グラフに含まれる誘導マッチングの列挙

栗田 和宏[1,a)]　和佐 州洋[1,b)]　喜田 拓也[1,c)]　有村 博紀[1,d)]

概要：与えられたグラフ $G = (V, E)$ に対して，その誘導マッチングを列挙する問題について議論する．誘導マッチング $M \subseteq E$ とは，M に含まれるいかなる 2 つの辺も G の辺によって接続していないマッチングである．本稿では，最大次数 Δ のグラフ G に対し，G の誘導マッチングを $O(\Delta^2)$ 遅延時間と $O(|E|)$ 領域で列挙するアルゴリズムを提案する．さらに，長さ 6 以下の閉路を含まないグラフ G に対して，G の誘導マッチングをならし定数時間で列挙するアルゴリズムを提案する．

1. はじめに

1.1 本稿の目的

本稿では，以下のように定義される誘導マッチング列挙問題について考察する．与えられたグラフ $G = (V, E)$ と辺集合 M に対し，M に含まれるどの 2 つの辺も隣接していないとき，M をマッチングという．G 上のマッチング M のうち，誘導グラフの辺集合となっているものを誘導マッチングという [3]．これは，M に含まれるいかなる 2 つの辺も，G の辺によって接続していないようなマッチングである．誘導マッチング列挙問題とは，与えられたグラフ G に対して，すべての誘導マッチングを重複なしに出力する問題をいう．

1.2 主結果

はじめに，与えられたグラフに対して，$O(\Delta^2)$ 遅延時間でその誘導マッチングを列挙するアルゴリズムを与える．誘導マッチング中の任意の 2 辺はその間が辺 2 つ分以上離れているという性質を持つ．さらに，誘導マッチングの任意の部分集合は誘導マッチングである．これらの性質に基づいて，アルゴリズムは追加可能な辺の集合を管理し，分割法を用いてすべての誘導マッチングを列挙する．

次に，入力のグラフに長さ 6 以下の閉路を含まないという制約を加えた場合を考察する．このとき，最初のアルゴリズムを改良することですべての誘導マッチングをならし定数遅延時間で列挙可能なことを示す．

1.3 関連研究

通常の（誘導でない）最大マッチングは多項式時間で求めることができる [4]．マッチング列挙に関しては，以下の先行研究が知られている．Uno [6] は 2 部グラフ上の完全マッチング列挙を解 1 つあたり $O(\log |V|)$ 時間で解くアルゴリズムを与えた．宇野 [9] は，最大次数 Δ を持つ一般のグラフ上で，極大マッチングが解 1 つあたり $O(\Delta)$ 時間で列挙可能なことを示した．しかし，著者らの知る限りでは，誘導マッチング列挙問題の計算量に関して既存の研究は見当たらない．

最大の誘導マッチングを 1 つ求める問題は，たとえ入力グラフを二部グラフに制限しても NP 完全であることが示されている [1]．弦グラフは長さ 4 以上の誘導閉路を含まないグラフである．入力グラフを弦グラフに制限すると，最大の誘導マッチングが多項式時間で求められる [5]．誘導マッチングの最大サイズの下限について，最大次数 $\Delta \leq 3$, 内周 *1 $g \geq 6$，長さ 7 の閉路を持たないような頂点数 n の連結なグラフ G はサイズ $\frac{n-1}{5}$ 以上の誘導マッチングを持つことが示されている [5]．

1.4 本稿の構成

2 節では，本稿で扱う誘導マッチング列挙問題を導入する．3 節では，一般のグラフに対して，その誘導マッチングを列挙する $O(\Delta^2)$ 遅延アルゴリズムを与える．4 節では，長さ 6 以下の閉路を含まないグラフに対して，誘導マッチングを列挙するならし定数遅延のアルゴリズムを与える．

2. 準備

本節ではグラフと列挙問題に関する基本的な定義を与え

1 北海道大学 大学院情報科学研究科
Sapporo, Hokkaido 060–0814, Japan
a) k-kurita@ist.hokudai.ac.jp
b) wasa@ist.hokudai.ac.jp
c) kida@ist.hokudai.ac.jp
d) arim@ist.hokudai.ac.jp

