属性付きグラフに対するビームサーチを用いたコミュニティ検索

真次　彰平† 　塩川　浩昭†† 　北川　博之††

† 筑波大学情報学群情報科学類　〒 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ソーシャルネットワーク等のグラフ構造から問合せに合うコミュニティを見つけ出すグラフ分析手法である．本研究ではビームサーチを用いた属性値付きグラフに対するコミュニティ検索手法について提案する．従来の手法では頑健なトラス構造を持つコミュニティしか解として見つけることができず，グラフ構造への制約が柔軟でない．そこで本稿では解に求める構造的制約条件を緩和した上で，ビームサーチを用いてより多様なコミュニティを探索し，性能の向上を図る．本稿では提案手法の概要と評価実験の結果について述べる．

キーワード　コミュニティ検索，グラフマイニング

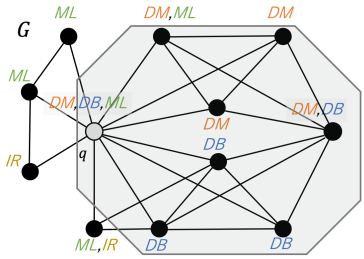

図 1　属性付きグラフに対するコミュニティ検索の例

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，それらのデータから有用な知識を抽出するデータ分析が注目されている．特に，各エンティティをノード，エンティティ間の関係をエッジとして表現するグラフを用いた様々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に対して適切なノード集合 (コミュニティ) を見つけ出す技術である．一般的にグラフはノードに属性値を持つ場合がある．具体的には，Facebook などの SNS ではユーザの出身地や，職業，興味のあるトピックなどの情報をノードに付随して持つことが考えられる．これらの属性値の付いたグラフを，属性付きグラフ [1] という．属性付きグラフに対するコミュニティ検索で与えるクエリはクエリノードとクエリ属性から構成される．クエリノードは自身が属するコミュニティを問い合わせる単一のデータエンティティを表し，クエリ属性はコミュニティ内の多くのノードが保持するべき属性を指定する．属性付きグラフに対するコミュニティ検索はこのクエリに対してクエリノード近傍のクエリ属性に関係のあるコミュニティを見つけ出すことが目的となる．

属性付きグラフの例として図 1 に共著ネットワークを示す．図 1 のグラフに対して，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 において灰色で示した領域がクエリ Q に適合するコミュニティに該当し，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．これにより，著者間の関係の概略を把握することが可能となる．

グラフに対するコミュニティ検索を実現する先行研究は，グラフの構造的な特徴に基づく手法 [2–4] が主流であった．しかしながら，これらの手法はグラフの属性を考慮しておらず，図 1 に示したような属性をもったグラフ構造に対するコミュニティ検索を実現することができなかった．これに対して，Huang らはグラフの構造とノードの属性の両方の観点からコミュニティ検索手法として LocATC [5] を提案し，属性付きグラフに対するコミュニティ検索を実現した．LocATC ではクエリに対する適切なコミュニティを定義するため，Attribute Truss Community (ATC) 問題を定式化している．LocATC はこの ATC 問題を最適化するコミュニティを貪欲法により探索する手法であり，いくつかのベンチマークデータセットにおいて高い精度でコミュニティを検索できることが報告されている．

しかしながら，ATC 問題は解となるコミュニティに頑健な truss 構造 [6] が含まれることを条件とする．truss 構造とはグラフを構成するエッジによって作られる三角形の集まりによる構造である．ゆえに，ATC 問題ではコミュニティは必ず truss 構造を持つ必要があるため，LocATC によって取得されるコミュニティは強い構造的な制約を課せられている．その結果として LocATC によって取得されるコミュニティは柔軟性を欠くという性質を持つ．しかしながら，実世界に存在するコミュニティは必ずしも密な truss 構造によって構成される訳ではない．ゆえに，先行研究である LocATC は，実データにおいて十分な精度を得られないという問題がある．

本研究では属性付きグラフから高精度なコミュニティを検索

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
$\mathcal{A}$	全属性集合
Q	与えられるクエリ
v_q	クエリに含まれるノード
W_q	クエリに含まれる属性の集合
H	G の部分グラフ
B	ビームサーチにおけるビーム幅

する手法を提案する．本研究ではまず，ATC 問題の構造的制約条件を緩和することで，柔軟なコミュニティ構造を取得可能な新しい問題である緩和 ATC 問題を定義する．また，この緩和 ATC 問題に対する解法として新たなコミュニティ検索手法を提案する．提案手法では多様な構造を持つ実グラフに対応するため，多様なコミュニティの状態を保持した探索を実現可能なビームサーチ [7] を採用する．提案手法は緩和 ATC 問題をビームサーチを用いて解くことにより，実グラフに truss 構造が多く含まれない場合においても，従来手法 LocATC と比較して高精度にコミュニティを検索することができる．また，提案手法で採用したビームサーチのビーム幅を大きくした場合には，より大きな実行時間を掛けて更に高い精度のコミュニティを求めることが可能である．本稿では提案手法の有効性について実データを用いた実験により検証を行う．

