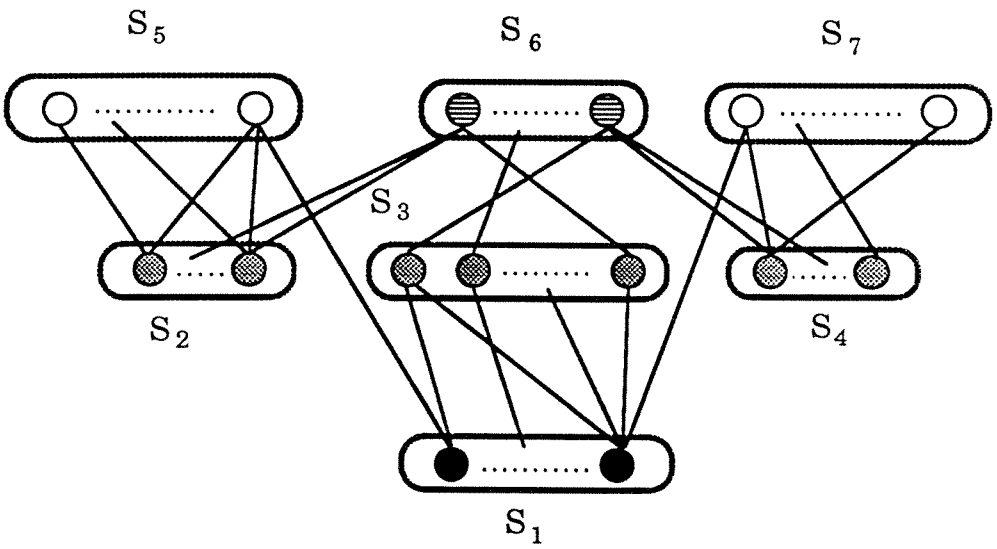
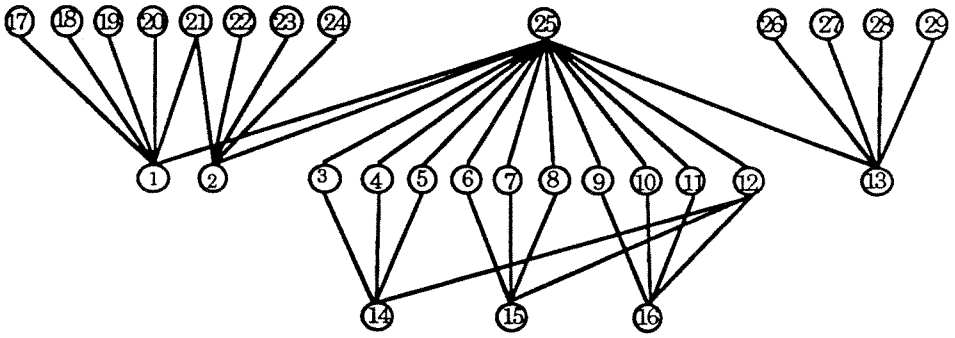
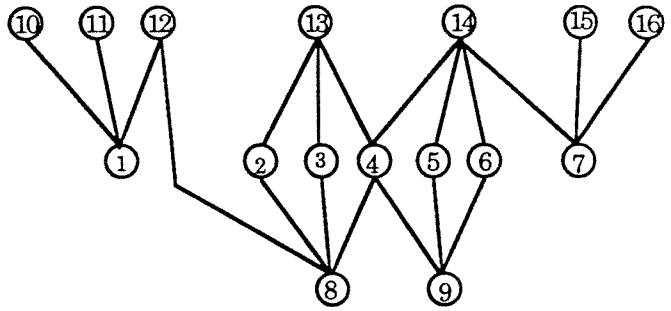


Figure 3.7: Schematic Hasse diagram of a facet-producing poset.



