アルゴリズム 50－5
（1996．3．15）

重みつきマトロイドの K 番目に重い基を求める

松井 知己
東京大学 工学系研究科 計数工学専攻

松井 泰子
東京都立大学 理学部 数学教室

本稿では、マトロイドの基を生成する算法をいくつか提案する。提案する基をすべて列挙する算法は、時間計算量が $O(\beta(T+r^2))$ 記憶容量は $O(r^3)$ である。ただし、β は基の個数、r はマトロイドのランク、$O(T+r)$ はサーキットオラクルの計算量である。重みつきマトロイドに対しては、基を重みの重い順に列挙する算法の提案を行なう。この算法で K 番目に重い基を求めるのに必要な時間計算量は $O(n^2+K(T+r^2+\log n))$ であり、必要記憶容量は $O(Krn)$ である。ただし n は台集合の大きさである。また重みが全て整数のときは、基数ヒープを用いることにより、時間計算量は $O(K(T+r^2+\log W))$ となり空間計算量は $O(\log W+Krn)$ となる。ただし W は重みの絶対値の最大の値である。

K Best Bases of a Weighted Matroid

Tomomi Matsui
Department of Mathematical Engineering and Information Physics,
Faculty of Engineering, University of Tokyo,
Bunkyo-ku, Tokyo 113, Japan. tomomi@misojiro.t.u-tokyo.ac.jp

Yasuko Matsui
Department of Mathematics,
Faculty of Science, Tokyo Metropolitan University,
Minami-Ohsawa, Hachioji, Tokyo 192-03, Japan. yasuko@math.metro-u.ac.jp

In this paper, we propose an algorithm which generates the bases of matroids. The enumeration algorithm requires $O(\beta(T+r^2))$ time and $O(r^3)$ space where β is the number of bases, r is the rank of a given matroid and $O(T+r)$ is the time complexity of circuit oracle. In the last section, we describe a ranking algorithm which generates all the bases in order of nonincreasing order of weight. The algorithm requires $O(n^2+K(T+r^2+\log n))$ time and $O(Krn)$ space for finding K best bases. When the weight vector is integer valued, time complexity becomes $O(K(T+r^2+\log W))$ time and the space complexity becomes $O(\log W+Krn)$ space by employing the radix heap structure where W is the maximum absolute value of weights.

1 Introduction

In this paper, we propose an algorithm which generates the bases of matroids. The enumeration algorithm requires $O(\beta(T + r^2))$ time where β is the number of bases, r is the rank of a given matroid and $O(T + r)$ is the time complexity of circuit oracle. In the last section, we describe a ranking algorithm which generates all the bases in order of nonincreasing order of weight. If we employ the Kapoor and Ramesh's data structure [4], then the algorithm requires $O(n^2 + K(T + r^2 + \log n))$ time and $O(Krn)$ space for finding K best bases. When the weight vector is integer valued, time complexity becomes $O(K(T + r^2 + \log W))$ time and the space complexity becomes $O(\log W + Krn)$ space by employing the radix heap structure [1,2] where W is the maximum absolute value of weights.

2 Main Framework

In this section, we give some definitions and describe the main framework of our algorithm.

Let us consider a matroid $M = (E, \mathcal{B})$ with a finite set E and a collection of bases $\mathcal{B}$. We denote $|E|$ by n and $|\mathcal{B}|$ by β. The rank of the matroid M is denoted by r. For any base B of M and for any element $e \notin B$, $C(B|e) \subseteq E$ denotes the unique circuit contained in $B + e$. For any element $e \in B$, the unique cocircuit contained in $(E \setminus B) + e$ is denoted by co-$C(B|e)$.

Throughout this paper, we assign a linear ordering on the set E by setting $E = \{e_1, e_2, \ldots, e_m\}$. For any element e_j, we say that the *index* of e_j is j. We denote the index of an element e by $\mathbf{index}(e)$. Given a subset $E' \subseteq E$, the element in E' with the smallest index is called the *top element* of E' and denoted by $\mathrm{top}(E')$. Similarly, we call the element in E' with the largest index the *bottom element* of E' and denote the element by $\mathrm{btm}(E')$. For any pair of subsets E' and E'', we say E' is lexicographically greater than E'', denoted by $E' >_{\mathrm{lex}} E''$ or $E'' <_{\mathrm{lex}} E'$, when there exists an integer j satisfying that (1) for any k with $0 \leq k < j \leq m$, $e_k \in E'$ if and only if $e_k \in E''$ and (2) $E' \ni e_j \notin E''$.

