コードレスサイクルを列挙する線形時間アルゴリズム

宇野 毅明
国立情報学研究所 〒101-8430 東京都千代田区一ツ橋 2-1-2 e-mail: uno@nii. jp

抄録: グラフのサイクル C の頂点集合が誘導する部分グラフが閉路となるとき、C をコードレスサイクルとよぶ。本稿では、与えられたグラフ $G=(V,E)$ のコードレスサイクルを列挙する問題を考え、コードレスサイクル 1 つあたり $O(|E|)$ 時間のアルゴリズムを提案する。また、計算機実験により、枝の密度がそれほど濃くないランダムグラフでは、1 つあたり定数時間しかかからないこと、ランダムグラフのコードレスサイクル数はサイクルの数に比べてはるかに少ないことを示す。

An Output Linear Time Algorithm for Enumerating Chordless Cycles

Takeaki Uno
National Institute or Informatics
2-1-2 Hitotsubashi, Chiyoda-ku, Tokyo 101-8430, JAPAN, e-mail: uno@nii. jp

Abstract: A cycle C of a graph is called a *chordless cycle* if the induced subgraph of its vertices is a cycle. In this paper, we consider a problem of enumerating chordless cycles of a given graph $G=(V,E)$, and propose an algorithm taking $O(|E|)$ time for each chordless cycle. We evaluate the performance of the algorithm by computational experiments for random graphs, and show that the computation time is constant per chordless cycle for not so dense random graphs. We also show the number of chordless cycles is quite smaller than the number of cycles for those random graphs.

1 はじめに

グラフ $G=(V,E)$ に対して, 頂点と枝の列 $v_1,(v1,v_2),v_2,(v_2,v_3),...,(v_{k-1},v_k)$ で $v_i \neq v_j, i \neq j$ が成り立つものを**をパス**という。パスの最初の頂点と最後の頂点に注目し、とくに v_1 と v_k を結ぶパスともよぶ。また、パスの始点 v_1 と終点 v_k が一致しているものを**サイクル**とよぶ。サイクル、あるいはパスの 2 頂点を結ぶ、サイクル・パス自身に含まれない枝を**コード** といい、コードがないパスを**コードレスパス**、コードがないサイクルを**コードレスサイクル**とよぶ。図 1 にコードレスパスの例を示す。サイクルの頂点集合からなる集合族を考えたとき、コードレスサイクルの頂点集合は、この集合族の極小元となる。

本稿では、与えられたグラフ G のコードレスサイクルを列挙する問題を考える。有機化合物の分析を行う研究分野では、化合物の結合が導くグラフのサイクルは環構造とよばれ、化合物の性質を解析する上で重要な働きをする。グラフには多くのサイクルが含まれるため、特にコードレスサイクルのみに注目し、コードレスサイクルを列挙して化合物の性質を解析する研究も行われている。化学シフト値とよばれる指標を予測する際に、コードレスサイクルを列挙して予測制度をあげたという研究もある [2]。

グラフのサイクルを列挙するアルゴリズムの研究は、1970 年代に Read, Tarjan により行われている [1]。彼らは、入力したグラフ $G=(V,E)$ のサイクルを 1 つあたり $O(|V|+|E|)$ 時間で列挙する出力線形時間アルゴリズムを提案している。出力線形時間アルゴリズムとは、出力する解 1 つあたりの計算時間が入力の多項式であるようなアルゴリズムのことをいう。しかし、現在までにコードレスサイクルの列挙を行うアルゴリズムは研究されていない。本稿では、コードレスサイクルを

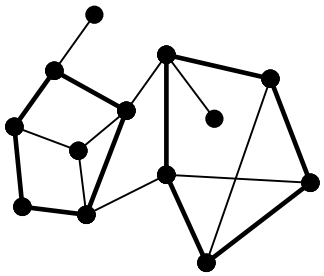

図 1: 左のサイクルはコードレスサイクル、右のサイクルは 2 本のコードを持つ

1 つあたり $O(|V|+|E|)$ 時間で列挙するアルゴリズムを提案する。また、計算機実験により、このアルゴリズムのランダムグラフに対する計算時間を調べ、それと同時に、ランダムグラフがどの程度の数のコードレスサイクルを含むかを調べる。

