平成 30 年度
卒業論文

タイトル

名古屋工業大学　情報工学科

所属: 泉研究室

平成 26 年度入学　26115142 水谷　龍誠

目 次

第1章

はじめに

1.1 研究背景

あるグラフ $G = (V, E)$ が与えられたとき, グラフを非連結な二つの頂点集合に分割する小さな”セパレータ”の存在について考えられることがある. このセパレータの存在は, 高速なグラフアルゴリズム設計において非常に重要である. しかし, 一般のグラフに対して最小サイズのセパレータを求める問題は NP 困難であるため, 小さなセパレータを求める近似アルゴリズムがこれまで研究されている.

現在, 一般のグラフに対して小さなセパレータを近似する集中型のアルゴリズムはいくつか知られているが, 分散環境におけるアルゴリズムはまだあまり知られていない. この論文では, 既存の近似アルゴリズムをベースとする分散セパレータ近似アルゴリズムを提示する.

1.2 研究結果

第2章

準備

アルゴリズムは分散システムにおけるCONGESTモデルの下で動作する.n個の計算機ノード集合Vと通信リンクの集合Eである無向ネットワークグラフ$G=(V,E)$があるとする.CONGESTモデルにおいて計算機はラウンドに従って同期して動作を行う.各ラウンドにおいてノードは最大$O(\log n)$ビットのメッセージを各隣接ノードに送信,各隣接ノードからメッセージの受信,無制限の内部計算を行う事が出来る.
分散設定では,ノードは一意なID($O(\log n)$ビット)を持っており,隣接ノードのIDは知っているものとする.

グラフ$G=(V,E)$に対して,Dはその直径を表すとする.頂点$s,t\in V$に対して,sとtを結ぶ経路をs-tパスと呼ぶ.特に,いくつかのs-tパスに対して,それぞれのパスが同じ頂点を共有しないとき,これらをs-t点素パスと呼ぶ.また,sとtの間にパスが存在しなくなるように取り除かれた頂点集合をs-tカットと呼ぶ.

s-t点素パスとカットについて以下の定理が示されている.

定理 1 (メンガーの定理)**.** グラフGに対して,s,tのカットの最小値はs-t点素パスの最大本数と等しい

s-t点素パスの最大本数について考える.メンガーの定理によって各点素パスから頂点を一個づつ適切に取り出すと,その頂点集合はグラフGの最小s-tカットになる.もちろん,任意に各パスから頂点を取り出すとs-tカットにならない場合もある.この各パスから任意に取り出した頂点集合を特にスライスと呼び,以下のように定義する.

定義 1. Gをグラフとし,$s,t\in V$とする.$\{f_1,\ldots,f_k\}$をGにおける点素s-tパスの集合とする. この時,すべての$1\leq i\leq k$に対して,$w_i\in f_i$、$s\neq wi\neq t$であれば,$(f_1,\ldots,f_k)$に

関してタプル $(w_1, \ldots, w_i)$ をスライスと呼ぶ. X を V の任意の部分集合とし、$s, t \notin X$ とする。$G[V \backslash X]$ に s-t パスがない場合,X は s と t を分離すると言う。s と t を分離するスライスをカットと呼ぶ.

スライスの集合は,s に対する相対的な"近さ"によって部分的に並べることができる. ある s-t 点素パスの頂点について s 方向を前方,t 方向を後方として向きと順序を定義する.

定義 2. $\{f_1, \ldots, f_k\}$ を s-t 点素パスの集合とする. $U = (u_1, \ldots, u_k)$ および $W = (w_1, \ldots, w_k)$ を $(f_1, \ldots, f_k)$ に関するスライスとする. すべての $1 \leq i \leq k$ について、u_i は f_i における w_i の前方または $u_i = w_i$ とする. この時 U は W より s に近いと言い,$U \preceq W$ と書く.$1 \leq i \leq k$ に対して, さらに $u_i \neq w_i$ ならば,U は W より厳密に s に近いと言い,$U \prec W$ と書く. 同様に、W は U より厳密に t に近いと言う. 便宜上, タプル $(s, s, \ldots, s)$ と $(t, t, \ldots, t)$ についても同様に上記を定義. したがって, 例えば,$(s, s, ...s)$ はどのスライスよりも s に近いと言える." $\preceq$ " は一般的な順序の合計を定義するものではない.

