Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

COMPUTING MAXIMUM WEIGHTED K-FAMILIES AND K-COFAMILIES IN PARTIALLY ORDERED SETS

J. Cong

May 1993
CSD-930014

Computing Maximum Weighted k-Families and k-Cofamilies in Partially Ordered Sets

Jason Cong

Department of Computer Science
University of California at Los Angeles
Los Angeles, CA 90024
(E.mail: cong@cs.ucla.edu, Tel: 310-206-2775)

Abstract

A k-family in a partially ordered set is the union of at most k antichains. A k-cofamily in a partially ordered set is the union of at most k chains. In this paper, we show that Greene and Kleitman's results on the k-families and k-cofamilies can be generalized to partially ordered sets with positive weights. Moreover, we present an algorithm for computing a maximum weighted k-family in $O(n^2 m \log n^2/m)$ time and an algorithm for computing a maximum weighted k-cofamily in $O(n^2 \log n + mn)$ time. These algorithms lead to efficient solutions to a number graph theory problems, such as the maximum weighted k-colorable subgraph problem for comparability graphs, and VLSI routing problems, such as the two-row planar routing problem with fixed density and the multi-layer planar routing problem for channels.

1. Introduction

A *partially ordered set* (poset) is a set of elements together with a binary relation defined on this set which is reflexive, antisymmetric, and transitive. Let P be a finite poset. A *chain* in P is a totally ordered subset of P. An *antichain* in P is a set of mutually unrelated elements of P. Let d_1 and $\hat{d}_1$ denote the maximum size of an antichain and a chain, respectively. The well-known Dilworth's theorem [Di50] states that P can be partitioned into d_1 chains. The dual of Dilworth's theorem states that P can be partitioned into $\hat{d}_1$ antichains. In order to generalize the Dilworth's Theorem, the notion of k-families and k-cofamilies were introduced. A *k-family* in P is a subset of P which contains no chain of size $k+1$. A *k-cofamily* is a subset of P which contains no antichain of size $k+1$. It is easy to see that a k-family can be represented as the union of at most k antichains, and a k-cofamily can be represented as the union of at most k chains. Greene and Kleitman showed that the Dilworth's theorem could be generalized naturally as follows [GrKl76, Gr76].

Let d_k denote the maximum size of a k-family. Let $\Gamma = \{C_1, C_2, ..., C_q\}$ be a chain partition of P. Let $m_k(\Gamma) = \sum_{i=1}^{q} \min(|C_i|, k)$. It is easy to see that $d_k \le m_k(\Gamma)$ for any chain partition Γ since the intersection of a k-family with any chain contains no more than k elements. We say that a chain partition Γ is *k-saturated* if $m_k(\Gamma) = d_k$. Greene and Kleitman [GrKl76] showed that

Theorem 1.1 (Greene and Kleitman, 1976) For any poset P, there always exists a k-saturated chain partition of P, i.e.

$$d_k = \min_{all\ \Gamma} m_k(\Gamma).$$

Similarly, let $\hat{d}_k$ denote the maximum size of a k-cofamily. Let $\Lambda = \{A_1, A_2, ..., A_q\}$ be an antichain partition of P. Let $\hat{m}_k(\Lambda) = \sum_{i=1}^{q} \min(|A_i|, k)$. It is easy to see that $\hat{d}_k \le \hat{m}_k(\Lambda)$ for any antichain partition Λ. We say that an antichain partition Λ is *k-saturated* if $\hat{m}_k(\Lambda) = \hat{d}_k$. The dual of the Dilworth's theorem can be generalized as follows [Gr76]:

Theorem 1.2 (Greene, 1976) For any poset P, there always exists a k-saturated antichain partition of P, i.e.

$$\hat{d}_k = \min_{all\ \Lambda} m_k(\Lambda).$$

Theorem 1.1 and 1.2 was proved originally by Greene and Kleitman based on lattice theory methods. Hoffman and Schwartz [HoSc77] showed a generalization of Theorem 1.1 using linear programming duality. Later on, Saks gave a simply and elegant combinatorial proof of Theorem 1.1 by applying Dilworth's theorem to the product poset of P and a chain of length k. Frank [Fr80] showed a nice common generalization of Theorem 1.1 and 1.2 based on the notion of orthogonal chain families and antichain families. Cameron [Ca82] proved an important bijection theorem for maximal weighted k-families, which shall be used later in this paper.

