修士論文

制約を満たすグラフの高速列挙

山崎　一明

主指導教員　上原 隆平

北陸先端科学技術大学院大学
先端科学技術研究科
（情報科学）

令和 6 年 8 月

Abstract

In the research field of graph algorithms, there are several researches which achieve efficiently solving problems that it is difficult to solve for general graphs by restricting input graph. In the situation to design/implement such algorithms, it is useful for testing algorithms if there is a catalog of elements of the target graph class. Furthermore, since the graphs which are isomorphic are essentially the same, it is ideal if those catalogs contain isomorphic graph exactly once. On the other hand, since it is widely known that graph isomorphism problem is GI-complete for many graph classes, it is thought that it is difficult to solve efficiently. However, there are exist some graph classes that graph isomorphism problem can be solved efficiently for whose elements.

From the viewpoint of graph enumeration, there are several preceding researches which achieve efficient enumeration by restricting the target graph class. Two of these researches, they proposed algorithms which enumerate proper interval graphs and bipartite permutation graphs efficiently each. However, since these researches use characteristics of the target graph classes to achieve efficiency, it is difficult to extend the algorithms for the other graph classes. Then, in this research, we focus on the graph classes that we can solve graph recognition problem and graph isomorphism problem for its elements, and propose the general framework to construct algorithms which efficiently enumerate elements of the target graph class with certain number of vertices. For the graph classes that we can solve graph recognition problem and graph isomorphism problem efficiently, there are many cases that there is tree structure which corresponds to a graph in the graph class, and then, we can solve graph isomorphism problem efficiently by solving tree isomorphism problem instead of general.

As the graph classes that we can efficiently solve graph recognition problem and graph isomorphism problem, interval graphs and permutation graphs are known. There are many applicative researches for these graph classes, however, there are never been graph catalogs for them.

In this research, first, we propose the general framework to construct enumeration algorithm for the graph class that we can efficiently solve graph recognition problem and graph isomorphism problem. Next, as concrete applications of the framework, we construct enumeration algorithms for interval graphs and permutation graphs. And finally, we implement the algorithms on a real computer and show the experimental results.

目 次

図 目 次

表 目 次

第1章　はじめに

グラフアルゴリズムの分野において，一般のグラフに対しては難しい問題であっても，入力となるグラフのクラスを制限することで効率性を達成する研究がなされている [1, 2]．そういったアルゴリズムを設計・実装してテストする状況を考ええると，対象とするグラフクラスの元の一覧があればテストデータとして有用であると考えられる．また，同型なグラフ同士は本質的に同じものであるため，同型なグラフを取り除いた一覧があると望ましい．一方で，グラフの同型性判定問題は多くのグラフクラスで GI-完全であることが知られており，効率的に解くことは難しいと考えられている．しかし，同型性判定問題を効率的に解くことができるグラフクラスも存在している．

認識問題と同型性判定問題を効率的に解くことができるグラフクラスとして，区間グラフと置換グラフが知られている [3, 4]．これらのグラフクラスについては多くの応用が研究されており，また，グラフ理論・グラフアルゴリズムの観点においても基本的なグラフクラスである．従って，これらのグラフクラスに関してアルゴリズム設計に寄与する性質が多く発見され，また，それらを利用して効率性を達成するアルゴリズムが多く提案されてきた [5, 6, 7]．実践的な観点からは，こうしたアルゴリズムを設計・実装する際には前述のように，そのテストのために対象とするグラフクラスに属するグラフの多くの実例が必要となる．従って，主要なグラフクラスに対しては，効率的な列挙アルゴリズムが求められる [8]．しかし，グラフの具体的な一覧は提供されていなかった．

グラフ列挙の観点では，対象を特定のグラフクラスに制限して効率性を達成した先行研究がある [9, 10]．これらの研究では，proper な区間グラフおよび二部置換グラフを効率的に列挙するアルゴリズムが示されている．しかし，これらのアルゴリズムは対象のグラフクラスがもつ特徴を利用して効率性を達成しており，他のグラフクラスへの拡張が難しい．そこで本研究では，認識問題と同型性判定問題が効率的に解けるグラフクラスに着目し，そのクラスの元の内で特定の頂点数をもつものを効率的に列挙するアルゴリズムを構築する一般性のあるフレームワークを提案する．認識問題と同型性判定問題を効率的に解けるグラフクラスでは，グラフに対応する木構造が存在することが多く，木構造に対する同型性判定問題を解くことで対応するグラフの同型性判定問題を効率的に解くことができる [11]．

本研究ではまず，認識問題と同型性判定問題が効率的に解けるグラフクラスを対象として，その元を効率的に列挙するアルゴリズムを構成する一般性のあるフレームワークを提案する．続いて，フレームワークを区間グラフと置換グラフに

対して適用し，実際に列挙アルゴリズムを構築する．そして，実際の計算機上にアルゴリズムを実装した実験結果を示す．

第2章　準備

2.1　グラフ

$V = \{1, 2, \ldots, n\}$ とする．無向グラフ $G = (V, E)$ とは，**頂点** $(vertex)$ 集合 V と**辺** $(edge)$ 集合 E からなる順序対である．ここで，E の元は V の元からなる非順序対 $\{u, v\}$ $(u, v \in V)$ である．$\forall\{u, v\} \in E, u \neq v$ のとき G は**単純** $(simple)$ であるという．単純かつ無向なグラフを**単純無向グラフ**と呼ぶ．

有向グラフ $G = (V, E)$ とは，頂点集合 V と辺集合 E からなる順序対であるが，E の元は V の元からなる順序対である点が無向グラフと異なる．本研究では主に単純無向グラフを扱うため，断りなくグラフと書いた場合は単純無向グラフを指す．

グラフ $G = (V, E)$ と非順序対 e $(e \notin E)$ について，e を G の辺集合に追加したグラフ $(V, E \cup \{e\})$ を $G + e$ と表記する．同様に，$G = (V, E)$ とその辺 e $(e \in E)$ について，G の辺集合から e を取り除いたグラフ $(V, E \setminus \{e\})$ を $G - e$ と表記する．

頂点集合 V について，$(V, \{\{u, v\} \mid u, v \in V, u \neq v\})$ で定まるグラフを**完全グラフ**という．頂点数が n である完全グラフを記号 K_n で表す．

グラフ $G = (V, E)$ について，頂点集合 V の元の列 $\mathrm{P} = \langle p_0, p_1, \ldots, p_k \rangle$ であって，$\forall i \in \{0, 1, \ldots, k-1\}, \{p_i, p_{i+1}\} \in E$ を満たすとき，P を**パス** $(path)$ という．p_0, p_k をパス P の**端点**といい，k をパス P の**長さ**という．2 頂点 $u, v \in V$ について，u, v 間の**距離** $d(u, v)$ を，u, v を端点とする任意のパスの内で，長さが最小なものの長さで定義する．

グラフ $G = (V, E)$ について，任意の 2 頂点 $u, v \in V$ に対して u, v を端点とするパスが存在するとき，G は**連結**であるという．連結で $|V| - 1 = |E|$ であるようなグラフを**木**という

木 $T = (V, E)$ について，**根** $(root)$ と呼ばれる頂点 $r \in V$ を固定する．更に，E の元 $\{u, v\}$ を順序対 (u, v) $(d(r, u) < d(r, v))$ で置き換えて得られる有向グラフ $T' = (V, E')$ を**根付き木**という．

2 つのグラフ $G_1 = (V_1, E_1), G_2 = (V_2, E_2)$ について，全単射 $\phi\colon V_1 \to V_2$ が存在し，

$$\forall u, v \in V_1, \{u, v\} \in E_1 \iff \{\phi(u), \phi(v)\} \in E_2$$

を満たすとき，G_1 と G_2 は**同型** $(isomorphic)$ であるといい，$G_1 \simeq G_2$ で表す．

2.1.1 区間グラフ

グラフ $G = (V, E)$ について，$\mathbb{R}$ 上の区間集合 $\mathcal{I} = \{I_u = [l, r] \mid u \in V, l, r \in \mathbb{R}, l \leq r\}$ であって，

$$\{u, v\} \in E \iff I_u \cap I_v \neq \emptyset$$

を満たすものが存在するとき，G は**区間グラフ** (*interval graph*) であるといい，I を G の**区間表現** (*interval representation*) という．

2.1.2 置換グラフ

グラフ $G = (V, E)$ について，V 上の置換 π_1, π_2 であって，

$$\{u, v\} \in E \iff (\pi_1(u) - \pi_1(v))(\pi_2(u) - \pi_2(v)) < 0$$

を満たすものが存在するとき，G は**置換グラフ** (*permutation graph*) であるという．置換グラフは，並行する 2 つの線分 L_1, L_2 と異なる線分上の点同士を結ぶ $|V|$ 本の線分に対応付けることができる．具体的に述べる．L_1, L_2 を水平に描画するとして，L_1, L_2 それぞれの上に相異なる n 個の点を配置し，それぞれ左から順に点 $1, 2, \ldots, n$ と呼ぶ．更に，各頂点 $u \in V$ に対し，L_1 上の点 $\pi_1(u)$ と L_2 上の点 $\pi_2(u)$ を結ぶ線分 l_u を引く．このとき，2 頂点 $u, v \in V$ について $\{u, v\} \in E$ であることと，2 線分 l_u, l_v が交点をもつことは同値である．これは，$\pi_1(u), \pi_1(v)$ の大小関係と $\pi_2(u), \pi_2(v)$ の大小関係が反転していることから確かめられる．こうして得られる表現を置換グラフの**線分表現**と呼ぶ．今，置換グラフを 2 つの置換 π_1, π_2 で特徴付けているが，頂点の番号を π_1 が恒等置換 $\begin{pmatrix} 1 & 2 & \ldots & n \\ 1 & 2 & \ldots & n \end{pmatrix}$ となるように付け替えることができる．この付け替えを行った後の π_2 を置換グラフを表す置換であるとする．

2.1.3 グラフクラス

特定の条件を満たすグラフをすべて集めてできる集合を**グラフクラス**という．例えば,「すべての区間グラフからなる集合」「すべての置換グラフからなる集合」などである．特定のグラフクラスを記号 $\mathcal{C}$ で表す.

