DEIM Forum 2022 G34-2

ウェッジ分割を用いた大規模グラフの要約手法

真次　彰平† 　塩川　浩昭††

† 筑波大学大学院理工情報生命学術院　〒 305-8571 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　〒 305-8571 茨城県つくば市天王台 1-1-1
E-mail: †matsugu@kde.cs.tsukuba.ac.jp, ††shiokawa@cs.tsukuba.ac.jp

あらまし　グラフ要約はグラフ中のいくつかのノード，およびエッジをひとつにまとめることにより，グラフサイズを削減する技術である．グラフ要約では (1) 情報の欠損が無い， (2) 圧縮率が高い， (3) 実装が容易であるという三つの要件が求められるが，従来の手法ではそれらを同時に満たすことができない．そこで本稿では，情報の欠損が無く高圧縮率なグラフ要約手法を提案する．提案手法は連結した 3 ノードに着目し要約を行い，それらの接続関係をビット列に変換することで．元のグラフに存在する全ての情報を高い圧縮率で要約する．本稿では実データを用いた実験を行い，従来手法と比較して高圧縮率な要約が行えることを確認した．

キーワード　グラフ，グラフ要約，圧縮

1　はじめに

近年ではウェブやソーシャルネットワーキングサービスの発展に伴い，大規模グラフが数多く登場している [1,2]．例えば，2020 年時点での Facebook のユーザ数は 27 億人を超えたと報告されており，これら実世界に存在するグラフの規模は現在もなお増加し続けている [3]．これらの大規模グラフから有用な知識を抽出する大規模グラフ分析 [4–7] は重要な研究課題である一方で，グラフの保持に要する空間コストが大きく，コモディティの計算機ではそのようなグラフ分析アルゴリズムを実行できないという問題がある [8–10]．

この問題を解決するために，グラフ要約 [11,12] が近年期待されている．グラフ要約はグラフ中のノードやエッジをひとつにまとめることによりグラフサイズを削減する技術である．図 1 はグラフ要約の概念図を示している．グラフ要約では図 1 左側に示すグラフを，図 1 右側の S_1〜S_6 ように，特定のノード集合とエッジ集合をまとめたスーパーノード，スーパーエッジに変換する．これにより，元のグラフの特徴を残しつつ，グラフサイズを削減する．グラフ要約はグラフの構造を保ちつつデータ容量を圧縮できるため，クエリ応答への効率化 [13–15] やグラフデータの可視化 [16,17] など幅広く応用されてきた．

しかしながら，既存のグラフ要約手法には二つの問題点が存在する．一つ目の問題点は計算コストが大きいという点である．グラフ要約において各スーパーノードに集約されるノード集合を計算する過程では，膨大な組み合わせが考えられ，実用的なグラフ要約のためには効率的な計算手法の設計が求められる．またこの問題は NP 困難であることが知られており，最新のグラフ要約手法を用いた場合でも 1 億エッジ規模のグラフに数時間の計算時間を必要とする．二つ目の問題点は情報の欠損が発生する場合があるという点である．グラフ要約手法には要約グラフから元のグラフを再構築できないものがあるが，このような情報の欠損は，後のグラフ分析の結果に悪影響を及ぼすため最小限に抑えなくてはならない．

