A-017

グラフの各頂点を高々2 回まで通る経路数の下界の改善

A Lower Bound of the Number of Graph Walks Visiting Each Vertex at Most Twice

ムハマド・ホリルロハマン‡　　湊真一‡
Muhammad Kholilurrohman　　Shin-ichi Minato

1．まえがき

SAW (Self-avoiding Walk) はグラフ上の同じ頂点を 2 度通らない経路であり，その経路を列挙することは，分子構造のモデル化 [1]にも使われているグラフ理論上の重要な問題である．一般に格子グラフの最短路の数え上げは簡単な公式で求められるが，遠回りを許すと急に難しくなり，網羅的に列挙する以外に有効な方法は知られていない．

SAW 列挙問題に関しては，Knuth らはゼロサプレス型二分決定グラフ (ZDD) [2]を用いた動的計画法による高速な列挙法 simpath [3]を提案している．簡単に説明すれば，simpath 法はグラフの各辺を通るかどうかのすべての組み合わせ(辺の数を m として 2^m 通り存在する全ての通り方のパターン)を圧縮して表現する ZDD を効率よく構築するアルゴリズムである．

本研究では，SAW 問題を拡張し，同じ頂点を高々2 回まで 通ることを許す経路の数え上げを考える．この拡張は，複数の経路を通すことを許すような通信ルータの資源割り当て問題などに応用することができる．

筆者らは以前に，経路を無向グラフとして扱い，各辺の通過回数のパターンを 4 分木を使って高速に数え上げるアルゴリズムを提案した [4]．しかし，この方法では同じパターンを持つ経路が複数存在する場合があり，正確な経路数ではなく，その経路の下界を求めてしまう．本稿では無向グラフの代わりに有向グラフを用いることによって，経路の数をより正確に数えるアルゴリズムを提案する．本手法でも経路数の下界の値を計算していることに変わりはないが，計算機実験により既存手法と比べてより良い下界の値が得られることを確認した．

本稿では，まず川原 [5]がグラフを 2 重化した方法で真の経路数の上界の値を求める方法から始まり，筆者らが以前に提案した 4 分木を用いた方法，そして，本研究で開発した 8 分木を用いた方法について述べ，それぞれの方法の実験結果を示す．

2．準備

例を用いて，本研究で扱う問題を説明する．図 1 の三角形状のグラフの例では，始点 A と終点 C の間の単純経路は AC と ABC の 2 通りしかないが，各頂点を高々2 回まで通ることを許すと $ABABC$, $ABAC$, $ABACBC$, ABC, $ABCABC$, $ABCAC$, $ABCBAC$, $ABCBC$, AC, $ACABC$, $ACAC$, $ACBABC$, $ACBAC$, $ACBC$ と，合計 14 通りもの経路が存在する．

図 2 の 2×2 の格子グラフの例では，最短路が 6 通り，遠回りを許す単純経路(SAW)が 12 通りであることは各頂点を高々2 回まで通る経路は，実に 9,482 通りもある．このような簡単なグラフでさえも膨大な数の経路が存在し，数え上げが難しい問題となる．

‡　北海道大学大学院情報科学研究科

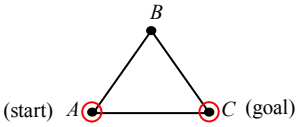

図 1．三角形グラフ

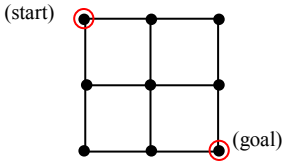

図 2．2×2 の格子グラフ

3．従来の研究と提案手法

グラフの各頂点を高々2 回まで通る経路数は素朴なバックトラック法を使えば求めることができるが，解の個数に比例する時間がかかるため，少し大きいグラフになると時間がかかり過ぎる．

そこで，川原はグラフを 2 重化することで，各頂点を 2 回まで通る経路を SAW の列挙問題に帰着し，これを Knuth が提案した simpath 法を使って経路数を高速に列挙する方法を提案した．