グラフ G とグラフクラス $\mathcal{C}$ について，$G \in \mathcal{C}$ かどうかを判定する問題をグラフの **認識問題** (*recognition problem*) という.

2.2 列挙問題

列挙問題とは，指定された集合のすべての要素を重複なく出力する問題であり，列挙問題を解くアルゴリズムを**列挙アルゴリズム**という [12].

本研究では，頂点数が n であるようなグラフ（の部分集合）を列挙することに着目するが，一般には，出力するべきグラフの個数 N が n の多項式で抑えられるとは限らない．よって，以下の 3 つに着目する．

- アルゴリズムの実行開始から，最初の出力が得られるまでの時間
- $i \in 1, 2, \ldots, N-1$ について，i 番目の出力が得られてから $i+1$ 番目の出力が得られるまでの時間
- N 個目の出力が得られてから，アルゴリズムが終了するまでの時間

これらすべてが多項式時間で抑えられているとき，その列挙アルゴリズムは**最悪時多項式時間遅延アルゴリズム**であるという．

第3章　列挙フレームワーク

指定されたグラフクラス $\mathcal{C}$ について，$\mathcal{C}$ に属するグラフに関するグラフ認識問題とグラフ同型性判定問題を多項式時間で解くことができると仮定する．$\mathcal{C}$ に属する n のグラフの認識問題を解く時間計算量を $\mathrm{Rec}(n)$ と書き，$\mathcal{C}$ に属する n 頂点のグラフ $G_1, G_2 \in \mathcal{C}$ についての同型性判定問題を解く時間計算量を $\mathrm{Iso}(n)$ と書く．

$\mathcal{C}$ に属する任意のグラフ $G \in \mathcal{C}$ の**標準形**を定義する．まず，$\mathcal{C}$ 中の同型なグラフについて，全順序 $<$ が定義可能であるとする．すなわち，グラフ $G_1 = (V, E_1), G_2 = (V, E_2), G_3 = (V, E_3) \in \mathcal{C}$ $(G_1 \sim G_2 \sim G_3)$ について，

- $G_1 \neq G_2$ のとき，$G_1 < G_2$ または $G_2 < G_1$ のいずれかが（排他的に）成り立つ．
- $G_1 < G_2$ と $G_2 < G_3$ がともに成り立つとき，$G_1 < G_3$ が成り立つ．

このとき，$\{G' \mid G' \in \mathcal{C}, G' \sim G\}$ の**最小元**が一意的に定まる．これを G の標準形と呼ぶ．

本研究の目的は，$\mathcal{C}$ に属するグラフの内，($\mathcal{C}$ に属するいずれかのグラフの）標準形であるものすべてを列挙することであると言える．多項式時間遅延での列挙のために．次の性質を満たすように標準形を定める：

標準形の性質: 任意のグラフ $G \in \mathcal{C}$ について，G の標準形を多項式時間で計算できる．すなわち，多項式時間のコストをかけることで，取り扱うグラフを標準形に限定できる．

本研究では，指定されたグラフクラス $\mathcal{C}$ の要素を列挙するために逆探索法 [13] を用いる．逆探索法を適用するために，$\mathcal{C}$ の要素に親子関係を定義することで $\mathcal{C}$ 上の有向木を（仮想的に）構成する．より正確には，まず，木の根となるノード G_R を固定する．本研究では，$G_R = K_n$ とする．K_n は区間グラフでありかつ置換グラフでもある．各グラフ $G' \in \mathcal{C} \setminus \{G_R\}$ について，G' の親 G を一意的に定めることができ，多項式時間で計算できるとする[1]．この親子関係により，G_R を根とする $\mathcal{C}$ 上の有向全域木が定まる．この木を**家族木**と呼び，$\mathcal{T}$ と書く．列挙は，$\mathcal{T}$ の辺を辿ることにより全頂点を訪問することに対応する．

グラフ $G' \in \mathcal{C} \setminus \{G_R\}$ について，**基本操作**を適用することで $\mathcal{T}$ における G' の親を定める．本研究では，基本操作として「辺の追加」を用いる[2]．$G + e \in \mathcal{C}$ と

[1] 具体的なグラフクラスについて論じるときにそのように定義する．

[2] 区間グラフおよび置換グラフに適用する際に辺の追加が妥当か否かについては各章で述べる．

なるような e が複数存在する場合，その内ひとつを一意的に決定できる必要がある．そのために，操作は次の性質をもつ必要がある：

operational peoperty: G_R を $\mathcal{T}$ の根として，G' を $\mathcal{C} \setminus \{G_R\}$ に属する任意のグラフとする．このとき，G' に対して一回の基本操作を行うことで得られるグラフ G であって，$G \in \mathcal{C}$ であるようなものが少なくともひとつ存在する．加えて，そのようなグラフが複数ある場合は，その内ひとつを一意的に決定でき，かつ多項式時間で計算できる．

この性質を満たすように基本演算を定めれば，G' $(\in \mathcal{C} \setminus \{G_R\})$ の親 G を多項式時間で得ることができる．

逆探索法に基づくグラフ列挙アルゴリズムのフレームワークについて説明する．逆探索法は，$\mathcal{T}$ を $G_R \in \mathcal{T}$ を始点とする幅優先探索 (Breadth First Search; BFS) あるいは深さ優先探索 (Depth First Search; DFS) [3]を行うが，訪問先ノードの決定方法に特徴がある．逆探索法では，グラフ G の（$\mathcal{T}$ における）子（未来の訪問先）を決定するために，**子候補**の集合 $S(G)$ を生成する．そのためにまず，$G = (V, E)$ に対し基本操作の逆操作（辺の削除）を行って得られるグラフすべてからなる集合 $S'(G) = \{G - e \mid e \in E\}$ を得る．$S'(G)$ には G の子がすべて含まれていることは構成法から言えるが，そうでないものも含まれている可能性がある．漏れも重複も無く列挙するためには，真に子であるもののみをすべて抽出し，処理する必要がある．処理してはいけないケースとして，3 つのケースがある：

1. $G - e \notin \mathcal{C}$ である場合．この場合は，このグラフに対する処理をスキップする．

2. G に対する基本操作の逆操作によって，複数の同型なグラフが生成される場合．この場合は，標準形でないグラフをすべて取り除く．

3. $G' \in S'(G)$ が G の子ではない場合．この場合は，G' の親が G であるか（i.e., G の子候補の親が G 自身であるか）をチェックし，同型でないならば処理をスキップする．

上記 1., 2. によって得られる集合を $S(G)$ とする．

列挙アルゴリズムの内，列挙対象のグラフクラスに依存しない部分を抽出してできるフレームワークを以下に示す．

[3]BFS になるか DFS になるかは，未訪問ノードを管理するデータ構造としてキューを使うかスタックを使うかによる．

Algorithm 1 逆探索法に基づくグラフ列挙アルゴリズム

Input: 非負整数 n
Output: $\mathcal{C}$ に属する頂点数 n のグラフすべて

```
S ← {G_R}
while S が空でない do
    S の要素をひとつ取り除き，取り除いたグラフを G = (V, E) とする
    G を出力する
    S' ← {G − e | e ∈ E}
    // このとき，S' は S'(G)
    for G' ∈ S' do
        if G' ∉ C then
            S' から G' を取り除く
        else
            G' の標準形を計算し，G' を置き換える
            // S' は集合なので，重複要素は取り除く
        end if
    end for
    // この時点で，S' は S(G) になっている
    for G' ∈ S' do
        Ĝ' ← G'の親
        if Ĝ' ∼ G then
            S ← S ∪ {G'}
        end if
    end for
end while
```

グラフクラス $\mathcal{C}$ および基本操作が上述の性質をもつとき，以下の主定理を得る．

定理 1. $\mathcal{C}_n = \{G \mid G = (V, E) \in \mathcal{C}, |V| = n\}$ とし，$\mathrm{poly}(n)$ を n に関する多項式関数とする．このとき，アルゴリズム *1* は $\mathcal{C}_n$ の要素を多項式時間遅延で列挙する．すなわち，アルゴリズムの実行時間は $|\mathcal{C}_n|\mathrm{poly}(n)$ で抑えられる．

Proof. 読みやすさのために，一般性を損なわず $\mathcal{T}$ の根は K_n であるとし，基本操作は「辺の追加」であるとする．

まず，$\mathcal{C}_n$ 上に定義する親子関係が正しく定義されていれば，家族木も well-defined になることを示す．K_n を根と定めたので，$\mathcal{C}_n \setminus \{K_n\}$ の元は一意的に定まる親をもつ．また，辺の追加を基本操作としたことから，操作によって辺数が必ず増加するため，親子関係は循環しない．$G \in \mathcal{C} \setminus \{K_n\}$ の親を $\mathrm{par}(G)$ で表すとし，$A = \{(\mathrm{par}(G), G) \mid G \in \mathcal{C} \setminus \{K_n\}\}$ とする．このとき，有向グラフ $\mathcal{T} = (\mathcal{C}_n, A)$ が