本研究の貢献は 2 つある．本研究は (1) 従来手法より柔軟なコミュニティを検出可能とする緩和 ATC 問題を定義した．また (2) 実データに対して，提案手法は先行研究 LocATC より 18% から 60%程度高い精度でコミュニティを検索することができる．

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 LocATC 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．

2　前 提 知 識

本章では，本研究に関する基本事項について説明する．はじめに 2.1 節にて属性付きグラフについて延べ，2.2 節にて先行研究である LocATC について説明する．最後に，2.3 節にて LocATC の問題点について述べる．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　属性付きグラフ

本節では属性付きグラフについて定義する．

定義 2.1 (属性付きグラフ)　属性付きグラフを連結なグラフ $G(V, E)$ として与える．ここで，V はノード集合，E はエッジ集合である．また，全ての属性の集合を $\mathcal{A}$ とし，各ノード $v \in V$ の持つ属性値の集合を $attr(v) \subseteq \mathcal{A}$ で表す．

図 1 の共著ネットワークではノードは著者を，エッジは著者同士の共著関係を表す．ノードは各著者が取り扱うトピックを属性として持つ．例えば，著者 q はその近傍の 8 人の著者と共著の経験があり，*DB, DM, ML* の属性値を持つ．各ノードの持つ属性値の数は不定であり，共著関係にある著者が必ずしも同じ属性値を持つとは限らない．

2.2　先行研究：LocATC

本節では本研究の前提となる属性付きコミュニティ検索手法 LocATC について説明する．Huang らはまずクエリに対する適切なコミュニティを求める問題について，Attribute Truss Community (ATC) 問題として定式化を行い，その解法として LocATC を含む 3 つの手法を提案している．本節では，2.2.1 節にて ATC 問題の概要を述べ，2.2.2 節にて Huang らが提案した 3 つの手法を説明する．

2.2.1　ATC 問 題

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．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) 出力部分グラフ H は，全てのクエリ点を含む．

(2) 部分グラフ H は凝集性が高い，つまり H 内のノード間に含まれるエッジが多い．

(3) 部分グラフ H にはクエリ属性を持つノードを多く含む．

(4) 部分グラフ H はクエリ点から近い点から構成する．

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 2.2 (k-truss)　G の連結な部分グラフ H が k-truss [3] であるとき，H の任意のエッジの support は $k-2$ 以上である．すなわち，H は $H = \{G' \subseteq G | \forall e \in E(G'), sup_{G'}(e) > (k-2)\}$ を満たす．ただし，グラフ G 中のエッジ $e(u, v)$ に対し，e の support $sup_G(e)$ は，エッジ e を含む三角形の数，すなわち，$sup_G(e) = |\{\triangle_{uvw} | w \in V\}|$ を表す．ここで，$\triangle_{uvw}$ はノード u, v, w によって作られる三角形の集合を示す．

k-truss は部分グラフの凝集性を内部の辺が構成する三角形の構造によって評価する．

定義 2.3 (query distance)　グラフ G 中のノード 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，G の部分グラフ $H \subseteq G$ において，ノード $v \in V$ から H の各頂点 $u \in H$ への query distance は $dist_H(H, v) = \max_{u \in H} dist_H(u, v)$ となる．

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 2.4 ((k, d)-truss)　クエリノード v_q およびパラメータ k, d が与えられたとき，G の部分グラフ H が *(k, d)-truss* であるとは，H と v_q の query distance が d 以下かつ H が *k-truss* を満たすことを表す．つまり $H = \{G' \subseteq G | v_q \in V(G'), dist_{G'}(G', v_q) \geqq d, \forall e \in E(G'), sup_{G'}(e) \geqq (k-2)\}$ を満たすノード集合は *(k, d)-truss* である．

定義 2.4 に示した *(k, d)-truss* は，k が大きいとき，凝集性が高いこと (性質 2) を示し，d が小さいとき，クエリ点までの距離が近いこと (性質 4) を示す．

定義 2.4 では，性質 2 と 4 の指標を示した．性質 1 の評価は自明であるため，ここでは，残る性質 3 について指標を検討する．まず，性質 3 の指標となる関数 $f(H, W_q)$ を，以下の二つのアイデアを元に構成する．

(1) H 中のノードはできるだけ属性 $w \in W_q$を含むべきである．

(2) G 中の属性 $w \in W_q$を含むノードはできるだけ H に含まれるべきである．

1 番目のアイデアを評価する基準として，ある一つの属性 w について，その重み $\theta(H, w)$ を定義する．

定義 2.5 (θ 関数) 部分グラフ H，属性 w が与えられたとき，関数 $\theta(H, w)$ を，以下に定義する．

$$\theta(H, w) = \frac{|V_w(H) \cap V(H)|}{|V(H)|}.$$

これは，H 中の全ノードの数に対する属性 w を含むノードの割合を表す．ここで，$V_w(H)$ は H 中の w に属すノード集合，すなわち以下の式を満たす集合である．

