属性付きコミュニティ検索におけるビームサーチの高速化

著者	真次 彰平, 塩川 浩昭, 北川 博之
内容記述	第12回データ工学と情報マネジメントに関するフォーラム (DEIM2020) 日時：2020年3月2日～4日　オンライン開催
雑誌名	第12回データ工学と情報マネジメントに関するフォーラム (DEIM2020)
発行年	2020-03-02
URL	http://hdl.handle.net/2241/00160267

DEIM2020 D1-4(day1 p23)

属性付きコミュニティ検索におけるビームサーチの高速化

真次　彰平† 　塩川　浩昭†† 　北川　博之††

† 筑波大学 システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1-1-1
†† 筑波大学 計算科学研究センター　〒 305–8577 茨城県つくば市天王台 1-1-1
E-mail: †matsugu@kde.cs.tsukuba.ac.jp, ††{shiokawa,kitagawa}@cs.tsukuba.ac.jp

あらまし　コミュニティ検索はグラフ構造から問合せに対して特定の評価関数の値を最大化するコミュニティを見つけ出すグラフ分析手法である．従来提案されたビームサーチに基づくコミュニティ検索手法は実世界の大規模なグラフに対して大きな実行時間を要する．これは，ビームサーチの過程において，部分グラフの冪集合を近傍解として繰り返し列挙する必要があることに起因する．そこで本稿ではビームサーチにおける近傍解の高速列挙法を提案する．具体的には，評価関数の極値を導出することで評価関数の高くなる近傍解のみを選択的に列挙する．これにより将来的に棄却される可能性の高い近傍解を早期に枝刈りすることを可能とし，コミュニティ検索の高速化を実現する．本稿では 2 つの実データセットに対して実行速度とコミュニティ検索精度の評価を行い，提案手法は先行研究と同程度の精度を示しつつ，最大約 500 倍高速であることを確認した．

キーワード　コミュニティ検索，グラフマイニング

1 序　　論

Twitter や Facebook 等のソーシャルメディアの普及に伴い膨大なデータが生成されており，大量のデータから知識を抽出するデータ分析が注目されている．特に，各データエンティティをノード，データエンティティ間の関係をエッジとして表現するグラフ分析は，ネットワーク構造を持つデータから有益な情報を見つけ出す技術として重要である．とりわけ，近年の情報の多様化に伴い，ノードが属性値を持つ属性付きグラフ [1] が多く見られるようになった．例えば，SNS のユーザをノードとした場合，ノードにはユーザの出身地や，職業，興味のあるトピックといった属性値が付与されており，属性付きグラフの一種と考えることができる．ゆえに，属性付きグラフに対する分析技術は幅広い分野において重要な要素技術となっている．

グラフに対する代表的な分析手法のひとつに，コミュニティ検索がある．コミュニティ検索はグラフの中からクエリとなるノード集合を選択し，選択したクエリに対して類似度の高いノード集合 (コミュニティ) を見つけ出す技術である．従来のグラフクラスタリング手法 [2–4] ではグラフ全体からいくつかのコミュニティを検出するのに対して，コミュニティ検索ではクエリを含む密な部分グラフをひとつ見つけることが目的となる．コミュニティ検索手法はデータベース分野を中心に多くの手法が提案されてきた．代表的な手法として，グラフの構造的な特徴に基づく手法 [5–7] が挙げられる．これらはノードの次数や truss 構造 [6] などの数を判定する指標を用いてヒューリスティックにコミュニティを検索する．しかしながら，これらの手法は属性付きグラフ中のノードの持つ属性を考慮しておらず，属性を考慮したコミュニティを検索することができない．

この問題を解決するために，Huang らによって属性付きコミュニティ検索が提案された [8]．Huang らによる手法は指定したノード近傍の指定した属性を多く持つコミュニティを検索する．具体的には，クエリノードとクエリ属性からなるクエリに対し，属性付きコミュニティ検索はクエリノード近傍のクエリ属性に類似するコミュニティを見つけ出すことが目的となる．

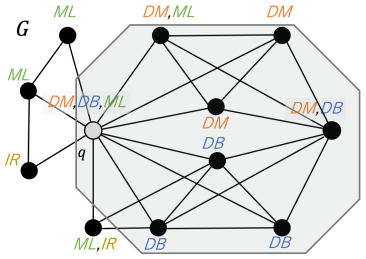


図 1　共著ネットワークにおける属性付きコミュニティ検索の例：ノードは著者，エッジは共著関係を示す．また各ノードのラベルは著者が専門とする分野 (DM：データマイニング，DB：データベース，ML：機械学習，IR:情報検索) を表す．

例えば，図 1 に示した論文の共著ネットワークを表した属性付きグラフに対して，クエリ $Q = \{q, \{DB, DM\}\}$ を与えた場合を考える．ここで q はクエリノード，$\{DB, DM\}$ はクエリ属性を示しており，クエリ Q は著者 q 近傍のデータベース (DB)，データマイニング (DM) 分野に関連のあるコミュニティを問い合わせる．図 1 において灰色の領域で示した部分グラフがクエリ Q に対する適切なコミュニティである．コミュニティ内部でノードが密に接続しているだけでなく，属性ラベル $DB \cdot DM$ を持つノードが多く存在することがわかる．このようなコミュニティを見つけ出すことで，ある著者と関わりの深く，かつ特定のトピックについて詳しい著者の集合を得られる．

Huang らによる研究では，属性付きコミュニティ検索問題を解くためにヒューリスティックなアルゴリズム LocATC [8] を提案している．LocATC は Attribute Score と呼ばれる指標を用

いてコミュニティとクエリ属性の類似度を評価する．Attribute Score が大きく truss 構造が多く含まれるような場合，LocATC はクエリに対するコミュニティを返す．しかしながら，LocATC はコミュニティが必ず truss 構造を持つという強い構造的制約を持っていることから，実データに対するコミュニティ検索の精度が低いことが報告されている [9].