得られるが，$|\mathcal{C}_n| - 1 = |A|$ であることからこのグラフは K_n を根とする $\mathcal{C}$ 上の有向全域木である．よって，$\mathcal{T}$ は $\mathcal{C}_n$ の家族木である．

$\mathcal{T}$ に属するグラフ G に**レベル**を定義する．$G_r = K_n$ のレベルは 0 である．各 $i = 1, 2, \ldots$ について，グラフ G がレベル $i-1$ のグラフの子であるとき，G はレベル i である．辺追加を基本操作としているので，G がレベル i をもつことは G の辺数が $\binom{n}{2} - i$ であることと同値である．

$\mathcal{C}$ の要素が丁度一回ずつ出力されることを，レベルに関する数学的帰納法で示す．帰納法の基底ケースはレベル 0 の場合である．K_n はアルゴリズムの開始直後に $\mathcal{S}$ に挿入され，while ループの一回目の実行で取り出され，出力される．while ループの内部では，取り出されたグラフの子を $\mathcal{S}$ に追加するが，K_n はどのグラフの子でもないので，再び挿入されることはない．よって，K_n は丁度一回だけ出力される．帰納法仮定は，$\mathcal{C}$ に属するレベル i 未満の標準形であるグラフは丁度一回ずつ出力され，標準形でないグラフは出力されないことである．$i > 0$ とする．G' をレベル i の任意のグラフとする．このとき，operational property から，G' に追加できる辺がひとつ以上存在する．従って，レベル $i-1$ をもつグラフからなる空でない集合をとることができ，その要素として，G' の親 $\hat{G}'$ を含む．帰納法仮定から，$\hat{G}'$ は丁度一回出力される．その際の while ループの実行において集合 $S(\hat{G}')$ が構成されるが，これは標準形であるグラフ G'' であって $G'' \sim G'$ であるものを含む．上述の標準形の性質より，$\hat{G}'$ が G''（および G'）の親であることから，G'' は $\mathcal{S}$ に挿入される．従って，G' に同型なグラフは少なくとも一回出力される．アルゴリズムが標準形のみを出力することと，標準形の親となるグラフはただひとつであることから，アルゴリズムは同型なグラフを複数回出力しない．従って，$\mathcal{C}$ の要素は丁度一回ずつ出力される．

次に，時間計算量について示す．$\mathcal{T}$ の各要素 G が多項式時間のみを消費することを示す．G がレベル 0 のとき，G に対する処理は $\mathcal{S}$ への挿入・取り出しと出力のみであるため，成り立つ．$G \not\sim K_n$ とする．$G = (V, E)$ が $\mathcal{S}$ から取り出された後，まず G が出力される．続いて，アルゴリズムは $S(G)$ を構成する．$S(G)$ が多項式時間で構成可能であることを示せばよい．基本操作が辺の追加であったので，その逆操作である辺の削除をする方法は $O(|E|)$ 通りであり，$S(G)$ の要素数も $O(|E|)$ で抑えられる．よって，アルゴリズムが G に基本操作の逆操作を適用して得られるグラフ G' をすべて生成し $S'(G)$ を生成する部分には多項式時間しかかからない．続いて，アルゴリズムは $S'(G)$ から冗長な要素を取り除いて $S(G)$ を構成する．このとき，各 $G' \in S'(G)$ に対して次の 2 つの処理が行われる：

1. $G' \in \mathcal{C}$ かどうかをチェックし，$G' \in \mathcal{C}$ であるもののみを残す．
2. G' の標準形を計算し，G' を置き換える．このとき，標準形が既に $S'(G)$ に含まれているなら置き換える代わりに G' を削除する．

1. では $S'(G)$ の各要素について認識問題を解くので，$O(|E|\mathrm{Rec}(n))$ 時間がかかる．2. では $S'(G)$ の各要素 G_1 の標準形について，$S'(G)$ の各要素 G_2 との同

型性判定問題を解く．同型性判定問題を解く回数は $O(\binom{|E|}{2})$ で抑えられるので，$O(\binom{|E|}{2}\mathrm{Iso}(n))$ 時間がかかる．仮定から，1., 2. のどちらも多項式時間である．続いて，各 $G' \in S(G)$ についてその親 $\hat{G}'$ を計算する．G' は G から辺を削除することで得られるグラフなので，G' の辺数は $|E|-1$ であり，G' に追加できる辺の候補は $\binom{n}{2} - |E| + 1$ 本である．それらの内 G' の親 $\hat{G}'$ を生成するものを選択することは，operational property から多項式時間で可能である．次に $\hat{G}' \sim G$ であるかをチェックするが，これは $\mathrm{Iso}(n)$ 時間で完了する．総じて，$\mathcal{C}_n$ の各要素あたりにアルゴリズムが消費する時間は多項式時間である． □

定理 1 により，いくつかのグラフクラスについて，その要素を多項式時間遅延で列挙するアルゴリズムを構成することができる．しかし，具体的な計算量は対象とするグラフクラスによって実装が変化する部分の実装に依存する．本研究では，代表的なグラフクラスである区間グラフと置換グラフについて，列挙アルゴリズムを構成する．

第4章　区間グラフの列挙

n 頂点の区間グラフを列挙するアルゴリズムについて論じる．本章では，$\mathcal{C}$ を n 頂点のすべての区間グラフからなる集合とする．$G_R = K_n$ とし，$\mathcal{C} \setminus \{G_R\}$ について operational property を満たすことを示す．

補題 2 ([14]). $G = (V, E)$ を K_n でない任意の区間グラフとする．このとき，$G+e$ が区間グラフとなるような辺 e が少なくともひとつ存在する．

Proof. 厳密な証明は [14] にあるのでここでは証明の概略の概略のみを示す．K_n ではない区間グラフ G の任意の区間表現を $\mathcal{I}_G$ とし，区間 $I_u \in \mathcal{I}_G$ の最小値・最大値（併せて端点と呼ぶ）をそれぞれ $L(I_u), R(I_u)$ と書く．このとき，頂点 $u, v \in V$ であって $R(I_u) < L(I_v)$ かつ区間 $[R(I_u), L(I_v)]$ 内に他の区間の端点を含まないようなものをとることができる．ここで，$R(I_v)$ と $L(I_v)$ の座標を交換し $L(I_v) < R(I_u)$ とすることで $I_u \cap I_v \neq \emptyset$ となり，辺 $e = \{u, v\}$ が G に加わり，新たな区間グラフ $G + e$ を得る．　□

4.1　標準形

列挙フレームワークを区間グラフに適用するために，区間グラフの標準形を定める必要がある．本節では，区間グラフの標準木表現を示した後，区間グラフの標準形を示す．

4.1.1　標準木表現

区間グラフに対応する木構造として，Korte と Möhring により導入された $\mathcal{MPQ}$-tree があり [15]，これは Booth と Lueker による $\mathcal{PQ}$-tree [16] に変更を加えたものである．ここでは，まず $\mathcal{PQ}$-tree を説明し，続いて $\mathcal{MPQ}$-tree への変更方法を示す．

$\mathcal{PQ}$-tree は Booth と Lueker によって導入され [16]，区間グラフの認識に用いることができる．$\mathcal{PQ}$-tree を $\mathcal{T}^*$ とする．$\mathcal{T}^*$ は根付き木で，内部ノードは $\mathcal{P}$-ノードと $\mathcal{Q}$-ノードの 2 種類からなる．図示する場合は，それぞれ円と長方形で描画する．$\mathcal{T}^*$ の葉ノードは区間グラフ G の極大クリークと 1 対 1 対応する．$\mathcal{T}^*$ の *frontier* とは，$\mathcal{T}^*$ の葉を左から右に並べることによって得られる，G の極大クリークの順

列である．2 つの $\mathcal{PQ}$-tree $\mathcal{T}^*$ と $\mathcal{T}'^*$ が**等価**であるとは，次の操作を有限回適用することで片方から他方を得られることである：

1. $\mathcal{P}$-ノードの子ノードを任意に並べ替える．

2. $\mathcal{Q}$-ノードの子ノードの順序を反転する．

[16] において Booth と Lueker は，グラフ G が区間グラフであることと，その frontier が G の極大クリークからなる連続する列であるような $\mathcal{PQ}$-tree $\mathcal{T}^*$ が存在することと同値であることを示した．また著者らは，G に対応する $\mathcal{PQ}$-tree を線形時間で構築するか，G が区間グラフでない場合はそのことを報告するアルゴリズムを提案した．区間グラフ G_1, G_2 が同型であることは，それらに対応するラベル付き $\mathcal{PQ}$-tree $\mathcal{T}_1^*, \mathcal{T}_2^*$ が同型であることと同値である．$\mathcal{T}_1^*, \mathcal{T}_2^*$ の同型性を判定できることから，区間グラフに関する同型性判定問題を線形時間で解くことができる（詳細は [16, 3, 17] を参照）．

$\mathcal{PQ}$-tree を単純化するために Korte と Möhring により提案されたのが $\mathcal{MPQ}$-tree であり [15]，名称は *modified* $\mathcal{PQ}$*-tree* に因む．$G = (V, E)$ に対応する $\mathcal{MPQ}$-tree $\mathcal{T}$ では，$\mathcal{PQ}$-tree $\mathcal{T}^*$ の各ノードに V の部分集合（空でもよい）を割り当てる．$\mathcal{P}$-ノードに対しては，ひとつの頂点集合を割り当て，$\mathcal{Q}$-ノードに対しては，その子それぞれに対して頂点集合を割り当てる．