*1 G の内周（girth）とは，G に含まれる閉路の長さの最小値をいう [2]．

る．ここで，グラフ $G = (V, E)$ とは二つの集合 V と E の組であり，E は V と V の直積の部分集合である．このとき，集合 V を**頂点集合**といい，集合 E を**辺集合**という．本稿において扱うグラフは，ループや多重辺を含まない単純無向グラフのみを扱うものとする．以下では，集合 S に対して，S の要素数を**サイズ**といい，$|S|$ で表す．

2.1 グラフの基本的な用語と記法

グラフ $G = (V, E)$ において，頂点集合 $V' \subseteq V$ と辺集合 $E' \subseteq E$ からなるグラフ $G' = (V', E')$ をグラフ G の**部分グラフ**という．頂点集合 $V_p = \{x_0, \ldots, x_k\} \quad \subseteq V$ と辺集合 $E_p = \{\{x_0, x_1\}, \{x_1, x_2\}, \ldots, \{x_{k-1}, x_k\}\} \in E$ からなる部分グラフ $P = (V_p, E_p)$ を**パス**という．また，$|E_p|$ を**パス長**という．頂点 $u, \quad v \in V$ に対して，それらを端点とする最短のパス長を u と v の**距離**といい，$d(u, v)$ と表す．ここで，**閉路**とは，$k \geq 2$ を満たす，パス $P = x_0, \ldots, x_k$ で $x_0 = x_k$ を満たすパスをいう．このとき，閉路に含まれる辺の数 k を**閉路の長さ**とする．長さ k の閉路を C^k と表す．

頂点 u, v が $d(u, v) = 1$ を満たす場合，それらは**隣接**しているという．このとき u と v の間にはある辺 $e \in E$ が存在し，e によって u と v は**接続**されている．頂点 v と隣接している頂点の集合を**近傍**といい，$N(v)$ と表す．頂点 $v \in V$ を端点とする辺の数を**頂点** v **の次数**といい，$d(v)$ と表す．また，$\Delta(G) = \max\{d(v) \mid v \in V\}$ を**グラフ** G **の次数**という．

頂点どうしの距離を基に，辺と頂点，辺と辺の距離を次のように定義する．まず，頂点を $w \in V$ とおき，辺を $e = \{u, v\}$ と $e' = \{u', v'\} \in E$ とおく．頂点 w と辺 e の距離を $d(w, e) = \min\{d(u, w), d(v, w)\}$ とし，辺 e と e' の距離を $d(e, e') = \min\{d(u, e'), d(v, e')\}$ とする．辺 $e, e' \in E$ が $d(e, e') = 0$ を満たす場合，それらは**隣接**しているという．

頂点 $v \in V$ に対して，距離が k 以内のすべての辺からなる集合を $D^k(v)$ と表す．すなわち，$D^k(v) = \{e \in E \mid d(v, e) \leq k\}$ である．同様に，辺 $e = \{x, y\} \in E$ に対して，$D^k(e) = D^k(x) \cup D^k(y)$ である．定義より，$D^0(v)$ は v に接続する辺の集合であり，$D^1(v)$ は v に直接接続しているか，または，他の辺を介して v に接続している辺の集合である．頂点 v から距離 k 以内の頂点の集合を $N^k(v) = \{w \in V \mid d(v, w) \leq k\}$ である．

2.2 誘導マッチング

辺集合 $M \subseteq E$ の任意の相異なる 2 辺が隣接していないとき，M を**マッチング**という．頂点集合 $U \subseteq V$ に対して，この U と辺集合 $F = \{\{u, v\} \in E \mid u, v \in U\}$ からなる G の部分グラフを，U により**誘導**された部分グラフと呼び，$G[U] = (U, F)$ と表す．$G[U]$ を単に**誘導部分グラフ**ともいう．特に混乱のない限り，頂点集合 $V \setminus U$ によって誘導される誘導部分グラフ $G[V \setminus U]$ を $G \setminus U$ と表す．