Although significant progress have been made to understand the structure of maximum k-families (k-cofamilies) and their relation to k-saturated chain (antichain) partitions, we see only a limited amount of work on the computational aspect of the maximum k-families or k-cofamilies in a poset. Gavril [Gr87] showed that a maximum (cardinality) k-family can be computed in $O(n^3 \log n)$ time in the worst case and that a maximum (cardinality) k-cofamily can be computed in $O(n^3)$ time in the worst case. For a poset with positive weights, Mohring [Mo85] showed that a maximum weighted antichain (1-family) can be computed in $O(n^3)$ time and a maximum weighted chain (1-cofamily) can be computed in $O(n^2)$ time in the worst case. Berenguer, Diaz and Harper showed that a maximum weighted k-family can be computed in $O(n^6)$ time in the worst case [BeDH]. Sarrafzadeh and Lou showed that a maximum weighted k-cofamily can be computed in $O(n^3)$ time in the worst case [SaLo90].

In this paper, we shall study the maximum weighted k-families and k-cofamilies in posets with positive weights. First, we show that the Greene and Kleitman's results (Theorem 1.1 and 1.2) can be generalized to posets with positive weights. Then, we show that the problem of computing a maximum weighted k-family can be reduced to the dual problem of computing a flow with maximum k-bounded gain. Based on this reduction, we obtain an $O(n^2 m \log n^2/m)$ time algorithm for computing a maximum weighted k-family in the worst case, where n is the number of elements in the poset and m is the number of related pairs in the poset (which is bounded by $O(n^2)$. Moreover, we show that the problem of computing a maximum weighted k-cofamily can be reduced to the problem of computing a minimum cost flow of fixed value. Based on this reduction, we obtain an $O(n^2 \log n + mn)$ time algorithm for computing a maximum

weighted k-cofamily in the worst case. Finally we show that these algorithms for computing a maximum weighted k-family or k-cofamily lead to efficient solutions to many interesting problem, such as the maximum weighted k-colorable subgraph problem for comparability graphs, the two-row planar routing problem with fixed density, and the multi-layer planar routing problem for channels.

2. Existence of Weighted k-Saturated Chain and Antichain Partitions

In this section, we generalize Theorem 1.1 and 1.2 to posets with integer weights. We show that for any poset with integer weights there always exists a weighted k-saturated chain or antichain partition.

Let P be a partially ordered set with a weight function w on the elements. Without loss of generality, we assume that w is an integer function (otherwise, we shall scale w to an integer function). The weight of a k-family (or k-cofamily) in P is the sum of the weights of the elements in the k-family (k-cofamily). A collection of chains $\Gamma^w = \{C_1, C_2, ..., C_q\}$ is called a *weighted chain partition* of P if every element p in P occurs in $w(p)$ chains in Γ^w. Let $m_k(\Gamma^w) = \sum_{i=1}^{q} \min(|C_i|, k)$. Let d_k^w denote the maximum weight of a k-family in P. Clearly, $d_k^w \le m_k(\Gamma^w)$ for any weighted chain partition Γ^w since the intersection of a k-family with any chain contains no more than k elements. We say that a weighted chain partition Γ^w is k–*saturated* if $m_k(\Gamma^w) = d_k^w$. We show that Theorem 1.1 can be generalized as follows:

Theorem 2.1 (Generalization of Theorem 1.1) For any poset P with an integer weight function w, there always exists a weighted k-saturated chain partition, i.e.

$$d_k^w = \min_{all\ \Gamma^w} m_k(\Gamma^w).$$

Similarly, a collection of antichains $\Lambda = \{A_1, A_2, ..., A_q\}$ is called a *weighted antichain partition* of P if every element p in P occurs in $w(p)$ antichains in Λ^w. Let $\hat{m}_k(\Lambda^w) = \sum_{i=1}^{q} \min(|A_i|, k)$. Let $\hat{d}_k^w$ denote the maximum weight of a k-cofamily in P. Then, $\hat{d}_k^w \le \hat{m}_k(\Lambda^w)$ for any weight antichain partition Λ^w. We say that a weighted antichain partition Λ^w is *k–saturated* if $\hat{m}_k(\Lambda^w) = \hat{d}_k^w$. We can generalize Theorem 1.2 as follows:

Theorem 2.2 For any poset P with an integer weight function w, there always exists a weighted k-saturated antichain partition, i.e.

$$\hat{d}_k^w = \min_{all\ \Lambda^w} m_k(\Lambda^w).$$

In order to prove Theorem 2.1, we first replace each element p in P by an antichain of $w(p)$ elements, each of them has the same relationship as p with respect to the rest of elements in the poset. Then, we apply Theorem 1.1 to the expanded poset to show the result. Similarly, in order to prove Theorem 2.2, we first replace each element p in P by a chain of $w(P)$ elements, each of them has the same relationship as p with respect to the rest of elements in the poset.

Then, we apply Theorem 1.2 to the expanded poset to show the result. The details of the proofs are left to the reader.

3. Computing a Maximum Weighted k-Family in a Poset

In this section, we present an strong polynomial time algorithm for computing a maximum weighted k-family in a poset in $O(n^2 m \log n^2/m)$ in the worst case, where n is the number of elements in the poset and m is the number of related pairs in the poset. We show that the problem of computing a maximum weighted k-family in a poset can be reduced to the one of computing a flow in a network with maximum k-bounded gain.

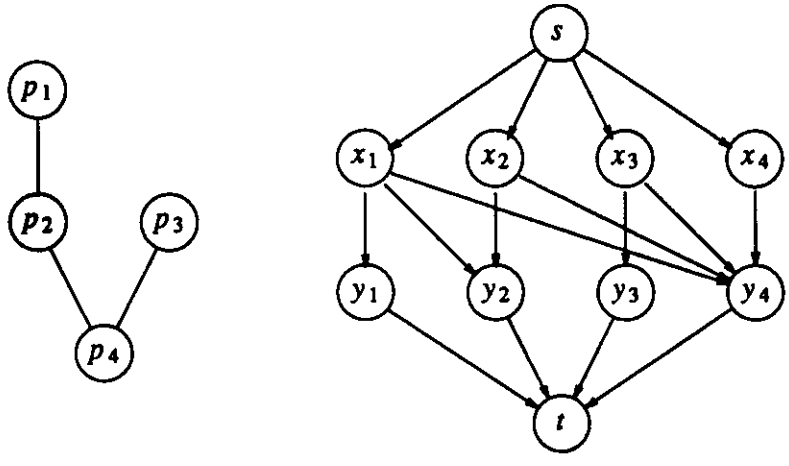
Let P be a poset with positive weights. Let $p_1, p_2, ..., p_n$ be the elements in P and $\leftarrow$ be the partial ordering relation. Let w_i denote the weight of p_i. First, we construct the *split graph* [We85] $G(P) = (V, E)$ associated with P as follows: For each element p_i in P, we introduce two vertices x_i and y_i in V. We introduce an direct edge (x_i, y_j) in E if $p_j \leftarrow p_i$. Moreover, we introduce two more vertices s (source) and t (sink) in V and add edges (s, x_i) and (y_i, t) for $1 \le i \le n$ in E. Fig. 3-1 shows an example of a poset and its corresponding split graph. Now we define the capacity of each edge (x, y), denoted $c(x, y)$, as follows:

$$c(s, x_i) = c(y_i, t) = w_i, \quad \text{for } 1 \le i \le n, \text{ and}$$
$$c(x_i, y_j) = \infty, \quad \text{for } 1 \le i, j \le n.$$

Furthermore, we define the cost of each edge (x, y), denoted $a(x, y)$, as follows:

$$a(s, x_i) = a(y_i, t) = 0, \quad \text{for } 1 \le i \le n, \text{ and}$$
$$a(x_i, y_j) = 0, \quad \text{for } 1 \le i \ne j \le n, \text{ and}$$
$$a(x_i, y_i) = 1, \quad \text{for } 1 \le i \le n.$$

So far, we have constructed a network $G(P)$ for the given poset P. Each edge in the network has



(a) (b)

Fig. 3-1 (a) A poset P. (b) Its split graph $G(P)$.

a non-negative capacity and a non-negative cost associated with it.

