Algorithms for the Minimax k-Ideal Problem

Toshio Nemoto

Faculty of Information and Communication
Bunkyo University
1100 Namegaya, Chigasaki 253, Japan
e-mail: *nemoto@shonan.bunkyo.ac.jp*

October, 1997

Abstract

Suppose we are given a poset(partially ordered set) $\mathcal{P} = (V, \preceq)$, a real-valued weight $w(e)$ associated with each element $e \in V$ and a positive integer k. We consider the problem which asks to find an ideal of size k of $\mathcal{P}$ such that the maximum element weight in the ideal is the minimum for all ideals that can be constructed from $\mathcal{P}$. We call this problem the minimax k-ideal problem. In this paper we propose two fast algorithms: a greedy algorithm and a threshold algorithm. Combining these algorithms, we accomplish the best available bound $\mathrm{O}(\min\{n \log n + m, (m+n)\log^* n\})$ for this problem; the two bounds in this expression are, respectively, due to the greedy algorithm and the threshold algorithm, where $|V| = n$ and m is the number of arcs of the Hasse diagram representing the given poset. This ruesult shows that this problem does not have an $\Omega(n \log n + m)$ lower bound in spite of the fact that the minimum-range k-ideal problem, which is a general problem of the minimax k-ideal problem, has an $\Omega(n \log n + m)$ lower bound.

1. Introduction

A binary relation $\preceq$ on a set V is called a *partial order* when it has the following properties:

- $\forall v \in V$: $v \preceq v$. (Reflexivity)
- $v \preceq w$, $w \preceq v \Rightarrow v = w$. (Antisymmetry)
- $u \preceq v$, $v \preceq w \Rightarrow u \preceq w$. (Transitivity)

We call the pair $(V, \preceq)$ a *partially ordered set* or a *poset* for short. For a poset $\mathcal{P} = (V, \preceq)$ a subset I of V is called an *ideal* of $\mathcal{P}$ if $u \preceq v \in I$ implies $u \in I$. Posets and ideals appear in numerous application settings and in many forms (see, e.g., [Ahuja+Magnanti+Orlin93], [Picard76], [Picard+Queyranne82]). In this paper we consider the *minimax k-ideal problem* defined as follows:

$$\mathrm{P}_{k\text{-minimax}}: \quad \text{Minimize} \quad \max_{e \in I} w(e) \tag{1.1a}$$

$$\text{subject to} \quad I \in \mathcal{I}(\mathcal{P}), \tag{1.1b}$$

$$|I| = k, \tag{1.1c}$$

where w is a weight function $w : V \to \mathbf{R}$ and assumed as $w(\emptyset) = -\infty$. When there is no possibility of confusion, an optimal ideal of Problem $\mathrm{P}_{k\text{-minimax}}$ is called a *minimax ideal* (*of size* k) of $\mathcal{P}$.

In general, for the minimax combinatorial optimization problems there are three main strategies (cf. [Pferschy95] which classifies into three main strategies for solving the bottleneck assignment problem): The first is based on a "greedy" principle — that is, it makes the cheapest choice at each step; the second is based on a "threshold" method which needs an efficient search method to find an optimal solution (see [Lawler76]); the third is a combination of the above two strategies (see [Punnen+Nair94] which succeeded in constructing an efficient algorithm for the bottleneck assignment problem by this strategy).

According to the first and the second general approaches we propose a greedy algorithm and a threshold algorithm for the minimax k-ideal problem. To the author's knowledge, no one has ever considered the minimax k-ideal problem. The greedy algorithm runs in $\mathrm{O}(n \log n + m)$ time and the threshold method can be implemented in $\mathrm{O}((m+n)\log^* n)$ time, where $\log^*$ is the *iterated logarithm*, defined by

$$\log^{(0)} x = x, \tag{1.2a}$$

$$\log^{(i+1)} x = \log \log^{(i)} x, \tag{1.2b}$$

$$\log^* x = \min\{i \mid \log^{(i)} x \leq 1\}. \tag{1.2c}$$

Notice that $\log^* x$ is a very slowly growing function. For example, if $x = 2^{65536}$, then $\log^* x = 5$. Therefore, the required time of the threshold method is shorter than that of the greedy algorithm when $m << (\frac{n}{2})^2$.

Combining these algorithms, we accomplish the best available bound $\mathrm{O}(\min\{n \log n+m, (m+n)\log^* n\})$ for this problem; the two bounds in this expression are, respectively, due to the greedy algorithm and the threshold algorithm. This time complexity shows that the minimax k-ideal problem does not have an $\Omega(n \log n+m)$ lower bound in spite of the fact that the minimum-range k-ideal problem, which is a general problem of the minimax k-ideal problem, has an $\Omega(n \log n+m)$ lower bound[Nemoto95].