$$V_w(H) \subseteq V(H)\ s.t.\ \forall v \in V_w(H), w \in attr(v).$$

また，2 番目のアイデアを評価する基準として，関数 $score(H, w)$ を定義する．

定義 2.6 (score 関数) 部分グラフ H，属性 w が与えられたとき，関数 $score(H, w)$ を，以下に定義する．

$$score(H, w) = |V_w \cap V(H)|.$$

これは，H が G 中の属性 w を含むノードをいくつ含むかを表す．関数 $f(H, W_q)$ は，W_q 中の各属性に対するこれらの積を足し合わせたもの，すなわち次の式で表す．

$$\begin{aligned} f(H, W_q) &= \sum_{w \in W_q} \{\theta(H, w) \times score(H, w)\} \\ &= \sum_{w \in W_q} \frac{|V_w(H) \cap V(H)|^2}{|V(H)|}. \end{aligned}$$

定義 2.7 (Attribute Score Function) 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 Function $f(H, W_q)$ を，以下に定義する．

$$\begin{aligned} f(H, W_q) &= \sum_{w \in W_q} \{\theta(H, w) \times score(H, w)\} \\ &= \sum_{w \in W_q} \frac{|V_w(H) \cap V(H)|^2}{|V(H)|}. \end{aligned}$$

ただし，$\theta(H, w) = \frac{|V_w \cap V(H)|}{|V(H)|}$，$score(H, w) = |V_w \cap V(H)|$ である．

ATC 問題では，定義 2.4 および定義 2.7 を用いて，次に定義するコミュニティを検索する．

定義 2.8 (Attribute Truss Community) 入力グラフ G，入力クエリ $Q = (v_q, W_q)$，*(k, d)-truss* のパラメータ k, d に対して，部分グラフ $H \subseteq G$ が以下を満たすとき，部分グラフ H は Attribute Truss Community (ATC) である．

(1) H は v_q を含む *(k, d)-truss* である．

(2) H は条件 1 を満たす中で最大の $f(H, W_q)$ を持つ．

最終的に ATC 問題は次のように定義する．

問題定義 2.1 (Attribute Truss Community(ATC) 問題) グラフ $G(V, E)$，クエリ $Q = (v_q, W_q)$，及びパラメータ k, d が与えられたとき，ATC，すなわち $f(H, W_q)$ が最大となる *(k, d)-truss* である部分グラフ H を見つける．ただし，$f(H, W_q) = \sum_{w \in W_q} \frac{|V_w \cap V(H)|^2}{|V(H)|}$ である．

2.2.2 従来手法

前節で述べた ATC 問題は NP 困難であることが先行文献により証明されている [5]．そこで Huang らは，BASIC，BULK，および LocATC と呼ばれる手法を提案している．本節では各種法について概説する．

a） BASIC

BASIC は貪欲法に基づく最も素朴な解法である．BASIC では，ある解 H において，ノード $v \in V(H)$ を削除した際の Attribute Score の低下量を考え，その低下量を v の Attirbute Score への貢献度として解釈する．この指標を用いて，貢献度の小さいノードから削除していく際に，この削除操作によって解が (k, d)-truss でなくなることが考えられる．そこで，メンテナンスと呼ばれる (k, d)-truss の整合性を維持する処理を行い，これらの処理を H が (k, d)-truss でなくなるまで繰り返す．最後にこの過程で得られた最も Attribute Score の高い部分グラフを解とする．

クエリ属性集合 W_q において，暫定解 H からあるノードを削除したときのスコアの低下量は次の式で計算する．

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

ここで，ノード v に関係のある項のみを，スコアの低下量を代表して v のスコアへの貢献度とすることにする．

定義 2.9 (Attribute Score 貢献度) 部分グラフ H とクエリ属性集合 W_q が与えられたとき，ノード $v \in V(H)$ の貢献度 $f_H(v, W_q)$ を，以下に定義する．

$$f_H(v, W_q) = \sum_{w \in W_q \cap attr(v)} 2|V_w(H) \cap V(H) - 1|$$

BASIC のアルゴリズムを，Algorithm 1 に示す．BASIC では，始めに query distance が d 以下となる全ノードからなるグラフを初期解とする．貢献度の小さいノードから随時削除を行い，その過程にある，最も Attribute Score が高い部分グラフを解とする．ただし，ノードの削除により，暫定解が *(k, d)-truss* でなくなった場合には，メンテナンスとして以下の 2 つの処理を，該当するノードまたはエッジが無くなるまで繰り返す．

- 処理 1：support が $(k - 2)$ 未満であるエッジを削除する．
- 処理 2：query distance が d より大きいノードを削除する．

Algorithm 1 BASIC(G, Q)

Require: グラフ $G = (V, E)$，クエリ $Q = (v_q, W_q)$，パラメータ k, d
Ensure: $f(H, W_q)$ が最大となる *(k, d)-truss* H