2 アルゴリズム

グラフ G に対して、G と頂点集合 S に対して、G から S に含まれる頂点と、S の頂点に接続する枝を取り除いて得られるグラフを $G \setminus S$ と表記する。グラフ G の頂点 v に対して、G のコードレスサイクルは、v を含むものと含まないものに分割される。v を含まないコードレスサイクルは、$G \setminus \{v\}$ のコードレスサイクルと 1 対 1 対応するため、$G \setminus \{v\}$ を入力として再帰的に列挙できる。

v に接続する枝 $e = (s, t)$ に対して、v を含むコードレスサイクルは e を含むものと含まないものに分割される。e を含むコードレスサイクル C に対し、$C \setminus \{e\}$ はグラフ $G \setminus \{e\}$ の頂点 s と t を結ぶコードレスパスになっている。よって、G の e を含むコードレスサイクルと、$G \setminus \{v\}$ の s と t を結ぶコードレスパスは 1 対 1 対応する。また、v を含み、$e = (s, t)$ を含まないコードレスサイクルは、頂点 t を含まない（t を含むと、e がコードになるため）。よって、s を含み、$e = (s, t)$ を含まないコードレスサイクルは、$G \setminus \{t\}$ の v を含むコードレスパスとは 1 対 1 対応する。

以上より、 コードレスサイクルの列挙は、コードレスパスの列挙を子問題として、以下のアルゴリズムで行える。そこで 以下では、頂点 s と t を結ぶコードレスパスを列挙するアルゴリズムについて議論する。

Enum_Chordless_Cycle ($G = (V, E)$)
1. G がサイクルを含まなければ、終了
2. $s := G$ のサイクルに含まれる頂点
3. **Call Enum_Chordless_Cycle** ($G \setminus \{s\}$)
4. **For each** s に隣接する頂点 t
5. $\quad G \setminus \{(s, t)\} = (V, E \setminus \{e\})$ の s と t を結ぶコードレスパスを列挙する
6. $\quad G := G \setminus \{t\}$
7. **End for**

以後、簡単のため、頂点 s と t を結ぶコードレスパスを、s-t **コードレスパス**とよぶ。また、パス P を、P に含まれる頂点の列で表す。頂点 s に対して、s に隣接する頂点の集合を $N(s)$ と表記する。以下は、コードレスパスの列挙の中核をなす補題である。

補題 1 グラフ $G = (V, E)$ の頂点 $v \in N(s)$ に対して、P が v を含む s-t コードレスパスであるとき、またそのときに限り、$P \setminus \{s\}$ は、グラフ $G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパスである。

Proof: $P \setminus \{s\}$ が、グラフ $G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパスであるとする。このとき、P は G の s と t を結ぶパスである。また、$P \setminus \{s\}$ は $N(s)$ の頂点を含まないことから、パス P の s に接続するコードは存在しない。よって P は G のコードレスパスである。
P が G の $v \in N(s)$ を含む s-t コードレスパスであるとする。このとき、$N(s)$ の頂点で P に含まれるものは v だけである。それ以外の頂点が P に含まれれば、s とその頂点をつなぐ枝がコードになる。よって、P は $G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパスとなる。∎

この補題より、G の s-t コードレスパスの集合は、$N(s)$ のどの頂点を含むかでグループ分けでき、さらに v を含む s-t コードレスパスは、$G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパスを列挙することにより、列挙できることが分かった。また、$G \setminus (N(s) \setminus \{v\})$ が v-t コードレスパスを含むか含まないかは、以下の補題により簡単に判定できる。