これに対し，Matsugu らは先行研究 [9] でより柔軟に多様なコミュニティを検索すべく F-ATC 問題を定義した．F-ATC 問題はクエリ属性を持つノードと多くの 3 部クリークに属すエッジをなるべく多く含むようなノード集合を見つける問題である．Matsugu らはビームサーチ [10] を用いた F-ATC に対するヒューリスティックなコミュニティ検索手法を提案し，LocATC よりも精度の良いコミュニティを検索できることを示した [9]．しかしながら，その時間計算量はノード数 $|V|$, エッジ数 $|E|$, ビーム幅 β，パラメータ d に対して $\mathcal{O}(2^{(|E|/|V|)^d)}N^2\beta)$ となる．これはノード数 10,000 程度の共著ネットワークにおいても 1 日以上の実行時間を要する計算コストであり，大規模グラフに対する適用は現実的ではない．

そこで本研究では属性付きコミュニティ検索問題である F-ATC 問題の高速な解法を提案する．先行研究 [9] では部分グラフの隣接ノード集合による冪集合を全て列挙した上で，Attribute Score の大きな上位の要素のみを検索結果候補として採用する．ところが，冪集合の列挙は部分グラフの大きさ n に対して $\mathcal{O}(2^n)$ の計算量を要するため，この処理はコミュニティ検索において大きなボトルネックとなる．そのため，提案手法では前述した冪集合の中から Attribute Score の大きな要素のみを高速に列挙する．具体的には，Attribute Score 関数の高くなるような近傍解のみを選択的に列挙し，解になり得ないような検索結果候補をコミュニティ検索の早い段階で枝刈りすることで高速化を図る．

本研究の貢献は以下の 3 つである．

- **高速性**：計算量を $\mathcal{O}(d\beta^2\eta\log\beta\log(\beta\eta))$(ただし，$\eta = \min(|V|,(|E|/|V|)^d)$) に抑えた高速なコミュニティ検索アルゴリズムを提案し (定理 1)，実験により先行研究に比べて最大 500 倍の高速化を実現した (4.2.1，4.2.3 節).
- **正確性**：現実的な実行時間でより高精度なコミュニティ検索，および，ノード数百万程度の大規模なグラフに対してコミュニティ検索が可能であることを実証した (4.2.2 節，4.2.4 節).
- **省メモリ**：各ノードを探索および保持する回数を減らし，同じパラメータにおいて同程度の精度を保ったままメモリ使用量を削減した (定理 2).

本研究の提案手法を用いることで，ノード数百万程度の規模のグラフにおいて属性を指定して現実的な時間でコミュニティを検索することが可能となった．これには，DBLP のような共著ネットワークや，Wikipedia のリンク構造などがこの規模に該当し，より多くの場面において高精度なコミュニティ検索を実現できることを示唆している．

本稿の構成は以下の通りである．2 節で前提となる知識および先行研究について説明し，3 節で提案手法について述べる．4 節では評価実験の結果を示し，5 節では関連研究を紹介する．最後に，6 節で本稿のまとめを述べる．

表 1　本稿で用いる記号とその定義

記号	定義
G	計算対象のグラフ
V	G に含まれるノードの集合
E	G に含まれるエッジの集合
H	G の部分グラフ
$V(H)$	H に含まれるノードの集合
$E(H)$	H に含まれるエッジの集合
$\mathcal{A}$	全属性集合
Q	与えられるクエリ
v_q	クエリノード
W_q	クエリ属性の集合
$V_w(w)$	属性 w を含むノードの集合
β	ビームサーチにおけるビーム幅
Δ_k	k-truss の集合
$\Delta_{*,d}$	$(*,d)$-truss の集合

2 前提知識

本研究に関する基本事項について説明する．本研究で対象とするグラフは連結な属性付きグラフ $G(V,E,\mathcal{A})$ である．ここで，V はノード集合，E はエッジ集合である．また属性集合を $\mathcal{A}$ とし，各ノード $v \in V$ は属性集合 $attr(v) \subseteq \mathcal{A}$ を持つ．また，表 2 に本稿で用いる主な記号とその定義について示す．

2.1 問題設定

本節では本研究で扱う問題である F-ATC 問題 [9] を説明する．属性付きコミュニティ検索では，クエリ $Q = (v_q, W_q)$ が与えられたとき，クエリ点 v_q を含み，クエリ属性集合 W_q に対して最も適切な部分グラフ H を検索することを考える．F-ATC 問題ではコミュニティ検索によって獲得する部分グラフ H について，次の 4 つの性質を満たすことに主眼をおく．

(1) 部分グラフ H は，全てのクエリノードを含む．
(2) 部分グラフ H 内のノードは互いに密に接続する．
(3) 部分グラフ H はクエリ属性を持つノードを多く含む．
(4) 部分グラフ H はクエリノードの近傍から構成する．

事前準備として前節で述べた 4 つの性質について定量的な議論をするため，次の定義を導入する．

定義 1 (k-truss)
$sup_G(e)$ をエッジ $e(u,v)$ を辺に持つ三角形 (3 部クリーク) の数とする．k-truss 集合 [6]Δ_k は次の部分グラフ集合である．

$$\Delta_k = \{G' \subseteq G | \forall e \in E(G'), sup_G(e) > (k-2)\}.$$

また $H \in \Delta_k$ である部分グラフ H は k-truss であるという．

すなわち，*k-truss* は部分グラフ H の凝集性を内部の辺が構成する三角形の数によって評価する．

定義 2 (query distance)
ノード u, v 間の最短路の長さを，$dist_G(u,v)$ で表す．未接続の場合は，$+\infty$ と定義する．このとき，部分グラフ $H \subseteq G$ において，ノード $v \in V$ から H の各頂点 $u \in H$ への query distance を $dist_H(H,v) = \max_{u \in H} dist_H(u,v)$ とする．

query distance はクエリノードに対して部分グラフ中の最も離れたノードとの距離を表す．

定義 3 ((*, d)-truss)
クエリノード v_q およびパラメータ d が与えられたとき，*(*, d)-truss* 集合 $\Delta_{*,d}$ は以下のように定義する．

$$\Delta_{*,d} = \{G' \subseteq G | G' \in \Delta_k, v_q \in V(G'), dist_G(G', v_q) \leqq d, k \geqq 2\}.$$

また定義 1 に示した *k-truss* と同様に，部分グラフ H が $H \in \Delta_{*,d}$ であるならば，H は *(*, d)-truss* であるという．

提案手法では定義 3 を用いて，性質 (1)，(2)，および (4) の指標とする．最後に性質 (3) を示す属性の評価指標を定義する．

定義 4 (Attribute Score 関数)
部分グラフ H とクエリ属性集合 W_q が与えられたとき，Attribute Score 関数 $f(H, W_q)$ を，以下に定義する．

$$f(H, W_q) = \sum_{w \in W_q} \frac{|V_w(H) \cap V(H)|^2}{|V(H)|}.$$

ただし，$V_w(H)$ は属性 w を持つノードの集合である．

Attribute Score 関数は部分グラフ H にクエリ属性を持つノードが多く，そうでないノードが少ないとき大きな値をとる．

定義 3 および定義 4 を用いて，F-ATC 問題を定義する．

問題定義 1 (F-ATC 問題)
Given: $G(V, E, \mathcal{A})$，$Q = (v_q, W_q)$，d
Find: $H \in \Delta_{*,d} \arg\max f(H, W_q)$