$\mathcal{P}$-ノード P に割り当てられた頂点集合は，$\mathcal{T}$ において P を根とする部分木に対応する部分グラフ内の極大クリークすべてに含まれ，それ以外のクリークには含まれない頂点からなる．

$\mathcal{Q}$-ノード Q の子ノードを順に並べたものを $Q_1, Q_2, \ldots, Q_m$ とし，$\mathcal{T}_i$ を $\mathcal{T}$ において Q_i を根とする部分木であるとする．このとき，$m < 3$ であれば等価な $\mathcal{P}$-ノードに変換できるので，$m \geq 3$ とする．Q に対しては，各 Q_i に V の部分集合 S_i を割り当て，これらを**セクション**と呼ぶ．セクション S_i は，$\mathcal{T}_i$ および他の $\mathcal{T}_j$ の部分木に対応する部分グラウ内の極大クリークすべてに含まれるが，Q の子孫でない $\mathcal{T}$ の他の部分木に対応する部分グラフ内の極大クリークには含まれない頂点からなる．

$\mathcal{MPQ}$-Tree の重要な性質は以下である：

定理 3 ([15, Theorem 2.1])**.** 区間グラフ $G = (V, E)$ に対応する $\mathcal{PQ}$*-tree* を $\mathcal{T}^*$ とし，それに対応する $\mathcal{MPQ}$*-tree* を $\mathcal{T}$ とする．このとき，次が成り立つ：

(a) $\mathcal{T}$ と $\mathcal{T}^*$ は $O(|V| + |E|)$ 時間・空間で計算できる．

(b) G の各極大クリークは $\mathcal{T}$ における根から葉へのパスに対応し，各頂点 $v \in V$ は可能な限り根に近いノードに配置される．

(c) $\mathcal{T}$ において，頂点 v はひとつの葉か，ひとつの $\mathcal{P}$-ノードか，ある $\mathcal{Q}$-ノードの連続するセクション $S_i, S_{i+1}, \ldots, S_{i+j}$ $(j > 0)$ に配置される．

性質 (b) は重要である．例えば，$\mathcal{T}$ の根に属する頂点は G のすべての極大なクリークに含まれ，葉に属する頂点は単体的な頂点である．[15] では，定理 4.1.1(c) は明示的には示されていない．(c) は，$\mathcal{T}$ の葉を順に並べたとき，特定の頂点は連続する葉に含まれることから従う．

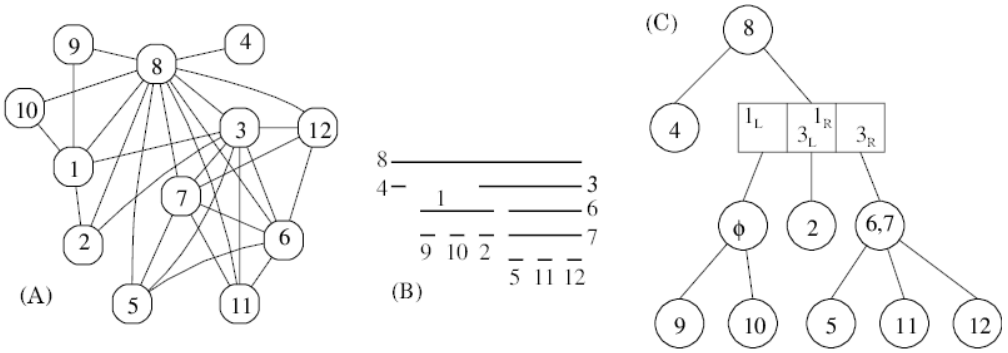

図 4.1: (A) 区間グラフの例．(B) 区間グラフに対応する区間表現．(C) 区間グラフに対応する $\mathcal{MPQ}$-tree

例を図 4.1 に示す．図 4.1(A) に示したグラフ G は区間グラフである．図 4.1(B) は G に対応する区間表現で，図 4.1(C) は G に対応する $\mathcal{MPQ}$-tree である．

4.1.2 $\mathcal{MPQ}$-tree の順序付け

本節では，すべての $\mathcal{MPQ}$-tree からなる集合上に全順序を定義する．この順序は推移的である．

$\mathcal{MPQ}$-tree $\mathcal{T}$ に対して，すべての $\mathcal{MPQ}$-tree の集合を整列したときのその添字を $\mathrm{Ind}(\mathcal{T})$ で表す．よって，次が成り立つ：

1. 任意の $\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2$ について，$\mathrm{Ind}(\mathcal{T}_1) = \mathrm{Ind}(\mathcal{T}_2) \Leftrightarrow \mathcal{T}_1 \sim \mathcal{T}_2$.

2. 任意の $\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3$ について，$\mathrm{Ind}(\mathcal{T}_1) < \mathrm{Ind}(\mathcal{T}_2)$ かつ $\mathrm{Ind}(\mathcal{T}_2) < \mathrm{Ind}(\mathcal{T}_3)$ ならば，$\mathrm{Ind}(\mathcal{T}_1) < \mathrm{Ind}(\mathcal{T}_3)$ が従う．

3. $\mathcal{T}_1 \not\sim \mathcal{T}_2$ である任意の $\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2$ について，$\mathrm{Ind}(\mathcal{T}_1) < \mathrm{Ind}(\mathcal{T}_2)$ または $\mathrm{Ind}(\mathcal{T}_1) > \mathrm{Ind}(\mathcal{T}_2)$ のいずれかが（排他的に）成り立つ．

本研究では，この順序によって定まる最小元をもって $\mathcal{MPQ}$-tree および対応する区間グラフの標準形を求める．よって，以降は添字を具体的に求めるのではなく，順序を決定する方法を定義する．

n_1 頂点からなる $\mathcal{MPQ}$-tree $\mathcal{T}_1$ と n_2 頂点からなる $\mathcal{MPQ}$-tree $\mathcal{T}_2$ の順序を定義する．まず，頂点数が異なる場合はその大小をもって $\mathcal{MPQ}$-tree の大小とする．すなわち，$n_1 < n_2$ ならば $\mathrm{Ind}(\mathcal{T}_1) < \mathrm{Ind}(\mathcal{T}_2)$ であり，$n_1 > n_2$ ならば

$\mathrm{Ind}(\mathcal{T}_1) > \mathrm{Ind}(\mathcal{T}_2)$ であるとする．従って，以降は $n_1 = n_2$ と仮定する．続いて，$\mathcal{MPQ}$-tree の（根とは限らない）ノードに対して次の 2 つの大小関係を定義する：

1. $\mathcal{P}$-ノードは $\mathcal{Q}$-ノードより小さいとする．そうでなく，

2. 部分木の頂点数が異なるならば，頂点数が少ない方が小さいとする．

以上で $\mathcal{T}_1, \mathcal{T}_2$ の根ノードの順序が定まるならば，それをもって $\mathcal{T}_1, \mathcal{T}_2$ の順序とする．大小が定まらなかった場合，2 つの根ノードは同種（$\mathcal{P}$-ノードまたは $\mathcal{Q}$-ノード）であり，頂点数も等しい．このとき，考慮するべきケースが 2 つある．

ケース (a) は $\mathcal{P}$-ノード同士の場合である．$\mathcal{T}_1$ の $\mathcal{P}$-ノードの子ノードの個数を k_1 とし，$\mathcal{T}_2$ の $\mathcal{P}$-ノードの子ノードの個数を k_2 とする．$k_1 \neq k_2$ の場合，子ノードが少ない方の $\mathcal{P}$-ノードの方が小さいと定義する．$k_1 = k_2$ の場合，それぞれの子ノードを添字をキーにして再帰的に昇順ソートし，子の列同士を辞書式順序で比較する．より正確には，$\mathcal{T}_1$ の $\mathcal{P}$-ノードの子ノードをソートした列を C とし，$\mathcal{T}_2$ の $\mathcal{P}$-ノードの子ノードをソートした列を C' とする．続いて，$C_i \not\sim C'_i$ となる最小の i を求める．そのような i が存在しない場合，$\mathcal{T}_1 \sim \mathcal{T}_2$ である．そうでない場合，C_i と C'_i の順序をもって $\mathcal{P}$-ノード同士の順序を定義する．

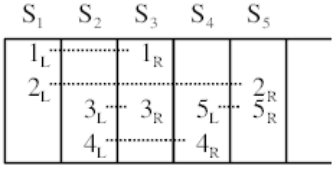

図 4.2: $\mathcal{Q}$-ノードの例

ケース (b) は $\mathcal{Q}$-ノード同士の場合である．まず 2 つの $\mathcal{Q}$-ノード Q_1, Q_2 の子の順序が固定されている場合の順序を定義する．Q_1 のセクションが左から順に $S_1, S_2, \ldots, S_k$ であり，Q_2 のセクションが左から順に $S'_1, S'_2, \ldots, S'_{k'}$ であるとする．$k \neq k'$ のとき，セクションの個数が少ない方が小さいと定義する．よって，以降は $k = k'$ と仮定する．説明のため，いずれかのセクションに含まれる各頂点 $u \in \bigcup_{1 \leq i \leq k} S_i$ に対し，**端点**を定義する．i を $u \in S_i$ であるような最小の i とし，セクション S_i を u の**左端点**と呼ぶ．同様に，i を $u \in S_i$ であるような最大の i とし，セクション S_i を u の**右端点**と呼ぶ．セクションを順序付けるために，セクション S_i に対してベクトル $(x_1, x_2, \ldots, x_{i-1}, y)$ を次のように定義する．各 $j = 1, 2, \ldots, i-1$ に対し，x_j は S_i が右端点であり，左端点が S_j であるような頂点の個数である．y は左端点が S_i であるような頂点の個数である．図 4.2 を例に具体例を示す．S_1 に対しては，S_2 を左端点とする頂点の個数が 2 であるので，ベクトルは (2) である．S_2 に対しては，S_2 を右端点とする頂点は無く，S_2 を左端点とする頂点の個数が 2 であるので，ベクトルは $(0, 2)$ である．S_3 に対しては，S_3 を右端点とする頂点はそれぞれ S_1, S_2 を左端点とする頂点 $1, 3$ であり，S_3

