2006－AL－108 (6)
2006／9／27

集合の被覆の列挙

菊地 洋右[1] 中野 眞一[2]

[1] 津山工業高等専門学校情報工学科岡山県津山市沼 624-1
kikuchi@tsuyama-ct.ac.jp
[2] 群馬大学工学部情報工学科群馬県桐生市天神町 1-5-1
nakano@cs.gunma-u.ac.jp

概要： 本文では集合 S と S の被覆 $\mathcal{C}$ が与えられたときに $\mathcal{C}$ の部分集合で S の被覆であるものを効率的に列挙することを考える。この列挙においては逆探索法を用いる。本文で提案するデータ構造と手続きにより集合の被覆は $|S||C|$ 時間の初期化の後、1被覆あたり $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$ 時間で列挙できる。ここで C_i は $\mathcal{C}$ の要素とする。

On the generation of set cover

Yosuke Kikuchi[1] Shin-ichi Nakano[2]

[1] Department of Electronics and Computer Science
Tsuyama National College of Technology
kikuchi@tsuyama-ct.ac.jp
[2] Department of Computer Science, Gunma University
nakano@cs.gunma-u.ac.jp

Abstract : Let S be a set and $\mathcal{C}$ the cover of S. This paper deals with a generation algorithm for all set covers of S that are sub set of $\mathcal{C}$. This generation algorithm emplys reverse search method. After $O(|S||C|)$ time preprocessing the algorithm generates each set cover in $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$, where $C_i\in\mathcal{C}$.

1 はじめに

集合 S とその部分集合族 $\mathcal{C}$ が与えられ、$S\subseteq\bigcup_{C\in\mathcal{C}}C$ であるとき $\mathcal{C}$ は S の被覆という。本文では集合 S と S の被覆 $\mathcal{C}$ が与えられたときに $\mathcal{C}$ の部分集合で S の被覆であるものを列挙することを考える。S の被覆の中で、その被覆の濃度が最小のものを最小被覆という。集合 S とその部分集合族 $\mathcal{C}$ に対して最小被覆を求める問題は最小被覆問題とよばれ NP-完全な問題として知られている。また、極小被覆を列挙する問題は #P-完全な問題であることも知られている [3]。例えば集合 $S=\{a,b,c,d,e,f\}$, S の部分集合族 $\mathcal{C}$ として $\{C_1=\{a,b,c\},C_2=\{a,c,d\},C_3=\{a,e\},C_4=\{b,c,d\},C_5=\{e\},C_6=\{c,d,e,f\},C_7=\{c,f\}\}$ が与えられているとする。このとき、$S=\bigcup_{i=1}^{7}C_i$ より $\mathcal{C}$ は S の被覆である。また、$\mathcal{C}'=\{C_1,C_3,C_5,C_6,C_7\}$ も S の被覆となっており、最小被覆は $\mathcal{C}''=\{C_1,C_6\}$ である。

本文は、集合 S とその部分集合族 $\mathcal{C}=\{C_1,C_2,...,C_k\}$ が与えられたとき、$\mathcal{C}$ の部分集合で S の被覆であるものの全てを、重複なく、抜けなく、高速に列挙するアルゴリズムを与える。本文

のアルゴリズムは、$O(|S||\mathcal{C}|)$ 時間の前処理の後、すべての被覆を1個あたり、$O(\max_{1\leq i\leq|\mathcal{C}|} C_i)$ 時間で列挙する。

本文の主なアイディアは、次の通りである。まず、$\mathcal{C}$ の部分集合で S の被覆であるものの全ての間に、木構造を定義する。一般にこの木は膨大なサイズとなり、全体を生成するためには、膨大な時間と膨大な記憶領域を必要とする。そこで、この木全体を生成せずに、上手に巡回することにより、高速に、被覆の列挙を行なう。このような手法は、逆探索法 [1, 2] を工夫したものであり、同様の手法により、多くの列挙問題を解く高速なアルゴリズムが幾つも知られている [4]。