In this paper, we assume that the following property holds

Assumption 1 *There exists a sequence of integers $(j_0, j_1, \ldots, j_t)$ satisfying that (1) $0 = j_0 < j_1 < \cdots < j_t = m$, and (2) the partition $(F_1, F_2, \ldots, F_t)$ of E where the subset $F_s = \{e_{j_{s-1}+1}, e_{j_{s-1}+2}, \ldots, e_{j_s}\}$, satisfies the conditions that F_1 is a base of M and:*

$$\forall s \in \{2, \ldots, r\},\ F_s \text{ is a maximal independent set in } E \setminus (F_1 \cup F_2 \cup \cdots \cup F_{s-1}).$$

We denote the base F_1 by B^*. Clearly, B^* is the lexicographically maximum base of M. The partition $(F_1, \ldots, F_t)$ defined in Assumption 1 is called *greedy partition* of E and each subset in the greedy partition is called *greedy set*.

For any base B' different from B^*, $\phi(B')$ denotes the base $B'-f+g$ where $f = \mathrm{btm}(B')$ and $g = \mathrm{top}(\text{co-}\mathcal{C}(B'|f))$.

Lemma 1 *Let B' be a base different from B^*. If $\phi(B') = B' - f + g$, then $g \in B^* \not\ni f$.*

It is clear from Assumption 1. and so proof is omitted. The above lemma implies that for any base $B' \neq B^*$, $\phi(B') >_{\mathrm{lex}} B'$ and there exists a positive integer t satisfying $\phi^t(B') = B^*$.

Given a base $B'(\neq B^*)$, we say B' is a *child* of $\phi(B')$ and $\phi(B')$ is the *parent* of B'. Then, we can construct a tree structure, denoted by $(\mathcal{B}, \phi)$, with the node set $\mathcal{B}$ and the arc set $\{(B, B') \in \mathcal{B} \times (\mathcal{B} \setminus \{B^*\}) \mid B = \phi(B')\}$. Clearly, the base B^* corresponds to the root of this tree structure. Our algorithm traverses this tree structure by using a suitable tree search rule and outputs all the bases of M.

3 Finding All the Bases of a Matroid

In this section, we propose a procedure for finding all the bases of a matroid.

Let B be a base of M. We say that an element f is a *child pivot element* of B, if there exists a child B' of B such that $B' \ni f \notin B$. An element g in B is called a *parent pivot element* of B, if there exists a child B' of B such that $B \not\ni g \in B'$. Let $B' = B - g + f$ be a child of B.

From the definition of the parent-children relation, the parent pivot element g is the top edge of co-$\mathcal{C}(B'|f)$. Since co-$\mathcal{C}(B'|f) =$ co-$\mathcal{C}(B|g)$, g is also the top edge of co-$\mathcal{C}(B|g)$. We denote the edge-subset $\{e' \in B \mid e'$ is the top edge of co-$\mathcal{C}(B|e')\}$ by $\mathrm{Bst}(B)$.

Claim 2 *Any parent pivot element of a base B is contained in $Bst(B)$.*

Next, we give an essential characterization of the child pivot element.

Lemma 3 *Let B be a base M. An element f is a child pivot element of B if and only if (1) $\mathtt{index}(btm(B)) < \mathtt{index}(f)$ and (2) there exists an element $g \in Bst(B)$ such that $B - g + f$ is a base.*

The following lemma characterizes the set of children.