マッチング M 中の任意の相異なる 2 辺 $e, e' \in M$ のそれぞれの端点を結ぶ辺がグラフ G に含まれない場合，これを**誘導マッチング**という．すなわち，誘導マッチングとは，任意の $U \subseteq V$ に対して，$G[U]$ で得られる辺集合 F が G のマッチングとなるような辺集合 F である．

本稿で扱う問題を以下に与える．

誘導マッチング列挙問題

入力： グラフ $G = (V, E)$

仕事： G に含まれるすべての誘導マッチング M を列挙せよ

次の節のアルゴリズムを説明するために有用な補題を示す．誘導マッチングの定義より，次の補題 1 が得られる．

補題 1 グラフ $G = (V, E)$ と辺集合 $M \subseteq E$ に対して，M が G の誘導マッチングであるならば，またそのときに限り，M に含まれる任意の相異なる 2 辺 $e, e' \in M$ の距離は，$d(e, e') \geq 2$ を満たす．

証明 辺集合 $M \subseteq E$ が G の誘導マッチングであるならば，定義より任意の相異なる 2 辺 $e, e' \in M$ のそれぞれの端点を結ぶ辺がグラフ G に含まれない．すなわち，e と e' の両方に接続する辺は存在しないので，$d(e, e') \geq 2$ である．逆に，マッチング M について，任意の相異なる 2 辺 $e, e' \in M$ の距離が 2 以上（$d(e, e') \geq 2$）ならば，それらの辺 e と e' の両方に接続する辺はグラフ G に存在しないので，そのような M は誘導マッチングである．□

上の補題 1 より，さらに次の補題 2 を得ることができる．

補題 2 辺集合 $M \subseteq E$ をグラフ G の誘導マッチングとする．このとき，M の任意の部分集合 $M' \subseteq M$ は誘導マッチングである．

証明 $e, e' \in M'$ は，$M' \subseteq M$ より明らかに $e, e' \in M$ である．補題 1 より，$d(e, e') \geq 2$ である．よって，M' は誘導マッチングである．□

2.3 列挙アルゴリズム

列挙アルゴリズムとは，与えられた入力に対して，その解を重複なく出力するアルゴリズムである [10]．一般に列挙アルゴリズムの時間計算量は，入力と出力の両方に関して測られる．特に本稿では，時間計算量を解 1 個あたりの出力にかかる時間で測る．

列挙アルゴリズムが 1 つ目の解を得るまでの時間，および任意の $i \in \{1, \ldots, s-1\}$ 番目の解から $i+1$ 番目の解を得るまでの時間，さらに，最後の s 番目の解を得てから停止するまでにかかる時間のそれぞれが n の多項式で抑えられる場合，これを**最悪時多項式時間遅延**という．また，列挙アルゴリズムが，前処理にかかる時間を除いて，解をすべて出力するまでにかかる時間が n について多項式，かつ s に関して線形である場合，これを**ならし多項式時間遅**

Algorithm 1 G の誘導マッチングを列挙するアルゴリズム

1: **procedure EnumIMatch**$(G = (V, E))$
2: **RecIMatch**$(\emptyset,\ E,\ G)$;
3: **end procedure**
4: **procedure RecIMatch**$(M,\ AddList,\ G)$
5: 入力：誘導マッチング M, 候補リスト $AddList$, グラフ G
6: **if** $AddList = \emptyset$; **then**
7: M を出力する;
8: **return** ;
9: **end if**
10: $AddList \leftarrow AddList \setminus \{e\}$
11: **RecIMatch**$(M \cup \{e\},\ AddList \setminus D^1(e),\ G \setminus D^1(e))$;
12: **RecIMatch**$(M,\ AddList \setminus \{e\},\ G \setminus \{e\})$;
13: **end procedure**

延という．

3. 提案アルゴリズム

本節では，誘導マッチングの列挙を $O(\Delta^2)$ 遅延時間で列挙するアルゴリズムを与える．我々は列挙アルゴリズムを構築するための方法として**分割法** [?] を用いる．分割法は，解空間をある要素を含む部分空間と，含まない部分空間に分割し，これを繰り返すことで解を列挙するアルゴリズムである．本稿では辺に着目して分割法を構築する．