つまり F-ATC 問題は，与えられたクエリに対して Attribute Score 関数が最も大きな *(*, d)-truss* を見つける問題である．

2.2 F-ATC 問題の解法

先行研究 [9] における F-ATC 問題の解法を概説する．

Matsugu らは F-ATC 問題の解法として，ビームサーチ [10] と呼ばれる探索木において解の多様性を保持しながら探索する手法を応用した．ビームサーチでは幅優先探索のように，ある状態から遷移先の状態をキューにプッシュすることで逐次的に探索木上を走査する．探索木にビーム幅 β というパラメータを加え，Attribute Score 関数の暫定評価値が最も良い β 個のみを探索し，残りの状態は枝刈りする．これにより，探索木の各レベルにおいて多様な遷移の可能性を保持しつつ，高速な探索を実現する．

Algorithm1 に先行研究で提案されたアルゴリズムの概要を示す．先行研究 [9] では，ビームサーチの初期状態はクエリノードのみを要素とした部分グラフとする．次に優先度付きキューからビーム幅 β に応じて最良の β 個だけ部分グラフをポップする．それぞれの部分グラフついてその部分グラフとその隣接ノードからなるあらゆる *(*, d)-truss* を計算し，個別に優先度付きキューにプッシュする．これを繰り返し，その過程で最も Attribute Score 関数が高かった部分グラフを解として出力する．

この探索手法はノードの追加を行うたびに部分グラフの Attribute Score 関数を計算し直す必要がある．しかし，この計算は効率的ではない．そこで，部分グラフにノードを追加する際の元のグラフとの Attribute Score 関数の差分を定義 5 に示す Attribute Expansion Gain を用いて計算する．

Algorithm 1 従来手法

Require: $G = (V, E), Q = (v_q, attr(v_q)), d, \beta$
Ensure: H
1: 優先度付きキュー P, P^{prime} を $P \leftarrow \emptyset, P' \leftarrow \emptyset$ として初期化;
2: $G_0 \leftarrow (\{v_q\}, \emptyset)$, P.enqueue(G_0), *count* $\leftarrow 0$, and *depth* $\leftarrow 0$;
3: **while** $P \neq \emptyset$ **and** *depth* $< d$ **do**
4: $G' \leftarrow P$.dequeue(), *count* $\leftarrow$ *count* $+ 1$;
5: $N_{G'} \leftarrow \bigcup_{u \in V(G')} \{v \in V | (u, v) \in E\} \backslash V(G')$;
6: **for each** $v \in N_{G'}$ **do**
7: **if** $\triangle f(G', v, attr(v_q)) > 0$ **then**
8: Construct G'' from G' and $\{v\}$;
9: **for each** $(*, d)$-truss H' in G'' **do**
10: P'.enqueue(H');
11: **end for**
12: **end if**
13: **end for**
14: **if** $P = \emptyset$ **or** *count* $\geqq \beta$ **then**
15: $P \leftarrow P', P' \leftarrow \emptyset$, *count* $\leftarrow 0$;
16: *depth* $\leftarrow$ *depth* $+ 1$;
17: **end if**
18: **end while**
19: $H \leftarrow P$.dequeue();

定義 5 (Attribute Expansion Gain)
部分グラフ H，ノード $v \notin V(H)$，及びクエリ $Q = (v_q, attr(v_q))$ が与えられたとき，Attribute Expansion Gain $\triangle f(H, v, attr(v_q))$ は次のように定義される．

$$\triangle f(H, v, attr(v_q)) =$$
$$\sum_{w \in \mathcal{A}_H \cap attr(v)} \frac{2|V_w \cap V(H)| + 1}{|V(H) + 1|} - \sum_{w \in \mathcal{A}_H} \frac{|V_w \cap V(H)|^2}{|V(H)|^2 + |V(H)|},$$

ここで，$\mathcal{A}_H = \bigcup_{u \in V(H)} attr(u)$, 及び $V_w = \{u \in V | w \in attr(u)\}$ である．

先行研究で提案されたアルゴリズムは，ビームサーチを導入したことによりコミュニティ候補の多様性を保持することで，既存の手法よりも高い精度でのコミュニティ検索を可能としている．しかしながら，より精度の良いコミュニティを検索するためには，コミュニティ候補の網羅的な列挙が必要となり，多くの実行時間を要する．具体的には，ある部分グラフの隣接ノード集合の大きさを n とすると，それから派生するコミュニティ候補の数は 2^n 個存在し，先行研究で提案されたアルゴリズムは，2^n 個のパターン全てについて定義 5 で与えた Attribute Expansion Gain の計算を行い，最良の β 個の候補を保存する必要がある．これは大規模なグラフにおけるコミュニティの検索効率を著しく低下させるボトルネックである．

3 提 案 手 法

本節では提案手法について説明する．提案手法は F-ATC 問題に対して先行研究 [9] と同等のコミュニティ検索結果をより高速に検索することを目指す．最初に提案手法を構成する基本的なアイデアについて述べ，その詳細を 3.2 節以降で説明する．

3.1 基本アイデア

2.2 節で示した通り，先行研究 [9] で提案されたビームサーチは，Attribute Score の大きな上位 β 個の部分グラフを計算するために，近傍の部分グラフとして存在し得る全ての部分グラフを計算する必要がある．これには隣接ノード集合の大きさ n に対し，$\mathcal{O}(2^n)$ の時間計算量を要するため，計算コストが大きい．

本稿では，先行研究のボトルネックである冪集合の列挙を行わずに上位 β 個の部分グラフのみを列挙する手法について提案する．まず Attribute Score 関数の極値を求めることで，最も Attribute Score の大きな部分グラフを導出する．次に，2 番目に Attribute Score の大きな部分グラフを見つけるために，得られた部分グラフから Attribute Score を僅かに低下させるような操作を行う．具体的には，部分グラフにクエリ属性を持たないノードを一つ追加する，または部分グラフからクエリ属性を持つノードを一つ削除するという操作を行う．この処理を β 回繰り返すことで，高速に上位 β 個の部分グラフを計算する．

この操作は優先度付きキューを用いて実装することができる．Attribute Score を僅かに低下させる操作は $\Omega(n)$ 通り存在するため，提案手法のビームサーチにおける近傍状態の列挙は $\mathcal{O}(\beta n \log n)$ に抑えられる．

