平成 30 年度
卒業論文

平衡分離集合を近似する分散アルゴリズム

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

ある n 頂点のグラフ $G = (V, E)$ が与えられたとき, 頂点の部分集合 $S \subset V$ が，それを取り除くとグラフが非連結な二つ以上の部分グラフに分けられるとき，その頂点集合 S は G の分離集合と呼ぶ．特に，S を取り除いた後におけるグラフの各連結成分がいずれも高々 αn 個の頂点しか含まないとき S をグラフ G の α-平衡分離集合と呼ぶ．α が定数（$\alpha = \Theta(1)$）であるようなサイズの小さい分離集合を発見できるとき，元のグラフ G に対する何らかの問題を，分離後の αn 頂点サイズのグラフにおける問題に分割統治法を用いて帰着できる場合がしばしば存在する．実際に，平面的グラフのような，小さい平衡分離集合をもつグラフクラスに対して，そのような手法で高速なグラフアルゴリズムが設計可能である．一般のグラフに対して，最小サイズの α-平衡分離集合を求める問題は NP 困難であることが知られているが [1]，いくつかの近似アルゴリズムの存在が知られており，現時点では近似率 $O(\log n)$ のアルゴリズムが存在する []．

本研究では，特に分散システム上の平衡分離集合発見問題を考える．すなわち，ネットワークのトポロジを問題の入力とみなし，その上での小さい平衡分離集合を発見するアルゴリズムを考える．分散システムのモデルとしては，単位時間当たりに 1 通信リンクあたりに伝送可能な情報量を $O(\log n)$ ビットに制限した *CONGEST* モデルを考える．同モデルにおいては通信リンクの帯域が制限されているため，ある 1 台の計算機がネットワーク全体のトポロジ情報を収集して，その上で逐次アルゴリズムを実行するというアプローチでは最悪時に $\Omega(m)$ ラウンド (m はグラフの辺数) の実行時間を必要とする．本研究での提案アルゴリズムの基本アイデアは，Brandt と Wattenhofer らによる，一般のグラフに対する平衡分離集合計算のための近似アルゴリズム [2] を分散システム上に実現することである．同アルゴリズムはサイズ K の α-平衡分離集合を持つ

ような入力インスタンスに対して，サイズ $O(\varepsilon^{-1}K^2\log^{1+o(1)} n)$ の $(\alpha+\varepsilon)$-平衡分離集合を計算する．提案アルゴリズムは，このアルゴリズムと同等の近似性能を持つ解を $\tilde{O}(\varepsilon^{-1}K^3(K^{O(1)}+\ell)(\sqrt{n}+D))$ ラウンド (で合ってる？) で出力する．XXXXXXXXX のところは適切に埋める．．平衡分離集合を近似的に計算する *CONGEST* モデル上の分散アルゴリズムはこれまでに知られておらず，提案手法は非自明な計算時間上界を持つ初めてのアルゴリズムである．

1.1 論文構成

本論文は全 5 章で構成される.2 章ではアルゴリズムのために必要な諸定義や補題等の準備を行う．3 章では既存の集中型でのアルゴリズムの概要を説明し，4 章では提案する分散アルゴリズムの説明を行う.5 章で研究のまとめを記述する.

第2章

準備

2.1 分散システム

2.1.1 *CONGEST* モデル

本稿で考える分散システム (*CONGEST* モデル) は，単純無向連結グラフ $G = (V, E)$ により表現される．ここで V はノードの集合であり，E は通信リンクの集合である．また，$|V| = n$ とする．*CONGEST* モデルでは計算機はラウンドに従って同期して動作するものとする．1 ラウンド内で，隣接頂点へのメッセージ送信，隣接頂点からのメッセージ受信，内部計算 (多項式時間) をおこなう．各辺は単位ラウンドあたり $O(\log n)$ ビットを双方向に伝送可能であり，各ノードは異なる接続辺に異なる内容のメッセージを同一ラウンドに送信可能である．各ノードはまた，$O(\log n)$ ビットの自然数値よる ID が付与されており，自身の隣接頂点全ての ID を既知であるとする．各ノードはグラフのトポロジに関する事前知識を持たないものとする．なお，本稿を通じて，G の直径を D で表すこととする．

2.1.2 同時部分収集問題

本研究で検討される分散アルゴリズムの動作は，同時部分収集問題（Subgraph Aggregation: SA）と呼ばれる抽象化された集合通信操作をプリミティブとして設計されている．同時部分収集問題は Ghaffari と Haeupler らによる低競合ショートカット [3] の枠組みにおいて初めて提案されており．具体的には以下のような問題として定義される．

定義 1 (Subgraph Aggregation [4] [5]). $G = (V, E)$ をネットワークとし,$\mathcal{P} = (P_1, ...P_{|\mathcal{P}|})$ を G の連結な部分グラフの集合 (パートと呼ぶ) とする．ただし各パートの頂点集合は互いに素であるとする．各 P_i について、そこに含まれるノード $v \in V(P_i)$ は自身および自身の隣接ノードがどのパートに属しているかは既知であるとする．すべてのノード $v \in \bigcup_i P_i$ が $O(logn)$ ビットの値 $x_v \in X$ を持つとし，$\oplus$ を X 上の可換かつ結合的な任意の 2 項演算とする．演算 $\oplus$ に対する同時部分収集問題は，P_i 内の各ノードに値 $\bigoplus_{v \in P_i} x_v$、すなわち P_i 内のすべての値 x_v に対する ($\oplus$ の下での) 和を通知する問題と定義される．