カットを取り除くことでは G を少なくとも 2 つの連結成分に分離する. あるカットによって分離した各連結成分に対して以下の定義をする.

定義 3. U を任意のカットとする.s を含む $G[V \backslash U]$ の連結成分の頂点セットとして $V_s(U)$ を定義し,t を含む $G[V \backslash U]$ の連結成分の頂点セットとして $V_t(U)$ を定義し,V_r を $G[V \backslash U]$ の残りの連結成分の頂点集合の和集合として定義する. すなわち、$V_r(U)$ は s も t も含まない連結成分である.（したがって、$V_r(u)$ は空であり得る）

2.1 セパレータ

頂点集合 V の分割 $(A, S, B) : V = A \cup S \cup B$ について,$|A| \leq \frac{2}{3}|V|$ かつ $|B| \leq \frac{2}{3}|V|$ であり, 集合 A, B 間に辺が存在しないとき, 集合 S を G のセパレータと呼ぶ.$\frac{2}{3} \leq \alpha < 1$ とし, 上記の分割 (A, S, B) に対して $|A|, |B| \leq \alpha|V|$ の時,S を α-セパレータと呼ぶ.

セパレータ, またはカットはある値 α に関して, セパレータやカットによって分割された全ての頂点集合のサイズが高々$\alpha|V|$ 以下である時, セパレータやカットについてバランスが取れていると言うことにする.

2.2 Subgraph Aggregation

Ghaffari と Haeupler [1] のショートカットフレームワークは, 制限されたグラフ族の分散アルゴリズムを設計する上で有益であることが証明されている. さらに, このショートカットフレームワークの制約をさらに弱めて改良されたタスクが Li [2] によって示されている. この論文では, このショートカットのフレームワークの内部動作には触れずに Subgraph Aggregation として定義されているタスクを利用する.

定義 4 (Subgraph Aggregation [2] [3])**.** $G = (V, E)$ をネットワークグラフとし,$\mathcal{P} = (P_1, ...P_{|\mathcal{P}|})$ をパートの集合, 各 P_i について H_i を P_i のノード上の G の連結部分グラフとする. 必ずしもグラフ $G[P_i]$ から誘導されるとは限らない. 各部分グラフ H_i について、$V(H_i)$ 内のすべてのノードが部分グラフ H_i 内の隣接ノードを認識し、それ以外は何も知らないと仮定する. すべてのノード $v \in \bigcup_i P_i$ が O(log n) ビットの整数 x_v を持ち,$\oplus$ を長さ O（log n）の整数に作用する結合関数とする.P_i 内の各ノードは値 $\bigoplus_{v \in P_i} x_v$、すなわち P_i 内のすべての値 x_v の集合 $\oplus$ を知りたいとする. このようなタスクをオペレーター $\oplus$ における Subgraph Aggregation と呼ぶ.

SA ラウンドを,Subgraph Aggregation（SA）において各パートがその収集値 $\bigoplus$ を学習する一回の反復とする.SA は特別なグラフ構造に関して以下の定理が成り立つ.

定理 2. 結合演算子 $\oplus$ について、Q_G がグラフ G とその直径 D に依存するパラメータである場合,$\tilde{O}(Q_G)$ ラウンドで Subgraph Aggregation 問題を解くことができる.

全てのグラフ G : $Q_G = O(\sqrt{n} + D)$

種数 g のグラフ G : $Q_G = O(\sqrt{g+1}D)$

木幅 k のグラフ G : $Q_G = \tilde{O}(kD)$

H をマイナーとして含まないグラフ G : $Q_G = \tilde{O}(f(H) \cdot D^2)$,$f$ は H にのみ依存する関数

2.3 s-t 点素パス探索アルゴリズム

グラフプロパティの一種である木幅と呼ばれる値 k に対して以下の補題が示されている.

補題 1 ([2])**.** 木幅が高々kのグラフ$G=(V,E)$と二つの頂点$s,t\subseteq V$を与えると,k点素 s-t パスを見つけるか, サイズk以下の s-t ノードカットを$\tilde{O}(k^{O(1)}D)$ラウンドで出力する分散アルゴリズムが存在する. 前者の場合, 各ノードは自身がパス上にあるかどうかを知っており、そうであれば、そのパス上のその前方と後方を知る. 後者の場合、k点素パスが存在しないという事実と, カットに含まれるかどうかを各ノードが知っている.