Lemma 4 *Let B be a base M. The subset $B' = B - g + f$ is a child of B if and only if B' satisfies the following conditions; (1) B' is a base, (2) $g \in Bst(B)$, (3) $\mathtt{index}(btm(B)) < \mathtt{index}(f)$.*

Now we introduce the *matroid tableau*. For any base B of $M = (E, \mathcal{B})$ and a subset F of E, the *(matroid) tableau* $\mathrm{Tbl}(B|F)$ is a 0-1 valued matrix $X = (x_{fg})$ satisfying that rows are indexed by B, columns are indexed by F and each component x_{fg} satisfies;

$$x_{fg} = \begin{cases} 1 & \text{if } B - f + g \text{ is a base,} \\ 0 & \text{otherwise.} \end{cases}$$

For any element $e \in E$, we denote the tableau $\mathrm{Tbl}(B|\{e\})$ by $\mathrm{Tbl}(B|e)$. Clearly from the definition, for any element $e \notin B$), $\mathrm{Tbl}(B|e)$ is the characteristic vector of the subset $C(B|e) - e$ indexed by B.

Then, Lemma 4 is equivalent to the following.

Theorem 5 *Let B be a base M and X be the submatrix of $Tbl(B|E)$ satisfying that rows are indexed by $Bst(B)$ and columns are indexed by $\{f \in E|\mathtt{index}(btm(B)) < \mathtt{index}(f)\}$. Then, subset $B' = B - g + f$ is a child of B if and only if the element of X indexed by (g, f) is equal to 1.*

The above theorem give an idea for generating all the children of the current base B. Let e_k be the bottom edge of B. For each element f such as $\mathtt{index}(f) > k$, we check whether $\mathrm{Tbl}(B|f)$ is the zero vector or not. If it is not the zero vector, we generate the base $B - g + f$ for each non-zero element of $\mathrm{Tbl}(B|f)$.

Although the above algorithm finds all the children exactly, it is not so efficient. In the worst case, the algorithm checks $O(n - r)$ column vectors and and finds that the current base B has no child. In the following, we give an idea for finding all the pivot elements efficiently.

Assumption 1 implies the following lemma.

Lemma 6 *Any cocircuit C^* of M satisfies the property that if $C^* \cap F_{s+1} \neq \emptyset$, then $C^* \cap F_s \neq \emptyset$, where F_s and F_{s+1} are greedy sets .*

From the above, we can show the following.

Theorem 7 *Let B be a base of M and e the bottom element of B.*

If a greedy set F_s contains a child pivot edge of B, then F_{s-1} contains the bottom element of B or a child pivot element of B.

Corollary 8 *Let B be a base of M and f be a child pivot element of B. Then $\mathtt{index}(f) \leq \mathtt{index}(btm(B)) + 2r - 1$ or there exists a child pivot element f' of B such that $\mathtt{index}(f) - (2r - 1) \leq \mathtt{index}(f') < \mathtt{index}(f)$.*

The above property gives an algorithm for finding all the child pivot elements efficiently. Denote the bottom element of B by e_k. In our algorithm, we check each element in $(e_{k+1}, \ldots, e_n)$ in this order whether it is a child pivot element or not. When a consecutive $2r-1$ elements are not child pivot element, we can stop scanning the element.

Now we have an algorithm for finding all the bases.

Algorithm A

input: a base B

output: family of bases $\{B' \in \mathcal{B} | \phi(B') = B\}$

```
A1: begin
A2:   construct the subset H = Bst(B) from Tbl(B|B*);
A3:   let e_k be the bottom element of B;
A4:   set n' = min{k + 2r - 1, n};
A5:   construct the matroid tableau Tbl(B|F) where F = {e_{k+1}, e_{k+2}, ..., e_{n'}};
A6:   let X be the submatrix of Tbl(B|F) whose rows are indexed by Bst(B);
A7:   set Q be the sequence of column vectors of X which is arranged in increasing
      order of column indices;
A8:   set q be the number of non-zero (column) vectors of X;
A9:   while q ≠ 0 do
A10:    begin
A11:      remove the first vector v of Q;
A12:      if v is not the zero vector then
A13:        begin
A14:          set q := q - 1;
A15:          for each nonzero element v_g of v do output the base B - g + f;
A16:        end;
A17:      set e_{k'} be the index of the vector v;
A18:      if k' + 2r - 1 < n then
A19:        begin
A20:          construct the vector Tbl(B|e_{k'+2r-1});
A21:          set v' be the subvector of Tbl(B|e_{k'+2r-1}) indexed by Bst(B);
A22:          add v' to the last of Q;
A23:          if v' is not the zero vector then set q := q + 1;
A24:        end;
A25:    end;
A26: end
```

Now we discuss the time complexity of Algorithm A. For any base B and an element $f \notin B$, we assume that the tableau $\mathrm{Tbl}(B|f)$ is obtained in $O(T+r)$. Then the above algorithm requires $O(r^2 + (\beta(B)+1)(T+r))$ where $\beta(B)$ is the number of children of B. The memory complexity is $O(r^2)$.