本稿を通じて，分散アルゴリズムの実行時間評価を，そのアルゴリズムが駆動する SA の回数 (SA ラウンド) により評価することとする．SA の 1 回の実行に必要な (通常の意味での) ラウンド数は一般のグラフに関しては $O(\sqrt{n} + D)$ ラウンドであり，この実行時間はタイトである．すなわち，SA ラウンド数 $s(n)$ の分散アルゴリズムは，自動的にラウンド数 $O(s(n)(\sqrt{n} + D))$ の分散アルゴリズムを与える．一方で，いくつかの特殊なグラフクラスを仮定した場合，より少ないラウンド数で SA を実現するアルゴリズムが構成可能であることが知られている．具体的には，以下の定理が存在する．

定理 1 (Ghaffari and Haeupler- [3], Haeupler et al. [**?**], Li et al. [], Kitagawa et al. []). 可換かつ結合的な任意の演算子 $\oplus$ について、各グラフクラスに対して以下に示すラウンド数で SA を実現するアルゴリズムが存在する．

一般のグラフ：$O(\sqrt{n} + D)$ ラウンド,

種数 g のグラフ：$O(\sqrt{g + 1}D)$ ラウンド,

木幅 k のグラフ：$\tilde{O}(kD)$ ラウンド,

k-コーダルグラフ：$O(kD)$ ラウンド,

グラフ H をマイナーとして含まないグラフ：$\tilde{O}(f(H) \cdot D^2)$ ラウンド．ここで f は H の大きさにのみ依存する関数.

本稿で提案するアルゴリズムは，これらのグラフクラスに対して，それに対応する SA の実現ラウンド数に応じた実行時間で動作することが保証される．

2.2 α-平衡分離集合問題

頂点集合 V の分割 $(A, S, B) : V = A \cup S \cup B$ について, 集合 A, B 間に辺が存在しないとき, 集合 S を G の分離集合と呼ぶ.$0 < \alpha < 1$ とし, 上記の分割 (A, S, B) に対して $|A|, |B| \leq \alpha|V|$ の時,S を α-平衡分離集合と呼ぶ．また，V 中の頂点 s，t が与えられたとき，$s \in A$，$t \in B$ であるような分離集合 S を，G の s-t 点カットと呼ぶ.

s-t 点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. U を任意の s-t 点カットとするとき，$V \setminus U$ により誘導される部分グラフにおいて s，t を含む連結成分の頂点集合をそれぞれ $V_s(U)$，$V_t(U)$ とする．また，$V_r(U)$ をそれ以外の連結成分すべてからなる部分グラフと定義する．なお，$V_r(u)$ は空グラフとなる場合もあり得る.

2.3 s-t 点素パスとメンガーの定理

$s, t \in V$ が与えられたとき，s と t を結ぶパスで，s, t 以外に互いに頂点を共有していないパスの集合 P を s-t 点素パスと呼ぶ．s-t 点素パスと分離集合 (s-t 点カット) について，以下の定理がよく知られている.

定理 2 (メンガーの定理). グラフ G に対して, 最小サイズの s-t 点カットにおける頂点数と，s-t 点素パスの最大本数は等しい.

与えられた s-t 点素パス集合に対して，スライスと呼ばれる概念を定義する.

定義 2. 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 s-t 点素パス集合とする. この時, すべての $1 \leq i \leq k$ に対して,$w_i \in f_i$、$s \neq wi \neq t$ であるような頂点列 $(w_1, \ldots, w_i)$ を,$(f_1, \ldots, f_k)$ に関してのスライスと呼ぶ. X を V の任意の部分集合とし、$s, t \notin X$ とする。$G[V \setminus X]$ 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.

定義より明らかに，任意の s-t 点素パス P に対して，任意の s-t カットは P 中の各パス内の頂点を少なくとも一つ含む．すなわち，任意の s-t カットは P のスライスを含む．また，メンガーの定理は，最大本数の s-t 点素パス集合を取ったとき，そのスライスで s-t 点素カットとなるようなものが存在することを示している．s-t 点素パス集合 P に関

するスライスは，P 中の各パスに対して，その前方側（s に近い側）と後方側（t に近い側）を自然に定義する．$(f_1, \ldots, f_k)$ に関する 2 つのスライス $U = (u_1, \ldots, u_k)$ および $W = (w_1, \ldots, w_k)$ に対して，すべての $1 \leq i \leq k$ について、u_i が f_i における w_i の前方に存在するか，または $u_i = w_i$ が成立するとき，U は W より s に近いと言い，$U \preceq W$ と記述する．任意の i（$1 \leq i \leq k$）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，$U \prec W$ と記述する．このあたり，図解を入れては？