3.1 候補リスト

誘導マッチング M に対して次のような辺集合 $C(M)$ を**候補集合**という．$C(M) = \{e \in E \mid M \cup \{e\}$ は誘導マッチング $\}$ また，候補集合 $C(M)$ の部分集合 $AddList(M) \subseteq C(M)$ を $C(M)$ の**候補リスト**という．

3.2 $O(\Delta^2)$ 遅延アルゴリズム

Algorithm1 では候補集合の部分集合である**候補リスト**を更新しながら誘導マッチングを列挙する．空集合は誘導マッチングであり，空集合に対する候補集合は E なので，空集合と E を引数にして Algorithm1 で誘導マッチングを列挙できることを証明する．

定理 1 Algorithm1 はグラフ中の誘導マッチングをすべて出力し，出力した解はすべて誘導マッチングである．

証明 まず，Algorithm1 の出力は必ず解であることを証明する．**RecIMatch** 関数は $e \in AddList$ を用いて，受け取った誘導マッチング M から誘導マッチング $M \cup \{e\}$ を作る．誘導マッチング $M \cup \{e\}$ を用いて再帰的に **RecIMatch** 関数を呼ぶ．ここで，$AddList$ は $C(M)$ の部分集合になるため，**RecIMatch** 関数が呼び出す関数が持つ $M \cup \{e\}$ は誘導マッチングである．また，**RecIMatch** 関数が $M \cup \{e\}$ を受け取った場合，それ以降の再帰呼び出しでは e を含まない解しか出力しないので，同じ解が 2 度出力されることはない．次に，Algorithm1 がすべての誘導マッチングを出力することを数学的帰納法を用いて証明する．サイズが 0 である誘導マッチングは空集合のみであり，空集合は明らかに出力される．ここで，サイズ k の誘導マッチングがすべて出力されていることを仮定する．サイズ k の誘導マッチングがすべて出力されているのですべてのサイズ k の誘導マッチングに対して **RecIMatch**$(G,$ $AddList,\ M)$ が呼び出されている．ここで，辺集合 F を $F = \{e \in E \mid e \in C(M) \setminus AddList\}$ とする．このとき F 中の任意の辺 e_f を含む任意の誘導マッチング M' ははすでに出力されている．なぜなら，e_f は再帰計算木の先祖が 11 行目すでに選んだ辺だからである．また，明らかに任意の辺 $e \in AddList$ を用いた誘導マッチング $M \cup \{e\}$ は出力される．よって，サイズ k の誘導マッチング M に $C(M)$ 中の任意の辺を加えた誘導マッチングは出力される．ここで，補題 2 より，サイズ k の誘導マッチングを部分集合に含まないサイズ $k+1$ の誘導マッチングは存在しないので，すべてのサイズ $k+1$ の誘導マッチングは出力される． □

Algorithm1 の，再帰呼び出しが作る再帰計算木を用いて，Algorithm1 の時間計算量の評価を行う．関数 **RecIMatch** は 11 行目で $O(\Delta^2)$ 時間かかっている．これは $D^1(e)$ を計算するのに必要な計算時間である．それ以外の計算は定数時間で計算可能であるため，1 回の **RecIMatch** の計算は $O(\Delta^2)$ 時間で計算できる．また，すべての **RecIdMatch** は常に 1 つの解を出力する．そのため，アルゴリズムの遅延は 1 回の **RecIMatch** にかかる時間計算量 $O(\Delta^2)$ と等しい．

4. 長さ 6 以下の閉路を含まないグラフに対するならし定数時間列挙アルゴリズム

3 節では Algorithm1 が $O(\Delta^2)$ 遅延列挙アルゴリズムであることを与えた．本節では入力グラフに制約を加えた場合の提案アルゴリズムの高速化を与える．本節では，長さ 6 以下の閉路を含まない入力グラフを考える．

4.1 修正したアルゴリズム EnumIMatchConst

Algorithm2 に改良したならし定数時間列挙アルゴリズム **EnumIMatchConst** を示す．Algorithm1 では誘導マッチング M に加える辺の選び方を任意に選んでいたが，改良した Algorithm2 ではこの選択の仕方を注意深く行うことで，計算時間を削減する．