Now let us consider a flow f from s to t in $G(P)$. (For convenience, any flow referred in the rest of this paper is a flow from s to t in $G(P)$.) We use $f(x,y)$ to denote the value of the flow on edge (x,y). Let $v(f)$ denote the total value of flow f from s to t, i.e., $v(f) = \sum_{(s,x)\in E} f(s,x) = \sum_{(y,t)\in E} f(y,t)$. For a positive integer k, we define the *k-bounded gain of flow f* to be $k\cdot v(f) - \sum_{(x,y)\in E} a(x,y)\cdot f(x,y)$. We are interested in finding a flow f with the maximum k-bounded gain. We shall show later that a flow with the maximum k-bounded gain in $G(P)$ leads to a maximum weighted k-family in P.

In fact, the problem of computing a flow with the maximum k-bounded gain can be formulated as the following linear programming problem:

$$\sum_{(s,x)\in E} f(s,x) - v(f) \le 0, \tag{3.1}$$

$$\sum_{(x,y)\in E} f(x,y) - \sum_{(y,x)\in E} f(y,x) \le 0, \qquad x \in V,\ x \ne s,t, \tag{3.2}$$

$$v(f) - \sum_{(y,t)\in E} f(y,t) \le 0, \tag{3.3}$$

$$0 \le f(x,y) \le c(x,y), \quad (x,y) \in E, \tag{3.4}$$

$$\text{maximize } k\cdot v(f) - \sum_{(x,y)\in E} a(x,y)\cdot f(x,y). \tag{3.5}$$

Note that although we use inequalities in equations (3.1) - (3.3), it is easy to see that the equalities are implied for these equations. (These equations enforce the flow conservation property.) Now let us consider the dual problem of this linear programming problem. For each flow conservation equation stated in (3.1) - (3.3), we introduce a dual variable $\pi(x)$. For each capacity constraint stated in (3.4), we introduce a dual variable $\gamma(x,y)$. Then, it is easy to derive that the dual problem has the following formulation (see [FoFu62]):

$$-\pi(s) + \pi(t) = k, \tag{3.6}$$

$$\pi(x) - \pi(y) + \gamma(x,y) \ge -a(x,y), \quad (x,y) \in E, \tag{3.7}$$

$$\pi(x) \ge 0, \quad x \in V, \tag{3.8}$$

$$\gamma(x,y) \ge 0, \quad (x,y) \in E, \tag{3.9}$$

$$\text{minimize } \sum_{(x,y)\in P(E)} c(x,y)\cdot\gamma(x,y). \tag{3.10}$$

Each dual variable of the form $\pi(x)$ is called the *node potential* of vertex x. And each dual

variable of the form $\gamma(x, y)$ is called the *edge slack* of edge (x, y). Then, we have the following important result:

Theorem 3.1 Suppose that $\{\hat{\pi}(x) | x \in V\}$ is the set of node potentials in an optimal solution to (3.6) - (3.10). Then, we have:

(1) $0 \le k - \hat{\pi}(x_i) + \hat{\pi}(s) \le k$ for $1 \le i \le n$; and

(2) $A_h = \{p_i | k - \hat{\pi}(x_i) + \hat{\pi}(s) = h$ and $\hat{\pi}(y_i) - \hat{\pi}(x_i) = 1\}$ is an antichain in P for each $0 \le h \le k$; and

(3) $A_1 \cup A_2 \cup \cdots \cup A_k$ is a maximum k-family is P.

The proof of Theorem 3.1 is similar to the proof of the bijection result regarding the k-family in Chapter of [Ca82]. We leave out the details of the proof. According to this theorem, in order to compute a maximum weighted k-family in a poset P, we need to solve the dual problem defined in (3.6) - (3.10) of computing a flow with maximum k-bounded gain in $G(P)$. In the remainder of this section, we describe a primal-dual algorithm for solving both the primal and dual problem of computing a flow with the maximum k-bounded gain in a network. Such an algorithm was originally used by Ford and Fulkerson [FoFu62] for computing a minimum cost maximum flow in a network.