補題 2 グラフ G が s と t を結ぶパスを含むなら、またそのときに限り、G は s-t コードレスパスを含む

Proof: G の s と t を結ぶパスの中で最も枝数の少ないものは、s-t コードレスパスになる。このことより、題意は即座に導かれる。∎

以上の補題から、以下のアルゴリズムが導かれる。なお、Q は今までに解に含まれることが確定している頂点の集合である。

Enum_Chordles_Path ($G = (V, E), s, t, Q$)
1. **If** s が t と隣接 **then output** $Q \cup \{t\}$; **return**
2. **For each** $v \in N(s)$ **do**
3. **If** $G \setminus (N(s) \setminus \{v\})$ に v-t パスが存在する **then**
4. **Call Enum_Chordles_Path** ($G \setminus (N(s) \setminus \{v\}), v, t, Q \cup \{v\}$)
5. **End for**

このアルゴリズムの各反復で、再帰呼び出しを実行している間保持する必要があるものは、定数個の局所変数とグラフから除去した頂点と枝の情報だけである。よって、このアルゴリズムの空間計算量は $O(|V| + |E|)$ である。このアルゴリズムを素直に実装すると、1 反復は $O(|V| + |E|)$ 時間となる。このアルゴリズムは、再帰呼び出しを行わない反復、つまり再帰構造の葉に相当する反復でのみ出力を行う。再帰の深さは最大で $O(|V|)$ となるため、コードレスパス 1 つあたりの計算時間は $O(|V|(|V| + |E|))$ となる。しかし、この計算時間は以下の工夫により短縮することができる。
まず、上記のアルゴリズムを以下のように書き換える。パス P において、頂点 v の次に位置する頂点を $nxt(v)$ と表記する。いくつか新しい変数が導入されているが、アルゴリズムの内容としては等価である。

Enum_Chordles_Path2 ($G = (V, E), s, t, Q$)
1. **If** s が t と隣接 **then output** $Q \cup \{t\}$; **return**
2. $P :=$ G の s-t コードレスパス
3. **Call Enum_Chordles_Path2** ($G \setminus (N(s) \setminus \{v\}), nxt(s), t, Q \cup \{v\}$)
4. **For each** $v \in N(s), v \neq nxt(s)$ **do**
5. **If** $G \setminus (N(s) \setminus \{v\})$ に v-t パスが存在する **then**
6. **Call Enum_Chordles_Path2** ($G \setminus (N(s) \setminus \{v\}), v, t, Q \cup \{v\}$)
7. **End for**