1: query distance $\leqq d$ となるノード集合 $S_0 = u : dist_G(u, Q) \leqq d$ を見つける．
2: $G_0 = (S_0, E(S_0))$, ただし，$E(S_0) = \{(v, u) : v, u \in S_0, (v, u) \in E\}$
3: G_0 が *(k, d)-truss* となるよう，メンテナンスを行う．
4: Let $l \leftarrow 0$;
5: **while** G_lにおいて全ての Q が接続 **do**
6: $f(G_l, W_q)$ を求める．
7: 貢献度の最も小さいノード $u*$ を求める．
8: $u*$ とそれに接続するエッジを G_l から削除する．
9: G_l が (k, d)-truss となるよう，メンテナンスを行う．
10: $G_{l+1} \leftarrow G_l; l \leftarrow l + 1$;
11: **end while**

b） BULK

BULK は BASIC を効率化した手法である．BASIC はメンテナンスによって削除されるノードについては貢献度を考慮しないため，重要なノードを削除してしまう可能性がある．また，1 ループで削除可能なノード数が少ないため非効率である欠点がある．そこで，一度メンテナンスを行う毎に複数のノードを同時に削除することで，この効率の改善を図る．そのために BULK では定義 2.10，定義 2.11 に示す，Attribute Margin Gain という指標を導入する．

定義 2.10 (Attribute Margin Gain) Attribute Margin Gain $gain_H(v, W_q)$ を，以下に定義する．

$$gain_H(v, W_q) = f(H, W_q) - f(H - S_H(v), W_q)$$

ここで，$S_H(v)$ は，v の削除に伴って削除されるノード集合である．

定義 2.10 に示した Attribute Margin Gain は $S_H(v)$ の計算が非効率であるため近似を行う．v の隣接点のうち次数が $k-1$ の点は必ず削除されることを用いた近似を，次に定義する．

定義 2.11 (近似 Attribute Margin Gain) Attribute Margin Gain の近似 $gain_H(v, W_q)$ を，式 (2.11) にて定義する．

$$\overline{gain}_H(v, W_q) = f(H, W_q) - f(H - P_H(v), W_q)$$

ただし，$P_H(v) = \{u \in N(v) : deg_H(u) = k - 1\}$ とする．以降，単に Attibute Margin Gain と表すときは，式 (2.11) を指す．

BULK は Algorithm 2 に示すように式 (2.11) を用い，ある閾値 ϵ を設定した上で，Attribute Margin Gain が小さい方から $|S| = \frac{\epsilon}{1+\epsilon}|V(G_i)|$ 個を各ステップにて一度に削除する．

c） LocATC

LocATC は BULK に前処理を加えて性能を向上させ，入力のパラメータ k, d を自動的に決定する手法である．BULK および BASIC では，グラフ G のサイズが大きくなると非常に計算時間が長くなってしまうという欠点がある．これは，始めに解に含まれ得る全てのノードを含んだ状態から処理を開始することに起因する．そこで LocATC ではまず小さな初期グラフ G_t を作り，初期グラフへノードを追加した後 BULK を適用する．これにより本来の BULK より小さなグラフから処理を行うため処理の高速化をが実現できる．まず前処理による初期グラフ G_t を作るために，以下の定義を行う．

Algorithm 2 BULK(G, Q)

Require: グラフ $G = (V, E)$，クエリ $Q = (v, W_q)$，パラメータ k, d, ϵ
Ensure: $f(H, W_q)$ が最大となる (k, d)-truss H

1: 最大の $(k, d) - truss$ である G_0を求める．
2: Let $l \leftarrow 0$;
3: **while** G_lにおいて全ての Q が接続 **do**
4: $\overline{gain}_{G_l}(v, W_q)$ が最も小さいものから $|S| = \frac{\epsilon}{1+\epsilon}|V(G_i)|$ 個の集合 S を求める．
5: S とそれに接続するエッジを G_l から削除する．
6: G_l が (k, d)-truss となるよう，メンテナンスを行う．
7: $G_{l+1} \leftarrow G_l; l \leftarrow l + 1$;
8: **end while**

Algorithm 3 LocATC(G, Q)

Require: グラフ $G = (V, E)$，クエリ $Q = (v_q, W_q)$
Ensure: $f(H, W_q)$ が最大となる *(k, d)-truss* H

1: $G_0 \leftarrow \{v_q\}$.
2: $|V(G_0)| > \eta$ となるまで拡張可能ノード v を T へ追加し，グラフ G_t へ拡張する．
3: G_t 中の v_q を含む最大の k-truss を新たに G_t とし，そのときの k を k_{max} とする．
4: $k = k_{max}, d = dist_{G_t}(G_t, v_q)$ として G_t に Algorithm 2 を適用する．

定義 2.12 (拡張可能ノード) グラフ G, クエリ $Q = \{v_q, W_q\}$ において，以下の式を満たすノード $v \in V(H)$ を拡張可能ノードと呼ぶ．

$$\theta(G, W_q \cap attr(v)) \geqq \frac{f(G, W_q)}{2|V(G)|}$$

定義 2.12 を満たすノードは，$f(G \cup \{v\}, W_q) \geqq f(G, W_q)$ となることが文献 [5] により示されている．ここで，初期グラフ G_0 を $G_0 = \{v_q\}$ として，G_0 の拡張可能ノードを逐次追加し，$|V(G_0)| \leqq \eta$ である範囲で拡張したグラフを G_t とする．ここで η は経験的に求められる値である．