3.2 候補コミュニティ集合の高速計算

本節では，ビームサーチにおいて隣接ノード集合からなる上位 β 個の部分グラフを高速に計算する方法について述べる．まず，部分グラフの隣接ノード集合を次のように定義する．

定義 6 (隣接ノード集合)
グラフ G，および，部分グラフ $H \subset G$ が与えられたとき，H の隣接ノード集合 $N(H)$ を以下に定義する．

$$N(H) = \{v \in V(G \setminus H) | \exists u \in V(H), (u, v) \in E(G)\}.$$

また，ビームサーチにおける次の探索の候補となる β 個の部分グラフの集合を，候補コミュニティ集合として次に定義する．

定義 7 (候補コミュニティ集合)
グラフ G の部分グラフ $H \subseteq G$，および，ビーム幅 β に対して，$C_i(H) \subseteq N(H)(i = 1, 2, ..., \beta)$ を候補コミュニティとする．このとき，H に対する候補コミュニティ集合 $C(H)$ とは，以下を満たす大きさ β の部分グラフの集合である．

$$C(H) = \{C_1(H), C_2(H), ..., C_\beta(H)\} \subseteq 2^{N(H)}.$$

ただし，$C_i(H) \subseteq N(H)$，$i \neq j$ のとき $C_i(H) \neq C_j(H)$．
候補コミュニティ集合は，H の隣接ノード集合 $N(H)$ からいくつかのノードを H に追加する $2^{|N(H)|}$ 通りの候補コミュニティを，β 個選んでできる集合である．

提案手法では最も Attribute Score の大きい上位 β 個の部分グラフからなる候補コミュニティ集合を求める必要がある．そのため，部分グラフ H に対する最良の候補コミュニティ集合 $C^*(H)$ を求める問題を以下に定義する．

問題定義 2 (Attribute Score 関数の総和の最大化問題)
Given:グラフ H，クエリ属性集合 W_q，ビーム幅 β
Find:以下の値が最大となるような候補コミュニティ集合 $C^*(H)$

$$\sum_{i=1}^{\beta} f(H \cup C_i^*(H), W_q).$$

問題定義 2 では，β 件による Attirbute Score 関数の総和の最大化を問題としている．この解となる候補コミュニティ集合の β 個の要素となる部分グラフは明らかに Attribute Score 関数の大きな上位 β 個であるため，問題定義 2 を解くことで，H に対する最良の候補コミュニティ集合が求まる．

問題定義 2 を解くため，Attribute Score 関数の凸性に着目しながら，以下の二つの処理を用いて逐次 $C_i^*(H)$ を決定する．ただし，$C_i^*(H)$ は $f(C_1^*(H), W_q) \geq f(C_2^*(H), W_q) \geq ... \geq f(C_\beta^*(H), W_q)$ を満たすものとする．

- 最も Attribute Score 関数の大きな部分グラフ $C_1^*(H)$ の候補を列挙し，$C_1^*(H)$ を決定する．
- 残った $C_i^*(H)$ の候補，および，$C_i^*(H)$ にノードを一つ追加あるいは削除した部分グラフを，$C_{i+1}^*(H)$ の候補とし，最も Attribute Score の大きな部分グラフを $C_{i+1}^*(H)$ に決定する．

これらのアイデアを実現するために，部分グラフ H にノードを追加するときの Attribute Score の振る舞いを分析する．

はじめに，$N(H)$ 内のノードは，次の 3 つに分類される．

（1）クエリ属性集合 W_q のいずれにも属さないノード
（2）クエリ属性集合 W_q の全てに属すノード
（3）クエリ属性集合 W_q の一部にのみ属すノード

ここで，(3) について記号を定義する．

定義 8 (一部のクエリ属性のみを持つ隣接ノード集合)
クエリ属性集合 W_q の部分集合 $W_q' \subseteq W_q$ が与えられたとき，W_q' のみに属す，すなわち W_q' の全てに属し，$W_q \setminus W_q'$ の全てに属さない隣接ノード集合を $N_{W_q'}(H)$ と表す．

次に，(1),(2),(3) について順に振る舞いを確認する．

補題 1 (全てのクエリ属性を持つノード)
全てのクエリ属性を持つ隣接ノード $v \in N_{W_q}(H)$ について，$f(H) < f(H \cup \{v\})$ が成り立つ．

証明
$f(H \cup \{v\}) - f(H) > 0$ を示す．

$$\begin{aligned}
&f(H \cup \{v\}) - f(H) \\
&= \frac{(|V_w \cap V(H)| + 1)^2}{|V(H)| + 1} - \frac{|V_w \cap V(H)|^2}{|V(H)|} \\
&= \frac{|V(H)| \sum_{w \in W_q} (|V_w \cap V(H)|^2 + 2|V_w \cap V(H)| + 1)}{|V(H)|(|V(H)| + 1)} \\
&\quad - \frac{(|V(H)| + 1)|V_w \cap V(H)|^2}{|V(H)|(|V(H)| + 1)} \\
&= \frac{|V(H)| \sum_{w \in W_q} (2|V_w \cap V(H)| + 1)}{|V(H)|(|V(H)| + 1)} \\
&\quad - \frac{\sum_{w \in W_q} |V_w \cap V(H)|^2}{|V(H)|(|V(H)| + 1)}
\end{aligned}$$

ここで，$V(H) > |V_w \cap V(H)|$ であるため，

$$2|V(H)||V_w \cap V(H)| - |V_w \cap V(H)|^2 > 0.$$

よって，

$$\begin{aligned}
&f(H \cup \{v\}) - f(H) \\
&= \frac{|V(H)| \sum_{w \in W_q} (2|V_w \cap V(H)| + 1)}{|V(H)|(|V(H)| + 1)} \\
&\quad - \frac{\sum_{w \in W_q} |V_w \cap V(H)|^2}{|V(H)|(|V(H)| + 1)} > 0.
\end{aligned}$$

であり，補題は成り立つ． □

補題 2 (クエリ属性を持たないノード)
クエリ属性を持たない隣接ノード $v \in N_{\emptyset}(H)$ について，$f(H) > f(H \cup \{v\})$ が成り立つ．

証明
$f(H \cup \{v\}) - f(H) < 0$ を示す．

$$\begin{aligned} & f(H \cup \{v\}) - f(H) \\ & = \frac{|V_w \cap V(H)|^2}{|V(H)|+1} - \frac{|V_w \cap V(H)|^2}{|V(H)|} < 0. \end{aligned}$$