さらにこのアルゴリズムの 2 で求めている s-t コードレスパスを、その反復の親反復で求め、引数として渡すようにすると、以下のアルゴリズムが得られる。

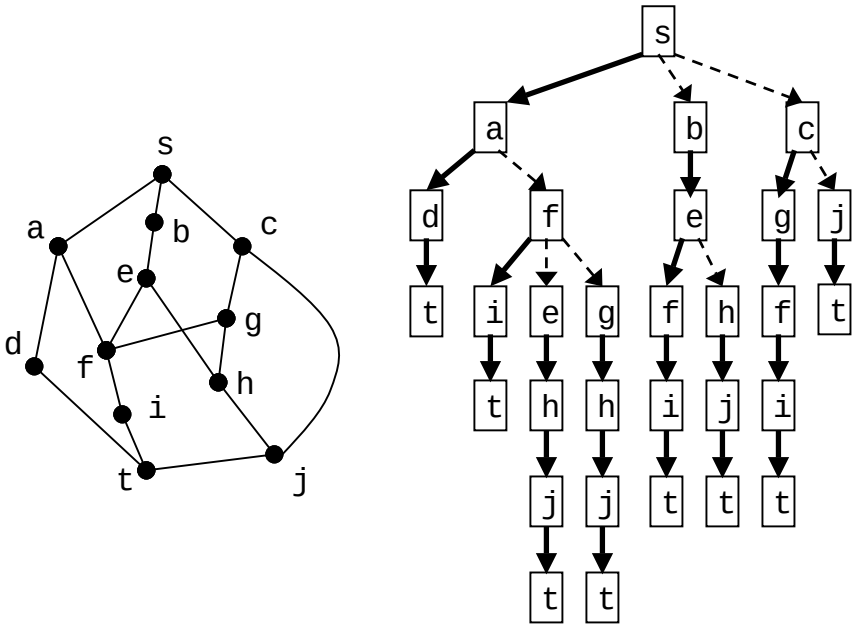

図 2: 左のグラフの s-t コードレスパスを列挙する再帰構造。太線が 2 での再帰呼び出し, 破線がそれ以外の呼び出しに対応

Enum_Chordles_Path3 ($G = (V, E), s, t, Q, P$)
1. **If** s が t と隣接 **then output** $Q \cup \{t\}$; **return**
2. **Call Enum_Chordles_Path3** ($G \setminus (N(s) \setminus \{v\}), nxt(s), t, Q \cup \{v\}, P \setminus \{s\}$)
3. **For each** $v \in N(s), v \neq nxt(s)$ **do**
4. **If** $G \setminus (N(s) \setminus \{v\})$ に v-t パスが存在する **then**
5. $P := G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパス
6. **Call Enum_Chordles_Path3** ($G \setminus (N(s) \setminus \{v\}), v, t, Q \cup \{v\}, P$)
7. **End if**
8. **End for**

このアルゴリズムを実行した際に得られる再帰構造の 1 例を表したものが図 2 である。各頂点が各反復に対応し、矢印の根元の頂点が矢印の先にある頂点を呼び出したという関係になっている。また、2 で行われた再帰呼び出しに対応する矢印は太くなっている。この太い矢印は、いくつかのパスを構成する。このパスを**直系パス**とよぶ。直系パスの一番下の頂点は葉であることから、直系パスの数は葉の数、つまりコードレスパスの数と一致する。また、太線でない矢印の先は直系パスの最も上の頂点であることから、太線でない矢印の数は葉の数を超えないことも分かる。よって、太線でない矢印、つまり 6. で行われた再帰呼び出しの総数はコードレスパスの総数を超えない。

以上の観察から、このアルゴリズムの各反復の計算時間を解析すると、

・ 1 でパスを出力した場合、計算時間は $O(|E| + |V|)$ である。よって、コードレスパス 1 つあたり 1 に費やす時間は $O(|E| + |V|)$ である。
・ 2,6 は定数時間である。
・ 3 のループの回数は s の次数を超えない。そのため、任意の直系パスについて、その反復において 3 で行うループの回数を合計すると、それはグラフの枝数を越えない。
・ 5 でコードレスパスを求める時間は $O(|E| + |V|)$ であるが、上記の観察より、アルゴリズム全体で 5 に費やす時間はコードレスパス 1 つあたり $O(|E| + |V|)$ である。

・ 4 の実行時間は $O(|E|+|V|)$ である。

となり、ボトルネックとなる部分は 4 の計算時間であることが分かる。そこで以下では、4 を短時間で行う方法について解説する。まず、以下の性質を記する。

性質 1 グラフ G が s-t パスを含むなら、またそのときに限り、$G \setminus \{s\}$ はある頂点 $v \in N(s)$ から t へのパスが存在する。∎

この性質を用いて、以下のようなアルゴリズムを考える。このアルゴリズムでは、4 の判定を迅速に行うため、$G \setminus \{N(s)\}$ で t から到達可能な頂点にマークを付けるようにしており、そのマークは 2 で行われる再帰呼び出しで呼び出された反復においてつけている。

Enum_Chordles_Path4 ($G=(V,E), s, t, Q, P$)
1. **If** s が t と隣接 **then output** $Q \cup \{t\}$; **goto** 11
2. **Call Enum_Chordles_Path** ($G \setminus (N(s) \setminus \{v\}), nxt(s), t, Q \cup \{v\}, P \setminus \{s\}$)
3. **For each** $v \in N(s), v \neq nxt(s)$ **do**
4. **If** v に隣接する頂点にマークが付いている **then**
5. 全ての頂点のマークを消す
6. $P := G \setminus (N(s) \setminus \{v\})$ の v-t コードレスパス
7. **Call Enum_Chordles_Path4** ($G \setminus (N(s) \setminus \{v\}), v, t, Q \cup \{v\}, P$)
8. 5 で消したマークを付け直す
9. **End if**
10. **End for**
11. $G \setminus \{s\}$ で t から到達可能な頂点にマークを付ける