アルゴリズムの詳細については触れないが, 点素パスの探索ステップ自体には木幅の値は関わっていない. 木幅が影響するのは SA ラウンドに関してのみである. 上記の補題と定理 2 より一般のグラフに置き換えた以下の系が成り立つ.

系 1. 補題 1 のグラフGを一般のグラフに置き換えると, 最大k本の s-t 点素パスは$\tilde{O}(k^{O(1)}(\sqrt{n}+D))$ラウンドで見つけることができる. 終了時, すべての頂点は自身がパス上にあるかどうか, そしてもし存在すればそのパスの前方と後方を知っている.

また, [2] では以下の系が示されている.

系 2 ([2])**.** 木幅kのグラフ$G=(V,E)$で,$G[U]$が連結であるような集合$U\subseteq V$および 3 つの互いに素な頂点集合$A,B\subseteq V-U$および$X\subseteq U$を考えると,$\tilde{O}(k^{O(1)}D)$ラウンドで内部ノードが$G[U]-X$に属するk個のA-B点素パスを見つけることができるか,k個の点素パスは存在しないと結論付ける. 前者の場合, すべての頂点は自身がパス上にあるかどうか, そしてもし存在すればそのパスの前と後を知っています. 後者の場合, 全ての頂点は,k個の点素パスが存在しないという事実を知っている.

この系についても, 先程と同様にして以下のように修正する.

系 3. 系 2 のグラフを一般のグラフ$G=(V,E)$に置き換える.$G[U]$が連結であるような集合$U\subseteq V$および 3 つの互いに素な頂点集合$A,B\subseteq V-U$および$X\subseteq U$を考えると,$\tilde{O}(k^{O(1)}(\sqrt{n}+D))$ラウンドで内部ノードが$G[U]-X$に属する最大$k$本の$A$-$B$点素パスを見つけることができる. 終了時, すべての頂点は自身がパス上にあるかどうか, そしてもし存在すればそのパスの前方と後方を知っている.

第3章

既存手法の概要

この章では,Brandt と Wattenhofer の近似アルゴリズム [4] のプロセスを簡略化して説明する.

アルゴリズムのアプローチは最大 s-t フローに基づいている. 小さいセパレータを含むグラフ G が与えられ, そのセパレータを跨ぐ”異なる側”に頂点 s と t があるとする. この時, メンガーの定理により,s-t 点素パスの最大数も少なくなる.

アルゴリズムは s-t 点素パスの最大本数の集合を計算することから始める. 最大本数 k 本の各パスから 1 つの頂点を取ることによって,k 個の s-t カットを見つける. これらの頂点は s と t を分離するように選ぶ必要がある.s-t カットは, メンガーの定理によって存在することは明らかである.

次に, 二分探索法により全ての s-t カットの中で現時点で最良のバランスがとれた 2 つのカットを決定する. これら 2 つのカットのうちの 1 つが十分にバランスが取れていれば, 目的の小さいセパレータが発見できたことになる. バランスが取れていないとき, 連結成分が 2 つの s-t カットで分断されているされていると見なす. そして,s を含む連結成分を新たな頂点 s' に,t を含む成分を新たな頂点 t' に縮約する. 新しく得られたグラフのすべての s'-t' 頂点カットも G の s-t カットであり, さらに上記の 2 つの s-t カットよりもバランスが取れていることが示されている. そのため, 本数がある値 K に等しい s-t カットを得るまで点素パスを探索し, 最良のバランスのとれた s-t カットをいくつか発見し, 頂点集合を縮約する上記のプロセスを繰り返す.