本文で定義した木構造を図4に示す。3章で説明するように、被覆 $\mathcal{C}_p$ が被覆 $\mathcal{C}_c$ の親であるとき、$\mathcal{C}_c \subset \mathcal{C}_p$ かつ $|\mathcal{C}_c| = |\mathcal{C}_p| - 1$ である。しかし、$\mathcal{C}_c \subset \mathcal{C}_p$ かつ $|\mathcal{C}_c| = |\mathcal{C}_p| - 1$ であっても被覆 $\mathcal{C}_p$ は被覆 $\mathcal{C}_c$ の親であるとは限らないことに注意しよう。(そのような $\mathcal{C}_p$ は複数存在するかもしれないので、親を唯一に定めるために工夫が必要である。) また、3章では、各被覆から、その被覆の全ての子の被覆を高速に列挙する方法を説明する。本文のアルゴリズムは、これを再帰的に行なうことにより、全ての被覆を高速に列挙する。

2 データ構造

逆探索を行うためにはグラフ G のある頂点 v が与えられたときに次に訪れる頂点を v のデータから決める必要がある。そのためのデータ構造について述べる。本論文では被覆を2次元配列、被覆の要素を格納したリスト構造、S の各要素に対してそれを含む $\mathcal{C}$ の要素を格納した1次元配列の3つのデータで表現する。2次元配列だけでも被覆を表現できるが、列挙の効率化のためにリスト構造, 1次元配列を用いる。

各被覆のデータ構造の2次元配列は集合 S の各要素が各行に対応し、集合族 $\mathcal{C}$ の各要素が各列に対応する。この配列を A^2_{SC} とする。A^2_{SC} において集合 S の要素 x が $\mathcal{C}$ の要素 C_i に含まれるとき x 行 i 列の値は1、含まれないときは0である。また、集合 S の要素が $\mathcal{C}$ の要素に含まれる回数を2次元配列の最後の列に格納する。図1の下は1章で例に挙げた S と $\mathcal{C}$ に対する2次元配列を表したものである。図中には示していないが、この2次元配列では列ごとに1の要素はポインタで連結されている。C_3 列では第1行目の要素の1と第5行目の要素の1がポインタで連結されている。2次元配列の右には S の要素が $\mathcal{C}$ の要素に含まれる回数を管理するデータがある。

図1の上段にリスト構造 L_{SC} を示す。被覆を表すリストにおいてリンクは、2種類のリンクがある。一つは要素を順次連結したものであり、もう一つはその被覆からその要素を取り除いても被覆であるような要素を順に連結している。

さらに A^2_{SC} の右に S の各要素を含む $\mathcal{C}$ の要素をポインタで連結した1次元配列 A^1_{SC} を置いている。このポインタでは L_{SC} と同様に取り除くことができる最小の添字をもつ要素に A^2_{SC} からポインタで連結されている。この例では a は C_1, C_2, C_3 に含まれているので C_1, C_2, C_3 がポインタで連結されている。

この図1の場合、2種類のリンクは同一のものとなっている。

先の例では S の全ての要素が2つ以上の $\mathcal{C}$ の要素に含まれていた。ここで、被覆の2次元配列が表1であった場合は図2になる。表1では b は C_4 のみに含まれている。よって、表1の被覆を親とする全ての被覆は C_4 を含まなければならない。つまり、C_4 を被覆から取り除くと被覆ではなくなる。そのことを表わすため、図2において C_4 の背景を斜線としている。このとき C_3 から C_5 へのリンクが張られている。被覆を表すリストにおいてリンクは、その被覆からその要素を取り除いても被覆であるような要素を順に結んでいる。

3 被覆の家系木

1章で述べた木構造をここでは家系木とよび、この家系木を用いて逆探索を行う。集合 S とその部分集合族 $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ について $\mathcal{C}$ は S の被覆であり、これを根被覆とよぶ。また、被覆