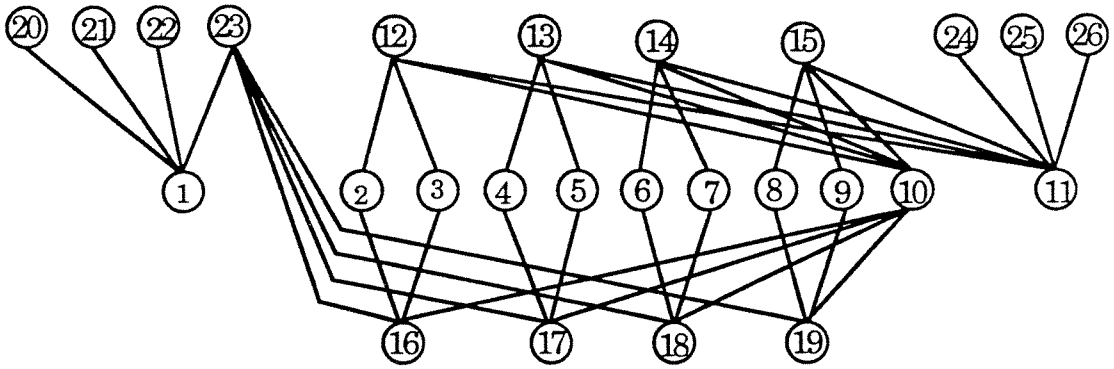
Facet: $\sum_{j=1}^{13} 4x_j + \sum_{j=14}^{16} 12x_j + \sum_{j=17}^{24} x_j + 37x_{25} + \sum_{j=26}^{29} x_j \leq 48$

Figure 3.8: First example of the facet class of Figure 3.7; $|S_6| \neq |S_1|$, $\kappa = 12$.



$$\text{Facet}: \sum_{j=1}^{7} 2x_j + \sum_{j=8}^{9} 4x_j + \sum_{j=10}^{12} x_j + \sum_{j=13}^{14} 4x_j + \sum_{j=15}^{16} x_j \le 12$$

Figure 3.9: Second example of the facet class of Figure 3.7; $|S_6| = |S_1| = 2$, $\kappa = 6$.



$$\text{Facet}: \sum_{j=1}^{11} 3x_j + \sum_{j=12}^{19} 6x_j + \sum_{j=20}^{26} x_j \le 30$$

Figure 3.10 : Third example of the facet class of Figure 3.7; $|S_6| = |S_1| = 4$, $\kappa = 10$.

4. A Strong Cutting Plane Algorithm for MWLA

We now describe a linear programming–based algorithm for solving MWLA. We assume throughout the discussion that P denotes the antichain polytope for the poset S defined on elements 1,2,...,n. The algorithm described here makes explicit use of the maximal chain facets and the $(\kappa + 1)$–antichain facets developed in the previous section. A basic component of the approach is the *separation algorithm*; i.e., given a point $x^* \notin P$, we would like to determine a facet from one of these two facet classes which *separates* x^* from P, i.e., an inequality violated by x^*.

Separation algorithm for maximal chain facets. Suppose $x^* \notin P$. We interpret component x_j^* as the weight associated with node j in the poset. The form of inequality (3.1) shows that if some maximal chain facet is violated by x^*, then there is a chain of weight greater than one in the poset. In particular, the weight of the heaviest chain must exceed one, so any algorithm for determining a maximum weight chain provides a separation algorithm for maximal chain facets. If a chain of maximum weight (with respect to x^*) has weight less than one, then x^* satisfies relation (3.1) for *every* chain of the poset and hence no maximal chain facet separates x^* from the antichain polytope. Otherwise, if C is a maximal chain whose weight exceeds unity, then x^* violates the facet given in relation (3.1).

A chain of maximum weight can be determined in polynomial-time using a simple dynamic programming algorithm. Assuming that the nodes of the Hasse diagram for S are indexed so that $(i, j) \in E \Rightarrow i < j$, we consider the recursive relation $U(j) = \max \{U(i) + w(j) : (i, j) \in E\}$, where U(i) is the weight of the maximum weight chain in $P(i) = \{i\} \cup \{j : j \text{ dominates } i\}$, w(i) is the weight of node i, E is the edge set for the Hasse diagram and U(i) = w(i) if i is a source node. Applying this recursion to the nodes in the order of

increasing index yields a chain of maximum weight. Since the degree of each node is at most n, this algorithm will terminate in at most $O(n^2)$ steps.