Algorithm 3 に，LocATC のアルゴリズムを示す．まず，LocATC はクエリノード近傍の拡張可能ノードを逐次追加して初期グラフ G_t を作る．次に，G_t 中の v_q を含む最も大きな k-truss を新たに G_t とし，併せてパラメータ k および d を決定する．最後に G_t に BULK を適用して解を出力する．

LocATC はグラフ全体を入力とする BULK と比較して，対象とする初期グラフが小さいため高速な計算が可能である．

2.3 LocATC の問題点

本節では LocATC の問題点について説明する．前節で述べた通り，従来の ATC 問題はその構造的制約条件として定義 2.4 を満たす必要があり，LocATC においては k はクエリノード周辺の truss 構造の頑健さによって定まる．しかし，現実のコミュニティ構造を考えた場合，コミュニティ構造が常に *(k, d)-truss* のような強い制約を満たすとは考えにくい．従って，LocATC では頑健な truss 構造を持つコミュニティしか見つけられず，例えばスター型のようなコミュニティの内部におけるエッジの密度が低いコミュニティを検索することができない．

3 提 案 手 法

3.1 提案手法の概要

提案手法では ATC 問題の構造的制約条件を緩和し，より高精度なコミュニティを検索することを目指す．前節で述べた通り，従来の ATC 問題は構造的制約条件として定義 2.4 を満たす必要があるが，これは現実的ではない．具体的には現実のコミュニティ構造においては *(k, d)-truss* のような強い制約を満たさないコミュニティが多く存在する．そこでまず，本研究では取り得る k の値を可変とすることにより，多様なコミュニティを解の候補とすることを考える．次に，その解の探索においてビームサーチを採用し，柔軟なコミュニティ構造を発見する．

本節では 3.2 節にて緩和した ATC 問題について示し，3.2.2 節にてその解法において重要なビームサーチを説明する．最後に 3.2.3 節にてビームサーチを用いた解法について述べる．

3.2 緩和 ATC 問題

前述した通り，ATC 問題は多くの truss 構造を含むコミュニティしか見つけることができないため，k を可変とした (k, d)-truss，すなわち (*, d)-truss を定義する．また，(*, d)-truss を用いて緩和した ATC 問題についてもここで定義する．

3.2.1 *(*, d)-truss*

従来の ATC 問題においてはその解は *(k, d)-truss* の条件を満たす必要があったが，ここではその truss 構造に関する制約を緩和し，より多様な形態のコミュニティを見つけ出すことを目指す．具体的には，本研究では *(k, d)-truss* における k を可変とする *(*, d)-truss* を考えることで，より多様な構造を持つコミュニティを対象とする．*(*, d)-truss* の詳細な定義を次に示す．

定義 3.1 ((*, 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，G の部分グラフ H が *(*, d)-truss* であるとは，query distance が d 以下となるノード集合である．つまり以下を満たすノード集合 H は *(*, d)-truss* である．

$$H = \{G' \subseteq G | v_q \in V(G'), dist_{G'}(G', v_q) \geqq d\}.$$

定義 3.1 より，*(*, d)-truss* は truss 構造の数が任意であるため，truss 構造の少ないコミュニティも解の候補となる．

定義 3.1 に示した *(*, d)-truss* に基づき，本研究では属性付きグラフに対する新たなコミュニティ検索問題として，緩和 ATC 問題を次のように定義する．

問題定義 3.1 (緩和 ATC 問題) グラフ $G(V, E)$，クエリ $Q = (v_q, W_q)$，及びパラメータ d が与えられたとき，ATC，すなわち $f(H, W_q)$ が最大となる *(*, d)-truss* である部分グラフ H を見つける．ただし，$f(H, W_q) = \sum_{w \in W_q} \frac{|V_w(H) \cap V(H)|^2}{|V(H)|}$ である．

3.2.2 ビームサーチを用いた検索

ビームサーチは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である．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．そこに，ビーム幅 B というパラメータを加え，遷移先の状態数，つまりキューのサイズが B を超えた場合，暫定評価の最も良い B 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，全探索より高速な探索を実現する．

Algorithm 4 ビームサーチの例

Require: 探索木 T，パラメータ B, 根ノード t_0
Ensure: **argmax**$_{t \in T} F(t)$
Let 優先度付きキュー $P \leftarrow \{t_0\}$
Let 優先度付きキュー $NextP \leftarrow \emptyset$
Let $maxValue \leftarrow 0$
while $P \neq \emptyset$ **do**
  Let $b \leftarrow 0$
  while $P \neq \emptyset$ **and** $b \leqq B$ **do**
    $t \leftarrow PopP()$
    $maxValue =$ **max**$(maxValue, F(t))$
    $T_t \leftarrow t$ の隣接ノード集合
    $PushNextP(T_t)$
    $b \leftarrow b + 1$
  end while
  $P \leftarrow MextP$
  $NextP \leftarrow \emptyset$
end while