を左端点とする頂点は無いので，ベクトルは $(1,1,0)$ となる．同様にして，S_4, S_5 に対するベクトルはそれぞれ $(0,1,0,1)$ と $(1,0,0,1,0)$ である．$\mathcal{Q}$-ノードの順序を定義するには，2 つのセクション列 $S_1, S_2, \ldots, S_k$ と $S'_1, S'_2, \ldots, S'_{k'}$ をそれぞれ上で定義したベクトルの列に変換し，辞書式順序で比較する．ベクトル列の辞書式順序が等しかった場合は，子の列同士の辞書式順序で順序を定義する．それも等しい場合は，2 つの $\mathcal{Q}$-ノードは等価である．

子の順序の反転が可能な場合に 2 つの $\mathcal{Q}$-ノード Q, Q' の順序を決定する問題に戻る．まず，片方の $\mathcal{Q}$-ノード Q をとる．更に，Q のセクション列と子の列をそれぞれ反転して構成できる $\mathcal{Q}$-ノードを $\hat{Q}$ とする．Q と $\hat{Q}$ を上述の方法で比較し，大きくない方を $\min(Q, \hat{Q})$ とする．他方の $\mathcal{Q}$-ノード Q' についても同様に，反転して構成できる $\mathcal{Q}$-ノードを $\hat{Q'}$ とし，大きくない方を $\min(Q', \hat{Q'})$ とする．$\min(Q, \hat{Q})$ と $\min(Q', \hat{Q'})$ を比較し，その結果をもって Q, Q' の順序を定める．

上述の方法によって定まる順序がすべての $\mathcal{MPQ}$-tree 上の全順序になっていることは，$\mathcal{MPQ}$-tree の高さに関する帰納法で示すことができる．

4.1.3 標準文字列表現

任意の同型な区間グラフ G_1, G_2 $(G_1 \sim G_2)$ について，それらに対応する $\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2$ も同型であり，それらを用いて G_1, G_2 に関する同型性判定問題を線形時間で解くことができることから，区間グラフ $G = (V, E)$ に対する $\mathcal{MPQ}$-tree $\mathcal{T}$ は G の標準形であると言える．ここで,（K_n を除く）区間グラフの親を定めるために，区間グラフに対する**標準文字列表現**を導入する．すなわち，区間グラフの文字列表現であって，区間グラフ同士が同型ならば文字列同士も等しくなるようなものを定義する．

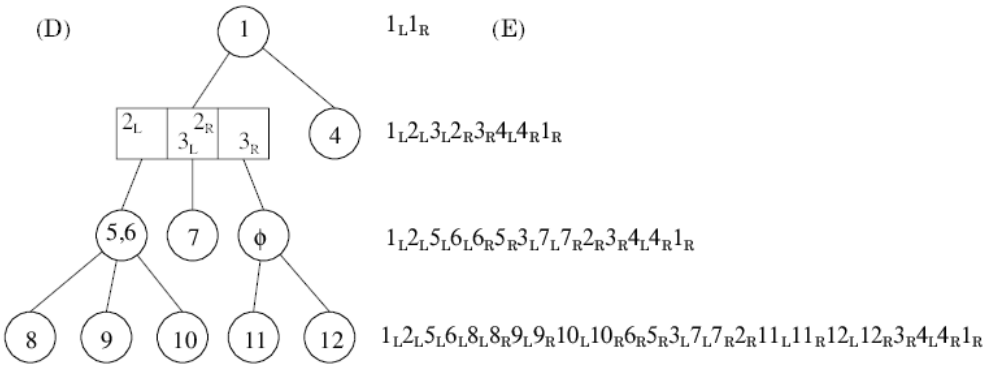

図 4.3: (D) left-heavy に描画した $\mathcal{MPQ}$-tree．(E) それに対応する文字列表現

$\mathcal{MPQ}$-tree から文字列表現を得る，3 段階からなる手続きを定める．まず，$\mathcal{MPQ}$-tree の各ノードについて，子ノードを “left-heavy” に並べ替えて描画する．この技法は [18] に見られる．ここでは，並べ替えのために 4.1.2 節で定義した順序を

用いる．$\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2$ の各ノードをそれぞれ並べ替えると，$\mathcal{T}_1 \sim \mathcal{T}_2$ であることと $\mathcal{T}_1, \mathcal{T}_2$ の描画が頂点への番号付けを除いて一致することが同値となる．

次に，頂点集合 $V = \{1, 2, \ldots, n\}$ の番号付けを，$\mathcal{MPQ}$-tree の根を始点とする BFS を行った際に $\mathcal{P}$-ノードに紐付いた頂点集合または $\mathcal{Q}$-ノードのセクションの要素として新たに発見される順序に従って付け替える．（子ノードの訪問順序は左から右であるとする．）$\mathcal{Q}$-ノードにおいて頂点が新たに発見されるのはその頂点の左端点となるセクションで起こるが，同時に複数頂点が新たに発見される場合は，右端点がより左にある頂点に若い番号を与える．これをリラベルと呼ぶ．リラベルを行うと，2 つの $\mathcal{MPQ}$-tree $\mathcal{T}_1, \mathcal{T}_2$ が同型であることと，それらの描画が頂点の番号付けを含めて一致することが同値となる．このことから，区間グラフ G に対応する $\mathcal{MPQ}$-tree $\mathcal{T}$ をリラベルしたものは，G **の標準** *$\mathcal{MPQ}$-tree* であると言える．例として，図 4.1(C) の $\mathcal{MPQ}$-tree に上記の手続きを適用して得られる $\mathcal{MPQ}$-tree を図 4.3(D) に示す．

最後に，標準 $\mathcal{MPQ}$-tree $\mathcal{T}$ を DFS 順に辿り，$\mathcal{T}$ の文字列表現を構築する．基本方針としては，探索中に訪問した各ノードあるいはセクションで，そこを左端点とする頂点の番号を昇順に並べたものを pre-order で出力し，そこを右端点とする頂点の番号を降順に並べたものを post-order で出力する．（$\mathcal{P}$-ノード P に属する頂点の両端点は P と見なす.）子ノードについては再帰的に文字列化して連結したものを in-order で出力する．$\mathcal{Q}$-ノードについては，セクション毎に上記の処理で文字列化し，それらを連結したものを子ノード列の文字列表現とする．各工程の結果は一意的であるため，得られる文字列は標準形と見なせる．

区間グラフ G について，上記の手続きで得られる文字列を $\mathrm{Str}_I(G)$ とする．例として，図 4.3(D) の標準 $\mathcal{MPQ}$-tree から得られる文字列表現は “1 2 5 6 8 8 9 9 10 10 6 5 3 7 7 2 11 11 12 12 3 4 4 1” である．（図 4.3(E) では，見やすさのために添え字 $_{L,R}$ を加えている．実際の文字列表現には含まれない.）また，レベルを制限して得られる部分木の文字列表現も併記している．[15] の結果と上述の定義から，次の定理を得る：

定理 4. $G = (V, E)$ を任意の区間グラフとし，$n = |V|, m = |E|$ とする．このとき，次が成り立つ：

1. 標準 *$\mathcal{MPQ}$-tree* および $\mathrm{Str}_I(G)$ は $O(n + m)$ 時間で計算できる．
2. $\mathrm{Str}_I(G)$ を（文字列ではなく）頂点番号の列と解釈したとき，$|\mathrm{Str}_I(G)| = 2n$ である．
3. *2* つの区間グラフ G_1, G_2 が同型であることと，$\mathrm{Str}_I(G_1) = \mathrm{Str}_I(G_2)$ は同値である．

4.2 親子関係

K_n ではない区間グラフを $G = (V, E)$ とし，G の標準文字列表現を S とする．補題 2 の証明から，順序対 (i, j) であって，S 中に連続して出現し，かつ i の出現が 2 回目で j の出現が 1 回目であるようなものが存在する．すなわち，S に添字 $_{L,R}$ を付けて表したとき，連続する部分列として $\langle i_R, j_L \rangle$ を含む．そのような S での出現が最も左であるようなものを (i, j) とする．S が G の標準表現であることと，S の構成法から頂点 i, j は G で隣接しないことにより，そのような (i, j) は一意的に定まる．G の親を $G + \{i, j\}$ で定義する．グラフの構造が変化することで標準形における頂点の番号付けが変わる可能性があるが，親に対応する $\mathcal{MPQ}$-tree を構築して再計算することで線形時間でリラベルできる．

定理 5. $G = (V, E)$ を任意の区間グラフとし，$n = |V|, m = |E|$ とする．このとき，G の親は $O(n + m)$ 時間で計算できる．