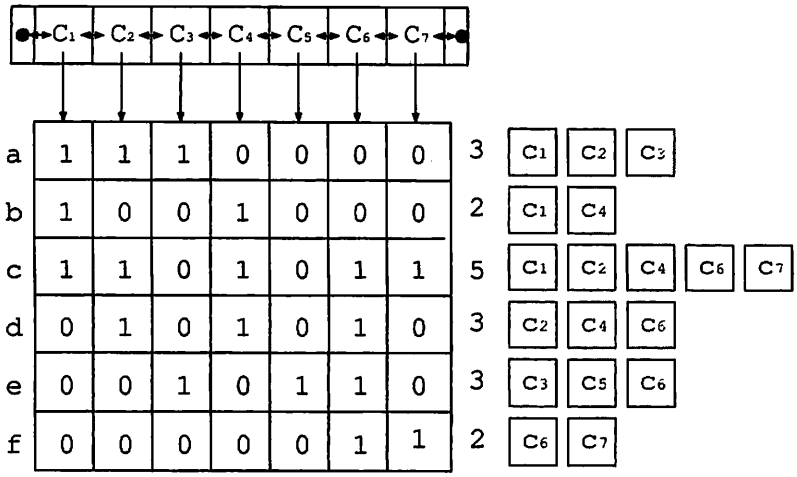

Figure 1: 被覆を表すリストと 2 次元配列

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	
a	1	1	1	0	0	0	0	3
b	0	0	0	1	0	0	0	1
c	1	1	0	1	0	1	1	5
d	0	1	0	1	0	1	0	3
e	0	0	1	0	1	1	0	3
f	0	0	0	0	0	1	1	2

Table 1: 被覆の例

$\mathcal{C}' = \{C'_1, C'_2, \ldots, C'_k\}$ に対して C_m を集合 $\mathcal{C} \backslash \mathcal{C}'$ の中で最大の添字をもつ要素とする。$\mathcal{C}_p = \mathcal{C}' \cup \{C_m\}$ を構成する。この $\mathcal{C}_p$ を $\mathcal{C}'$ の親被覆とよぶ。$\mathcal{C}' \neq \mathcal{C}$ であるならば $\mathcal{C} \backslash \mathcal{C}' \neq \emptyset$ となるので必ず $\mathcal{C}'$ に対して $\mathcal{C}_p$ は存在し、一意に定まる。1 章の例 $\mathcal{C}' = \{C_1, C_3, C_5, C_6, C_7\}$ では $C_m = C_4$ となるので、親被覆は $\mathcal{C}_p = \{C_1, C_3, C_4, C_5, C_6, C_7\}$ である。

$\mathcal{C}'$ に対して $\mathcal{C}_c = \mathcal{C}' \setminus C_l$ を構成する。ここで、C_l は $\mathcal{C} \setminus \mathcal{C}'$ の中で最大の添字をもつ要素 C_m より大きい添字をもつ要素とする。この $\mathcal{C}_c$ が被覆であるとき、$\mathcal{C}_c$ を $\mathcal{C}'$ の子被覆とよぶ。$\mathcal{C}' = \{C_1, C_3, C_5, C_6, C_7\}$ の場合、$C_m = C_4$ であるから $\mathcal{C}'$ の子被覆の候補として $\{C_1, C_3, C_6, C_7\}$, $\{C_1, C_3, C_5, C_7\}$, $\{C_1, C_3, C_5, C_6\}$ が挙げられる。このうち、$\{C_1, C_3, C_6, C_7\}$, $\{C_1, C_3, C_5, C_6\}$ は S の被覆となっているので $\mathcal{C}'$ の子被覆である。一方、$\{C_1, C_3, C_5, C_7\}$ は $d \notin C_1 \cup C_3 \cup C_5 \cup C_7$ となり、S を被覆していないので子被覆ではない。このような子被覆とはならない子被覆の候補を生成しないためのデータ構造が 2 章で述べたものである。