ここで，例として有限な探索木 T 上のノード $t \in T$ の中から，最も高い評価値 $F(t)$ を探索するアルゴリズムを Algorithm 4 に示す．ただし，$F(t)$ は非負であるとする．

3.2.3 提案手法: ビームサーチを用いた緩和 ATC 問題の解法

ここではビームサーチを応用し緩和 ATC 問題を解く手法について説明する．詳細なアルゴリズムを Algorithm 5 に示す．

Algorithm 5 の概要： まず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 B に応じて最良の B 個だけ部分グラフをポップし，それぞれについてその部分グラフとその隣接ノードからなる *(k, d)-truss* を $2 \leqq k \leqq k_{max}$ の範囲で計算し，見つかったそれぞれの *(k, d)-truss* を個別に優先度付きキューにプッシュする．ただし，k_{max} は G 中の最も大きな *(k, d)-truss* における k の値とする．これを繰り返し，その過程で最も Attribute Score Function が高かった部分グラフを解として出力する．提案手法はこのように多様な解を評価しつつ枝刈りにより探索にかかるコストを低く抑えた手法である．

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 Function 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 Function の差分を Attribute Expand Gain として定義 3.2 に定義する．

定義 3.2 (Attribute Expand Gain) 部分グラフ H と，ノード $v \notin V(H)$ が与えられたとき，Attribute Expand Gain $f_v(H, W_q)$ を以下の式 (4) に定義する．

$$f_v(H, W_q) = \sum_{w \in W_q \cap attr(v)} \frac{2|V_w \cap (V(H))| + 1}{|V(H)| + 1} - \sum_{w \in W_q} \frac{|V_w \cap (V(H))|^2}{|V(H)|^2 + |V(H)|}$$

補題 3.1 (ノードを追加したグラフの Attribute Score Function) 部分グラフ H と，ノード $v \notin V(H)$ が与えられたとき，H に v を追加したグラフの Attribute Score Function $f(H+\{v\}, W_q)$ は，以下で表される．

$$f(H+\{v\}, W_q) = \sum_{w \in W_q} \frac{|V_w \cap (V(H))|^2}{|V(H)+1} + \sum_{w \in W_q \cap attr(v)} \frac{2|V_w \cap (V(H))|+1}{|V(H)|+1}$$

証明 1 まず，定義 2.7 より，部分グラフ H にノード v を追加したグラフの Attribute Score Function $f(H+\{v\}, W_q)$ は以下のように表せる．

$$f(H+\{v\}, W_q) = \sum_{w \in W_q} \frac{|V_w \cap (V(H)+\{v\})|^2}{|V(H)|+1}$$

ここで，v の持つ属性の集合 $attr(v)$ について着目する．$attr(v)$ の要素のうち W_q に含まれない要素については変化を考慮しなくて良いことを利用して，以下のように変形できる．

$$f(H+\{v\}, W_q) = \sum_{w \in W_q \setminus attr(v)} \frac{|V_w \cap (V(H))|^2}{|V(H)|+1} + \sum_{w \in W_q \cap attr(v)} \frac{(|V_w \cap V(H)|+1)^2}{|V(H)|+1}$$

次に，右辺の 1 番目の項を，$W_q \setminus attr(v) = W_q \setminus \{W_q \cap attr(v)\}$ であることを用いて展開する．

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

そして項をまとめ，$f(H+\{v\}, W_q)$ は以下のように表せる．

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

よって，補題 3.1 は示された． □

補題 3.2 (ノードの追加による Attribute Score Function の変化量) 部分グラフ H と，ノード $v \notin V(H)$ が与えられたとき，H に v を追加したグラフと元のグラフとの Attribute Score Function の差 $f_v(H, W_q)$ は，式 (4) となる．

$$f_v(H, W_q) = \sum_{w \in W_q \cap attr(v)} \frac{2|V_w \cap (V(H))|+1}{|V(H)|+1} - \sum_{w \in W_q} \frac{|V_w \cap (V(H))|^2}{|V(H)|^2+|V(H)|}$$

Algorithm 5 提案手法

Require: グラフ $G=(V,E)$，クエリ $Q=(v_q, W_q)$，パラメータ d, B
Ensure: $f(H, W_q)$ が最大となる *(*, d)-truss* H

```
Let 優先度付きキュー P ← {v_q}
Let 優先度付きキュー NextP ← ∅
Let bestAttributeScore ← 0
Let bestGraph ← ∅
while P ≠ ∅ do
    Let b ← 0
    while P ≠ ∅ and b ≦ B do
        graph ← PopP()
        if bestAttributeScore > f(graph) then
            bestGraph ← graph
            bestAttributeScore = max(maxValue, F(t))
        end if
        nextGraph ← graph の近傍の (*, d)-truss
        PushNextP(nextGraph)
        b ← b + 1
    end while
    P ← MextP
    NextP ← ∅
end while
```