Separation algorithm for (κ + 1)–antichain facets. No polynomial–time separation algorithm is known for (κ + 1)–antichain facets. We indicate now a heuristic separation algorithm for this facet class. Let $x^* \notin P$; to improve computational efficiency, we remove each node j with $x_j^* = 0$ from the poset. Thus for the present discussion, we assume $x_j^* > 0$, $1 \le j \le n$.

We attempt to discover an antichain of width $\kappa + 1$, all of whose members are related to a single element of the poset, say j. This will insure that x_j receives a large coefficient (value κ) as a result of the lifting process (see below). The algorithm first sorts the nodes by decreasing degree into the list L, with the node of highest degree, say j^*, being the first element in L. Let R be the set of elements related to j^*. If R has fewer than $\kappa + 1$ nodes, then j^* is deleted from L and the whole process repeats with the (new) first node in L. If R has more than $\kappa + 1$ nodes, but is not an antichain, then the node with the highest degree in R is deleted from R and the nodes of this smaller set are re–examined for independence. The size of R continues to decrease until either R is an antichain of size greater than κ or R contains fewer than $\kappa + 1$ elements. In the latter case, j^* is deleted from L and the procedure repeats from the beginning. In the former case, the inequality is lifted (see discussion below) to the remaining nodes in the order of decreasing degree. If x^* violates the resulting inequality, it is added to the current LP and the new LP is reoptimized; otherwise, j^* is deleted from L. This procedure continues until either $L = \varnothing$ or the degree of the first node in L is less than two.

Generally, the lifting procedure for strengthening a supporting hyperplane to a facet is computationally prohibitive, requiring the solution of an integer programming problem. We approximate this by solving the associated

linear programming relaxation. Crowder, Johnson and Padberg [1983] have demonstrated practical effectiveness of this approach in the context of general 0–1 programming problems. Furthermore, only the constraints of the current LP formulation (see below) are included in this relaxation and coefficients of nodes j with $x_j^* = 0$ are assumed to be zero. These approximations result in an underestimate of actual coefficient values for variables in the $(\kappa + 1)$–antichain facets. Certainly, this diminishes the effectiveness of the $(\kappa + 1)$–antichain inequalities generated, but this approximation achieves an overall gain in computational efficiency.

Computational implementation. The implementation of the strong cutting plane algorithm is based on the code XMP and is written in FORTRAN. XMP is a simplex-based linear programming software package (see Marsten[1981]). All problems were run on a SUN 3/60 equipped with a 68881 floating point processor and 8Mb of memory. We primarily used dual simplex pivots, as this approach turns out to be more efficient than primal simplex pivoting for our problems. This is often the case for combinatorial problems which suffer from severe degeneracy.

The implementation begins with a heuristic procedure for determining an initial solution; this heuristic is based on the fact that MWLA without the cardinality restriction can be solved in polynomial-time using a max-flow algorithm. We assume $q \geq 2$ is a given integer and let S' be the poset generated by the q "heaviest" elements of S. We first find an antichain of maximum weight in S', say A. If $|A| > \kappa$, then only the κ heaviest elements of A are retained, so that $|A| = \kappa$. On the other hand, when $|A| \leq \kappa$, all of A is retained and we delete A and all elements related to members of A in S to obtain a smaller poset T. As before, we find the heaviest antichain (without cardinality restriction) in T, say A'. Now $A \cup A'$ is an antichain and we take (at most) κ elements from $A \cup A'$ as a trial solution. When this

procedure is applied, different values of q may, of course, result in different trial solutions. Our implementation considers seven values of q determined by rounding up the quantity $|S|/k$, for $k = 4,5,...,10$; we then take as an initial solution the heaviest antichain among these candidates. For our test problems, this procedure was extremely effective, often producing an optimal solution for the MWLA problem under consideration.