2. A Greedy Algorithm

We propose an $\mathrm{O}(n \log n + m)$ algorithm for Problem $\mathrm{P}_{k\text{-minimax}}$ based on the greedy principle. The algorithm enlarges an element set J from an evident ideal $\emptyset$ with the property that J is an ideal of $\mathcal{P}$ keeping. For the set J the algorithm maintains the set C of all the upper neighbors of each element in J, which are candidates for inclusion in J. In choosing an element from the set

C, we use a greedy principle on weights. This approach yields an $O(n \log n + m)$ time complexity. The algorithm is shown precisely in Figure 1.

Algorithm MINIMAX-GREEDY($\mathcal{P}, k$)

Input: A poset $\mathcal{P} = (V, \preceq)$ and a positive integer k.

Output: A minimax ideal of size k of $\mathcal{P}$.

Step 1: Put $J := \emptyset$.

Step 2: Repeat the following $(*)$ k times.

$(*)$ Put $C := \{v \mid v$ is a minimal element of $\mathcal{P}(V - J)\}$. If $C = \emptyset$, then stop (there is no feasible ideal of $\mathcal{P}$); otherwise find a minimum-weight element $\hat{v}$ in C and put $J := J \cup \{\hat{v}\}$.

Step 3: The current J is a minimax ideal of size k.

(End)

Figure 1: A greedy algorithm for Problem $\mathrm{P}_{k\text{-minimax}}$.

The validity of this algorithm is shown below.

Lemma 2. 1 : *The algorithm* MINIMAX-GREEDY($\mathcal{P}, k$) *computes a minimax ideal of size* k *of a poset* $\mathcal{P}$.

(Proof) For the ideal $\hat{I}$ found by the algorithm MINIMAX-GREEDY($\mathcal{P}, k$), let $\hat{e}$ be a maximum-weight element in $\hat{I}$, and let C' and J' be the set C and J, respectively, in the algorithm MINIMAX-GREEDY($\mathcal{P}, k$) just before $\hat{v}$ is chosen. Similarly, for a minimax ideal I^* of size k, let v^* be a maximum-weight element in I^*. Suppose $w(\hat{v}) > w(v^*)$. If $v^* \notin J'$, i.e., $v^* \in V - J'$, then $I^* \cap C' \neq \emptyset$ from the fact that C' is the set of all the minimal elements of $\mathcal{P}(V - J')$. Hence, I^* has an element $\tilde{v} \in I^* \cap C'$ such that $w(v^*) < (w(\hat{v}) \leq) w(\tilde{v})$, contradicting the fact that e^* is the maximum weight element in I^*. If $v^* \in J'$, then there exists an element $\tilde{v} \in C' \cap I^*$ such that $w(\tilde{v}) \leq w(v^*)(< w(\hat{v}))$ because $|I^*| > |J'|$ and e^* has the maximum weight in I^*. This contradicts the choice of $\hat{v}$. Consequently, we have $w(\hat{v}) = w(v^*)$. □

We now turn to the time complexity analysis. In Step 2-$(*)$, it is not difficult to renew the set C by making use of the list of all the lower neighbors of each element in $\mathcal{P}(V - J)$. Notice that the list of all the lower neighbors is controlled in the complete list representation of the given Hasse diagram $G(\mathcal{P})$. Suppose C_i is the set C at the ith iteration, we can get $C_{i+1} = (C_i - \{\hat{v}\}) \cup \{v \mid$

the list of arcs δ^+v has just become empty by removing $\hat{v}$ from $G(\mathcal{P}(V - J))\}$. Finding C_1 and identifying new elements added to C_{i+1} at the end of the ith iteration, require $\mathrm{O}(m+n)$ time in the whole of the algorithm. By having the heap data structure (see [Ahuja+Magnanti+Orlin93]) for C, finding a minimum-weight element $\hat{v}$ in C, inserting new members to C and deleting $\hat{v}$ from C are carried out in $\mathrm{O}(\log n)$, respectively. Since each operation is done for an element at most once, it takes $\mathrm{O}(n \log n)$ time. In total, this algorithm requires $\mathrm{O}(n \log n + m)$ time. Consequently, we have the following theorem.