Given a direct graph $G = (V, E)$ in which each edge has a non-negative capacity and a non-negative cost, We starts with the initial solution $f(x, y) = 0$ (for all $(x, y) \in E$) and $\pi(x) = 0$ (for all $x \in V$). In order to compute a flow with the maximum k-bounded gain, the algorithm goes through k iterations. During each iteration, first, we construct the admissible graph $H = (V, E')$ of G. The admissible graph H has the same vertex set as G. The edge set E' in H is defined as follows: An edge (x, y) belongs to E' if and only if

(i) $a(x, y) + \pi(x) - \pi(y) = 0$ and $f(x, y) < c(x, y)$ in G; or
(ii) $a(x, y) + \pi(x) - \pi(y) = 0$ and $f(y, x) > 0$ in G

In case (i), we define the capacity of edge (x, y) in H to be $c(x, y) - f(x, y)$. In case (ii), we define the capacity of edge (x, y) in H to be $f(y, x)$. Next, we compute a maximum flow f' in H from s to t. Then, we augment the flow f in G by f'. Moreover, let $R_{f'}(H)$ be the residual graph for flow f' in H. We increase the node potential $\pi(x)$ by one if x is not reachable from s in $R_{f'}(H)$. (Note that at least t is not reachable from s in $R_{f'}(H)$ since f' is a maximum flow in H.) The updated f and π are used in the construction of the admissible graph in the next iteration. At the end of k-th iteration, we can show that f is a flow with the maximum k-bounded gain and π is the node potential function in an optimal solution to the dual problem defined in (3.6) - (3.10). (The proof of the correctness of the algorithm can be found in [FoFu62, pp.113-127].) Note that during each iteration, the most time-consuming step is to compute a maximum flow in the admissible graph H, which can be carried out in $O(mn \log n^2/m)$ time using an algorithm by Goldberg and Tarjan [GoTa86]. Thus, this algorithm computes a flow with the maximum k-bounded gain and the corresponding dual variables in a network in

$O(k \cdot mn \log n^2/m)$ time. According to Theorem 3.1, such an algorithm leads to an $O(k \cdot mn \log n^2/m)$ time algorithm for computing a maximum weighted k-family. Therefore, we have

Theorem 3.2 For a poset with positive weights, a maximum weighted k-family can be computed in $O(k \cdot mn \log n^2/m) = O(n^2 m \log n^2/m)$ time in the worst case, where n is the number of elements in the poset and m is the number of related pairs in the poset.

4. Computing a Maximum Weighted k-cofamily in a Poset

In this section, we present a strong polynomial time algorithm for computing a maximum weighted k-cofamily in a poset with positive weights in $O(n^2 \log n + mn)$ time in the worst case. We show that the problem of computing a maximum weighted k-cofamily in a poset is equivalent to the problem of computing a minimum cost flow of a fixed value in a network.

Let P be a poset with positive weights. We construct the corresponding split graph $G(P)$ the same way as specified in Section 3. However, we assign the capacities and costs of the edges in $G(P)$ differently. We define the capacity of each edge e, denoted $c(e)$, to be 1. We define the cost of each edge e, denoted $a(e)$, to be:

$$a(e) = \begin{cases} w_i & \text{if } e = (x_i, y_i) \\ 0 & \text{otherwise} \end{cases}$$

We shall show that a maximum weighted k-cofamily in P corresponds to minimum cost flows of value $n - k$ in $G(P)$.

According to Dilworth's Theorem, any k-cofamily can be partitioned into no more than k chains. A k-cofamily is said to be *non-trivial* if it can be partitioned into exactly k chains. For a poset with positive weights, it is easy to see that any maximum weighted k-cofamily is a non-trivial k-cofamily. The following theorem shows that the connection between the non-trivial k-cofamilies in P and the $(n-k)$-flows in $G(P)$. (For convenience, we use f-flow to refer to a flow of value f from s to t in $G(P)$.) According to the result in [CoLi91] (Theorem 7, p. 978), we have

Theorem 4.1 Let P be a poset of n elements with positive weights. Then, P has a non-trivial k-cofamily of weight D if and only if $G(P)$ has a $(n-k)$-flow of cost $W - D$, where W is the sum of the weights of all the elements in P. (Clearly, it is also equal to the sum of the costs of all the edges in $G(P)$.)

Since every maximum weighted k-cofamily is a non-trivial k-cofamily, according to Theorem 4.1, we conclude that the problem of computing a maximum weighted k-cofamily in P is equivalent to the problem of computing a minimum cost $(n - k)$-flow in $G(P)$.