証明 2 証明 (1) にて求めた Attribute Score Function $f(H+\{v\}, W_q)$ と，元のグラフの Attribute Score Function $f(H, W_q)$ の差を求める．

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

よって補題 3.2 は示された． □

ビームサーチによる解法はパラメータ k の調整が不要であるが，代わりにビーム幅 B を導入する．しかし k の最適な値はグラフの形に大きく依存し調整が容易でないのに対し，B は増加に伴いより時間を掛けて正確なコミュニティを計算することに期待できるため，調整が容易であるという利点がある．

Algorithm 5 はボトムアップに *(*, d)-truss* を構築しながら多様な可能性を保持して探索を行うため，貪欲的な部分グラフの構築である LocATC と比較してより正確なコミュニティを推定できる．また k の値を単一に決定しないため，truss 構造の少ないグラフにおいても効果を期待できる．

4 評価実験

本章では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 節では実験の結果について述べる．

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	5038	154	224
Cornell	195	304	94	1588
Texas	187	328	104	1501

4.1 実験方法

実験は従来手法である LocATC [5] と，提案手法を $B = \{10, 30\}$ と設定した手法に加え，*(*, d)-truss* を求める素朴な手法として LocATC における k を $2 \leqq k \leqq k_{max}$ の全てにおいて実行し，最も Attribute Score Function の高い解を出力する手法である Naïve の 4 つを比較して行う．本実験は Mac OS 10.14, intel Core i5(3.3GHz)，メインメモリ 16GB にて行った．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．

また本稿では 3 つの実データを対象に LocATC と提案手法を比較する．使用するデータは Facebook，Cornell，Texas である．

Facebook はソーシャルネットワークにおけるグラフであり，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．

Cornell と Texas は Web のハイパーリンクにおけるグラフであり，ノードは Web ページを表し，エッジは Web ページ間のリンクを示す．また Cornell と Texas は属性として Web ページ内に 10 回以上出現した単語の集合を用い，その Web ページが {”course”, ”student”, ”faculty”, ”project”, ”staff”} のいずれであるかをコミュニティとする．

Facebook のデータセットは Stanford Network Analysis Project [8] より，Cornell と Texas のデータセットは LINQS の Web ページより参照できる．[1] 表 4.1 にデータセットの詳細を示す．また，表 4.1 中の d_{max} はグラフの最大次数を表す．

入力するクエリは LocATC 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．またクエリノード集合はここでは単一のノードとし，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．

評価は以下に定義する F1-Score を用いて行う．

定義 4.1 (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．

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．

1：https://linqs.soe.ucsc.edu/

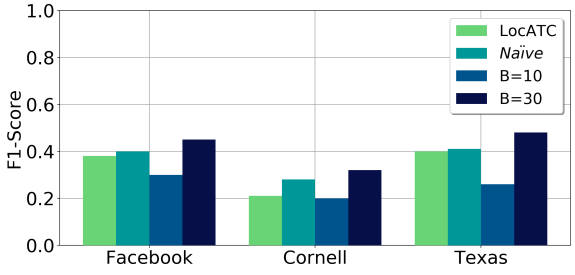

図 2　コミュニティ検索精度 (F1-Score) の比較

4.2 実験結果

本研究では，精度についての先行研究との比較と，ビーム幅による提案手法の精度の変化の 2 つを対象に実験を行なった．4.2.1 節，4.2.2 節にてそれぞれ説明する．

4.2.1 コミュニティ検索精度の比較

図 2 に精度についての実験結果を示す．図 2 より提案手法はビーム幅を十分に大きく確保した場合，先行研究 LocATC と比較して精度を 18%から 60%程度向上させることがわかる．これは，提案手法が *(*, d)-truss* を考慮することで，多様な構造を持つコミュニティを捉えられていることによるものであると考えられる．一方でビーム幅が小さな場合，先行研究と比較して 5%程度精度が低下する．この理由はビーム幅の小ささにより，提案手法が保持する探索の多様性が減少していることに起因する．探索の多様性が減少した結果，多くの有効な解の候補をその過程で枝刈りしていると考えられる．また Naïve は LocATC の精度をやや上回っており，k を唯一に決定することが精度の低下に繋がることがわかる．加えてビーム幅を大きく設定した提案手法は Naïve より精度が高い．この理由は提案手法が構造的制約の緩和のみでなく，多様な解の候補を保持した探索を行うためであると考えられる．以上の結果より，ビーム幅を十分に大きくすることで，提案手法は大幅にコミュニティ検索精度を向上させる可能性があることが示唆された．

4.2.2 ビーム幅の変化による精度への影響

本節では Facebook のデータセットを用いて，ビーム幅を $B = \{1, 5, 10, 20, 30\}$ と変化させたときの精度への影響を評価する．図 3 にビーム幅の変化による精度への影響についての実験結果を示す．図 3 より，本実験における B の範囲では提案手法はビーム幅を大きくするほど精度が向上することがわかる．これは LocATC のパラメータ η が経験的に求める必要があるのに対し，直感的な制御が可能であるという点で優れているといえる．