Theorem 2. 2: *The algorithm* MINIMAX-GREEDY$(\mathcal{P}, k)$ *computes a minimax ideal of size* k *of* $\mathcal{P}$ *in* $\mathrm{O}(n \log n + m)$. □

Example: Consider Problem $\mathrm{P}_{k\text{-minimax}}$ of $k = 5$ on the poset $\mathcal{P}$ represented by the Hasse diagram shown in Figure 2. The weight of each element is attached at the lower left of the element. The algorithm MINIMAX-GREEDY$(\mathcal{P}, 5)$ might execute as indicated in Table 1. At termination, we have a minimax ideal $\{v_1, v_4, v_8, v_{13}, v_{14}\}$.

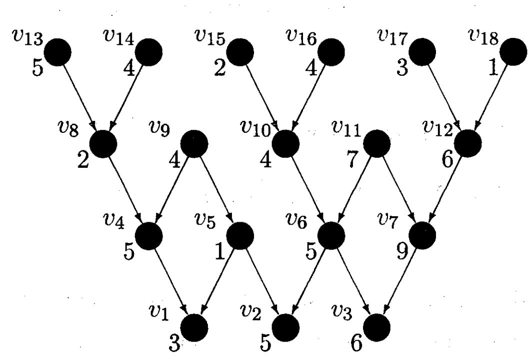


Figure 2: The Hasse diagram $G(\mathcal{P})$ representing a poset $\mathcal{P} = (V, \preceq)$ with the weights.

3. A Threshold Algorithm

Combining an efficient search method and a threshold method, we describe the other algorithm for the minimax cardinality-restricted ideal problem. Basically, the framework of the algorithm described here is the same as that of the algorithm proposed by Gabow and Tarjan [Gabow+Tarjan88] for the bottleneck spanning tree problem. However, to make use of the framework it is necessary to give some new ideas.

Table 1: The algorithm MINIMAX-GREEDY($\mathcal{P}$, 5) applied to the poset shown in Figure 2.

	Iter.	C	$\hat{v}$	J
Step 1				$\emptyset$
Step 2	1	$\{v_1, v_2, v_3\}$	v_1	$\{v_1\}$
	2	$\{v_2, v_3, v_4\}$	v_4	$\{v_1, v_4\}$
	3	$\{v_2, v_3, v_8\}$	v_8	$\{v_1, v_4, v_8\}$
	4	$\{v_2, v_3, v_{13}, v_{14}\}$	v_{14}	$\{v_1, v_4, v_8, v_{14}\}$
	5	$\{v_2, v_3, v_{13}\}$	v_{13}	$\{v_1, v_4, v_8, v_{14}, v_{13}\}$

First we show two lemmas which play a fundamental role in the threshold approach for the ideal problems.

Lemma 3. 1 : *For a poset $\mathcal{P} = (V, \preceq)$ and a subset U of V let*

$$W = V - \bigcup_{v \in U} F(v), \tag{3.1}$$

where $F(v) = \{w \mid w \in V, v \preceq w\}$ which is called the principal filter generated by v. Then $W \subseteq V - U$ and W is an ideal of $\mathcal{P}$.

(Proof) It is clear that $W \subseteq V - U$. Suppose W is not an ideal of $\mathcal{P}$, that is, there exist two elements $w \in W$ and $u \notin W$ ($u \in \bigcup_{v \in U} F(v)$) such that $u \preceq w$ on $\mathcal{P}$. Since $w \in F(u) \subseteq \bigcup_{v \in U} F(v)$, we have $w \notin W$, contradicting the fact $w \in W$. □

Lemma 3. 2 : *For a poset $\mathcal{P} = (V, \preceq)$ let I be an ideal of $\mathcal{P}$. Then, for a subset $U \subseteq V$, U is an ideal of the subposet $\mathcal{P}(I)$ if and only if U is an ideal of $\mathcal{P}$ and $U \subseteq I$.*

(Proof) ($\Rightarrow$) Suppose U is not an ideal of $\mathcal{P}$. Hence, there exist two elements u and v such that $u \preceq v$, $v \in U$ and $u \notin U$. If $u \in I - U$, then it contradicts the fact that U is an ideal of $\mathcal{P}(I)$. If $u \in V - I$, then it contradicts the fact that I is an ideal of $\mathcal{P}$. Therefore, U is an ideal of $\mathcal{P}$.
($\Leftarrow$) Suppose U is not an ideal of $\mathcal{P}(I)$, i.e., there exist two elements u and v such that $u \preceq v$, $v \in U$ and $u \notin U$, i.e., $v \in U$ and $u \in I - U \subseteq V - U$. This contradicts that U is an ideal of $\mathcal{P}$. □

For any real value β, let $H(\beta) = \{e \mid e \in E, w(e) > \beta\}$. Then, the set

$$E - \cup_{e \in H(\beta)} F(e) \tag{3.2}$$

is an ideal of $\mathcal{P}$ due to Lemma 3. 1 . We abbreviate the subposet $\mathcal{P}(E - \cup_{e \in H(\beta)} F(e))$ induced by (3.2) to $\mathcal{P}(-\infty, \beta]$. Now, let

$$\beta^* = \min\{ \beta \mid \text{ there is an ideal of size } k \text{ of } \mathcal{P}(-\infty, \beta]\}. \tag{3.3}$$

Then, we have the following lemma.

