修士論文 要旨

Fully dynamic algorithms for graph connectivity and spanning forests under general vertex updates

（グラフ連結性と全域森問題に対する頂点更新に対応した完全動的アルゴリズム）

数理情報学専攻 48-166224　中村 健吾
指導教員　定兼 邦彦 教授

1　はじめに

現実世界に現れるグラフ構造は，時間とともに徐々に変化するものが多い．この構造の変化は，グラフ理論では頂点や辺の追加・削除としてモデル化され，そのような変化 (**更新**) が一つずつ起こるようなグラフを**動的グラフ**とよぶ．動的グラフの上では，通常のグラフアルゴリズムは更新のたびに一から走らせる必要があり，効率が悪い．そこで，動的グラフの上で効率よく動く**動的グラフアルゴリズム**を考える．動的グラフアルゴリズムはここ 2,30 年で広く研究され，全点対間最短パスや (近似) 最大マッチング，最小カットなど多くの問題 (の動的版) に対して，アルゴリズムが提案されてきた．

しかし、動的グラフに関する多くの研究は**辺更新**，すなわち辺の追加や削除しか取り扱っていない．そこで本研究では，頂点の削除および，頂点とその隣接辺を一度に追加するという更新ができる設定を考える．これらの更新をここでは (**拡張**) **頂点更新** (general vertex update) とよぶ[*1]．頂点更新は辺更新を模倣できる上に，グラフの頂点数が変化しうるという辺更新設定にはない特徴があり，辺更新より広範な変化を扱える．

一方で，辺更新設定でも頂点更新設定をある程度模倣できる．まず前処理の際に，孤立頂点をいくつかグラフに追加しておく．すると，各頂点追加は，追加した孤立頂点のうち 1 つを追加頂点とみなすことで対応でき，1 回の頂点追加や削除は，それに隣接する $\mathrm{O}(n)$ 本[*2]の辺の追加や削除で対応できる．しかし，この手法では最初に追加した孤立頂点の数しか頂点追加できず，任意個の頂点追加に対応することはできない．そこで，本研究では任意個の頂点追加に対応した頂点更新動的グラフアルゴリズムを開発することを目指す．

本研究では，動的グラフ問題の中で，**動的 *DFS* 問題**および**動的連結性・全域森問題**を扱った．これらはいずれも，動的グラフの中でも特に基本的な問題である．

[*1] 頂点更新を考えた設定には他に**部分グラフ設定**とよばれるものもあるが，この設定も (拡張) 頂点更新設定で模倣できる．その意味で呼び名に “general” という単語を入れている．

[*2] 本稿では n はグラフの頂点数，m は辺数を表すものとする．

2　動的 DFS

深さ優先探索 (Depth-First Search, DFS) はよく知られたグラフ探索の手法で，線形時間 ($\mathrm{O}(m+n)$ 時間) で実行できて，探索の結果として ***DFS* 木** (**森**) とよばれる根付き全域木 (森) が生成される．DFS 木は有向グラフの強連結成分分解や無向グラフの頂点間の 2-(辺) 連結性判定など，様々なグラフアルゴリズムの道具として使われる．本研究で扱うのは，その動的グラフ版である**動的 *DFS* 問題**であり，その目標は，グラフ G への頂点や辺の追加，削除のたびに G の DFS 木 (森) T を一つ出力できるデータ構造を構築することである．

この問題は，2015 年以前に開発されたアルゴリズムはいずれも，辺追加か辺削除のどちらかしか対応せず，しかも複数回の更新に渡ったコストの平均 (**ならし計算量**) は通常の DFS を繰り返し行うより良いものの，1 更新あたり最悪でかかるコスト (**最悪計算量**) は通常の DFS と変わらないという問題点があった．一方，2016 年に Baswana ら [1] が開発した無向グラフ動的 DFS アルゴリズムは，頂点や辺の追加・削除全てに対応し，かつ 1 更新あたりの最悪計算量が $\mathrm{o}(m)$ になり，両方の問題を解決した．のちにその時間計算量は Chen ら [2] により少し改善された．