よって，補題は成り立つ． □

補題 1，補題 2 より，クエリ属性を全て持つ隣接ノードは部分グラフ H に追加することで Attribute Score を必ず増大させ，クエリ属性を全く持たない隣接ノードは必ず減少させることがわかる．これは，最良の候補コミュニティ $C_1^*(H)$ を探索する上で，これらのノードを追加する是非が自明であることを示す．

補題 3 (クエリ属性を一部のみ持つノード)
クエリ属性を一部のみ持つノード $v \in N_{W_q'}(H)$ を追加した部分グラフ $H \cup \{v\}$ の Attribute Score 関数は $|V_w \cap V(H)|$，すなわち H 内の属性 w を持つノードの数に対して下に凸である．

証明
属性 $w' \in W_q'$ に対して，$|V_w' \cap V(H)|$ を W とおく．このとき，$|V(H)| = W + |V(H) \setminus V_w'|$ であり，$f(H)$ は，

$$\begin{aligned} f(H) &= \sum_{w \in W_q} \frac{|V_w \cap V(H)|^2}{|V(H)|} \\ &= \frac{W^2}{W + |V(H) \setminus V_w'|} + \sum_{w \in W_q \setminus w^*} \frac{|V_w \cap V(H)|^2}{W + |V(H) \setminus V_{w'}|}. \end{aligned}$$

また，$f(H)$ の W についての 2 次偏導関数は，

$$\begin{aligned} \frac{\partial^2 f(H)}{\partial W^2} &= \frac{2(W + |V(H) \setminus V_w'|)(W + |V(H) \setminus V_{w'}|)^2}{(W + |V(H) \setminus V_{w'}|)^4} \\ &- \frac{2(W + |V(H) \setminus V_w'|)(W^2 + 2W|V(H) \setminus V_{w'}|)}{(W + |V(H) \setminus V_{w'}|)^4} \\ &+ \frac{\sum_{w \in W_q \setminus w'} |V_w \cap N(H)|^2}{(W + |V(H) \setminus V_{w'}|)^4}. \end{aligned}$$

これが常に正であることを背理法によって示す．$\frac{\partial^2 f(H)}{\partial W^2} < 0$ であると仮定すると，分母は明らかに正である．よって分子に着目すれば良い．

$$\begin{aligned} & 2(W + |V(H) \setminus V_w'|)(W + |V(H) \setminus V_{w'}|)^2 \\ & - 2(W + |V(H) \setminus V_w'|)(W^2 + 2W|V(H) \setminus V_{w'}|) \\ & + \sum_{w \in W_q \setminus w'} |V_w \cap N(H)|^2 < 0. \end{aligned}$$

となるため，第 3 項は明らかに正であることから，

$$\begin{aligned} & (W + |V(H) \setminus V_{w'}|)^2 - (W^2 + 2W|V(H) \setminus V_{w'}|) \\ & = |V(H) \setminus V_{w'}|^2 < 0. \end{aligned}$$

となり矛盾する．よって補題は成立する． □

Algorithm 2 ASF-総和の最大化問題の解法

Require: グラフ H，H の隣接ノード集合 $V_{AB}, V_A, V_B, V_{\emptyset}$，パラメータ β
Ensure: $\mathbf{argmax}_{S_h \subset S} f(H \cup S_h, W_q)$
1: Let 優先度付きキュー $P \leftarrow \{V_{AB} \cup V_A, V_{AB} \cup V_B, V_{AB} \cup V_A \cup V_B\}$
2: Let 出力集合 $Out \leftarrow \emptyset$
3: Let $cnt \leftarrow 0$
4: **while** $P \neq \emptyset or cnt < \beta$ **do**
5: Let $h \leftarrow PopP()$
6: AddOut(h)
7: PushP(GetNeighborState(h))
8: $cnt \leftarrow cnt + 1$
9: **end while**

Algorithm 3 GetNeighborState(h)

Require: グラフ h，H の隣接ノード集合 $V_{AB}, V_A, V_B, V_{\emptyset}$
Ensure: グラフ集合 GNS
1: Let $GNS \leftarrow \emptyset$
2: **for all** $v \in h$ **do**
3: AddGNS($h \setminus v$)
4: **end for**
5: **for all** $v \in (V_A \cup V_B \cup V_{\emptyset}) \setminus h$ **do**
6: AddGNS($h \cup v$)
7: **end for**

補題 3 に示した Attribute Score 関数は，$|V_w \cap V(H)| = 0$，または $|N_{W_q'}(H)|$ のいずれかのとき最大値をとる．

補題 1，2，3 により，$C_1^*(H)$ はいずれかのクエリ属性集合の部分集合 W_q' における，$N_{W_q'}(H)$ であることがわかる．また，$C_2^*(H)$ は $C_1^*(H)$ の残った候補に加え，「$C_1^*(H)$ にないノードを一つ追加した部分グラフ」，「$C_1^*(H)$ から一つのノードを削除した部分グラフ」のいずれかである．提案手法は同様の操作を高々 β 回繰り返すことで，$C^*(H)$ を高速に求める．

詳細なアルゴリズムを Algorithm 2 に示す．Algorithm 2 では，まず優先度付きキュー P に $C_1^*(H)$ の候補を追加する．次に，P 内にある $C_1^*(H)$ を取り出し，その近傍状態，つまり $C_2^*(H)$ の候補を Algorithm 3 により計算し，P に追加する．これを β 回または P が空になるまで繰り返し，$C^*(H)$ を出力する．

この処理の時間計算量を補題 4 にて示す．

補題 4 (Algorithm 2,3 の時間計算量)
Algorithm2,3 は $\mathcal{O}(\beta|N(H)|\log(\beta|N(H)|))$ 時間で計算できる．

証明
提案手法が $C^*(H)$ を求めるアルゴリズムは，優先度付きキューを用いて実装できる．まず，$C_i^*(H)$ が決定したとき，$C_i^*(H)$ を僅かに更新して $C_{i+1}^*(H)$ の候補を優先度付きキューに追加するコストは $\mathcal{O}(|N(H)|)$ である．これはノードを一つ追加または削除する処理は高々 $|N(H)|$ 回しか行われないためである．また優先度付きキューにおいて，$C_i^*(H)$ をその候補集合から取り出すコストは優先度付きキューのサイズの対数に比例する．優先度付きキューのサイズは高々 $\beta|N(H)|$ 個であるため，このコストは $\mathcal{O}(\log(\beta|N(H)|))$ である．この処理を β 回繰り返すため，全体のコストは $\mathcal{O}(\beta|N(H)|\log(\beta|N(H)|))$ である． □