Lemma 3. 3 : *Any ideal of size k of $\mathcal{P}(-\infty, \beta^*]$ is a minimax ideal of size k of $\mathcal{P}$.*

(Proof) Suppose that there exists an ideal I' of size k of $\mathcal{P}$ such that $\max\{w(e) \mid e \in I'\} < \beta^*$. Then, I' is also the ideal of $\mathcal{P}(-\infty, \beta^*]$ from the fact that $I' \subseteq E - \cup_{e \in H(\beta^*)} F(e)$ and Lemma 3. 2 . This contradicts the minimality of β^*. □

To find a minimax ideal of size k of $\mathcal{P}$ it suffices to compute β^* since an ideal of size k of $\mathcal{P}(-\infty, \beta^*]$ can be found in $O(m + n)$ time by the breath(depth)-first search method.

We find β^* by repeatedly splitting and narrowing the interval of possible values of β. The number of intervals which the current interval is split into is given by a function $A(2, i) : \{1, 2, \cdots\} \rightarrow \mathbf{Z}_+$ defined by

$$A(2, 1) = 2, \tag{3.4a}$$

$$A(2, i) = 2^{A(2, i-1)}. \tag{3.4b}$$

This is the *Ackermann's function* with a slight change (see [Tarjan83] for Ackermann's function). See Figure 3 for the description of a threshold algorithm for Problem $P_{k\text{-minimax}}$.
Here, for any real number x, $\lfloor x \rfloor$ denotes the maximum integer less than or equal to x, and $\lceil x \rceil$ denotes the minimum integer larger than or equal to x. Though it is not necessary to construct the set $S_{A(2,i)+1}$ in Step 2-(a) of the algorithm MINIMAX-THRESHOLD($\mathcal{P}, k$), we need it to implement this algorithm efficiently. The efficient implementation will be described later. The validity of the above algorithm is shown below.

Theorem 3. 4 : *The algorithm* MINIMAX-THRESHOLD($\mathcal{P}, k$) *computes a minimax ideal correctly.*

(Proof) This algorithm terminates in a finite number of iterations since the cardinality of the finite set U is strictly decreasing in Step 2-(d). It is clear $\beta^* = \lambda$ when $\lambda = \mu$. From the property that $\mathcal{P}(S_0 \cup \cdots \cup S_j)$ has an ideal of size k of $\mathcal{P}$ in Step 2-(c), we have $\beta^* = \lambda (= \min\{w(v) \mid v \in U\} = \min\{w(v) \mid v \in U_1\})$ if $j = 0$. □

We now consider an efficient implementation of this algorithm and the time complexity analysis. It is easy to implement Step 1, Step 2-(a) and Step 2-(d). Therefore the remaining part in this section is devoted to the implementation of Step 2-(b) and (c).

Algorithm MINIMAX-THRESHOLD($\mathcal{P}, k$)

Input: A poset $\mathcal{P} = (V, \preceq)$ and a positive integer k.

Output: A minimax ideal of size k of $\mathcal{P}$.

Step 1: Put $\lambda := \min\{w(v) \mid v \in V\}$, $\mu := \max\{w(v) \mid v \in V\}$ and $i := 1$.

Step 2: Repeat the following (a),(b),(c) and (d).

- **(a)** Let $S_0 := \{v \mid v \in V, w(v) \leq \lambda\}$ and $U := \{v \mid v \in V, \lambda < w(v) \leq \mu\}$. (Let $S_{A(2,i)+1} := \{v \mid v \in V, \mu < w(v)\}$.)
- **(b)** Partition U into $A(2,i)$ subsets $S_1, S_2, \cdots, S_{A(2,i)}$, each of size $\lfloor \frac{|U|}{A(2,i)} \rfloor$ or $\lceil \frac{|U|}{A(2,i)} \rceil$, such that if $v \in S_j$ and $u \in S_{j+1}$ $(j = 1, \cdots, A(2,i) - 1)$, then $w(v) \leq w(u)$.
- **(c)** Find $j^* = \min\{j \mid \mathcal{P}(S_0 \cup S_1 \cup \cdots \cup S_j)$has an ideal of size k of $\mathcal{P}\}$.
- **(d)** Put $\lambda := \min\{w(v) \mid v \in S_{j^*}\}$, $\mu := \max\{w(v) \mid v \in S_{j^*}\}$ and $i := i + 1$. If $j^* = 0$ or $\lambda = \mu$, then put $\beta^* := \lambda$ and stop.