We begin the actual cutting plane procedure by including only the cardinality constraint of relation (3.2) and the bounds $0 \leq x_j \leq 1$, $j = 1, 2,...,n$, in the linear programming (LP) formulation; suppose x^* is the resulting optimal LP solution. We first attempt to find a maximal chain facet violated by x^* using the algorithm described above. This is carried out regardless of whether x^* is integer–valued or not. Note that even when x^* is integral, it may still not be the incidence vector of an antichain, since our initial LP formulation does not include *all* maximal chain constraints. If a separating maximal chain facet is found, it is appended to the current LP and the new LP is reoptimized. If no separating maximal chain facet can be found and x^* is integral, then x^* satisfies all maximal chain constraints and hence must be an optimal solution to the original problem. If x^* has fractional components, but no separating maximal chain facet is discovered, then we attempt to generate a separating $(\kappa + 1)$–antichain inequality using the heuristic separation algorithm described above. If we succeed in finding such an inequality, it is appended to the current LP and the new LP is reoptimized. Otherwise, no further $(\kappa + 1)$–antichain inequality can be generated and x^* yields an (infeasible) upper bound for the original problem. In such cases our algorithm terminates with a fractional LP solution; the initial (heuristic) solution provides the best known solution for these instances. At this point, one could use the entire procedure described here as the core of an enumerative algorithm for MWLA, but we have not programmed such an algorithm. Nevertheless, as we discuss more fully

below, our approach fails to produce an optimum solution for only about 3% of 2886 MWLA problems considered.

As the algorithm generates constraints *on the fly*, the LP formulation grows in size over time. In order to keep the formulation small, we eliminate all nonbinding constraints whenever the number of such constraints exceeds a certain limit. Our computational experience indicates that fixing this limiting value to twenty works reasonably well in practice.

Data generation and computational results. Our sample problems were produced by a program for random generation of Hasse diagrams of posets. For input, this program requires the length of a longest chain and the maximum number and minimum number of nodes in a *level*. A node j in a Hasse diagram is said to belong to level l if l is the length of a longest chain in the poset ending at j. The length of a longest chain in a Hasse diagram is the number of levels in the diagram. The actual number of nodes in each level is then randomly generated within these limits. Edges between nodes belonging to successive levels are generated at random with Bernoulli trials; the probability that there exists an edge between a pair of nodes is 0.5. Then (non-transitive) edges between nodes in non–consecutive levels are randomly generated in a similar manner. Finally, we generate positive integral weights uniformly distributed between 1 and 99 for the nodes; three sets of weights were generated for each Hasse diagram. These posets have sizes ranging from 100 to 1005 nodes, with the number of edges in Hasse diagrams ranging from 503 to 21044. Altogether, 74 such posets were generated. We have run our algorithms on each poset for each set of weights with thirteen different values of the cardinality limit ranging from $\kappa = 2$ to $\kappa = 30$.

Computational results for the posets described in Table 4.1 are given in Tables 4.2–4.5. Under the "heur. opt." column, the (feasible) lower bounds

resulting from the heuristic procedure described above are shown. In the following three columns, "1" stands for the maximal chain facets and "2" stands for the $(\kappa + 1)$–antichain inequalities. Note that in the cutting plane algorithm, $(\kappa + 1)$–antichain inequalities are only generated when maximal chains facets fail to produce an integral optimal solution. For example, in row 4 of Table 4.4, poset i105a has a heuristic optimum of 1618. When the cutting plane algorithm is applied to the same problem using only maximal chain facets, we obtain an upper bound of 1633.8, 137 cuts are generated and it takes 578 seconds to complete the run. In this case, the resulting solution (value 1633.8) is not integral, so we attempt to generate violated $(\kappa+1)$–antichain inequalities. When both maximal chain facets and $(\kappa+1)$–antichain inequalities are generated, the upper bound decreases to 1632.6, 144 cuts remain in the final LP formulation and the computational time increases to 7508 seconds. Under the "remark" column, "1" indicates that using maximal chain facets alone results in a fractional upper bound solution, "2" indicates that both types of inequalities are generated and the cutting plane algorithm terminates because it can no longer generate violated inequalities, although the final solution remains fractional, and "3" indicates that the problem is abandoned due to severe degeneracy. The value of the cardinality limit is indicated at the top of each table.