C_1 が C'_1 の親被覆であるならば C'_1 は C_1 の子被覆である。また C'_1 が C_1 の子被覆であるならば C_1 は C'_1 の親被覆である。この 親被覆, 子被覆の関係を逆探索に用いる有向辺とする。さらにこの有向辺により被覆の木構造を構成でき、この木構造の根は根被覆 $\mathcal{C}$ である。この木を家系木とよぶ。1 章の例の家系木を図 4 に示す。

集合 S と S の部分集合族 $\mathcal{C}$ において根被覆 $\mathcal{C}$ から上記で定義した親子関係をもとに全ての被覆を生成することを考える。ある被覆 $\mathcal{C}_0 = \{C_1, C_2, \ldots, C_k\}$ において C_m は集合 $\mathcal{C} \setminus \mathcal{C}_0$ の中で最大の添字をもつ要素と定義した。さらに $\mathcal{C}_0$ において C_m より大きい添字をもち、かつその要素を $\mathcal{C}_0$ から取り除いても被覆であるような要素の集合を $\mathcal{C}_0(m) = \{C_1(m), C_2(m), \ldots, C_l(m)\}$ とす

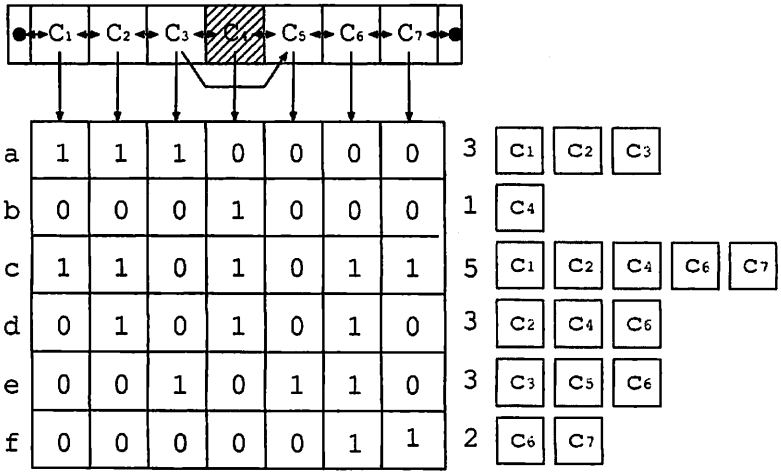

Figure 2: 表 1 に対するデータ構造

る。図 4 の $\{C_1, C_3, C_6, C_7\}$ の場合、$C_m = C_5$ であり、$C_0(1) = \{C_7\}$ である。このときのデータ構造を図 3 に示す。リストにおいて、その要素をリストから取り除くと被覆でなくなる要素に対して背景をドットで表わす。この図のリストでは C_1, C_6 の背景はドットとなっている。先頭から C_6 へのポインタは C_{m+1} へのポインタである。

	C_1	C_2	C_3	C_4	C_5	C_6	C_7	
a	1	1	1	0	0	0	0	2
b	1	0	0	1	0	0	0	1
c	1	1	0	1	0	1	1	3
d	0	1	0	1	0	1	0	1
e	0	0	1	0	1	1	0	2
f	0	0	0	0	0	1	1	2

Table 2: $C_0 = \{C_1, C_3, C_6, C_7\}$ をあらわす表

これらのデータ構造を用いて与えられた被覆 C_0 から C_0 の子被覆を生成する手続きは以下のようになる。

Procedure find_all_child_cover
input cover C_0
for(i=1;i $\leq$ l; i++){
  generate $C_0 \setminus \{C_i(m)\}$
}