Algorithm3 と Algorithm4 にそれぞれ副手続きの **RecNormal** と **RecManyChildren** を示す．仮定として，**RecNormal** の中で集合 $D^0(x)$，$D^1(x)$，$D^1(v)$ は常に $AddList$ との共通部分を表すよう管理する．これは，連結リストと配列を組み合わせて実現できる．この仮定は補題 3 と補題 4 の証明に本質的である．これらの手続きは M に加える辺を次のように選択する．候補リストの辺

Algorithm 2 G の誘導マッチングをならし定数時間で列挙するアルゴリズム

```
procedure EnumIMatchConst(G = (V, E))
    RecIMatchConst(∅, E, G);
end procedure
procedure RecIMatchConst(M, AddList, G)
    入力：誘導マッチング M, 候補リスト AddList, グラフ G
    if AddList = ∅; then
        M を出力する．;
        return ;
    end if
    AddList 中の任意の辺 e = {x, y} を選ぶ．ただし，d(x) ≥ d(y) とする．;
    if |D^1(x)| ≥ |AddList|/2 then
        RecManyChildren(M, x, AddList, G);
    else
        RecNormal(M, x, AddList, G);
    end if
    RecIMatchConst(M, AddList, G);
end procedure
```

$e = \{x, y\}$ を任意に選択し，$d(x) \geq d(y)$ となる端点 x を選ぶ．以降の辺は再び候補リスト中の x と隣接している辺から順に選択する．次に，x と隣接するすべての辺を選択し終えたら，再び任意に辺を候補リスト $AddList$ 中から選択する．$AddList$ が $\emptyset$ になるまで以上を繰り返す．

定理 2　長さ 6 以下の閉路を含まないグラフに対して，Algorithm2 はならし定数時間遅延アルゴリズムで，誘導マッチングを列挙する．以下では定理 2 の証明を与える．

補題 3　1 回の **RecNormal** の繰り返しでかかる計算時間は $O(\left|AddList \cap D^2(x)\right|$ である．

証明　手続き **RecNormal** の行う辺の除去のコストを見積もる．以下では先に仮定したように，**RecNormal** の中で集合 $D^0(x)$，$D^1(x)$，$D^1(v)$ は常に $AddList$ との共通部分を表す．4 行目の $D^1(x)$ の辺の除去と，5 行目の $N(x)$ の除去，9 行目の $D^0(x)$ の除去，$D^1(x)$ の追加，10 行目の $N(x)$ の追加に関しては，それぞれ，各集合のサイズで実行できる．したがってこれらのコストは $O(\left|D^1(x)\right|)$ で抑えられる．注意が必要なのは，6 行目から 8 行目の for 文の中ですべての $v \in N(x)$ に対して，7 行目の $D^1(v)$ 中の辺の除去と，$N(v)$ の頂点の除去を行うコストである．これらの辺と頂点からなる集合は，それぞれ $D^2(x) = \bigcup_{v \in N(x)} D^1(v)$ と $N^2(x) = \bigcup_{v \in N(x)} N^1(v)$ の部分集合であるが，問題なのは $D^2(x)$ 中の枝と $N^2(x)$ 中の頂点が $O(|N(x)|) = O(\Delta)$ 個の異なる v に対して重複して処理される可能性があることである．

そこで，背理法により，異なる $v \neq v'$ とある頂点 $w \in G$ に対して，$w \in N^1(v) \cap N^1(v') \subseteq N^2(x)$ と仮定すると，x, v, v', w を含む長さ 4 以下の閉路が存在する．しかし，仮定より入力グラフ G は長さ 6 以下の閉路を含まないので，矛盾が導かれる．同様にして，異なる $v \neq v'$ とある辺 $e \in AddList$ に対して，$e \in D^1(v) \cap D^1(v')$ に含まれると仮定すると，e の片方または両方の端点と，v，v'，x を含む長さ 5 以下の閉路が存在する．しかし，仮定よりこれは矛盾である．

以上から，上記に述べた辺と頂点の重複した処理はありえないとわかる．よって，除去のコスト総和は $O(\left|AddList \cap D^2(x)\right|)$ のサイズより小さい．よって，合わせると補題 3 が証明された．　□