4.3 解析

本節ではアルゴリズムの解析を行い，各グラフが多項式時間遅延で出力されることを示す．各グラフ $G = (V, E) \in \mathcal{C}$ についてそれが消費する計算時間を出力，$S(G)$ の計算，各 $G' \in S(G)$ の処理にかかる時間のそれぞれで評価する．$n = |V|, m = |E|$ とする．G の出力には，$O(n + m)$ 時間がかかる．基本操作を用いる本フレームワークでは，$S(G)$ の要素数は高々 m であり，それぞれの要素は G から一本の辺を取り除くことで得られる．グラフを文字列化できることを利用してグラフの集合を文字列集合と見なし prefix tree (trie) で実装することで，重複除去を高速化できる．これにより，$S(G)$ の計算は $O(m(n + m))$ 時間で完了する．結果として $O(m)$ 個のグラフからなる集合が得られ，これに属する各グラフ G' は n 頂点 $m - 1$ 辺からなる．アルゴリズムは各 G' の親が G であるかをチェックするが，定理 5 によりそれぞれについて $O(n + m)$ 時間で完了する．よって，これらの処理は併せて $O(m(n + m))$ 時間である．従って，各グラフは $O(m(n + m))$ 時間を消費する．単純グラフにおいては一般に $m \in O(n^2)$ であることから，本列挙アルゴリズムはグラフひとつあたり $O(n^4)$ 時間で動作する．

本章における主定理は以下である：

定理 6. n 頂点からなるすべての非同型な区間グラフは，グラフひとつあたり $O(n^4)$ 時間で列挙できる．

4.4 2 つの派生アルゴリズム

アルゴリズムに軽微な変更を加えることで，2 つの派生版アルゴリズムを得ることができる．

系 7. 定理 *6* のアルゴリズムは，次の条件を満たす非同型な区間グラフすべてを列挙するように変更できる：

1. 連結なものに限定する．*and/or*
2. 頂点数を高々 n とする．

どの派生においても，出力するグラフの頂点数を n' として遅延は $O(n'^4)$ 時間である．

Proof. $\mathcal{MPQ}$-tree の定義から，区間グラフ $G = (V, E)$ が非連結であることと，対応する $\mathcal{MPQ}$-tree の根ノードに割り当てられる頂点集合が空集合であることは同値である．従って，各 $G' \in S(G)$ を処理する際，G' に対応する $\mathcal{MPQ}$-tree の根ノードがそのような $\mathcal{P}$-ノードであった場合は処理をスキップすることで，出力を連結なグラフに限定できる．基本操作の定義から，非連結なグラフの子孫はすべて非連結なグラフなので，これによって出力するべきグラフが出力されなくなることはない．また，この判定は線形時間で完了し，グラフあたりの遅延には影響しない．

頂点数が高々 n の区間グラフを列挙するには，アルゴリズムを繰り返し実行し，頂点数 $1, 2, \ldots, n$ のグラフをそれぞれ列挙すればよい． □

第5章　置換グラフの列挙

本章では，置換グラフの列挙について論じる．本章では，$\mathcal{C}$ を n 頂点のすべての置換グラフからなる集合とする．$G_R = K_n$ とする．まずは $\mathcal{C} \setminus \{G_R\}$ の元が operational property をもつことを示す．

補題 8. $G = (V, E)$ を K_n でない任意の置換グラフとする．このとき，辺 e であって $G + e$ が置換グラフとなるものが少なくともひとつ存在する．

Proof. 置換 π で表現される置換グラフ $G = (V, E)$ の線分表現 $\mathcal{L}_G$ を考える．$G \neq K_n$ であることから，頂点 $u, v \in V$ であって $\{u, v\} \notin E$ であるものがひとつ以上存在する．一般性を損なわず $u < v$ であると仮定する．すると，$\{u, v\} \notin E$ であることから $\pi(u) < \pi(v)$ となる．G で隣接しない頂点対 $\{u, v\}$ の内,「最も近い」ものを次のように求める．u, v を $u < v$ かつ $\pi(u) < \pi(v)$ を満たす任意の 2 頂点とする．まず，$v - u \leq \pi(v) - \pi(u)$ の場合を考える．$v - u > 1$ のとき，$\mathcal{L}_G$ の線分 L_1 に着目すると，$u < w < v$ を満たす w が存在する．このとき，3 つのケースがある：

1. $\pi(u) < \pi(w) < \pi(v)$
2. $\pi(w) < \pi(u) < \pi(v)$
3. $\pi(u) < \pi(v) < \pi(w)$

1., 2. の場合，u を w で置き換え，$\{u, v\}$ より $\{w, v\}$ の方が近いとする．3. の場合，v を w で置き換え，$\{u, v\}$ より $\{v, w\}$ の方が近いとする．次に，$v-u > \pi(v)-\pi(u)$ の場合を考える．この場合は，L_1 に着目した代わりに L_2 に着目し，上記と対称的な同様の処理を行う．いずれの場合でも，置き換えによって $\min(|v-u|, |\pi(v)-\pi(u)|)$ は真に減少する．従って，上記手続きを繰り返し適用することで，$v - u = 1$ または $\pi(v) - \pi(u) = 1$ を満たす頂点対 u, v が得られる．ここで，新たな置換 π' を次のように定義する．$w \in V \setminus \{u, v\}$ に対しては $\pi'(w) = \pi(w)$ とし，$\pi'(v) = \pi(u), \pi'(u) = \pi(v)$ とする．u, v の取り方から 2 線分 u-$\pi(u)$ と v-$\pi(v)$ は交点をもたないが，u-$\pi'(u)$ と v-$\pi'(v)$ は交点をもつ．更に，線分の端点は L_1 か L_2 のいずれかの上で隣接しているので，他の線分との間に新たな交点は生まれない．よって，π' に対応するグラフは新たな置換グラフ $G + \{u, v\}$ である．　□

5.1 標準表現

本節では，置換グラフの標準表現について論じる．方針は区間グラフに対するものと同様で，まず同型な置換グラフに対して一意的に定まる木構造である *Modular decomposition tree* を導入する．続いて，木の描画を標準化し，標準化した木から文字列表現を構築する．

5.1.1 標準木表現

グラフ $G = (V, E)$ について，頂点の部分集合 $X \subseteq V$ が**モジュール**であるとは，各頂点 $u \in V \setminus X$ について $\forall x \in X, \{u, x\} \in E$ または $\forall x \in X, \{u, x\} \notin E$ のいずれかが成り立つことを言う.（詳細は [19] を参照.）**自明なモジュール**とは，$\emptyset, V$ および各シングルトン $\{u\}$ $(u \in V)$ のことを言う．グラフが *prime* であるとは，そのモジュールがすべて自明なモジュールであることを言う．任意の置換グラフ G に対して，G が prime であるとき，かつそのときに限り（反転の意味での）対称性を除いて一意な線分表現をもつ [20].

[20] において，Gallai はグラフの *Modular decomposition* を再帰的に定義した．Modular decomposition の概略は，極大なモジュールによる頂点集合 V の分割を再帰的に適用するものである．ここで，分割前の頂点集合を親とし，それを分割して得られる各頂点集合を子とする親子関係が定まり，これによって定まる木構造を *Modular decomposition tree* という．グラフ $G = (V, E)$ に対応する Modular decomposition tree $\mathcal{T}$ の各内部ノードは，次の 3 種に分類される：

1. すべての子ノードの対について，それらに対応するモジュール（の元）同士を結ぶ辺が E に含まれるならば，そのノードは *series* ノードという．
2. すべての子ノードの対について，それらに対応するモジュール（の元）同士を結ぶ辺が E に含まれないならば，そのノードは *parallel* ノードという．
3. いずれでもないとき，そのノードは *prime* ノードという．

区間グラフに対応する Modular decomposition tree について，次が知られている：

1. 同型なグラフの標準形となる [4].
2. 線形時間・空間で計算できる（[21] を参照）.

このことは，本研究において次のように利用できる．2 つの置換グラフを G_1, G_2 とし，それに対応する Modular decomposition tree を $\mathcal{T}_1, \mathcal{T}_2$ とする．このとき，$G_1 \sim G_2$ は次と同値である：

1. ラベル付き木として $\mathcal{T}_1 \sim \mathcal{T}_2$ である．かつ

2. 対応する 2 つの prime ノードにそれぞれ対応する部分グラフ同士が同型である．

続いて，すべての Modular decomposition tree からなる集合上に全順序を定義する．基本的な戦略は，前章において $\mathcal{MPQ}$-tree について行ったものと同じである．n_1 頂点からなる Modular decomposition tree $\mathcal{T}_1$ と n_2 頂点からなる Modular decomposition tree $\mathcal{T}_2$ があるとする．まず，頂点数が異なる場合はその大小をもって順序を定める．すなわち，$n_1 < n_2$ のときは $\mathrm{Ind}(\mathcal{T}_1) < \mathrm{Ind}(\mathcal{T}_2)$ であり，$n_1 > n_2$ のときは $\mathrm{Ind}(\mathcal{T}_1) > \mathrm{Ind}(\mathcal{T}_2)$ である．以降，$n_1 = n_2$ である．ここで，次の 2 つの順序を定める：

1. 葉は prime ノードより小さく，prime ノードは series ノードより小さく，series ノードは parallel ノードより小さい（この順序は推移的である）．