11 でのマーク付けでは、$N(s)$ からマークの付いていない頂点のみを経由して到達可能な頂点のみを探索し、マークをつければよい。この作業にかかる計算時間は、最悪で $O(|E|+|V|)$ であるが、1 つの直系パスに注目すると、1 度探索された枝、および 1 度マークを付けられた頂点は 2 度と探索されないため、直系パス 1 つあたりの 11 に費やす計算時間は $O(|V|+|E|)$ で押さえられる。よって、このアルゴリズムのコードレスパス 1 つあたりの計算時間は $O(|V|+|E|)$ である。これより、以下の定理を得る。

定理 1 グラフ $G=(V,E)$ の頂点 s と t を結ぶコードレスパスは、*1* つあたり $O(|V|+|E|)$ 時間で列挙できる。∎

定理 2 グラフ $G=(V,E)$ のコードレスサイクルは、*1* つあたり $O(|V|+|E|)$ 時間で列挙できる。∎

3 計算実験

この節では、前節で示したアルゴリズムの、ランダムグラフでの計算効率を検証し、また、グラフに含まれるサイクルの数とコードレスサイクルの数にどの程度の開きがあるのかを、ランダムグラフの例題について示す。実装は C を用いて行い、計算実験には, Pentium III 750MHz のノート PC を用いた. OS は Linux, コンパイラは gcc である.

まず、以下にランダムグラフに対する計算実験の結果を示す。密度は、グラフの頂点間に枝が存在する確率である。また、各頂点数・密度の組に対して、1 つだけ実験を行ったことを注意しておく。列挙問題は解が多く、そのため非常に解数・計算時間のばらつきが大きくなり、平均を取ることに余り意味がないと考えられるためである。各マスには、対応するグラフに対する、コードレスサイクル 1 万個あたりの計算時間を秒数で示した。なお、サイクルの数が多いものについては、100

万個出力した時点で打ち切り、その時点までの計算時間を記録した。

計算時間

頂点数＼密度	10%	20%	30%	40%	50%	60%	70%	80%	90%
50	0.18	0.12	0.098	0.089	0.082	0.08	0.085	0.1	0.11
75	0.17	0.12	0.099	0.088	0.079	0.074	0.077	0.088	0.1
100	0.17	0.12	0.099	0.09	0.083	0.081	0.089	0.095	0.12
150	0.2	0.12	0.099	0.098	0.077	0.075	0.083	0.103	0.14
200	0.18	0.12	0.1	0.088	0.081	0.078	0.085	0.11	0.17
300	0.19	0.12	0.1	0.087	0.082	0.083	0.091	0.12	0.21
400	0.17	0.12	0.1	0.089	0.08	0.086	0.1	0.15	0.26
600	0.18	0.11	0.12	0.12	0.11	0.1	0.13	0.23	0.42
800	0.2	0.12	0.14	0.13	0.11	0.11	0.13	0.26	0.54
1200	0.23	0.17	0.17	0.13	0.12	0.12	0.15	0.28	1
1600	0.24	0.19	0.14	0.13	0.13	0.14	0.21	0.29	1.3
2400	0.25	0.19	0.17	0.15	0.16	0.16	0.19	0.44	1.4
3200	0.29	0.23	0.2	0.19	0.18	0.2	0.25	0.61	1.79
4800	0.28	0.28	0.27						