$\frac{2}{3} \leq \alpha < 1$ の時、s と t を分離する最大で K のサイズの α-セパレータの存在を考える. 上述した反復プロセス終了時, バランスが少なくとも α と同等であるカットを生じない

場合,α-セパレータの少なくとも 1 つの頂点が実行された縮約のうちの 1 つに関与しているとみなす. したがって, プロセス全体を反復することによって, セパレータを探索する. ここまで, セパレータの「異なる側にある」頂点 s と t を見つけることができると仮定していたが, 実際には s と t を一様ランダムに選択している為, 上記の反復プロセスを適用してから得られた最大の連結成分に対して再度縮約アルゴリズムを反復する.
このようにして, 与えられたグラフが小さいセパレータを含んでいれば, 小さいセパレータを見つけるためのほぼ線形の実行時間を得る.

上記のプロセスをまとめると, ある値 $\frac{2}{3} \leq \alpha <$ と $0 < \varepsilon < 1 - \alpha$ について以下の結果が示される.
グラフ $G = (V, E), n \in V, m \in E$ について,G がサイズ K の α-セパレータを含んでいるとき, アルゴリズムはサイズ $O(\varepsilon^{-1}K^2 \log^{1+o(1)} n)$ の $(\alpha + \varepsilon)$-セパレータを高確率で $O(\varepsilon^{-1}K^3 m \log^{2+o(1)} n)$ 時間で見つけることができる.

第4章

分散実装

この章では, セパレータを近似するための分散アルゴリズムを提示する. 具体的には, [4] の 5 つのアルゴリズムそれぞれを分散ネットワークで実行できるように適応させていく.

4.1 カット探索アルゴリズム

あるスライスが与えられた時, そのスライスに”近い”カットについて定義する.

定義 5. $(f_1, \ldots, f_k)$ に関して U をスライスとする. $U \preceq X$ であり, かつ $U \preceq X' \prec X$ を満たすカット X' が存在しないようなカットを X とする. この時 $U^+ := X$ を定義する.
上記のような X が存在しない場合は、$U^+ := (t, t, \ldots, t)$ とする.
同様に,$Y \preceq U$ かつ $U \preceq Y' \prec Y$ を満たすカット Y' が存在しないようなカットを Y とする. この時,$U^- := Y$ を定義する.
上記のような Y が存在しない場合は $U^- := (s, s, \ldots, s)$ とする.

上記のカット U^+ を求めるアルゴリズムを以下に示す.
s-t 点素パスは計算済みであり, 点素パスの最大本数 k とする. パス上の各ノードは自身が点素パスの要素であることと, パス上の順序によるインデックスを知っているとする.
まず, 点素パス上の各ノードが隣接ノードに対してパス上でのインデックスをブロードキャストする.G から点素パスの要素を除くことで得られる各連結成分を B とし, 各 B 内で受け取ったパスのインデックスの最大値を収集する.B 内のブロードキャストを受け取った各ノードは, ブロードキャストを行わなかった隣接ノードが B 内のノードであると認識できるため,SA によって通信することができる. 最大 k 本のパスに対する最大値を

収集するので, 高々kSA ラウンドで収集完了する.
次に, スライスU よりも前方にあるパス上のノードは, あるノード集合X を示す ID をセットし, スライスU を除く隣接ノードに ID(X) をブロードキャストする.ID(X) を受け取ったB をX の部分集合とする.
以降,X にB を追加し,X 内で各パスの最大インデックス値を収集. 最大値が更新されたとき, そのインデックスを持つノードよりも前方にあるノードをX に追加して, 隣接ノードに ID(X) をブロードキャストしてB をX に追加して再度最大値を収集する反復を, 最大値が更新されなくなるまで繰り返す.X 内での最大値収集はkSA ラウンドで完了する.

反復終了後, 最終的な各パスの最大値が求めるU^+ である. また, 終了後のノード集合x は$V_s(U^+)$, 隣接ノードがU^+ のみである各B は$V_r(U^+)$, それ以外のノードは$V_t(U^+)$ である.

反復回数は, 最悪ケースの場合最も長いパスの長さ分だけ繰り返される. このパスの長さをℓ とするとアルゴリズムは$O(\ell k)$SA ラウンドで終了する.U^- の計算は,U^+ のアルゴリズムの前方後方を入れ替えることで同様に計算できる.

4.2 縮約アルゴリズム

まず系 1 を使用して最大本数k 本の s-t 点素パス集合$\{f_1, \ldots f_k\}$ を求める. 求めた点素パス上の各ノードは, 以下に示す補題によりパス上での順序に関するインデックスや前後方の収集値を$O(\log n)$SA ラウンドで知ることができる.