補題 4　1 回の **RecManyChildren** の繰り返しでかかる計算時間は $O(\left|AddList \cap D^3(x)\right|))$ である．

証明　**RecManyChildren** は，内部で頂点 x の隣接頂点 v に対して 6 行目で **RecNormal** を呼び出す．したがって，補題 3 の証明より，異なる **RecNormal** どうしで重複した処理があれば長さ $6 = 4 + 2$ 以下の閉路が存在する．しかし，これは入力グラフの仮定よりありえない．一方重複した頂点は $D^3(x)$ に含まれる．よって，$O(\left|AddList \cap D^3(x)\right|))$ である．同様に重複した辺の処理があれば単純に考えると，長さ $7 = 5 + 2$ の閉路が存在する．しかし，長さ 7 の閉路が存在しても Algorithm4 は $O(\left|AddList \cap D^3(x)\right|))$ である．

頂点 x を含む長さ 7 の閉路中の頂点で，x と距離が最も大きい頂点を y とする．このとき，$d(x, y) = 3$ である．x と y は長さ 7 の閉路中の頂点なので x と y を端点とするパス長 3 のパス 1 つしかない．はじめに，**RecManyChildren** の 7 行目で，x と y を端点とするパス長 3 のパスに含まれる頂点 u が **RecNormal** の引数になる場合を考える．このとき，y に接続する辺は 6 行目から 8 行目の **RecNormal** で除去される．次に，x と y を端点とするパス長 4 のパスに含まれる頂点 u が **RecNormal** の引数になる場合を考える．ここで，長さ 6 以下の閉路が存在しないことから，u と y を端点とするパスは一意に定まる．したがって，y に接続する辺を高々 1 個しか除去しない．最後に，x と y を端点とするパス長 5 以上のパスに含まれる頂点 u が **RecNormal** の引数になる場合を考える．この場合は y に接続する辺は，u と距離 4 以上なので **RecManyChildren** 中のどの **RecNormal** でも除去されない．これらのことより，頂点 y に隣接する辺は 2 回しか除去をされないことがわかった．

長さ 7 の閉路中の頂点 x と y で $d(x, y) = 3$ のとき，y と接続する辺で重複した処理は高々 2 回しか起こらないことがわかった．$d(x, y) = 2$ である頂点 y のときは，y に接続する辺は 1 回だけ除去される．これは補題 3 と同様に背理法で証明できる．$d(x, y) = 1$ である頂点 y のときは，y に接続する任意の辺は 4 行目で $AddList$ から除去され，9 行目で y に接続し，$D^1(x) \setminus D^0(x)$ である辺が $AddList$ に追加される．これらの処理ではそれぞれの除去または追加する辺のサイズの計算時間で可能である．以上より示された．　□

次に，以下では Uno [8] に基づき，アルゴリズムのならし計算量の解析を与える．論文の自己充足性のために同論

Algorithm 3 $d(x)$ 個の解を出力する．

procedure RecNormal(M, x, $AddList$, G)
  入力： 誘導マッチング M, 頂点 x, 候補リスト $AddList$, グラフ G;
  $V' \leftarrow N(x)$;
  $AddList \leftarrow AddList \setminus D^1(x)$;
  $G \leftarrow G \setminus (N(x) \cup \{x\})$;
  for $v \leftarrow V'$ **do**
    RecIMatchConst($M \cup \{\{x, v\}\}$, $AddList \setminus D^1(v)$, $G \setminus N(v)$);
  end for
  $AddList \leftarrow AddList \cup (D^1(x) \setminus D^0(x))$;
  $G \leftarrow (V(G) \cup N(x), AddList)$;
end procedure

Algorithm 4 $\left|D^1(x)\right|$ 個の解を出力する．

procedure RecManyChildren(M, x, $AddList$, G)
  入力：誘導マッチング M, 頂点 x, 候補リスト $AddList$, グラフ G;
  $V' \leftarrow N(x)$;
  RecNormal(M, x, $AddList$, G);
  for $v \leftarrow V'$ **do**
    RecNormal(M, v, $AddList$, G);
  end for
end procedure