本稿では，既存研究 [4] で示されている，本数最大の s-t 点素パスを発見する分散アルゴリズムをサブルーチンとして利用する．このアルゴリズムの仕様は以下の定理により形式化される．

補題 1 ([4])**.** ネットワーク $G = (V, E)$ 中の二つの頂点 $s, t \subseteq V$，および自然数 k に対して，k 本の点素 s-t パス，あるいはサイズ k 未満の s-t 点カットを $\tilde{O}(k^{O(1)})$ SA ラウンドで出力する *CONGEST* モデル上の分散アルゴリズムが存在する. このアルゴリズムは各ノードに，自身が点素パスにあるいは点カットに含まれるかどうかを出力する．また，点素パスが出力される場合，自身を含むパスにおける自身の直前，直後のノードについての情報も併せて出力する．

定理 1 より一般のグラフにおける以下の系が成り立つ.

系 1. ネットワーク $G = (V, E)$ 中の二つの頂点 $s, t \subseteq V$，および自然数 k に対して，k 本の点素 s-t パス，あるいはサイズ k 未満の s-t 点カットを $\tilde{O}(k^{O(1)}(\sqrt{N} + D))$ ラウンドで出力する *CONGEST* モデル上の分散アルゴリズムが存在する. このアルゴリズムは各ノードに，自身が点素パスにあるいは点カットに含まれるかどうかを出力する．また，点素パスが出力される場合，自身を含むパスにおける自身の直前，直後のノードについての情報も併せて出力する．

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 [2] 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. 小さい平衡分離集合を含むグラフ G が与えられ, その分離集合を跨ぐ”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 点カットの中で最良の平衡分離集合の候補である 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 つが元のグラフに対しても十分に平衡であれば, 目的の小さい平衡分離集合が発見できたことになる. そうでないとき, 連結成分が 2 つの s-t 点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' 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 カットよりも平衡であ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 s-t 点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 を分割するサイズ K の α-平衡分離集合の存在を考える. 上述した反復プロセス終了時, 少なくとも α-平衡分離集合と同等であるカットを生じない場合,α-平衡分離集合の少なくとも 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,α-平衡分離集合を探索する.

ここまで, 平衡分離集合の, 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最小の α-平衡分離集合を含んでいれば, 小さい近似平衡分離集合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 $G = (V, E), |V| = n, |E| = m$ について,G がサイズ K の α-平衡分離集合を持つとき, アルゴリズムはサイズ $O(\varepsilon^{-1}K^2 \log^{1+o(1)} n)$ の $(\alpha + \varepsilon)$-平衡分離集合を高確率で $O(\varepsilon^{-1}K^3 m \log^{2+o(1)} n)$ 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, [2] 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 3. $(f_1, \ldots, f_k)$ 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 [4])**.** 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 は整数 x_i と共通の二項演算子 $\oplus$ に対して, 各ノード v_i に前方側収集 $\bigoplus_{j\leq i} xj$ と後方側収集 $\bigoplus_{j\geq i} x_j$ 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 [2] 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**

for $i = 1$ to k **do**
  $c := 0$ ▷ f_i の始まり
  $d := \ell_i$ ▷ f_i の終わり
  while $d \neq c + 1$ **do**
    $e := \lceil \frac{c+d}{2} \rceil$ ▷ f_i 二分探索
    $W := w_1, ..., w_{i-1}, v_{ie}, w_{i+1}, ..., w_k$ ▷ i 番目を動かす
    if $W^+ \neq (t, t, ..., t)$and$|V_s(W^+) + g(s) \leq |V_r(W^+) \cup V_t(W^+)| + g(t)$ **then**
      $c := e$
      valid := **true**
    else
      $d := e$
    end if
  end while ▷ $O(\log n)$ 回反復
  $w_i := v_{ic}$
end for
if vaild **then**
  return $(w_1, ..., w_k)$
else
  return $(s, s, ..., s)$
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**??**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 $\tilde{O}(K^2(K^{O(1))} + \ell)$SA 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平衡分離集合を近似する分散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よってグラフが充分に小さくなったとき, 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平衡分離集合となる.

まず,$\frac{2}{3} \leq \alpha < 1, 0 < \varepsilon < 1-\alpha$ とし,G にサイズ K の α-平衡分離集合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平衡分離集合となる.

$|V(H)| \leq (\alpha+\varepsilon)|V|$ 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, 求めたい平衡分離集合が構成できた.

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. [2] 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 $(\alpha+\varepsilon)$-平衡分離集合を $\tilde{O}(\varepsilon^{-1}K^3(K^{O(1)}+\ell)$SA ラウンドで計算できる.

第5章

研究のまとめ

本研究では,SA を使用して, サイズ $O(\varepsilon^{-1}K^2\log^{1+o(1)} n)$ の $(\alpha+\varepsilon)$-平衡分離集合を高確率で $\tilde{O}(\varepsilon^{-1}K^3(K^{O(1)}+\ell)$SA ラウンドで求めることができた.

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] Thang Nguyen Bui and Curt Jones. Finding good approximate vertex and edge partitions is np-hard. *Information Processing Letters*, 42(3):153–159, 1992.

[2] 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.

[3] 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.

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

[5] 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.