3.3 アルゴリズム

提案手法のアルゴリズムについて説明する．提案手法は，まず優先度付きキューにクエリノードのみからなる部分グラフを追加し，d 回の反復によるビームサーチを行う．i 回目の反復

において，$i-1$ 回目の反復より得られた β 個の候補コミュニティのそれぞれについて，Algorithm 2,3 を適用する．この過程で得られた Attribute Score が最大のコミュニティを出力する．

最後に，提案手法の時間・空間計算量について解析する．

定理 1 (提案手法の時間計算量)
提案手法の時間計算量は $\mathcal{O}(d\beta^2\eta\log\beta\log(\beta\eta))$ である．(ただし，$\eta=\min(|V|,(|E|/|V|)^d)$．)

証明
提案手法のビームサーチは，優先度付きキューによって実装される．ビームサーチは d 回反復され ($\mathcal{O}(d)$)，それぞれの反復で β 回 Algorithm 2,3 を適用する．補題 4 より，Algorithm 2,3 の時間計算量は $\mathcal{O}(\beta|N(H)|\log(\beta|N(H)|))$ である．ここで，隣接ノード集合の大きさ $|N(H)|$ は，グラフ全体の平均次数を $|E|/|V|$ とすると，$\mathcal{O}((|E|/|V|)^d)$ と表せる．また $|N(H)|$ は $|V|$ を超えないため，より厳密には $\mathcal{O}(\min(|V|,(|E|/|V|)^d))$ であり，これを η とする．すると Algorithm 2,3 の時間計算量は η を用いて $\mathcal{O}(\beta\eta\log(\beta\eta))$ と言い換えられる．優先度付きキューのサイズは最大 β であるため，提案手法の時間計算量は $\mathcal{O}(d\beta^2\eta\log\beta\log(\beta\eta))$ である． □

定理 2 (提案手法の空間計算量)
提案手法の空間計算量は $\mathcal{O}((|E|/|V|))^{d+1})$ である．

証明
提案手法で用いる優先度付きキューには，最大で β 個の部分グラフが格納される．各部分グラフのノード数は平均次数を $|E|/|V|$ として $(|E|/|V|))^d$ 程度と表せる．また平均エッジ数はノード数と平均次数の積であるため，$(|E|/|V|))^{d+1}$ となる．隣接リストによるグラフの保持にかかるコストはノード数とエッジ数の和で表されるため，ノードの保持に掛かる空間計算量はエッジに比べて非常に軽微であり，無視できる．よって提案手法の空間計算量は $\mathcal{O}((|E|/|V|))^{d+1})$ である． □

従来手法の時間計算量は $\mathcal{O}(2^{(|E|/|V|)^d)}N^2\beta)$，および，空間計算量は $\mathcal{O}(2^{(|E|/|V|)^d+1)}\beta)$ を要する．それゆえ，提案手法は従来手法と比較して非常に高速かつ省メモリであるといえる．

4 評価実験

提案手法の有効性について実データを用いて評価を行う．まず 4.1 節では実験方法や実験に使用したデータセットについて述べ，4.2 節にて提案手法の実行時間について評価する．最後に，4.3 節にて提案手法のコミュニティ検索精度を検証する．

4.1 実験設定

表 2 にデータセットの統計情報を示す．表中の d_{max} はグラフの最大次数を表す．本稿では 2.2 節で述べた先行研究 [9](以後．Baseline と呼ぶ) と 3 節で述べた提案手法について実行速度とコミュニティ検索精度を比較する．実装には C++言語 (gcc 8.2.0) を用い，コンパイルオプションは g++ -g -std=c++11 -O3 とした．また，すべての実験は Intel(R) Xeon(R) E5-1620 v4(3.50GHz)，および，16GB のメインメモリを用いて行った．本実験では Stanford Network Analysis Project [11] から取得した 2 つの実データを用いて評価を行う．評価に用いたデータセットの詳細を以下に述べる．

表 2　データセットの詳細

データセット	$\|V\|$	$\|E\|$	d_{max}	$\|\mathcal{A}\|$
Facebook	347	5,038	154	224
DBLP	317,080	1,049,866	343	1,585

- **Facebook**：ソーシャルネットワークにおけるユーザ間の友人関係を表したデータセットである．ノードはユーザを表し，エッジはユーザ間の友人関係を示す．またユーザの属性として職業，居住地など 224 属性を用い，実験にはノード ID 0 のユーザ近傍のエゴネットワークを用いた．
- **DBLP**：　論文の共著関係を表したデータである．ノードは著者を表し，エッジは共著関係を示す．このデータセットには次のようにして人工的に各ノードに属性を割り振る．まず全属性数 $|\mathcal{A}|$ を，$|\mathcal{A}|=0.005|V|$ とする．これは，多くの実世界のグラフ [11] に見られるおおよその値に基づくものである．そして，各コミュニティはランダムに 3 つの属性を，コミュニティ内の 80%のノードに存在するようにそれぞれ割り当てる．

入力するクエリは従来手法 [9] の実験に基づき，ランダムなクエリ $Q=\{v_q,W_q\}$ を 100 回与える．このとき，クエリ属性集合はクエリノードが含まれるコミュニティに最も頻出する属性値と，その他のコミュニティに最も現れない属性値の 2 つを与える．本実験では 100 回与えたクエリの平均値を用いて提案手法を従来手法と比較し，有効性を議論する．

4.2 実行時間の比較

提案手法の実行時間について評価を行う．この実験では二つのデータセットに対して，d は共通して 1 から 5，β は Facebook データセットにおいては 1 から 50，DBLP データセットにおいては 1 から 500 まで変化させたときの実行時間を比較する．

図 2，3 に各データセットにおける実行時間の比較を示す．図 2 より我々の提案手法は，先行研究と比較して 40 から 500 倍程度高速であり，あらゆるパラメータにおいて実行速度の面で優れていることがわかる．一方で，図 3 に示した DBLP に対する結果では，先行研究はあらゆるパラメータにおいて，提案手法は $d=5$ かつ $\beta\leqq 100$ のときにメモリが不足し停止した，若しくは応答が 1 時間を超えたため，ここでは載せていない．提案手法は β に対して緩やかに低速になっていることがわかる．これは，定理 1 で示した提案手法の時間計算量において，β の変化に比べてノード数やエッジ数等の影響がより支配的であるためであると考えられる．これらの実験により我々の提案手法は，先行研究では現実的な時間で実行できなかった大規模なグラフにおいてもコミュニティを算出することができるといえる．