By using Algorithm A as a subprocedure, we can construct a base enumeration algorithm which traverses the tree structure $(\mathcal{T}, \phi)$ by using a tree search rule, e.g., depth-first search rule or breadth-first search rule. Thus, the overall time complexity of the enumeration algorithm becomes

$$O(\sum_{B\in\mathcal{B}}(r^2+(\beta(B)+1)(T+r))) = O(|\mathcal{B}|r^2+((|\mathcal{B}|-1)+|\mathcal{B}|)(T+r)) = O(|\mathcal{B}|(T+r^2)).$$

When we call Algorithm A recursively, it corresponds to a base enumeration algorithm which traverses the tree structure by using the depth-first search rule. Since the height of the tree structure is r, the depth-first search rule saves the memory space to $O(r^3)$.

4 Algorithm for Generating K best bases of a weighted matroid

Let $M = (E, \mathcal{B})$ be a matroid and $w : E \to Z$ be an integer weight vector indexed by E. For any subset $E' \subseteq E$, the *weight* of E, denoted by $w(E')$, is the value $\sum_{e\in E'} w(e)$. In the rest of this paper, we assume that for any pair of bases B, B' of M, $w(B) \neq w(B)$ if $B \neq B'$. This assumption holds by breaking ties lexicographically.

A base of M which maximizes the weight is called maximum base of M. It is well known that the greedy algorithm solves the problem for finding a maximum base. Here, we discuss the problem for generating K best bases of a weighted matroid.

In this section, we assume the following.

Assumption 2 *There exists a sequence of integers $(j_0, j_1, \ldots, j_t)$ satisfying that (1) $0 = j_0 < j_1 < \cdots < j_t = m$, and (2) the partition $(F_1, F_2, \ldots, F_t)$ of E where $F_s = \{e_{j_{s-1}+1}, e_{j_{s-1}+2}, \cdots, e_{j_s}$ satisfies the conditions that (a) F_1 is a maximum base of M and (b) $\forall s \in \{2, \ldots, r\}$, F_s is a maximal independent set in $E \setminus (F_1 \cup F_2 \cup \cdots \cup F_{s-1})$ which maximizes the weight $w(F_s)$.*

The following properties gives an idea for the problem.

Lemma 9 *For any base B and its child B', $w(B) < w(B')$.*

From the above lemma, if we traverse the tree structure by using best first search rule, we can generate all the bases in decreasing order of weight. More precisely, the ranking algorithm maintains all the scanned bases by a set Q. At the entrance of the algorithm, we insert the maximum base to Q. In each iteration, we remove and output the maximum weight base B in Q and insert all the children of B to Q. If we maintain the set Q by an ordinary heap structure, the algorithm requires $O(|\mathcal{B}|(T + r^2 + r\log n))$ time and $O(|\mathcal{B}|+r^2)$ space. By applying the radix heap structure [1,2], which is proposed for shortest path problem, it requires $O(|\mathcal{B}|(T+r^2+r\log W))$ time and $O(|\mathcal{B}|+r^2+\log W)$ space where $W = \max\{|w(e)| \mid e \in E\}$. If we use the Kapoor and Ramesh's data structure for K best

spanning tree problem [4], the time complexity becomes $O(|\mathcal{B}|(T + r^2 + \log n))$ and space complexity becomes $O(|\mathcal{B}| + r^2)$.

Now we consider the problem for finding K best bases.

Lemma 10 *Let B be a base of M. If $B' = B - g + f$ is the maximum weight child of B, then* $\mathtt{index}(f) < \mathtt{index}(btm(B)) + 2r - 1$.