A minimum cost $(n - k)$-flow in $G(P)$ can be computed as follows. We recall the result that *any flow obtained from a minimum cost flow by augmenting along an augmenting path of minimum cost is also a minimum cost flow* (Theorem 8.12 [Ta83]). A minimum augmenting path

can be found by finding a minimum cost path from s to t in the residual graph. Our algorithm works as follows: We start with a zero flow f in $G(P)$. Initially, the residual graph R is the same as $G(P)$. We find a minimum cost path from s to t in the residual graph R and augment the flow f in $G(P)$ by one (since each edge capacity is one). Next, we modify the costs of the edges in $G(P)$ such that

$$d'(v, w) = d(v, w) + cost(v) - cost(w), \tag{4.1}$$

where $cost(v)$ is the cost of a minimum cost path from s to v in the residual graph R (they were computed already as we compute the minimum cost path from s to t). Then, we update the residual graph R. We repeat the augmenting process until the value of the flow f reaches $n-k$. It is easy to show that modifying the costs of the edges in $G(P)$ according to (4.1) does not change the relative ordering of the augmenting paths (i.e., a minimum augmenting path still has the minimum cost among all the augmenting paths after we modify the edge costs). Moreover, such modification of the costs of the edges guarantees that the costs of the edges in the residual graph R are always non-negative. Therefore, we can compute a minimum cost path from s to t in R in $O(n \log n + m)$ time using Fibonacci Heaps [FrTa87], where m is the number of edges in R. Obviously, our algorithm goes through $n-k$ augmenting steps since the capacity of each edge in $G(P)$ is one. Therefore, the complexity for computing a minimum cost $(n-k)$-flow in $G(P)$ is $O((n-k)(n \log n + m)) = O(n^2 \log n + m)$. Based on these discussions, we have

Theorem 4.2 For a poset with positive weights, a maximum weighted k-cofamily can be computed in $O((n-k)(n \log n + m)) = O(n^2 \log n + mn)$ time in the worst case, where n is the number of elements in the poset and m is the number of related pairs in the poset.

5. Applications

In this section, we show that many interesting application problems can be formulated as the problem of computing the maximum weighted k-families or k-cofamilies of a poset. Therefore, we can apply the algorithms presented in the preceding sections to solve these problems efficiently. In the remainder of this section, we demonstrate a few such applications.

5.1. The k-Colorable Subgraph and k-Union of Clique Problems in Comparability and Incomparability Graphs

It is easy to see that the k-families in a poset correspond to the k-colorable subgraphs in the corresponding comparability graph of the poset (or the unions of k cliques in the corresponding incomparability graph of the poset). The k-cofamilies in a poset correspond to the union of k cliques in the corresponding comparability graph of the poset (or the k-colorable subgraphs in the corresponding incomparability graph of the poset). Therefore, the algorithms in Section 3 and 4 lead to efficient algorithms for computing the maximum weighted k-colorable subgraphs and the maximum weighted k-union of cliques in comparability and incomparability graphs, such as permutation graphs and interval graphs.

5.2. The Two-Row Planar Routing Problem with Fixed Density

Given two rows of terminals, a two-row planar routing solution is a set of planar connections in the routing area between the two rows. Given a two-row planar routing solution S, the two-row planar routing problem with fixed density is to choose a maximum weighted subset of connections S' from S such that the density of S' is no more than a given value. This problem occurs as one step in solving the over-the-cell routing problem for standard cell design [CoPL90]. We can show that given a two-row planar routing solution S, we can construct a poset $P(S)$ such that the problem of choose a subset of connections from S with density k can be reduced to the problem of computing a maximum weighted k-family in the corresponding poset $P(S)$. Details for the problem transformation can be found in [CoPL90] and [CoPL93].

5.3. The Multi-Layer Planar Routing Problem for Channels

A channel is a rectangular routing region with terminals on the lower and upper edges of the channel to be connected. The channel routing problem is an important problem in VLSI layout design. Assume that there are k routing layers in the channel and each net has terminals on both the lower edge and the upper edge of the channel. The k-layer planar routing problem is to choose a maximum weighted subset of nets such that each net can be routed entirely in one of the k routing layers (in this case, we do not have to use vias for these nets) [CoLi90]. We show that give a channel C, we can construct a poset $P(C)$ such that the k-layer planar routing problem for C is reduced to the problem of computing a maximum weighted k-cofamily in the corresponding poset $P(C)$. Details for the problem transformation can be found in [CoLi90] and [CoLi91].