しかし，この両者はいずれも必要な空間が $\mathrm{O}(m \log^2 n)$ ビットであるという欠点を抱えている．これはグラフの隣接リスト表現の $\mathrm{O}(\log n)$ 倍であり，巨大なグラフには適さない．そこで本研究では，Baswana ら [1] のアルゴリズムの空間の圧縮を図ると共に，更新時間計算量の削減に取り組んだ．

その結果が表 1 である．本研究で提案したアルゴリズム A,B は，頂点・辺の追加・削除が全てある設定の下ではいずれも既存のアルゴリズムより更新計算量が小さく，辺の追加のみの設定でも Chen らの結果 [2] に肉薄している．さらに，必要な空間が $\mathrm{O}(\log n)$ 倍圧縮されていて，アルゴリズム B ではさらに定数倍圧縮されている．提案アルゴリズムはいずれも，頂点・辺の追加・削除がある下で任意の 2 頂点間の *2*-(**辺**) **連結性**という高度な連結性を答えるデータ構造へ応用できる．

3 動的連結性・全域森

連結性クエリとは，入力された 2 頂点間がグラフで繋がっているかを判定するクエリで，グラフ中のクエリの中でも最も基礎的なものの一つである．グラフの更新とクエリのオンライン列を処理する問題を**動的連結性問題**，それを解くデータ構造を**動的連結性オラクル**とよぶ．一方，グラフ更新の下で更新の度にそのグラフの全域森に対する変化を出力する問題を**動的全域森問題**，それを解くデータ構造を**動的全域森データ構造**とよぶ．大まかに言えば，全域森を維持できれば連結性クエリは解けるので，動的連結性問題は動的全域森問題に帰着できる．一方で，DFS 森は全域森でもあるので，2 章の動的 DFS アルゴリズムで動的全域森問題は解ける．

動的連結性・全域森問題は動的グラフの中でも最も基礎的な問題で，辺更新設定の下では様々なアルゴリズムが提案されてきた．現在は，辺更新の下では乱択の種類 (決定的か Las Vegas か Monte Carlo か) や更新計算量の種類 (ならしか最悪か) によってそれぞれ異なる更新計算量上界が存在するほか，更新計算量とクエリ計算量の下界の関係も研究されている．しかし，頂点更新設定の下で，しかも任意個の頂点追加ができるようなアルゴリズムはほとんど提案されていない．Baswana ら [1] や本研究の動的 DFS アルゴリズムはその例外である．

そこで本研究では，辺更新のみに対応する動的連結性オラクルや動的全域森データ構造を，頂点更新かつ任意個の頂点追加に対応するよう変換する効率的な枠組みを提案した．この枠組みを用いて，Wulff-Nilsen [5] や Thorup [4] のならし更新計算量データ構造や，Kapron ら [3] の最悪更新計算量乱択データ構造[*3]を実際に変換した．変換前と変換後のアルゴリズムの更新・クエリ計算量を表 2 に示す．

提案したアルゴリズム X,Y,Z の更新計算量はいずれも l に依存している．l はある時点のグラフに対し計算された DFS 森の葉の数である．これは n 以下であり，1 頂点更新は $\mathrm{O}(n)$ 回の辺更新に相当するため，アルゴリズム X や Y の計算量は，元のデータ構造をそのまま使うより遅くはない．さらに，比較的密なグラフでは $l = \mathrm{o}(n)$ になりやすい．具体的には，ランダムグラフ (ER モデル) の下では平均辺数が $\omega(n \log n)$ であるとき高確率で $l = \mathrm{o}(n)$ となることが知られている．従って，比較的密なグラフに対しては各提案アルゴリズムの更新計算量は小さくなることが期待できる．連結性クエリに関してもアルゴリズム Y,Z では定数クエリ時間を達成した．繰り返しになるが，提案アルゴリズムは元のデータ構造をそのまま頂点更新に使うのと異なり，任意個の頂点追加に対応しているという利点がある．さらに，[4, 5] を含め多くの辺更新ならし計算量データ構造には初期グラフが空でなければならないという制約があるが，提案アルゴリズムにそのような制約は無い．