Step 3: Find an ideal of size k of $\mathcal{P}(-\infty, \beta^*]$. It is a minimax ideal of size k of $\mathcal{P}$.

(End)

Figure 3: A threshold algorithm for Problem $\mathrm{P}_{k\text{-minimax}}$.

First, we consider the implementation of Step 2-(b). It can be carried out by finding the median: Split U into a lower half and an upper half, then split each half into halves and so on. See Figure 4 which precisely describes the procedure to implement Step 2-(b). The technique used in (b1) of the procedure PARTITION-$A(2,i)$-SUBSETS is known as the procedure *select* [Aho+Hopcroft+Ullman83].

Lemma 3. 5: *The procedure* PARTITION-$A(2,i)$-SUBSETS *implements the required work in* Step 2-(b) *in* $\mathrm{O}(|U| \log A(2,i))$ *time.*

(Proof) The correctness of this procedure is obvious. The time complexity is derived from the following three facts: (1) finding the median in S_j and partitioning into two sets take $\mathrm{O}(|S_j|)$ time [Blum+Floyd+Pratt+Rivest+Tarjan73], (2) the total size of the sets S_j in each iteration of Step 2 is $|U|$ and (3) the number of the repetition of Step 2 is $\log A(2,i)$. □

Next, we consider the implementation of Step 2-(c). The following lemma demonstrates a good condition to test whether a given subposet has an ideal of size k.

Procedure PARTITION-$A(2,i)$-SUBSETS

Input: An element set U with weights w and a positive integer i.

Output: Sets $S_j (j = 1, \cdots, A(2,i))$ satisfying the condition in Step2-(b).

Step 1: Put $S_1 := U$ and $j := 1$.

Step 2: Repeat the following $(*1)$ and $(*2)$ until $j = \log A(2,i)$.

$(*1)$ Put $t = 1$. Repeat the following ($\flat$1) and ($\flat$2) until $t > A(2,i)$.

($\flat$1) If $|S_t| = 1$ or 0, then put $S_{t+\frac{A(2,i)}{2^j}} := \emptyset$; otherwise find the median ν in S_t, put

$$S^< := \{v \mid v \in S_t, w(v) < \nu\},$$
$$S^= := \{v \mid v \in S_t, w(v) = \nu\},$$
$$S^> := \{v \mid v \in S_t, w(v) > \nu\}.$$

and partition $S^=$ into two subsets $S^=_-$ and $S^=_+$ such that $|S^=_-| = \lfloor \frac{S_t}{2} \rfloor - |S^<|$. Put $S_t := S^< \cup S^=_-$ and $S_{t+\frac{A(2,i)}{2^j}} := S^=_+ \cup S^>$.

($\flat$2) Put $t := t + \frac{A(2,i)}{2^{j-1}}$.

$(*2)$ Put $j := j + 1$.

(End)

Figure 4: A procedure of MINIMAX-THRESHOLD($\mathcal{P}, k$).

Lemma 3. 6 : *For a poset $\mathcal{P} = (V, \preceq)$ and a subset U of V, there exists an ideal I of size k of $\mathcal{P}$ such that $I \subseteq U$ if and only if the following inequality holds:*

$$\left| \bigcup_{v \in V-U} F(v) \right| \leq |V| - k. \tag{3.5}$$

(Proof) ($\Rightarrow$) Suppose there is an ideal $I \subseteq U$ of size k of $\mathcal{P}$. Let C be the set of all the upper neighbors of the elements of I on $\mathcal{P}$. Then

$$\bigcup_{v \in C} F(e) = \bigcup_{v \in V-I} F(e) \supseteq \bigcup_{v \in V-U} F(e). \tag{3.6}$$

Therefore,

$$V - \bigcup_{v \in V-U} F(e) \supseteq V - \bigcup_{v \in C} F(e) = I. \tag{3.7}$$

Hence, we have

$$\left| V - \bigcup_{v \in V-U} F(e) \right| \geq |I| = k. \tag{3.8}$$

This means (3.5).
($\Leftarrow$) Let $W = V - \bigcup_{v \in V-U} F(v)$, then we have the facts that $W \subseteq U$ and W is an ideal of $\mathcal{P}$ due to Lemma 3. 1 , and that $|W| \geq k$ from the assumption. These imply that the subposet $\mathcal{P}(W)$ has an ideal I of size k. Thus, from Lemma 3. 2 , I is an ideal of size k of $\mathcal{P}$ such that $I \subseteq W \subseteq U$. □

The following result is an immediate consequence of the preceding lemma.