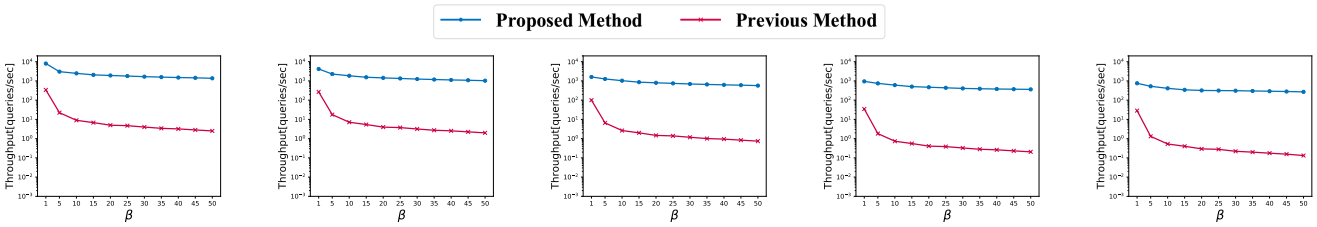
4.3 コミュニティ検索精度の比較

提案手法のコミュニティ検索精度について評価する．本実験ではコミュニティ精度の評価に F1-Score を用いる．

定義 9 (F1-Score)
各手法の解となるコミュニティを C, 真のコミュニティを $\hat{C}$ としたときの F1-Score を以下に定義する．

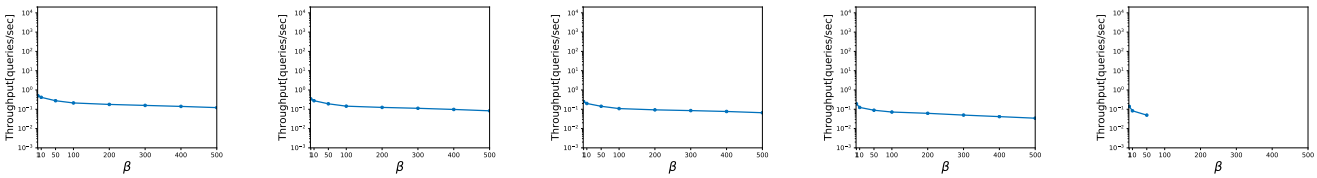
$$F1(C,\hat{C})=\frac{2\cdot prec(C,\hat{C})\cdot recall(C,\hat{C})}{prec(C,\hat{C})+recall(C,\hat{C})}$$

ただし，$prec(C,\hat{C})=\frac{|C\cap\hat{C}|}{|C|}, recall(C,\hat{C})=\frac{|C\cap\hat{C}|}{|\hat{C}|}$ とする．



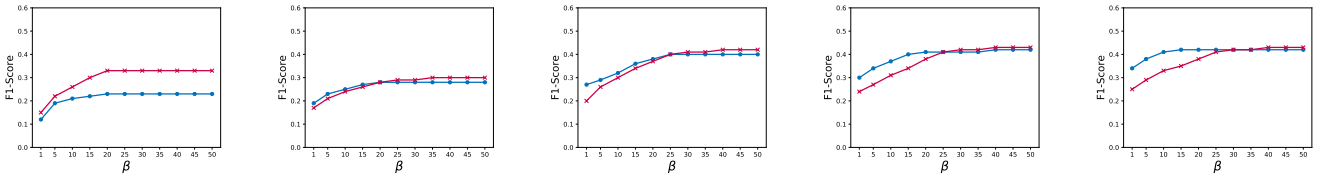
(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$ (e) $d = 5$

図 2　Facebook データセットにおける実行速度の比較



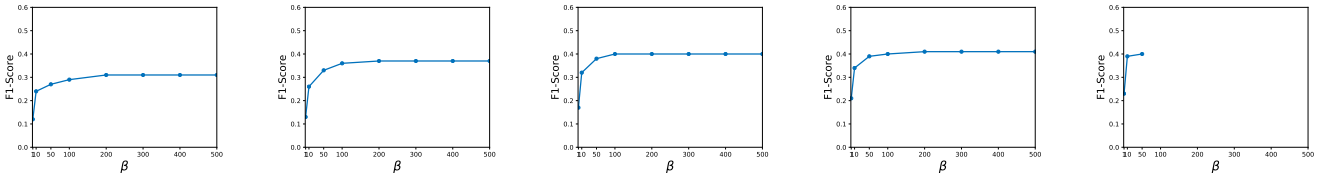
(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$ (e) $d = 5$

図 3　DBLP データセットにおける実行速度



(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$ (e) $d = 5$

図 4　Facebook データセットにおける F1-Score の比較



(a) $d = 1$ (b) $d = 2$ (c) $d = 3$ (d) $d = 4$ (e) $d = 5$

図 5　DBLP データセットにおける F1-Score

F1-Score は $[0, 1]$ の値を取る．F1-Score が高いほどその手法が見つけたコミュニティが真のコミュニティに近いことを示し，1 のとき両者は完全に一致する．

図 4，5 に，実験結果を示す．図 4 は，Facebook データセットに対する F1-Score の比較である．実験より提案手法 $d \geqq 2$ について，同程度の F1-Score を示していることがわかる．これは，補題 1，2，3 より導いた候補コミュニティ集合の探索アルゴリズムが，従来手法と同等のコミュニティを検索できていることを示唆している．一方で $d = 1$ の場合については，Baseline よりも提案手法の精度が低下している．これは提案手法は $d = 1$ においてはビームサーチによる反復を行わないのに対し，Baseline はビームサーチによる多様なコミュニティの検索を行えることが原因であると考えられる．しかしながら，$d = 1$ の実験結果は Baseline，提案手法共に $d \geqq 2$ の場合と比較して精度が低く有用ではないため，これは提案手法が先行研究に劣ることを示すものではない．

同様に図 5 では DBLP における F1-Score の値を示している．実行速度についての結果と同様に，先行研究と提案手法の一部の実験結果は測定できなかった，また，提案手法の精度は $\beta = 200$ を境にほぼ一定になっていることがわかる．これはビームサーチの各ステップにおいて，Attribute Score 関数の大きな上位 200 件程度の部分グラフを列挙することで，Attribute Score 関数値の最も大きな部分グラフに十分に近似しているためと考えられる．結論として，高精度にコミュニティを検索することを考えると，提案手法の精度面における性能は先行研究とほとんど差異が無いことを示す．