補題 2 (経路収集 [2])**.** G の有向パス$P = \{v_1, ..., v_\ell\}$ を考える. ここで、各ノードv_i はそのパスの前方ノードと後方ノードを知っている.$O(\log n)$SA ラウンドにおいて, 各ノードv_i はi の値と経路内のそのインデックスを知ることができる. さらに, 各ノードv_i が整数x_i と共通の結合演算子$\oplus$ を知っていれば、各ノードv_i に先頭集約$\bigoplus_{j\leq i} xj$ と後尾集約$\bigoplus_{j\geq i} x_j$ を学習させることができる.

次に, 最良のカット候補であるS, T を二分探索によって求める.$g(s), g(t)$ を, 縮約する集合の要素数とする. 以下に [4] の探索アルゴリズムを提示する.

Algorithm 最良のカット候補 S

初期設定:
$g(s), g(t) \in \mathbb{N}$
s-t 点素パス $f_1, ..., f_k$
ℓ_i : f_i の長さ ((s) は 0 番目)
$v_{ij}, 0 \leq j \leq \ell_i$ を f_i 上の j 番目の頂点とする
$w_i = v_{i1} (1 \leq i \leq k)$
valid := **false**

1: **for** $i = 1$ to k **do**
2: $c := 0$ ▷ f_i の始まり
3: $d := \ell_i$ ▷ f_i の終わり
4: **while** $d \neq c + 1$ **do**
5: $e := \lceil \frac{c+d}{2} \rceil$ ▷ f_i 二分探索
6: $W := w_1, ..., w_{i-1}, v_{ie}, w_{i+1}, ..., w_k$ ▷ i 番目を動かす
7: **if** $W^+ \neq (t, t, ..., t)$and$|V_s(W^+) + g(s) \leq |V_r(W^+) \cup V_t(W^+)| + g(t)$ **then**
8: $c := e$
9: valid := **true**
10: **else**
11: $d := e$
12: **end if**
13: **end while** ▷ $O(\log n)$ 回反復
14: $w_i := v_{ic}$
15: **end for**
16: **if** vaild **then**
17: **return** $(w_1, ..., w_k)$
18: **else**
19: **return** $(s, s, ..., s)$
20: **end if**

Algorithm 最良のカット候補 T

初期設定:
$g(s), g(t) \in \mathbb{N}$
s-t 点素パス $f_1, ..., f_k$
ℓ_i : f_i の長さ ((s) は 0 番目)
$v_{ij}, 0 \leq j \leq \ell_i$ を f_i 上の j 番目の頂点とする
$w_i = v_{i1} (1 \leq i \leq k)$
valid := **false**

1: **for** $i = 1$ to k **do**
2: $c := 0$ ▷ f_i の始まり
3: $d := \ell_i$ ▷ f_i の終わり
4: **while** $d \neq c + 1$ **do**
5: $e := \lfloor \frac{c+d}{2} \rfloor$ ▷ f_i 二分探索
6: $W := w_1, ..., w_{i-1}, v_{ie}, w_{i+1}, ..., w_k$ ▷ i 番目を動かす
7: **if** $W^- \neq (t, t, ..., t)$and$|V_t(W^-) + g(t) \leq |V_r(W^-) \cup V_s(W^-)| + g(s)$ **then**
8: $d := e$
9: vaild := **true**
10: **else**
11: $c := e$
12: **end if**
13: **end while** ▷ $O(\log n)$ 回反復
14: $w_i := v_{id}$
15: **end for**
16: **if** vaild **then**
17: **return** $(w_1, ..., w_k)$
18: **else**
19: **return** $(s, s, ..., s)$
20: **end if**

それぞれの while の反復は,f_i を二分探索する反復なので $O(\log n)$ 回で終了する. 前節で示した U^+, U^- の結果と合わせて,S, T はそれぞれ $O(\ell k^2 \log n)$SA ラウンドで計算できる.