5 関連研究

属性付きグラフ上のコミュニティ検索に関連するいくつかの研究について，そのアイデアと代表的な手法について述べる．

属性付きコミュニティ検索： 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唯一 LocATC [5] がある．

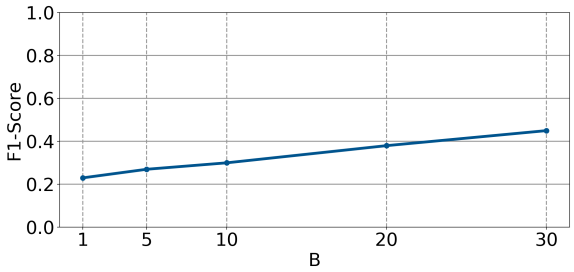

図 3　ビーム幅の変化による精度の変化

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を提案し，クエリ属性を持つノードが多く含まれるよう，コミュニティを拡張していく．その後，各ノードのコミュニティへの貢献度を定義し，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．本研究ではクエリノードは単一のノードとしたが，文献 [5] では複数のノードをクエリノードとする手法も提案されており，従来の属性を考慮しないコミュニティ検索に勝る精度と速度を実験にて示している．しかしながら，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．

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* [2, 9] を用いる手法や，*k-truss* [3,4] を用いる手法などがある．文献 [2] 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り，*k-core* を用いてグラフ中のノードの次数に着目し，貪欲的なアルゴリズムを提案した．文献 [9] 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 [3,4] は本研究の先行研究の著者である Huang らによって提案された．Huang らは *k-core* より頑健なグラフ構造を評価する指標として *k-truss* を採用し，グラフの truss 構造の中からコミュニティに不要なノードを計算し，優先度の低い順に貪欲的にノードを削除していく手法を提案し，実験により有効性を示した．いずれもグラフ構造に基づく手法であることから，頑健なコミュニティ構造を検索することに優れている．

6　結　　論

6.1　本研究のまとめ

本研究では大規模な属性付きグラフにおける高精度なコミュニティ検索手法を提案した．先行研究である LocATC は属性付きグラフにおけるコミュニティ検索問題をである ATC 問題を定式化し，その解法を提案した．しかし ATC 問題は構造的制約条件が強く，truss 構造を多く含むコミュニティしか解となり得ない．これは現実に存在するデータに対してコミュニティ検索をする上で好ましくない．そこで本稿では ATC 問題の構造的制約条件を緩和し，緩和した ATC 問題を効率的に計算する手法を提案した．提案手法では緩和した ATC 問題に対してビームサーチを属性付きコミュニティ検索に応用した．我々の実験において提案手法は新たに導入したビーム幅というパラメータを十分に大きくすることにより，従来手法である LocATC と比較して正確に真のコミュニティに近いコミュニティを算出でき，データセットに対して有効であることを確認した．

6.2　今後の課題

- **より大規模なグラフに向けての高速化** 提案手法は有効な精度を得るためには多大な計算時間を要する．本稿で実験に用いたデータセットは比較的小さなグラフであり，現実に存在するより大規模なグラフに対する実用は期待できない．そのためより大規模なグラフに向けて提案手法のボトルネックである隣接する *(*, d)-truss* の列挙を高速化する，あるいは行わなくて済む計算方法を提案し，計算量を改善する必要がある．
- **ビーム幅の変化後の精度と速度の分析** 4. 2. 2 節で示した通りビーム幅は要求に応じて制御しやすいパラメータである．そこで実験の回数を重ねビーム幅の変化と精度および速度の相関を調査することにより，実行時間の少ない小さなビーム幅でベンチマークテストを行い，その結果を用いて要求される精度あるいは速度に応じたビーム幅の調整が容易になることが期待できる．

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*, Vol. 2, No. 1, pp. 718–729, August 2009.

[2] 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)*, pp. 939–948, New York, NY, USA, 2010. ACM.

[3] Xin Huang, Laks V. S. Lakshmanan, Jeffrey Xu Yu, and Hong Cheng. Approximate Closest Community Search in Networks. *Proceedings of the VLDB Endowment*, Vol. 9, No. 4, pp. 276–287, December 2015.

[4] 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)*, 2014.

[5] Xin Huang and Laks Lakshmanan. Attribute-Driven Community Search. *Proceedings of the VLDB Endowment*, Vol. 10, No. 9, pp. 949–960, 2017.

[6] Jonathan D. Cohen. Trusses: Cohesive Subgraphs for Social Network Analysis. *National Security Agency*, 2008.

[7] Christian Blum, Maria J. Blesa, and Manuel López-Ibáñez. Beam search for the longest common subsequence problem. *Computers & Operations Research*, Vol. 36, No. 12, pp. 3178 – 3186, 2009. New developments on hub location.

[8] Jure Leskovec and Andrej Krevl. SNAP Datasets: Stanford Large Network Dataset Collection. `http://snap.stanford.edu/data`, June 2014.

[9] 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)*, pp. 991–1002, New York, NY, USA, 2014. ACM.