Acknowledgments

The author thanks Professors C. L. Liu and Douglas West for their helpful discussions. The author thanks Professor Michael Saks for providing the reference [BeDH].

References

[BeDH] Berenguer, X., J. Diaz and L. H. Harper, "A Solution of the Sperner-Erdos Problem". *Manuscript.*

[Ca82] Cameron, K. B., "Polyhedral and Algorithmic Ramifications of Antichains". *Ph. D. Thesis* (1982), University of Waterloo, Waterloo, Ontario.

[CoLi90] Cong, J. and C. L. Liu, "On k-Layer Planar Subset and Via Minimization Problems". *Proc. European Design Automation Conference*, pp. 459-463, Mar., 1990.

[CoLi91] Cong, J. and C. L. Liu, "On k-Layer Planar Subset and Via Minimization Problems". *IEEE Trans. on CAD*, pp. 972-981, Aug., 1991.

[CoPL90] Cong, J., B. Preas and C. L. Liu, "General Models and Algorithms for Over-the-Cell Routing in Standard Cell Design". *Proc. ACM/IEEE 27th Design Automation Conference*, pp.709-715, 1990.

[CoPL93] Cong, J., B. Preas and C. L. Liu, "Physical Models and Efficient Algorithms for Over-the-Cell Routing in Standard Cell Design". *IEEE Trans. on CAD*, May

1993.

[Di50] Dilworth, R. P., "A decomposition theorem for partially ordered set". *Ann. of Math* (1950) Vol. 51, pp. 161-166.

[FoFu62] Ford, L. R. and D. R. Fulkerson, *Flows in Networks*. Princeton Univ. Press, Princeton, N.J., 1962.

[Fr80] Frank, A., "On chain and antichain families of a partially ordered set". *J. of Combinatorial Theory, Ser. B* (1980) Vol. 29, pp. 176-184.

[FrTa87] Fredman, M. L. and R. E. Tarjan, "Fibonacci Heaps and Their Uses in Improved Network Optimization Algorithms". *Journal of the Association for Computing Machinery*, Vol. 34, No. 3, pp. 596-615, 1987.

[Ga87] Gavril, F., "Algorithms for Maximum k-Coloring and k-Covering of Transitive Graphs". *Networks*, Vol. 17 (1987), pp. 465-470.

[GoTa86] Goldberg, A. V. and R. E. Tarjan, "A new approach to the maximum flow problem". *Proc. Symposium on Theory of Computation* (1986) pp. 136-146.

[Gr76] Greene, C., "Some partitions associated with a partially ordered set". *J. Combinatorial Theory, Ser. A* (1976) Vol. 20, pp. 69-79.

[GrKl76] Greene, C. and D. Kleitman, "The structure of Sperner k-family". *J. Combinatorial Theory, Ser. A* (1976) Vol. 20, pp. 80-88.

[HoSc77] Hoffman, A. J. and D. E. Schwartz, "On partitions of a partially ordered set". *J. Combinatorial Theory, Ser. B* (1977) Vol. 23, pp. 3-13.

[Mo85] Mohring, R. H., "Algorithmic Aspect of Comparability Graphs and Interval Graphs". In: *Graphs and Orders*, I. Rival, ed. D. Reidel Publishing Company, 1985, pp. 41-102.

[ReND77] Reingold, E., J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1977.

[Sa79] Saks, M., "A short proof of the existence of k-saturated partitions of partially ordered sets". *Advances in Mathematics* (1979) Vol. 33, pp. 207-211.

[SaLo90] Sarrafzadeh, M. and R. D. Lou, "Maximum k-Coverings in Transitive Graphs". *IEEE Proc. Int'l Sym. on Circuits and Systems*, May, 1990, pp. 332-335.

[Ta83] Tarjan, R. E., *Data Structures and Network Algorithms*, Society fo r Industrial and Applied Mathematics, Philadelphia, Pennsylvania, 1983.

[We85] West, D. B., "Parameters of partial orders and graphs: packing, covering, and representation". In: *Graphs and Orders*, I. Rival, ed. D. Reidel Publishing Company, 1985, pp. 267-350.