2. ノードが同種のとき，頂点数が少ない方が小さい．

これに基づいて根ノード同士を比較して順序が定まるならば，それを $\mathcal{T}_1, \mathcal{T}_2$ の順序とする．そうでない場合，すなわち同じ頂点数からなる同種のノードの場合について述べる．このとき，着目しているノードが prime ノードでないならば $\mathcal{MPQ}$-tree における $\mathcal{P}$-ノードに対するものと同様の状況となる．この場合は子ノードの列をそれぞれ再帰的にソートし，それらを辞書式順序で比較することによって順序を定める．残るケースは，prime ノード同士を比較する場合である．[20] で示されたように，prime ノードに対応する部分グラフは対称性に関して一意な線分表現をもつ．ここで，5.1.2 節で詳しく論じるが，対応するグラフが同型となる 4 つの置換がある．それらの内で辞書式順序が最小のものを標準形とし，標準形同士の辞書式順序を用いて順序を定める．

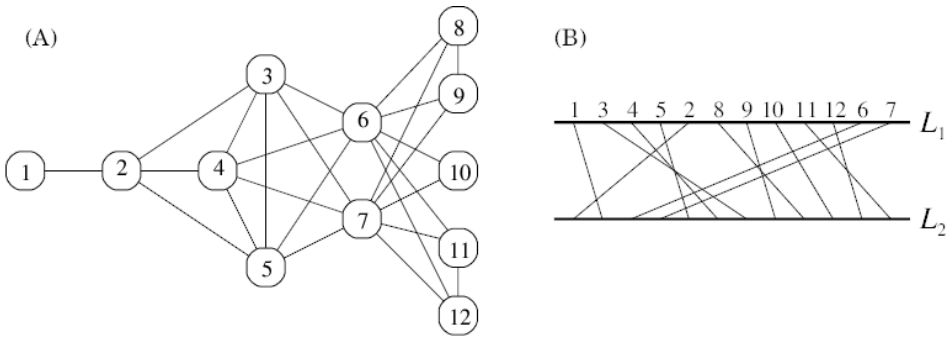

図 5.1: 置換グラフの例とその線分表現

ここで，置換グラフから木表現を得る様子の例を示す．図 5.1(A) は置換グラフの例であり，図 5.1(B) はその線分表現である．

図 5.2 がこのグラフのモジュールであり，得られる Modular decomposition tree が図 5.3 である．

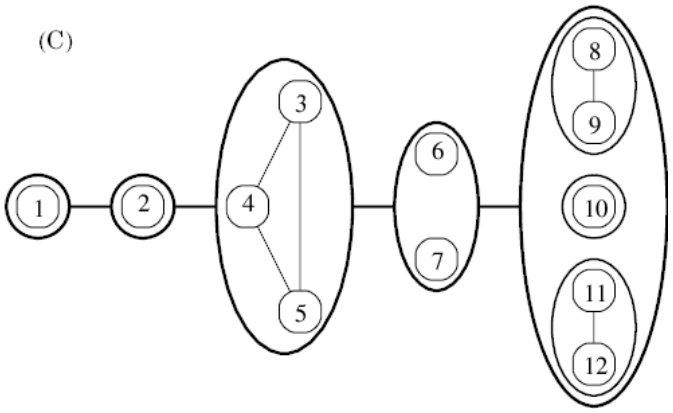

図 5.2: 図 5.1(A) のグラフのモジュール

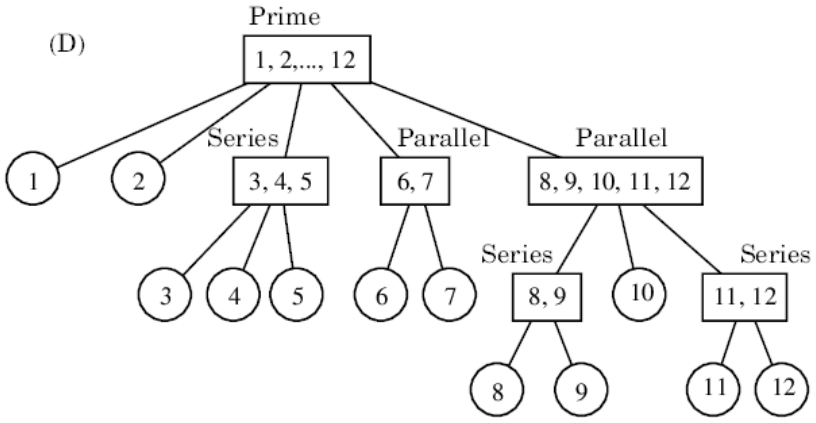

図 5.3: 図 5.1(A) のグラフに対応する Modular decomposition tree

得られた Modular decomposition tree を left-heavy に描画したものが図 5.4 である.

5.1.2 標準文字列表現

n 頂点の区間グラフ $G = (V, E)$ に対応する Modular decomposition tree $\mathcal{T}$ は G の標準形であり，$\mathcal{T}$ の各ノードの子ノードを left-heavy にソートすることで $\mathcal{T}$ の描画を標準化できる．区間グラフの章で行ったことと同様に，置換グラフの**標準文字列表現**を得ることができる.

まず，$G = (V, E)$ が prime モジュールである場合を考える．前述のように，この場合は G の線分表現は対称性を除いて一意に定まる．それに対応する置換を π とする．線分表現を垂直・水平・垂直と水平に反転することで得られる線分表現に対応する置換をそれぞれ π^V, π^H, π^R とする．各置換は，各 $i \in \{1, 2, \ldots, n\}$ が一度ずつ出現する列として表現でき，それを文字列として解釈することで文字列

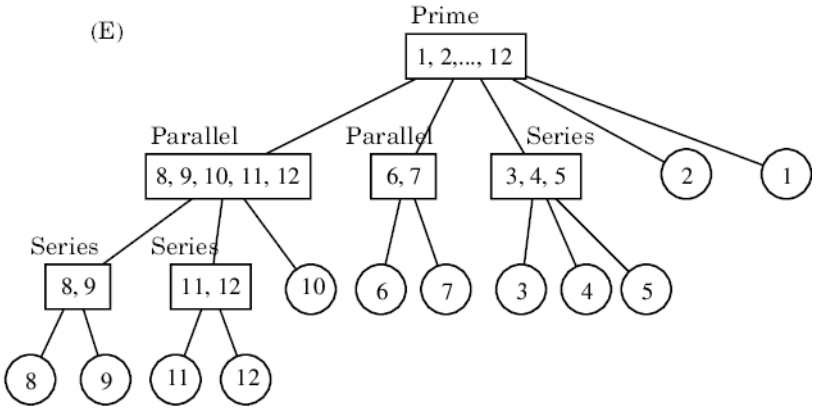

図 5.4: left-heavy に描画した Modular decomposition tree

表現を得ることができる．そうして得られる 4 つの文字列の内，辞書式順序で最小なものを G の標準文字列表現であると定める．グラフの例と，それに対応する 4 つの線分表現を図 5.5 に示す．

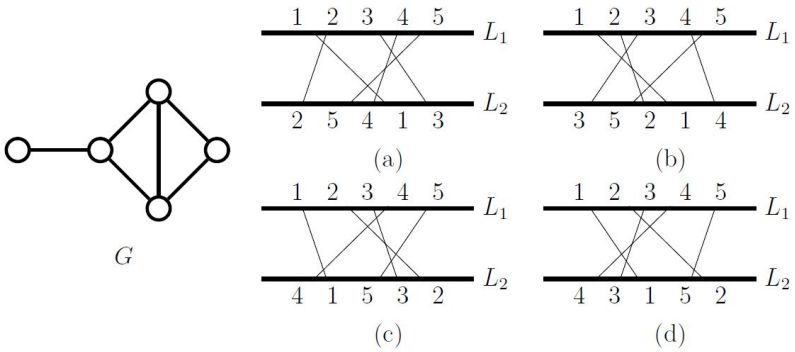

図 5.5: prime な置換グラフ G とそれに対応する線分表現．線分表現 (a) を π とすると，(b) が π^H, (c) が π^V, (d) が π^R である

一般のケースについて論じる．この場合は，G に対応する標準 Modular decomposition tree $\mathcal{T}$ の根ノードの種類によって処理が分かれる．根ノードの子の個数を n' とする．まず，G の極大モジュールを 1 頂点に縮約して得られる n' 頂点のグラフ G^s を考える．(極大モジュールを頂点とするグラフと同型である．辺の有無はモジュールの定義から自然に定まる.）根ノードが series ノードのとき G^s は $K_{n'}$ となり，parallel ノードのときは辺をもたないグラフとなる．前者に対応する置換は $\pi = \begin{pmatrix} 1 & 2 & \cdots & n' \\ n' & n'-1 & \cdots & 1 \end{pmatrix}$ であり，後者に対応する置換は $\pi = \begin{pmatrix} 1 & 2 & \cdots & n' \\ 1 & 2 & \cdots & n' \end{pmatrix}$ である．π に対応する線分表現を考え，各線分をそれに対応する G のモジュールの線分表現で置き換えることで G の線分表現が得られる．このとき，モジュールを線分に対応付ける順序によって異なる線分表現となるため，各モジュールが対

応する子ノードの順序に従って対応付けることで一意的に選択する．prime ノードの場合は G^s は prime モジュールとなっているので，上述の方法で線分表現を構成し，同様にして線分を対応するモジュールの標準線分表現で置き換えることで G の標準線分表現を得られる．最後に，得られた線分表現に対し L_1 側での頂点の出現順序が $1, 2, \ldots, n$ となるようにリラベリングを行うことで，G に対応する置換 π が L_2 側での出現順として得られる．この π を G の標準文字列表現であると定める．