カット S, T に対し,$M_s = V_s(S) \cup S, M_t = V_t(W) \cup W$ とする.$V_s(S), V_t(T)$ は U^+, U^- の計算時に同時に求めてある.$M_s \cap M_t \neq \emptyset$ または M_s, M_t 間に辺がある時, 結果をこの時の S, T として縮約アルゴリズムは終了する. この判定は S, T の各ノードが隣接ノードに各集合の ID をブロードキャストすることで判定できる.
そうでなければ,M_s, M_t をそれぞれ単一のノード s, t に縮約したとみなして反復を繰り返す. これは, 系 2 によって M_s, M_t 間の点素パスを求めることが可能であるため, 問題なく実行することができる.

縮約アルゴリズムは, 最大 K 回反復するので, アルゴリズムは $\tilde{O}(K^2(K^{O(1))} + \ell)(\sqrt{n} + D)$ ラウンドで終了する.

4.3 近似アルゴリズム

これまでのアルゴリズムを適用してセパレータ近似アルゴリズムを提示する. このアルゴリズムは, 縮約アルゴリズムを適用して得られたカットによってグラフ G を分割し, 分割後の各連結成分の中で一番ノード数が多い連結成分を H として縮約アルゴリズムを繰り返し適用する. 縮約アルゴリズムによって得られたカットについてバランスが取れているとき, 反復は終了する. 反復終了後, 各反復で得られたカットと s,t の和集合が求めたい近似セパレータとなる.

まず,$\frac{2}{3} \leq \alpha < 1, 0 < \varepsilon < 1-\alpha$ とし,G にサイズ K のセパレータが存在すると仮定する. 初期インスタンス $H := G, S' := \{\}$ とする.S' が最終的に求める近似セパレータ集合となる.

$|V(H)| \leq (\alpha+\varepsilon)|V|$ の時, 反復は終了する. 各反復において 2 つのノード $s,t \in V$ を一様ランダムに選択する.

選択された s,t が隣接しているかまたは $s=t$ の時,$H[V(H)\backslash(\{s\} \cup \{t\})]$ にあるノード数が一番多い連結成分 C を計算する.s,t が各連結成分のノード数は SA ラウンドでそれぞれ計算可能. 得られた C を H として反復を繰り返す.

s,t が離れているとき, 入力を (H,K,s,t) として縮約アルゴリズムを実行, 得られたカットを (S,T) とする. $H[V(H)\backslash(S \cup T \cup \{s\} \cup \{t\})]$ にある頂点数が一番多い連結成分 C を計算する. 各連結成分の頂点数は SA ラウンドでそれぞれ計算可能. 得られた C を H として反復を繰り返す.

以上の操作により, 求めたいセパレータが構成できた.

4.4 アルゴリズムの評価

近似アルゴリズムは,s,t を一様ランダムに選択して動作する, ラスベガス法による乱拓アルゴリズムである. [4] の補題 28 より, アルゴリズムは高確率で $O(\varepsilon^{-1}K\log^{1+o(1)} n)$ 回の反復で終了する. 各反復において,S' のサイズは最大 $2K+2$ 増える. これは, 縮約アルゴリズムの結果 (S,T) とノード s,t の個数を意味する. 従って, アルゴリズムはサイズ $O(\varepsilon^{-1}K^2\log^{1+o(1)} n)$ の $(\alpha+\varepsilon)$ セパレータを $\tilde{O}(\varepsilon^{-1}K^3(K^{O(1)}+\ell)(\sqrt{n}+D))$ ラウンドで計算できる.

第5章

まとめと今後の課題

第6章

謝辞

本研究の機会を与え，数々の御指導を賜りました泉泰介准教授に深く感謝致します．また，本研究を進めるにあたり多くの助言を頂き，様々な御協力を頂きました泉研究室の学生のみなさんに深く感謝致します．

参考文献

[1] Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks ii: Low-congestion shortcuts, mst, and min-cut. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 202–219. SIAM, 2016.

[2] Jason Li. Distributed treewidth computation. *arXiv preprint arXiv:1805.10708*, 2018.

[3] Bernhard Haeupler, D Ellis Hershkowitz, and David Wajc. Round-and message-optimal distributed graph algorithms. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing*, pages 119–128. ACM, 2018.

[4] Sebastian Brandt and Roger Wattenhofer. Approximating small balanced vertex separators in almost linear time. In *Workshop on Algorithms and Data Structures*, pages 229–240. Springer, 2017.