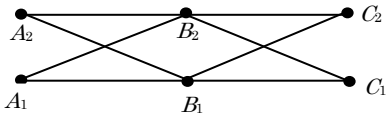

図 3．2 重化したグラ

例を用いてグラフを 2 重化する方法を説明する．例えば，3 つの頂点と 2 本の辺からなる直線状のグラフ A-B-C を 2 重化したグラフは図 3 のグラフになる．元のグラフの始点が A で，終点が C だとすると，2 重化したグラフの始点が A_1 で，終点が C_1 となる．そして，$A_1B_1A_2B_2C_1$ のような SAW の経路が元のグラフの $ABABC$ に対応することになる．

この方法は素朴なバックトラック法よりはるかに高速だが，元のグラフの経路が 2 重化したグラフで複数の経路に対応する場合があるので，重複して数えてしまう場合がある．例えば，$A_1B_1A_2B_2C_1$ と $A_1B_2A_2B_1C_1$ も同じく $ABABC$ を表しているのである．現在，この重複を取り除く方法は見つかっていない．つまり，真の経路数の上界の値しかわからない．

筆者らは以前の研究でグラフを 2 重化せず，simpath 法を拡張して経路を求める手法を提案した．この方法ではグラフを無向グラフとして扱っている点と，各頂点を高々2 回まで通るが，simpath 法とは異なり，各辺の通過回数は 0，1，2，3 回の 4 通りになる．そこで，4^m の組み合わせを表すために ZDD を用いず，代わりに 4 分木を用い，その 4 分木上で動的計画法を用いることにした．

この方法はグラフを 2 重化する方法と同じくらい高速だが，経路を 1 通りに特定できない場合がある．例えば，図 1 のグラフでは $ABCAC$ と $ACBAC$ の 2 つの異なる経路があるが，各辺の通過回数は，辺 AB と BC がそれぞれ 1 回，そして辺 AC が 2 回である．つまり 1 通りの通過回数パターンが複数の経路に対応するため，真の経路数の下界の値しかわからない．1 つの通過回数パタンで経路が何通り

表 1. $N \times N$ グラフに関する経路数の比較

格子サイズ	最短路	各頂点を高々1回まで通過回数パターン (SAW)	各頂点を高々2回まで通過回数パターン (無向グラフ) [4]	各頂点を高々2回まで通過回数パターン (本手法・有向グラフ)	各頂点を高々2回まで真の経路数 (バックトラック法)	2重化グラフでの SAW 経路数 (川原 [5])
1×1	2	2	16	18	22	76
2×2	6	12	2,756	4,334	9,482	967,552
3×3	20	184	6,675,314	19,052,512	112,269,228	1,358,152,089,472
4×4	70	8,512	250,879,228,802	1,569,281,791,868	Time out	Memory out

表 2. $N \times 3$ グラフに関する経路数と計算時間

格子サイズ	各辺の通過回数パターン（無向グラフ）	動的計画法状態数	計算時間（秒）	各辺の通過回数パターン（本手法・有向グラフ）	動的計画法状態数	計算時間（秒）	真の経路数（バックトラック法）	計算時間（秒）
1×3	812	604	0.02	1,140	1,337	0.06	2,100	0.02
2×3	71,252	6,463	0.19	140,892	16,292	0.74	457,712	1.30
3×3	6,675,314	24,531	0.74	19,052,512	57,796	2.69	112,269,228	2,940.43
4×3	666,976,269	64,742	1.95	2,656,295,182	143,372	6.83	-	Time out
5×3	68,515,294,778	139,429	4.32	374,178,848,172	293,045	14.09	-	Time out