5　関 連 研 究

本稿では属性付きグラフに対する効率的なコミュニティ検索手法を提案した．コミュニティ検索はデータベースコミュニティを中心にこれまで幅広く研究されている．本稿ではその中でも (1) 構成なしグラフに対するコミュニティ検索，ならびに (2) 属性付きグラフに対するコミュニティ検索について述べる．

5.1 属性なしグラフに対するコミュニティ検索

属性値を考慮しないコミュニティ検索手法としてはグラフ構造を評価する尺度として *k-core* [5, 12] を用いる手法や，*k-truss* [6, 7] を用いる手法などがある．文献 [5] は問合せ型のクラスタリング問題としてコミュニティ検索問題を提案した文献である．またその解法として *k-core* という次数が大きいノードらによって構成される部分グラフ構造に着目し，コミュニティを貪欲的に探索するアルゴリズムを提案した．文献 [12] はクエリノードの近傍でのみ探索を行う局所探索型の解法を提案し，大規模なグラフに対して実験にて有効性を示した．また，文献 [6, 7] は *k-core* より頑健なグラフ構造を評価する指標としてエッジによる 3 部クリークに着目した *k-truss* を採用した．これはグラフ中の truss 構造を検索し，更にその中からコミュニティに不要なノードを計算し，逐次ノードを削除する手法を提案した．これらの研究はいずれもグラフ構造のみに注目する手法であり，頑健なコミュニティ構造を検索することに優れている．しかしながら，これらの研究が対象とするグラフにはノードに属性が付与されていない，あるいはノードの属性を無視して計算を行うため精度を欠くという問題がある．それに対して本研究ではノードに与えられた属性を活用して計算を行うため，より高精度にコミュニティを検索できる．

5.2 属性付きグラフに対するコミュニティ検索

従来のコミュニティ検索は属性付きグラフに対応した手法として LocATC [8] や先行研究 [9] がある．LocATC はコミュニティ内のクエリ属性の分布を評価する指標として Attribute Score 関数とそれを最大化する ATC 問題を提案し，クエリ属性を持つノードが多く含まれるよう，ボトムアップにコミュニティを構成する．その後，コミュニティ内の各ノードの貢献度を定義し，貢献度の小さいノードから順に削除を行い，その過程で最も評価の高かった解を出力する．これは従来の構造にのみ着目したコミュニティ検索と比較して高速かつ高精度ではあったが，解となるコミュニティへの構造的な制約が強いため．エッジの密な構造を持つコミュニティしか検索することができないという欠点を持つ．また先行研究 [9] は LocATC の提案した ATC 問題の構造的な制約を緩和する F-ATC 問題を提案し，より高い精度でコミュニティを得られる探索アルゴリズムを提案したが，大規模なグラフに適用するには長い計算時間を要するという欠点がある．本研究では，先行研究のボトルネックである解候補の列挙を早期の枝刈りにより省略することで，先行研究と同等の精度でより高速にコミュニティを検索できる．

6 結　　論

本稿では大規模な属性付きグラフに対する高速なコミュニティ検索手法を提案した．提案手法は，先行研究のビームサーチにおけるボトルネックであった隣接ノード集合の冪集合による解候補の列挙を行わず，Attribute Score 関数の高くなる解のみを選択的に列挙する．その結果として，先行研究 [9] において定義された F-ATC 問題に対して，提案手法は先行研究と同等の精度を保ちつつ，コミュニティを最大 500 倍高速に検索することが可能である (定理 1，および，4.2 節)．

しかしながら，提案手法はパラメータを大きく設定すると依然として多大な計算時間を要する場合がある．本稿で実験に用いたデータセットはノード数が 40 万程度であるが，実世界にはより巨大なグラフが存在する．このような大規模なグラフに対して高速に解を算出しようとした場合，パラメータを小さくせざるを得ないためコミュニティ検索の精度を欠いてしまうという問題がある．そのため，更なる計算量の改善を行い，より効率的な計算方法を提案することが今後の課題である．

謝　　辞

本研究の一部は JST ACT-I ならびに JSPS 科研費 JP18K18057 による支援を受けたものである．

文　　献

[1] Yang Zhou, Hong Cheng, and Jeffrey Xu Yu. Graph Clustering Based on Structural/Attribute Similarities. *Proceedings of the VLDB Endowment*, 2(1):718–729, August 2009.

[2] Hiroaki Shiokawa, Toshiyuki Amagasa, and Hiroyuki Kitagawa. Scaling Fine-grained Modularity Clustering for Massive Graphs. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI2019)*, 2019.

[3] Hiroaki Shiokawa, Tomokatsu Takahashi, and Hiroyuki Kitagawa. ScaleSCAN: Scalable Density-based Graph Clustering. In *Proceedings of the 29th International Conference on Database and Expert Systems Applications*, DEXA, pages 18–34, 2018.

[4] A. Louni and K. P. Subbalakshmi. Who spread that rumor: Finding the source of information in large online social networks with probabilistically varying internode relationship strengths. *IEEE Transactions on Computational Social Systems*, 5(2):335–343, June 2018.

[5] Mauro Sozio and Aristides Gionis. The Community-Search Problem and How to Plan a Successful Cocktail Party. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2010)*, pages 939–948, New York, NY, USA, 2010. ACM.

[6] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. Approximate Closest Community Search in Networks. *Proceedings of the VLDB Endowment*, 9(4):276–287, December 2015.

[7] Xin Huang, Hong Cheng, Lu Qin, Wentao Tian, and Jeffrey Xu Yu. Querying K-truss Community in Large and Dynamic Graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, SIGMOD 2014, pages 1311–1322, 2014.

[8] Xin Huang and Laks Lakshmanan. Attribute-Driven Community Search. *Proceedings of the VLDB Endowment*, 10(9):949–960, 2017.

[9] Shohei Matsugu, Hiroaki Shiokawa, and Hiroyuki Kitagawa. Flexible Community Search Algorithm on Attributed Graphs. In *Proceedings of the 21st International Conference on Information Integration and Web-based Applications and Services (IIWAS 2019)*, December 2019.

[10] D. Raj. Reddy. Speech understanding systems: A summary of results of the five-year research effort. department of computer science. Technical report, Carnegie-Mellon University, 1977.

[11] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford large network dataset collection. `http://snap.stanford.edu/data`, June 2014.

[12] Wanyun Cui, Yanghua Xiao, Haixun Wang, and Wei Wang. Local Search of Communities in Large Graphs. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data (SIGMOD 2014)*, pages 991–1002, New York, NY, USA, 2014. ACM.