The reader should note that the cutting plane algorithm requires much longer to terminate when $(\kappa + 1)$–antichain inequalities are generated, primarily because each time the coefficient of a variable is determined by lifting, a linear programming problem must be solved. In most examples, however, maximal chain facets suffice to yield integral optimal solutions.

We have tested our procedure on 2886 randomly generated problems. A complete discussion of this computational testing is given in Shum[1989]; some typical instances are presented in Tables 4.2-4.5 below. Of these

problems, 2637 (91.4%) are solved to optimality using only maximal chain facets. When both maximal chain facets and $(\kappa + 1)$–antichain inequalities are used, 158 (5.5%) additional problems are solved to optimality. Thus the cutting plane algorithm solves 96.9% of the test problems to optimality. There are 60 (2.1%) problems for which the algorithm (generating both types of facets) terminates with fractional solutions and additionally 31 (1.1%) problems are abandoned due to degeneracy; i.e., only about 3% of the test problems cannot be solved to optimality using these routines. Furthermore, in some problem instances for which an integral solution is not produced by the cutting plane algorithm, the final LP value is actually within one unit of the corresponding (feasible) heuristic lower bound. For such problems, the heuristic has discovered, of course, an optimal solution.

Table 4.1: Sizes of some randomly generated Hasse diagrams.

problem #	# of nodes	# of edges	# of levels
i64a	208	1983	31
i82a	300	2563	16
i100a	416	9126	10
i105a	513	8226	15
i110a	605	4956	40
i115a	715	17860	14
i119a	792	11690	26
i121a	891	14598	29
i123a	1005	10973	47

Table 4.2 : Computational results for MWLA problems; $\kappa = 4$.

problem #	heur. opt.	cut opt. (1 only/1 & 2)	# of cuts (1 only/1 & 2)	CPU time (1 only/1 & 2)	Remarks
i64a	363.0	363.0/ –	37/ –	24/ –	
i82a	388.0	388.0/ –	7/ –	17/ –	
i100a	383.0	392.0/ –	21/ –	38/ –	
i105a	388.0	388.0/ –	31/ –	59/ –	
i110a	387.0	387.0/ –	56/ –	102/ –	
i115a	393.0	393.0/ –	18/ –	87/ –	
i119a	391.0	394.0/ –	20/ –	102/ –	
i121a	389.0	391.0/ –	23/ –	130/ –	
i123a	387.0	392.0/ –	59/ –	211/ –	

Table 4.3 : Computational results for MWLA problems; $\kappa = 12$.

problem #	heur. opt.	cut opt. (1 only/1 & 2)	# of cuts (1 only/1 & 2)	CPU time (1 only/1 & 2)	Remarks
i64a	621.0	621.0/ –	47/ –	73/ –	
i82a	1037.0	1037.0/ –	30/ –	31/ –	
i100a	1062.0	1079.5/ 1078.4	135/ 87	264/ 1564	1,2
i105a	1085.0	1085.0/ –	51/ –	110/ –	
i110a	1015.0	1026.0/ –	43/ –	170/ –	
i115a	1143.0	1143.0/ –	42/ –	125/ –	
i119a	1087.0	1087.0/ –	82/ –	518/ –	
i121a	1111.0	1111.0/ –	77/ –	291/ –	
i123a	1095.0	1095.0/ –	117/ –	726/ –	

Table 4.4 : Computational results for MWLA problems; $\kappa = 20$.

problem #	heur. opt.	cut opt. (1 only/1 & 2)	# of cuts (1 only/1 & 2)	CPU time (1 only/1 & 2)	Remarks
i64a	621.0	621.0/ –	44/ –	60/ –	
i82a	1537.0	1537.0/ –	57/ –	56/ –	
i100a	1597.0	1597.0/ –	119/ –	439/ –	
i105a	1618.0	1633.8/ 1632.6	137/ 144	578/ 7508	1,2
i110a	1451.0	1451.0/ –	71/ –	380/ –	
i115a	1819.0	1819.0/ –	75/ –	186/ –	
i119a	1637.0	1637.0/ –	111/ –	1100/ –	
i121a	1714.0	1714.0/ –	55/ –	449/ –	
i123a	1595.0	1595.0/ –	105/ –	1418/ –	