文から導出される証明も記載している．

再帰計算木の任意の内部ノード x の計算時間と，x の子 y の持つ計算時間の総和に対して，式 1 が成り立つならば，補題 5 が成り立つことを証明する．ここで，再帰計算木のノード x の子の集合を $children(x)$, x でかかる計算時間を $T(x)$，ノード x の子でかかる計算時間の総和を $\overline{T}(x)$, 再帰計算木の葉を $Leaf$ とする．

$$\alpha T(x) \leq \overline{T}(x)(\alpha > 1) \tag{1}$$

補題 5 再帰計算木のすべての内部ノードで式 1 を満たすならば，ノード x の子 y は x から，高々 $\frac{T(y)}{\alpha-1}$ しか計算時間を受け取らない．

証明 補題 5 が成り立つことを木の深さに対する帰納法によって証明する．まず，再帰計算木の根とその子について補題 5 が成り立つかどうかを考える．根は親を持たないので親から受け取る計算時間は 0 である．ここで，親 x がそれぞれの子 y に与える計算時間は式 2 とする．

$$\frac{T(x)}{\overline{T}(x)} T(y) \tag{2}$$

$\frac{T(y)}{\alpha} < \frac{T(y)}{\alpha-1}$ なので木の深さが 0 のときは補題 5 は成り立つ．深さが k のときに補題 5 が成り立つことを仮定する．ここで，あるノード x' が持つ計算時間は高々 $T(x') + \frac{T(x')}{\alpha-1}$ である．このとき，x' の子 y' が受け取る計算時間は式 2 より $\frac{T(x)+\frac{T(X)}{\alpha-1}}{\overline{T}(x)} T(y)$ となる．このとき，

$$\begin{aligned}\frac{T(x)+\frac{T(x)}{\alpha-1}}{\overline{T}(x)} T(y) &= \frac{1+\frac{1}{\alpha-1}}{\overline{T}(x)} T(x) * T(y) \\ &\leq \frac{1+\frac{1}{\alpha-1}}{\alpha T(x)} T(x) * T(y) \\ &= \frac{T(y)}{\alpha-1}\end{aligned}$$

となる．よって帰納法により補題 5 は成り立つ． □

再帰計算木の各ノードの計算時間は補題 3 もしくは補題 4 に従う．これからの解析を明らかにするため，各ノードの計算時間は $O(|AddList|)$ であるとする．ここで，再帰計算木のノード n が持つ $AddList$ を $AddList(n)$ とする．まず，親 n の計算時間より，子 $children(n)$ の計算時間の総和について以下の補題 6 が成り立つ．

補題 6 Algorithm2 で，ある頂点 x について，$D^2(x) \geq \frac{|AddList|}{2}$ であるとき，再帰計算木の親の計算時間と，子の計算時間の総和は親の計算時間の方が大きい．

証明 $children(n)$ のある子 m は $AddList(m) = AddList(n) \setminus D^0(x)$ を持つ．このような m は，Algorithm2 の 16 行目で必ず呼び出される．また，**RecNormal** で呼ばれるすべての子の持つ $AddList$ の和集合を考える．**RecNormal** で 2 つ以上の子 $m_1, \ldots, m_k$ を呼び出すとき，**RecNormal** に引数として渡す頂点をそれぞれ $v_1, \ldots, v_k \in N(x)$ とする．このとき，長さ 6 以下の閉路がないため，$D^1(x) = \bigcap_{vinN(x)} D^1(v)$ である．$D^1(x) = \bigcap_{vinN(x)} D^1(v)$ なので，$AddList()$ 和集合は $AddList(n) \setminus D^1(x)$ と一致する．そのため，子の計算時間は少なくとも $2(|AddList(n)| - \left|D^1(x)\right|$ である．これが親の計算時間 $|AddList(n)|$ より大きくときは $D^1(x) \geq \frac{|AddList(n)|}{2}$ を満たすときである．次に，**RecNormal** で子を 1 つしか持たない場合を考える．選んだ辺の端点の次数が大きい頂点 x を選択しているので，x の次数が 1 のときは，x の隣接頂点の次数も 1 である．