上記の **Procedure** find_all_child_cover において **generate** $C_0 \setminus \{C_i(m)\}$ では 2 次元配列 $A^2_{SC_0}$ とリスト L_{SC_0} の更新を行う必要がある。C_0 から C_i を取り除くとする。L_{SC_0} でポインタをたどり C_i を定数時間でみつけることができる。L_{SC_0} の C_i から $A^2_{SC_0}$ の C_i 列の最初の 1 をポインタでみつけ、$A^2_{SC_0}$ の最後の列の値を 1 減らし、$A^1_{SC_0}$ の配列から C_i を取り除く。$A^2_{SC_0}$ の最後の列の値と $A^1_{SC_0}$ の配列をもとに L_{SC_0} のポインタの更新を行う。これらの作業は定数時間で行え、C_i の要素全てに行うので $O(|C_i|)$ 時間かかる。よって、2 次元配列の更新に $O(\max_{1 \leq i \leq |C|} |C_i|)$ 時間、リストの

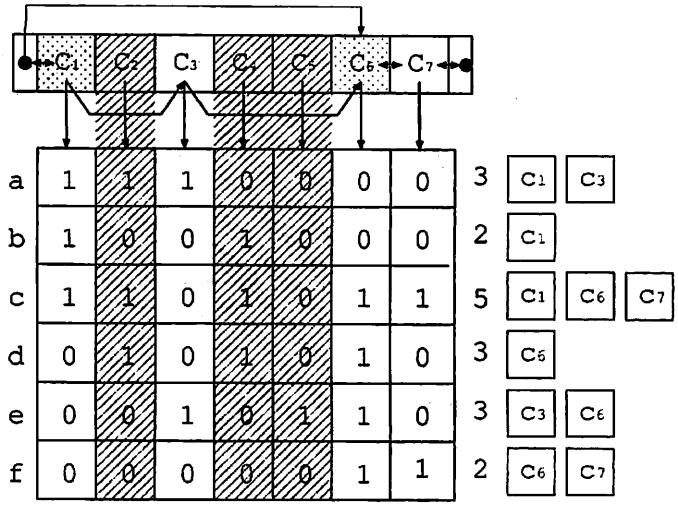

Figure 3: $\mathcal{C}_0 = \{C_1, C_3, C_6, C_7\}$ を表すデータ構造

更新は2次元配列の更新のたびに定数時間かかるので $\mathcal{C}_0$ の子被覆を一つあたり $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$ で生成できる。

ある被覆 $\mathcal{C}_0 = \{C_1, C_2, \ldots, C_k\}$ が与えられたとき、$\mathcal{C}_0$ の親被覆の2次元配列を得るためには $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$ 時間かかる。

Theorem 3.1 集合 S とその部分集合族 $\mathcal{C}$ が与えられたとき、上記のデータ構造と手続きにより集合の被覆は $O(|S||\mathcal{C}|)$ 時間の初期化の後、1被覆あたり $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$ 時間で列挙できる。

さらに上記のデータ構造では S の要素が被覆に含まれる回数を管理している。このデータ構造を用いることで S の各要素について少なくとも k 個の $\mathcal{C}$ の要素に被覆されるという条件の問題についても同様に列挙できる。

4 連続集合の被覆の列挙

前章において集合 S が離散集合の場合を扱った。本章では集合 S が1つの連続する区間である場合について考える。

連続集合では離散集合の場合のように1被覆あたり $O(\max_{1\leq i\leq|\mathcal{C}|}|C_i|)$ 時間での列挙はできない。本文では最初に与えられる被覆 $\mathcal{C}$ の細分 $\mathcal{I}$ を考え、その細分の被覆を列挙することで連続集合の被覆を列挙する。この方法により集合 S が1つの連続する区間の場合は高速に列挙できる。以下では細分の構成について述べる。区間の集合 $\mathcal{C} = \{C_1, C_2, \ldots C_n\}$ で、$\mathcal{C}$ は S の被覆となっている。また、各 $C_j \in \mathcal{C}$ は連続する一つの区間とする。$\mathcal{C}$ の各要素の上限、下限なる数列