枝の密度が 100%に近い場合、ほとんどのコードレスサイクルは、長さ 3 となる。この場合、$|V|$ 個の、長さ 3 のコードレスサイクルを見つけるために $\mathrm{O}(|E|) = \mathrm{O}(|V|^2)$ 時間を使うことになり、枝密度が濃い場合、例えば 80% 以上の場合、コードレスサイクル 1 つあたりの計算時間は枝数に比例する。枝の密度がそれほど濃くない場合は、計算時間は入力したグラフの頂点数・枝数によらずほぼ定数時間である。これは、再帰呼び出しを行うたびに頂点・枝が消去され、再帰木の葉に近い部分の反復は、大きさが定数のグラフを入力しているためと考えられる。また、頂点数が 1000 以上のところで急激に増加しているのは、CPU キャッシュに入力が入りきらなくなったためと思われる。また、グラフの密度が濃いほうが計算時間が若干短くなっているのは、コードレスサイクルの平均的な長さが短くなっているためと思われる。

次に、コードレスサイクル及びサイクルの個数を示す。“-” は数が多すぎて計算できなかった箇所である。なお、頂点数 25 以上のものについては、個数が多すぎるため、サイクルの個数は記載していない。

コードレスサイクルの個数（上段）・サイクルの個数（下段）

＼密度 頂点数	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
10	1	3	14	20	41	45	63	81	120	
	0	1	4	116	352	2302	3697	24016	108906	
15	0	10	34	116	165	193	247	297	350	455
	0	36	1470	613255	6620995	55525881	-	-	-	-
20	1	78	298	523	637	752	771	846	908	1140
	1	56348	9114083	-	-	-	-	-	-	-
25	8	1218	2049	2387	2099	1891	1775	1854	1928	2300
50	64383	395233	267435	146939	82607	49800	34137	23788	18873	19600
75	119379652	69357503	14504338	3679823	1158210	465368	221990	119719	73810	67525
100	-	-	-	40436716	8269935	2395333	877466	393549	199133	161700
150	-	-	-	-	149022507	27483087	6641331	2167686	854209	551300
200	-	-	-	-	-	167408239	30334867	7755298	2466694	1313400
300	-	-	-	-	-	-	-	51043836	11457502	4455100
400	-	-	-	-	-	-	-	-	35154412	10586800

この結果より、グラフのサイクルの数は、頂点数に対する枝数の増加にともない指数的に増える

が、コードレスサイクルの数は、指数的に増加するものの、サイクルほど大きくは増加しないことが観察される。次に、疎なグラフに対する実験結果を示す。このグラフは、頂点数 n のコードレスサイクルの各頂点に対して、その頂点に接続するコードをランダムに付け加えたものである。このグラフの平均次数は 4 である。

頂点数	10	20	30	40	50	60	70	80	90
コードレスサイクル数	12	90	743	5371	89164	853704	4194491	45634757	-
1 万個あたりの計算時間	16.6	3.33	0.53	0.22	0.18	0.2	0.23	0.24	-

この結果より、疎なグラフはとても多くのコードレスサイクルを含むことが分かる。実際に列挙が可能であるものは、頂点数が 100 程度までであろう。しかし、コードレスサイクル 1 つあたりの計算時間はほぼ一定である。

4 まとめ

本研究では, グラフの 2 頂点を結ぶコードレスパス・およびグラフのコードレスサイクルを列挙する、コードレスパス／コードレスサイクル 1 つあたりの計算時間がグラフの枝数に線形であるアルゴリズムを提案した。また、計算実験により、ランダムグラフに対する計算時間は、枝の密度が濃ければ頂点数に比例し、そうでなければほぼ定数であること、およびランダムグラフでのコードレスサイクルの数はサイクルの数ほど爆発的に増えないこと、特に疎なグラフでは爆発的に増えるが、密なグラフではそれほど大きくならないことを確認した。

参考文献

[1] R. C. Read and R. E. Tarjan, “Bounds on Backtrack Algorithms for Listing Cycles, Paths, and Spanning Trees,” Networks **5**, 237-252 (1975).

[2] 佐藤寛子，越野広雪，宇野毅明，中田忠, “CAST/CNMR による 13C-NMR 化学シフト高精度予測- 環構造の効果的な考慮と予測精度の向上”, 第 26 回情報化学討論会講演要旨集，講演番号 10, 2003