Corollary 3. 7 : *For sets* S_j $(j = 1, \cdots, A(2,i), A(2,i)+1)$ *obtained in* Step 2-(a) *and* (b) *of the algorithm* MINIMAX-THRESHOLD *and a positive integer* k, *we have*

$$j^* = \min\{j \mid \mathcal{P}(S_0 \cup S_1 \cup \cdots \cup S_j) \text{ has an ideal of size } k \text{ of } \mathcal{P}\} \tag{3.9}$$

$$= \max\left\{ j \;\middle|\; \left| \bigcup_{v \in S_j \cup S_{j+1} \cup \cdots \cup S_{A(2,i)} \cup S_{A(2,i)+1}} F(v) \right| > |V| - k \right\}. \tag{3.10}$$

(Proof) From Lemma 3. 6 and $V - (S_0 \cup S_1 \cup \cdots \cup S_j) = S_{j+1} \cup \cdots \cup S_{A(2,i)} \cup S_{A(2,i)+1}$,

$$(3.9) = \min\left\{ j \;\middle|\; \left| \bigcup_{v \in S_{j+1} \cup S_{j+2} \cup \cdots \cup S_{A(2,i)} \cup S_{A(2,i)+1}} F(v) \right| \leq |V| - k \right\} \tag{3.11}$$

$$= (3.10).$$

□

Due to Corollary 3. 7 we propose a procedure for finding j^* (see Figure 5).

Procedure FIND-j^*

Input: The sets S_j $(j = 1, \cdots, A(2,i)+1)$ and a positive integer k.

Output: The index j^* defined by (3.9).

Step 1: Put $j := A(2,i)+1$ and $Q := \bigcup_{v \in S_{A(2,i)+1}} F(v)$.

Step 2: While $|Q| \leq |V| - k$ do the following $(*)$.

$(*)$ Put $j := j - 1$ and $Q := Q \cup \bigcup_{v \in S_j} F(v)$.

Step 3: The current j is j^*.

(End)

Figure 5: A procedure of MINIMAX-THRESHOLD$(\mathcal{P}, k)$.

Notice that all that we need in Step2-($*$) is to identify the set $\bigcup_{v\in S_j} F(v) - Q = \bigcup_{v\in S_j} F(v) - \bigcup\{F(v) \mid v \in S_{j+1} \cup \cdots \cup S_{A(2,i)} \cup S_{A(2,i)+1}\}$ if we identify the set Q in the previous iteration. The depth(breath)-first search ([Ahuja+Magnanti+Orlin93]) works well for finding the set $\bigcup_{v\in S_j} F(v) - \bigcup_{v\in S_{j+1}\cup\cdots\cup S_{A(2,i)+1}} F(v)$ in Step 2-($*$). To implement this search we attach each element *label* which have one of two states: *unscanned* or *scanned*. See Figure 6 and Figure 7 for an implementation of this search.

Procedure FIND-Q

Input: A set S_j and the set $Q(= \bigcup_{v\in S_{j+1}\cup\cdots\cup S_{A(2,i)+1}} F(v))$ identified by *label*: if $label(v) = scanned$, then $v \in Q$, otherwise $v \notin Q$.

Output: The set $\bigcup_{v\in S_j} F(v) - Q$.

Step 1: Put $K := \emptyset$ and $L := S_j$.

Step 2: While $L \neq \emptyset$, do the following ($*$).

($*$) Select an element $v \in L$. Call DFS(v) and put $K := K \cup \{u \mid u \in V, label(u)$ is changed in DFS(v)$\}$ and $L := L - \{v\}$.

Step 3: The current K is the set $\bigcup_{v\in S_j} F(v) - Q$.

(End)

Figure 6: A procedure in the procedure FIND-j^*.