親と子 $AddList(n)$ のサイズの差は，辺を誘導マッチングに加えるかどうかにかかわらず 1 である．親と子の計算時間は $|AddList(n)|$ と $2(AddList(n) - 1)$ となり，$|AddList| = 1$ のときにしか親の計算時間が大きくならない．したがって，x の次数が 1 のときは常に $\left|D^1(x)\right| \geq \frac{|AddList(n)|}{2}$ を満たす． □

補題 6 より，$\left|D^1(x)\right| < \frac{|AddList(n)|}{2}$ の場合には $D^1(x)$ は式 1 を満たす．$\left|D^1(x)\right| \geq \frac{|AddList(n)|}{2}$ の場合を考える．このとき，$\left|D^1(x)\right| \geq \frac{|AddList(n)|}{2}$ なので，$O(\left|D^1(x)\right|) = O(|AddList(n)|)$ である．そのため，$|children(n)| = O(\left|D^1(x)\right|)$ を求めるのに $O(|AddList(n)|)$ 時間で求めることができれば，ならし定数時間で解を求められる．ここで補題 4 より，**RecManyChildren** を呼び出すことで $O(\left|AddList \cap D^3(x)\right|)$ 時間で $O(\left|D^1(x)\right|)$ 個の子を求められる．これは $\left|D^1(x)\right| \geq \frac{|AddList(n)|}{2}$ のとき，$O(|AddList|)$ 時間で $O(|AddList|)$ 個の解を求めている．ここで，Algorithm2 の再帰計算木の葉では解を出力するだけなので，定数時間で計算が終了する．そのため，ならし計算量は定数になる．したがって，これより定理 2 が証明された．

Algorithm2 は最悪遅延時間が $O(|E|)$ であるが，これは交代出力法 [7] を用いることで最悪遅延時間を定数にすることができる．

5. まとめ

本稿では，入力グラフに含まれる誘導マッチングの列挙問題について考察し，主結果として，分割法に基づく $O(\Delta^2)$ 遅延時間列挙アルゴリズムを与えた．さらに，長さ 6 以下の閉路を含まないグラフに対してはならし定数時間で列挙するアルゴリズムを与えた．

今後の課題としては，一般のグラフに対して真に $O(d^2)$ より小さい遅延時間で列挙するアルゴリズムの考案や，極大な誘導マッチングの列挙などが挙げられる．

謝辞

国立情報学研究所の宇野毅明先生には，列挙アルゴリズムのならし計算量解析に関して，スライドの提供，及びご教授をいただきました．髙木拓也さんには，本研究に関して貴重なコメントをいただきました．ここに，謝意を表します．

参考文献

[1] Cameron, K.: Induced matchings, *Discrete Applied Mathematics*, Vol. 24, No. 1-3, pp. 97–102 (online), DOI: 10.1016/0166-218X(92)90275-F (1989).
[2] Diestel, R.: Graph Theory, 4th (2010).
[3] Faudree, R. J., Gyárfás, A., Schelp, R. H. and Tuza, Z.: Induced matchings in bipartite graphs, *Discrete Mathematics*, Vol. 78, No. 1-2, pp. 83–87 (1989).
[4] Garey, M. R. and Johnson, D. S.: *Computers and intractability*, Vol. 29, wh freeman New York (2002).
[5] Henning, M. A. and Rautenbach, D.: Induced matchings in subcubic graphs without short cycles, *Discrete Mathematics*, Vol. 315, pp. 165–172 (2014).
[6] Takeaki, U.: A fast algorithm for enumerating bipartite perfect matchings, *Proceeding of Twelfth Annual International Symposium on Algorithms and Computation*, Citeseer, pp. 367–379 (2001).
[7] Takeaki, U.: Two General Methods to Reduce Delay and Change of Enumeration Algorithms (2003).
[8] Uno, T.: Constant Time Enumeration by Amortization, *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, pp. 593–605 (2015).
[9] 宇野毅明：一般グラフの極大マッチングを列挙する高速アルゴリズム，*NII journal*, Vol. 3, pp. 89–97 (2001).
[10] 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).