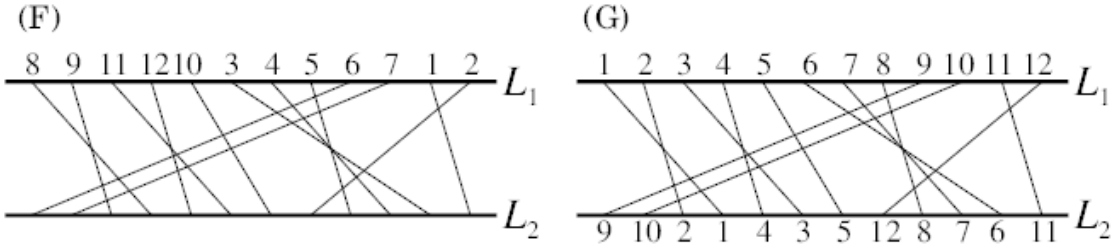

図 5.6: 線分表現からリラベリングを行う例

図 5.4 の Modular decomposition tree から，図 5.6 の線分表現が得られる．[20, 4, 21] の結果と上記の定義から，次の定理が得られる．

定理 9. $G = (V, E)$ を任意の置換グラフとし，$n = |V|, m = |E|$ とする．このとき，次が成り立つ：

1. G に対応する標準 *Modular decomposition tree* と標準文字列表現は $O(n+m)$ 時間計算できる．
2. 2 つの置換グラフ G_1, G_2 の標準文字列表現をそれぞれ π_1, π_2 とする．このとき，$G_1 \sim G_2 \Leftrightarrow \pi_1 = \pi_2$ である．

5.2 親子関係

K_n でない任意の置換グラフを $G = (V, E)$ とし，G の標準文字列表現を生成する置換を π とする．補題 8 の証明から，順序対 $(i, i+1)$ であって $\pi(i) < \pi(i+1)$ であるようなものが少なくともひとつ存在する．このとき，$\{i, i+1\} \notin E$ である．条件を満たす順序対の内，i が最小であるものを一意的に選び，G の親を $G+\{i, i+1\}$ で定義する．このとき，親は一意的に定まり，それに対応する置換を得るためのリラベリングは線形時間で可能である．

定理 10. $G = (V, E)$ を任意の置換グラフとし，$n = |V|, m = |E|$ とする．このとき，G の親は $O(n + m)$ 時間で計算できる．

5.3 アルゴリズムの解析

アルゴリズムの解析について記述する．区間グラフを列挙するアルゴリズムの解析における定理 5 を定理 10 で置き換えることで同様の結果を得る．従って，次の定理と系を得る．

定理 11. n 頂点からなる非同型なすべての置換グラフはグラフひとつあたり $O(n^4)$ 時間遅延で列挙できる．

系 12. 定理 *11* のアルゴリズムは，次の条件を満たす非同型な置換グラフすべてを列挙するように変更できる：

1. 連結なものに限定する．*and/or*
2. 頂点数を高々 n とする．

どの派生においても，出力するグラフの頂点数を n' として遅延は $O(n'^4)$ 時間である．

Proof. Modular decomposition tree の定義から，区間グラフ $G = (V, E)$ に対応する Modular decomposition tree の根ノードが parallel ノードであることと，G が非連結であることは同値である．従って，アルゴリズムが $G' \in S(G)$ を処理する際，G' に対応する Modular decomposition tree の根ノードが parallel ノードであった場合に処理をスキップすることで，出力を連結なグラフに限定できる．親子関係の定義から非連結なグラフの子孫となるグラフはすべて非連結なので，出力するべきグラフが出力されなくなることはない．

頂点数が高々 n の置換グラフを列挙するには，アルゴリズムを繰り返し実行し，頂点数 $1, 2, \ldots, n$ のグラフをそれぞれ列挙すればよい． □

第6章　実験結果

アルゴリズムを実際の計算機上に実装し，区間グラフと置換グラフを列挙した．結果を表 6.1 に示す．また，具体的なグラフの一覧は http://www.jaist.ac.jp/~uehara/graphs で公開されている．

表 6.1: 頂点数ごとの各グラフの個数

頂点数	区間グラフ	連結な区間グラフ	置換グラフ	連結な置換グラフ
1	1	1	1	1
2	2	1	2	1
3	4	2	4	2
4	10	5	11	6
5	27	15	33	20
6	92	56	142	99
7	369	250	776	600
8	1,807	1,328	5,699	4,753
9	10,344	8,069	-	-
10	67,659	54,962	-	-
11	491,347	410,330	-	-
12	3,894,446	3,317,302	-	-

第7章　おわりに

本研究では，グラフ認識問題とグラフ同型性判定問題が多項式時間で解けるグラフクラスの元を列挙するアルゴリズムを構成する一般性のあるフレームワークを提案した．更に，フレームワークの適用事例として区間グラフと置換グラフを列挙するアルゴリズムを構成・実装しグラフの列挙を行った．

未解決問題のひとつは効率性である．今回実装したアルゴリズムは $O(n^4)$ 時間遅延で実行できるが，これを改善することでより大きな n に対しても現実的な時間でグラフを列挙できる可能性がある．一方で，一般にグラフの個数は n に対して指数的に増大するため，グラフの一覧を公開するには出力サイズが問題となってくる．このことは，グラフ一覧を圧縮・展開するアルゴリズムの必要性を示唆している．

今後の展望としては，フレームワークを他のグラフクラスに適用することが挙げられる．加えて，構築されるアルゴリズムの実行時間を多項式時間遅延に限定しなければ，認識問題・同型性判定問題が多項式時間で解けないグラフクラスに対してもフレームワークの適用自体は可能である．それにより，ある n に対して現実的な時間でグラフを列挙できる可能性がある．

参考文献

[1] Peter Damaschke. Paths in interval graphs and circular arc graphs. *Discrete Mathematics*, Vol. 112, No. 1-3, pp. 49–64, 1993.

[2] Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, Vol. 1, No. 2, pp. 180–187, 1972.

[3] George S Lueker and Kellogg S Booth. A linear time algorithm for deciding interval graph isomorphism. *Journal of the ACM (JACM)*, Vol. 26, No. 2, pp. 183–195, 1979.

[4] Charles J Colbourn. On testing isomorphism of permutation graphs. *Networks*, Vol. 11, No. 1, pp. 13–21, 1981.

[5] Andreas Brandstädt, Van Bang Le, and Jeremy P Spinrad. *Graph classes: a survey*. SIAM, 1999.

[6] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*. Elsevier, 2004.

[7] Jeremy P Spinrad. *Efficient Graph Representations.: The Fields Institute for Research in Mathematical Sciences.*, Vol. 19. American Mathematical Soc., 2003.

[8] P. Heggernes. Personal communication, 2013.

[9] Toshiki Saitoh, Katsuhisa Yamanaka, Masashi Kiyomi, and Ryuhei Uehara. Random generation and enumeration of proper interval graphs. *IEICE TRANSACTIONS on Information and Systems*, Vol. 93, No. 7, pp. 1816–1823, 2010.

[10] Toshiki Saitoh, Yota Otachi, Katsuhisa Yamanaka, and Ryuhei Uehara. Random generation and enumeration of bipartite permutation graphs. *Journal of Discrete Algorithms*, Vol. 10, pp. 84–97, 2012.

[11] Johannes Kobler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Springer Science & Business Media, 2012.

[12] 岡本吉央. 列挙の基本と基礎的なアルゴリズム. 電子情報通信学会誌, Vol. 95, No. 6, pp. 477–483, 2012.

[13] David Avis and Komei Fukuda. Reverse search for enumeration. *Discrete applied mathematics*, Vol. 65, No. 1-3, pp. 21–46, 1996.

[14] Masashi Kiyomi, Shuji Kijima, and Takeaki Uno. Listing chordal graphs and interval graphs. In *Graph-Theoretic Concepts in Computer Science: 32nd International Workshop, WG 2006, Bergen, Norway, June 22-24, 2006 Revised Papers 32*, pp. 68–77. Springer, 2006.

[15] Norbert Korte and Rolf H Möhring. An incremental linear-time algorithm for recognizing interval graphs. *SIAM Journal on Computing*, Vol. 18, No. 1, pp. 68–81, 1989.

[16] Kellogg S Booth and George S Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using pq-tree algorithms. *Journal of computer and system sciences*, Vol. 13, No. 3, pp. 335–379, 1976.

[17] Charles J Colbourn and Kellogg S Booth. Linear time automorphism algorithms for trees, interval graphs, and planar graphs. *SIAM Journal on Computing*, Vol. 10, No. 1, pp. 203–225, 1981.

[18] Shin-ichi Nakano and Takeaki Uno. Constant time generation of trees with specified diameter. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pp. 33–45. Springer, 2004.

[19] Ross M McConnell and Jeremy P Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, Vol. 201, No. 1-3, pp. 189–241, 1999.

[20] T. Gallai. Transitiv orientierbare graphen. *Acta Mathematica Academiae Scientiarum Hungarica*, Vol. 18, pp. 25–66, 1967.

[21] Christophe Crespelle and Christophe Paul. Fully dynamic algorithm for recognition and modular decomposition of permutation graphs. *Algorithmica*, Vol. 58, pp. 405–432, 2010.

関連出版

本研究の内容は，*Theoretical Computer Science* に “Kazuaki Yamazaki, Toshiki Saitoh, Masashi Kiyomi and Ryuhei Uehara. (2020). Enumeration of nonisomorphic interval graphs and nonisomorphic permutation graphs. Theoretical Computer Science, 806, 310-322.” として掲載されている．本論文では掲載版と比して，置換グラフの標準文字列表現を得る手続きをより詳細に記述し，曖昧性を排し厳密性を高めている．