Lemma 3. 8 : *The procedure* FIND-j^* *which uses the procedure* FIND-Q *with the procedure* DFS(v) *as a subroutine finds* j^* *in* $O(m+n)$ *time.*

(Proof) The procedure FIND-j^* computes $|\cup_{v\in S_{A(2,i)+1}} F(v)|$, $|\cup_{v\in S_{A(2,i)}\cup S_{A(2,i)+1}} F(v)|$, $\cdots$, in turn, and finds j satisfying (3.10). Due to Corollary 3. 7 this j is j^*. The time complexity $O(m+n)$ is obtained from the following two facts: (1) if the procedure DFS(v) is executed for every element at most once, it requires $O(m+n)$ time in total since the total number of times the procedure DFS(v) tests whether $label(v)$ is *scanned* or *unscanned* is at most the number of arcs of the Hasse diagram, and (2) for each element the procedure DFS(v) is executed at most once since the sets S_t $(t = j, \ldots, A(2,i)+1)$ are disjoint. □

Theorem 3. 9 : *The algorithm* MINIMAX-THRESHOLD($\mathcal{P}, k$) *computes a minimax ideal of* $\mathcal{P}$ *in* $O((m+n)\log^* n)$ *time.*

Procedure DFS(v)

Input: A poset $\mathcal{P}$ with *label* and an element v.

Output: The poset $\mathcal{P}$ with *label* in which $label(u) = scanned$ for all $u \in F(v)$.

Step 1: Put $W := \{v\}$.

Step 2: While $W \neq \emptyset$, do the following (∗).

(∗) Select an element $u \in W$. If $label(u) = scanned$, then put $W := W - \{u\}$. If $label(u) = unscanned$, then put $label(u) := scanned$ and $W := W - \{u\} \cup \{\hat{u} \mid \hat{u}$ is an upper neighbor of $u\}$.

(End)

Figure 7: A procedure in the procedure FIND-Q.

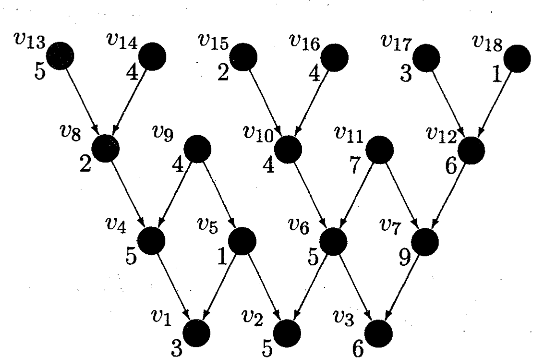
(Proof) In Step 2 each of (a) and (d) takes O(n) time per iteration. From Lemma 3. 5 Step 2-(b) requires O($|U| \log A(2, i)$) time per iteration and from Lemma 3. 8 Step 2-(c) needs O($m+n$) time. Hence, it requires O($|U| \log A(2, i)$+m+n) time in the ith iteration of Step 2. Notice that $|U| \leq \lceil \frac{n}{A(2,i-1)} \rceil$ in the ith iteration. Thus, O($|U| \log A(2, i) + m + n$)=O($m + n$) since

$$|U| \log A(2, i) \leq \lceil \frac{n}{A(2, i-1)} \rceil \log A(2, i) \leq 2\frac{n}{A(2, i-1)} \log 2^{A(2,i-1)} = 2n. \qquad (3.12)$$

The number of iterations of Step 2 is O($\log^* n$). Therefore the over all time bound is O($(m + n) \log^* n$). □

Example: Consider Problem $P_{k\text{-minimax}}$ of $k = 5$ on the poset $\mathcal{P}$ represented by the Hasse diagram which is the same as Figure 2. The algorithm MINIMAX-THRESHOLD($\mathcal{P}$, 5) might execute as indicated as follows. At termination, we have a minimax ideal.

Input:



Step 1: $\lambda = 1$, $\mu = 9$.

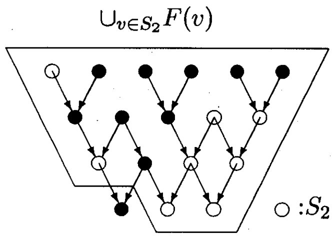
Step 2:

Iteration 1:

(a) $S_0 = \{v_5, v_{18}\}$, $U = \{v_1, v_2, v_3, v_4, v_6, v_7, v_8, v_9, v_{10}, v_{11}, v_{12}, v_{13}, v_{14}, v_{15}, v_{16}, v_{17}\}$, $S_3 = \emptyset$.

(b) $S_1 = \{v_1, v_8, v_9, v_{10}, v_{14}, v_{15}, v_{16}, v_{17}\}$, $S_2 = \{v_2, v_3, v_4, v_6, v_7, v_{11}, v_{12}, v_{13}\}$.

(c)



$|\cup_{v \in V} F(v)| = 17 \leq |V| - k = 13 \Rightarrow j^* = 2$.