$$a_1 \leq a_2 \leq \cdots \leq a_{2n}$$

を作る。C_j は開区間、閉区間、半開区間のいずれかであるので、a_i が C_j の上限または下限であったとすると、a_i は次の4つに分類できる。

1. C_j の上限であるが最大元ではない
2. C_j の最小元

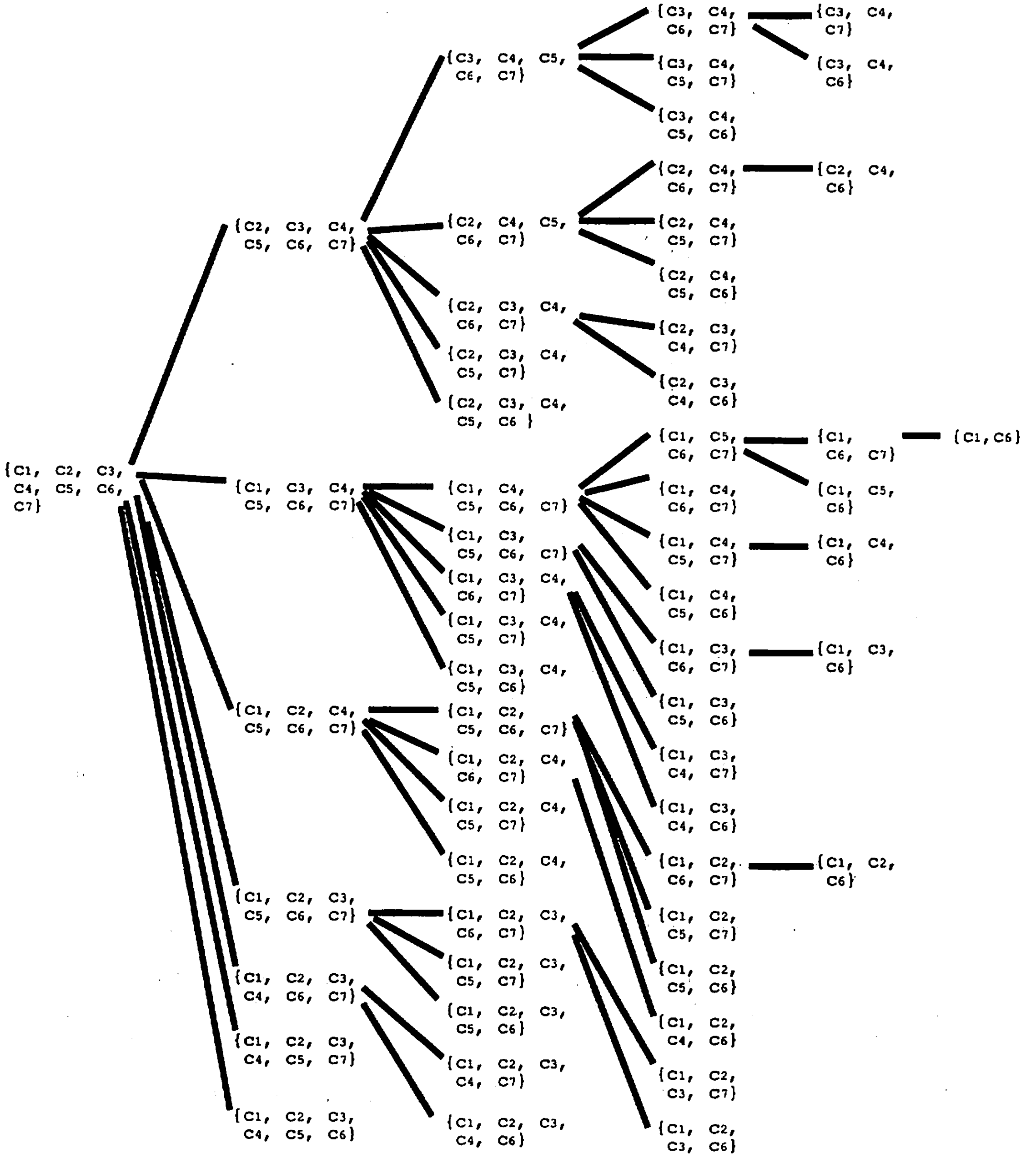

Figure 4: Family tree of covers

3. C_j の最大元

4. C_j の下限であるが最小元ではない

$a_i = a_{i+1}$ であるとき a_i, a_{i+1} が上の分類で同じであるときは a_i と a_{i+1} を同一視する。a_i,a_{i+1} が上の分類で異なるときは分類されている番号の小さい順に整列する。これにより得られた数列