Lemma 11 *Let B be a base of M. and $B'_1, B'_2, \ldots, B'_k$ be the set of children of B satisfying that $w(B_1) < w(B_2) < \cdots < w(B'_k)$. Then,* $\mathtt{index}(btm(B_{k'+1})) \leq \mathtt{index}(btm(B_{k'}) + 2r - 1)$ *for all $k = 1, 2, \ldots, k-1$.*

The above lemmas imply an algorithm for generating K best bases of a matroid.

For any base B of M, we denote the tableau $\mathrm{Tbl}(B|X)$ where $X = \{e \in E \mid i \leq \mathtt{index}(e) \leq j\}$ by $\mathrm{Tbl}[B|i,j]$.

Algorithm B

B1: **begin**
B2: set Q_{gf} be the empty queue for $(g,f) \in B \times E \setminus B$;
B3: output B^* as the maximum base;
B4: set $r' := \min\{3r+1, n\}$;
B5: construct the tableau $\mathrm{Tbl}[B^*|r+1, r']$;
B6: insert $(B^*, \mathrm{Tbl}(B^*|B^*) = I, \mathrm{Tbl}[B^*|r+1, r'])$ to Q_{out};
B7: generate the children $\mathcal{B}' = \{B' = B-g+f \in \mathcal{B} \mid g \in \mathrm{Bst}(B),\ r+1 \leq \mathtt{index}(f) \leq r'\}$;
B8: set $Q_{\mathrm{scan}} = \mathcal{B}'$;
B9: **while** $Q_{\mathrm{scan}} \neq \emptyset$ **do**
B10: **begin**
B11: set B be the base in Q_{scan} which minimizes the weight;
B12: remove the base B from Q_{scan}; output B;
B13: set $k := \mathtt{index}(\mathrm{btm}(B))$;
B14: set $k' := \min\{k + 2r - 1, n\}$;
B15: find the triplet $P = (\phi(B), \mathrm{Tbl}(\phi(B)|B^*), \mathrm{Tbl}[\phi(B)|i,j])$ in Q_{out};
B16: **if** $j < k'$ **then**
B17: **begin**
B18: construct the tableau $\mathrm{Tbl}[\phi(B)|j+1, k']$;
B19: generate $\mathcal{B}' = \{B' = B - g + f \in \mathcal{B} \mid g \in \mathrm{Bst}(B),\ j+1 \leq \mathtt{index}(f) \leq k'\}$;
B20: set $Q_{\mathrm{scan}} := Q_{\mathrm{scan}} \cup \mathcal{B}'$;
B21: replace the triplet P by $P = (\phi(B), \mathrm{Tbl}(\phi(B)|B^*), \mathrm{Tbl}[\phi(B)|i, k'])$;
B22: **end**;
B23: construct the tableau $\mathrm{Tbl}(B|B^*)$;
B24: construct the tableau $\mathrm{Tbl}[B|k+1, k']$;
B25: insert $(B, \mathrm{Tbl}(B|B^*), \mathrm{Tbl}[B|k+1, k']$ to Q_{out};
B26: **end**;

B27: **end**

In Algorithm B, we maintain all the bases in Q_{scan} by the pointer to the triplet of its parent base and the pivoting pair. Then it requires O(1) space. If we employ the Kapoor and Ramesh's data structure [4] for maintaining Q_{out}, then Algorithm B requires $O(n^2 + K(T + r^2 + \log n))$ time $O(Krn)$ space for finding K best bases. By using radix heap structure [1,2], time complexity becomes $O(K(T + r^2 + \log W))$ time and the space complexity becomes $O(\log W + Krn)$ space.

References

[1] R.K. AHUJA, K. MEHLHORN, J.B. ORLIN AND R.E. TARJAN, *Faster algorithms for the shortest path problem*, J. of ACM 37 (1990), pp.213–223.

[2] R.K. AHUJA, T.L. MAGNANTI AND J.B. ORLIN, *Network Flows: Theory, Algorithm and Applications*, Prentice Hall, 1993.

[3] D. AVIS AND K. FUKUDA, *Reverse search for enumeration*, research report 92-5, Graduate School of Systems Management, The University of Tsukuba, 1992.

[4] S. KAPOOR AND H. RAMESH, *Algorithms for enumerating all spanning trees of undirected and weighted graphs*, SIAM J. Computing 24 (1995), pp.247–265.