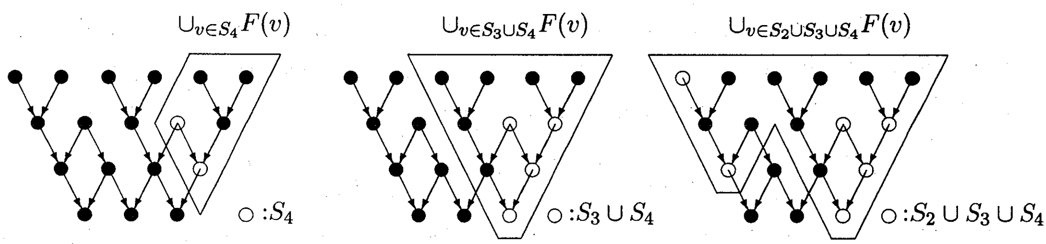
(d) $\lambda = 5$, $\mu = 9$.

Iteration 2:

(a) $S_0 = \{v_1, v_5, v_8, v_9, v_{10}, v_{14}, v_{15}, v_{16}, v_{17}, v_{18}\}$, $U = \{v_2, v_3, v_4, v_6, v_7, v_{11}, v_{12}, v_{13}\}$, $S_5 = \emptyset$.

(b) $S_1 = \{v_2, v_6\}$, $S_2 = \{v_4, v_{13}\}$, $S_3 = \{v_3, v_{12}\}$, $S_4 = \{v_7, v_{11}\}$.

(c)



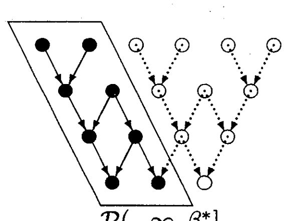
$|\cup_{v \in S_4} F(v)| < |V| - k \Rightarrow j^* \neq 4$.

$|\cup_{v \in S_3 \cup S_4} F(v)| < |V| - k \Rightarrow j^* \neq 3$.

$|\cup_{v \in S_2 \cup S_3 \cup S_4} F(v)| \geq |V| - k \Rightarrow j^* = 2$.

(d) $\lambda = \mu = 5 \Rightarrow \beta^* = \lambda = 5$.

Step 3:



Any ideal of size 5 of $\mathcal{P}(-\infty, \beta^*]$ is a minimax ideal of $\mathcal{P}$.

(Stop)

References

[Aho+Hopcroft+Ullman83] A. V. Aho, J. E. Hopcroft and J. D. Ullman: *Data structures and algorithms* (Addison-Wesley, MA, 1983).

[Ahuja+Magnanti+Orlin93] R. K. Ahuja, T. L. Magnanti and J. B. Orlin: *Network Flows: Theory, Algorithms, and Applications* (Prentice Hall, New York, 1993).

[Blum+Floyd+Pratt+Rivest+Tarjan73] M. Blum, P. Floyd, V. Pratt, R. L. Rivest and R. E. Tarjan: Time bounds for selection. *Journal of Computer and System Sciences* **7** (1973) 448-461.

[Gabow+Tarjan88] H. N. Gabow and R. T. Tarjan: Algorithms for two bottleneck optimization problems. *Journal of Algorithms* **9** (1988) 411-417.

[Lawler76] E. L. Lawler: *Combinatorial optimization: Networks and Matroids* (Holt, Reinehart and Winston, New York, 1976).

[Nemoto95] T. Nemoto: An efficient algorithm for the minimum-range ideal problem: In *Optimization — Modeling and Algorithm—* 7, (The Institute of Statistical Mathematics Cooperative Research Report 77, Tokyo, 1995), pp.26-41.

[Pferschy95] U. Pferschy: Solution methods and computational investigations for the linear bottleneck assignment problem. Technical Report 22-1995, Institute of Mathematics, University of Technology Graz, Austria, (1995).

[Picard76] J. C. Picard: Maximal closure of a graph and applications to combinatorial problems. *Management Science* **22** (1976) 1268-1272.

[Picard+Queyranne82] J. C. Picard and M. Queyranne: Selected applications on minimum cuts in networks. *Information Systems and Operational Research* **20** (1982) 394-422.

[Punnen+Nair94] A. P. Punnen and K. P. K. Nair: Improved complexity bound for the maximum cardinality bottleneck bipartite matching problem. *Discrete Applied Mathematics* **55** (1994) 91-93.

[Tarjan83] R. E. Tarjan: *Data Structures and Network Algorithm* (SIAM, Philadelphia, PA, 1983).