[*3] このデータ構造は Monte Carlo であり，僅かな誤り確率をもつ．連結性クエリに対し「連結」と答えた場合は必ず連結であるが，「非連結」と答えた場合でも確率 $1/\mathrm{poly}(n)$ 以下で連結である．変換後のアルゴリズム Z もこの性質をもつ．

表 1. 各動的 DFS アルゴリズムの必要な空間と 1 更新あたりの最悪時間計算量.

	空間 (bit)	更新毎最悪時間計算量 (頂点/辺の追加/削除)	更新毎最悪時間計算量 (辺追加のみ)
[1]	$\mathrm{O}(m \log^2 n)$	$\mathrm{O}(\sqrt{mn} \log^{2.5} n)$	$\mathrm{O}(n \log^3 n)$
[2]	$\mathrm{O}(m \log^2 n)$	$\mathrm{O}(\sqrt{mn} \log^{1.5} n)$	$\mathrm{O}(n)$
A	$\mathrm{O}(m \log n)$	$\mathrm{O}(\sqrt{mn} \log^{0.75+\varepsilon} n)$	$\mathrm{O}(n\sqrt{\log n})$*
B	$(3m + \mathrm{o}(m)) \cdot \log n$	$\mathrm{O}(\sqrt{mn} \log^{1.25} n)$	$\mathrm{O}(n \log n)$

*$m = \mathrm{O}(n^2/\sqrt{\log n})$ ならば $\mathrm{O}(n \log^{\varepsilon} n)$ まで高速化可能

表 2. 変換前の辺更新対応連結性・全域森データ構造と変換後の頂点更新対応連結性・全域森アルゴリズムの更新計算量とクエリ計算量．左列の更新計算量は 1 辺あたり，右列は 1 頂点あたり．

対応する辺更新データ構造	提案した頂点更新アルゴリズム
[5]	**X**
$\mathrm{O}(\frac{\log^2 n}{\log\log n})$/update	$\mathrm{O}(\sqrt{m} \log^{1.25} n + l\frac{\log^2 n}{\log\log n})$/update
$\mathrm{O}(\frac{\log n}{\log\log n})$/query	$\mathrm{O}(\frac{\log n}{\log\log n})$/query
動的連結性オラクル・ならし計算量・決定的	
[4]	**Y**
$\mathrm{O}(\log^2 n)$/update	$\mathrm{O}(\sqrt{m} \log^{1.25} n + l \log^2 n)$/update
$\mathrm{O}(\frac{\log n}{\log\log n})$/query	$\mathrm{O}(1)$/query
動的全域森データ構造・ならし計算量・決定的	
[3]	**Z**
$\mathrm{O}(\log^5 n)$/update	$\mathrm{O}(\sqrt{ml} \log^{2.75} n)$/update
$\mathrm{O}(\frac{\log n}{\log\log n})$/query	$\mathrm{O}(1)$/query
動的連結性オラクル・最悪計算量・Monte Carlo	

参考文献

[1] S. Baswana, S. R. Chaudhury, K. Choudhary, and S. Khan. Dynamic DFS in undirected graphs: breaking the $O(m)$ barrier. In *Proc. SODA*, pages 730–739, 2016.

[2] L. Chen, R. Duan, R. Wang, and H. Zhang. Improved algorithms for maintaining DFS tree in undirected graphs. arXiv:1607.04913v2, 2016.

[3] B. M. Kapron, V. King, and B. Mountjoy. Dynamic graph connectivity in polylogarithmic worst case time. In *Proc. SODA*, pages 1131–1142, 2013.

[4] M. Thorup. Near-optimal fully-dynamic graph connectivity. In *Proc. STOC*, pages 343–350, 2000.

[5] C. Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Proc. SODA*, pages 1757–1769, 2013.