Table 4.5 : Computational results for MWLA problems; $\kappa = 30$.

problem #	heur. opt.	cut opt. (1 only/1 & 2)	# of cuts (1 only/1 & 2)	CPU time (1 only/1 & 2)	Remarks
i64a	621.0	621.0/ –	38/ –	46/ –	
i82a	1934.0	1937.0/ 1935.4	83/ 107	131/ 6545	1,2,3
i100a	2151.0	2151.0/ –	124/ –	439/ –	
i105a	2152.0	2152.0/ –	86/ –	336/ –	
i110a	1625.0	1625.0/ –	105/ –	646/ –	
i115a	2540.0	2540.0/ –	59/ –	194/ –	
i119a	2136.0	2136.0/ –	133/ –	1761/ –	
i121a	2284.0	2284.0/ –	127/ –	853/ –	
i123a	2000.0	2000.0/ –	131/ –	2682/ –	

5. Concluding Remarks

We have restricted attention in the previous two sections to the polyhedral structure, i.e., derivation of specific facet classes, of the antichain polytope and the exploitation of these facets in a linear programming–based algorithm for optimization over this polytope (MWLA). Analogously, one can consider the chain polytope for the problem MWLC, i.e., the polytope whose extremal elements are the incidence vectors of chains of limited length in a given poset. Classes of facets for the chain polytope, analogous to those given here for the antichain polytope, are developed in detail in Shum [1989].

Shum [1989] also reports computational experience for solving MWLC by a strong cutting plane algorithm. As with MWLA, this algorithm for MWLC is quite effective, solving 99% of the test problems generated to integral optimality. Recall, however, that in Section 2 above we have given a dynamic programming recursion which solves MWLC in polynomial-time. In Shum [1989] it is shown that this polynomial–time algorithm is more efficient computationally than the linear programming–based approach; we note that this contrasts results of Grotschel and Holland [1985] for the matching problem.

Finally, we mention that the existence of a polynomial–time algorithm for MWLC suggests the existence of an explicit characterization of the (limited length) chain polytope, though the facets described in Shum [1989] suggest that such a description would necessarily be complex.

6. References

K. Cameron and J. Edmonds [1979], Algorithms for Optimal Anti–chains, *Research Report CORR 79–22*, Dept. of Combinatorics and Optimization, University of Waterloo, Ontario, Canada.

H. Crowder, E.L. Johnson and M.W. Padberg [1983], Solving Large Scale Zero–One Linear Programming Problems, *Oper. Res. 31,* 803–834.

L.R. Ford and D.R. Fulkerson [1962], *Flows in Networks*, Princeton University Press.

D.R. Fulkerson [1972], Anti-blocking Polyhedra, *J. Comb. Th. B 12,* 50-71.

M.R. Garey and D.S. Johnson [1979], *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman.

M. Grotschel and O. Holland [1985], Solving Matching Problems with Linear Programming, *Math. Prog. 33*, 243-259.

P. Hansen, A. Hertz and J. Kuplinsky [1990], Bounded Vertex Colorings of Graphs, preprint.

R.E. Marsten [1981], XMP: A Structured Library of Subroutines for Experimental Mathematical Programming, *ACM Trans. Math. Soft.* 7,481.

E.L. Lawler [1976], *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston.

G.L. Nemhauser and L.A. Wolsey [1988], *Integer and Combinatorial Optimization,* John Wiley & Sons.

A. Schrijver [1986], *Theory of Linear and Integer Programming*, John Wiley & Sons.

Henry Shum [1989], Chains of Bounded Length and Antichains of Bounded Width in Partially Ordered Sets, *Ph.D. dissertation*, Cornell University, Ithaca, New York.