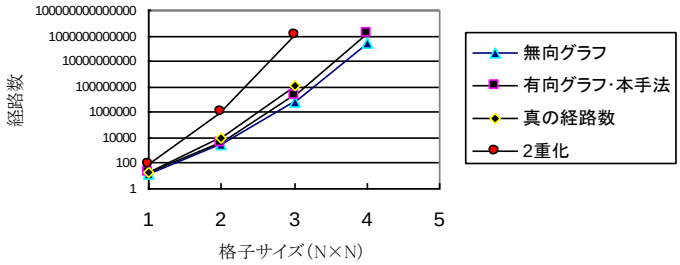

図 4. $N \times N$ グラフに関する経路数の比較

あるかは，オイラー路の数え上げが必要で，簡単な問題ではなさそうである．

グラフを無向グラフとして扱った方法ではうまく区別できない経路があるため，筆者らはグラフを有向グラフとして扱うことにし，本稿で新しい手法を提案する．本手法では各辺に対して図 5 の 8 つの通り方があるので，

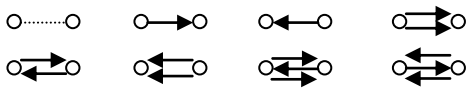

図 5. 有向グラフに現れる各辺のパターン

8^m の組み合わせを 8 分木で表した．例えば，図 1 のグラフで各辺の通過回数パターンの違いで区別できなかった経路 *ABCAC* と *ACBAC* が本手法では区別できるようになった．図 1 のグラフを有向グラフとしてみたとき，*ABCAC* で使った辺が *AB*, *BC*, *CA*, *AC* であるが，経路 *ACBAC* で使った辺は *BA*, *CB*, *AC*, *AC* となり，両方の経路の区別がつく．

しかし，この方法でも区別できないパターンが存在する．例えば，図 1 のグラフの *ABCAC* と *ACABC* が両方とも同じ辺 *AB*, *BC*, *CA*, *AC* を使っていて，この方法では経路を区別することができない．各辺の通過回数パターンで区別できた経路が必ず本手法でも区別できるので，従来の手法の経路数の下界の値を改善できる．

4. 実験結果

本実験では Intel® Core(TM) i3-2330M 2.20GHz, 2GB Memory, Ubuntu 13.10 32-bit の計算機を用いて実験を行った．プログラムは C++言語で実装した．

表 1 と表 2 に格子グラフに対する実験結果を示す．いずれもグラフの左上頂点から右下頂点に至る経路を数えている．動的計画法で辺を処理する順は，グラフの行ごとに上から下の順とした．

表 1 と図 4 から分かる通り，各頂点を通る回数を高々1 回から 2 回にすると，経路の数が爆発的に大きくなる．そして本手法で求めた値が経路数の下界の値で，2 重化グラフの手法で求めた値が経路数の上界の値となるが，両者の差は非常に大きい．

表 2 から本手法は，バックトラック法に比べてはるかに高速により膨大な数の経路を計算できていることが分かる．動的計画法の状態数は，パターン総数に比べると非常に小さい．本手法で求めた経路数の下界の値は筆者らが提案した 4 分木を用いた方法で求めた下界の値よりも精度がよいが，真の経路数との差はまだ大きい．

5. おわりに

本研究でグラフを有向グラフとして扱ったときに求めた経路数の下界の値は筆者らによるグラフを無向グラフとして扱って求めた数よりも改善しているが，それでもまだ真の経路数との差が大きい．今後は効率よく正確に数え上げることができるアルゴリズムを研究したい．

参考文献

[1] 白井伸宙: “拡張アンサンブル法を用いた self-avoiding walk の数の推定,” 京都大学数理解析研究所講究録，No. 1848-10, Oct. 2012.
[2] S. Minato: “Zero-Suppressed BDDs for Set Manipulation in Combinatorial Problems,” In Proc. of 30th ACM/IEEE Design Automation Conf. (DAC'93), pp. 272-277, Jun. 1993.
[3] D. E. Knuth: “The Art of Computer Programming,” volume 4, Binary Decision Diagrams, ZDDs to represent simple paths, pp.122, Addison-Wesley, 2008.
[4] ムハマド ホリルロハマン, 湊真一: “グラフの各頂点を高々2 回まで通る経路の数え上げ,” 電子情報通信学会 2014 総合大会, DS-1-10, pp. S-19～S-20, Mar. 2014.
[5] 川原純: “フロンティア法解説講演資料,” http://www-erato.ist.hokudai.ac.jp/~jkawahara/frontier/, 2012.