$$a'_1 \leq a'_2 \leq \cdots \leq a'_m$$

に対して a'_i から a'_{i+1} の区間を次のように構成する。
$a'_i < a'_{i+1}$ のとき

(i) a'_i が分類 1 のとき a'_i が最小元である区間

(ii) a'_i が分類 2 のとき a'_i が a'_i が最小元である区間

(iii) a'_i が分類 3 のとき a'_i が下限だが最小元でない区間

(iv) a'_i が分類 4 のとき a'_i が下限だが最小元でない区間

(v) a'_{i+1} が分類 1 のとき a'_{i+1} が上限だが最大元でない区間

(vi) a'_{i+1} が分類 2 のとき a'_{i+1} が上限だが最大元でない区間

(vii) a'_{i+1} が分類 3 のとき a'_{i+1} が最大元である区間

(viii) a'_{i+1} が分類 4 のとき a'_{i+1} が最大元である区間

上記 (i) から (viii) にしたがって a_i から a_{i+1} の区間を構成する。
$a'_i = a'_{i+1}$ のとき

(i) a'_i が分類 1,a'_{i+1} が分類 3 のとき

(ii) a'_i が分類 2,a'_{i+1} が分類 3 のとき

(iii) a'_i+ が分類 2,a'_{i+1} が分類 4 のとき

上記 (i) から (iii) のいずれかを満たす場合のみ a_i のみからなる区間を構成する。

$\mathcal{C} = \{(0,1],[1,2],(2,5],(3,4),(3,5]\}$ の場合、得られる区間は $(0,1)$, $\{1\}$, $(1,2]$, $(2,3]$, $(3,4)$, $[4,5]$ となる。この上限と下限の数列から得られた区間の集合を $\mathcal{I}$ とすると、$\mathcal{I}$ は $\mathcal{C}$ の細分となっており、S の被覆になっている。

よって、$\mathcal{I}$ を集合と考えると、$\mathcal{C}$ は $\mathcal{I}$ の被覆であるので $\mathcal{I}$ と $\mathcal{C}$ について離散集合の場合と同様に被覆を列挙できる。また、$|\mathcal{I}|$ は高々 $2|\mathcal{C}|-1$ であることから次が言える。

Theorem 4.1 区間 S とその部分区間の集合 $\mathcal{C}$ が与えられたとき、S の被覆は $|S||C|$ 時間の初期化の後、1 被覆あたり $O(|\mathcal{C}|)$ 時間で列挙できる。

5 まとめ

本文では集合被覆の列挙について考察した。この列挙において逆探索法を用いた。集合 S が区間の場合は細分により離散集合として列挙を考えた。しかし、S が区間の場合は区間グラフなどを用いてより効率的な列挙が考えられる。また、一般の連続集合での列挙についても考えられる。与えられた集合が区間であった場合は、その被覆の細分は被覆の定数倍であったが、一般の場合では定数倍とはならない。そのため細分による方法では効率的に列挙できないと考えられる。これらについては今後の課題としたい。

References

[1] D. Avis and K. Fukuda, Reverse Search for Enumeration, *Discrete Applied math* 65, 21–46, 1996.

[2] 福田公明, 逆探索とその応用, 離散構造とアルゴリズム II, 近代科学社, 47–78, 1993.

[3] M. L. Fredman and L. Khachiyan, On the complexity of dualization of monotone disjunctive normal forms, *Journal of Algorithms* 21, 618–628, 1996.

[4] S. Nakano, Efficient generation of triconnected plane triangulations, *Computational Geometry* 27, 109–122, 2004.