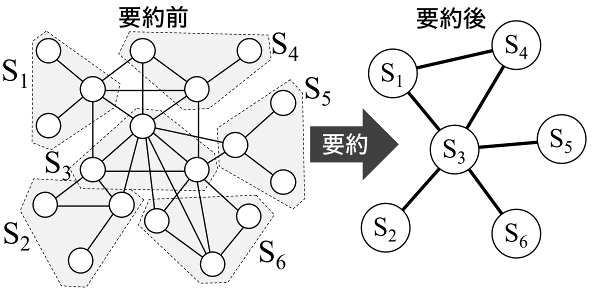

図 1　グラフ要約のイメージ図

1.1　既存研究と本研究の位置付け

上述の問題点を解消するために，これまで多くの手法が提案されてきた．既存手法には大きく分けて二つのアプローチが存在する．一つ目のアプローチは，補正セット法である．補正セット法ではスーパーノード内部のノード集合は完全グラフ，およびスーパーエッジにより隣接するノード集合の対は完全二部グラフとして元のグラフ上で表現されるという仮定をおく．しかしながら，そのような密構造のみで実世界のグラフをすべて表現することはできないため，補正セットと呼ばれる例外的なエッジ集合を用意することにより情報の欠損を補完する．補正セット法に基づく最新の手法である SWeG [12] は情報欠損が無いという点で優れているが，分散コンピューティングによる実装を主軸としており，単一の計算機による実行では速度・圧縮率の両面で性能が悪いという欠点がある．二つ目のアプローチは，効用制限法である．効用制限法では元のグラフのノードやエッジに効用 (Utility) と呼ばれるエッジの重要性を示す値を定義し，効用の大きなノード・エッジのみを要約する．具体的にはある閾値が与えられ，グラフ全体の効用がその閾値を下回らない範囲でノード数を最小にするよう手法を設計する．効用制限法に基づく最新の手法である UDS [11] は単一のマシンでも効率的な計算が可能であるが，情報の欠損が発生するため実用的でない．

以上のことから，これまでに提案されたグラフ要約手法は，(1) 効率的な要約を (2) 情報の欠損なく (3) 単一の計算機で実行可能であるという 3 つの望ましい要件を同時に満たさない．そのため本稿では，これら 3 つの要件を同時に満たす新たなアプローチ：**ウェッジ分割法**によるグラフ要約手法を提案する．

1.2 本研究の貢献

本稿では情報の欠損の無い高速・高圧縮率なグラフ要約手法を提案する．提案手法はウェッジと呼ばれる 3 ノードからなるパスあるいはサイクルに着目し，グラフ全体をウェッジによって分割する．このとき，要約グラフの圧縮率がウェッジ分割におけるスーパーエッジの重みの平均値に依存することを理論的に導出し (定理 1)，スーパーエッジの重みの平均値の最小化問題に対するヒューリスティックな解法を設計する．さらには，分割されたウェッジをそれぞれスーパーノードとみなし，スーパーノード間のエッジ集合をまとめてスーパーエッジとすることで，効率的なグラフ要約を行う．また情報の完全性を満たすための最適化として，スーパーノード内におけるローカルなノード id を定義することにより，スーパーエッジ内のエッジの組み合わせを 2 進数表現によって効率的に格納する．

本研究の貢献は以下の通りである．

- **高圧縮率**：ウェッジ分割法に基づいた提案手法は，実験により提案手法は既存手法 [11,12] と比較して最大 5 ポイント良い圧縮率を得られた (4.2 節).
- **高速**：提案手法は高速なスーパーノード候補の探索を行い，実験により既存手法と比較して最大 2.5 倍高速であることを示した (4.3 節).
- **正確**：提案手法は元のグラフにある全ての情報を要約グラフに保存しているため，情報の欠損が無い．
- **実用的**：提案手法はシングルスレッドマシンで動作するよう実装されているため，実用的である．

我々の知る限り，提案手法は前述した (1) 圧縮効率， (2) 情報の完全性， (3) 実装の容易さを兼ね備えた最初の手法である．本研究を通じて一般の大規模グラフ分析アルゴリズムは従来の空間消費量による適用可能性の限界を超え，より多様な計算環境でのグラフデータ分析を可能とする．

2 事前準備

本稿で用いる主な記号を 表 1 に定義する．本稿では連結な無向グラフ $G(V, E)$ を考える．V, E はそれぞれノード集合およびエッジ集合である．また，部分ノード集合 $V' \subseteq V$ の誘導部分グラフを $G[V']$，グラフ G 中のノード v の次数を $deg_G(v)$ と書き，文脈によって G が自明で有るとき単に $deg(v)$ と書く．本節ではまず要約を定義する．

定義 1 (要約)
グラフ $G(V, E)$ が与えられたとき，写像 $f : V \to S$ を G の要約と呼ぶ．ここで，S はスーパーノード集合である．また復元の写像 $F : S \to 2^V$ を $F(s) = \{v \in V | f(v) = s\}$ と定義し，$F(s)$ をスーパーノード s の元ノード集合と呼ぶ．

ここで，f は単射でないため F の像がノード集合となることに注意したい．次に，グラフ要約を定義する．

表 1　本稿で用いる主な記号

記号	意味
G	グラフ
V, E	ノード集合，エッジ集合
$V(g), E(g)$	グラフ g のノード集合，エッジ集合
$N(v)$	ノード v の隣接ノード集合
$G[V']$	ノード集合 V' の誘導部分グラフ
$deg(v)$	ノード v の次数
S	スーパーノード集合
P	スーパーエッジ集合
$\sigma : P \to \mathbb{R}$	スーパーエッジの重み
$f : V \to S$	ノード集合からスーパーノード集合への要約
$F : S \to 2^V$	スーパーノード集合からノード集合への要約
$\mathcal{F}$	$V \to S$ の要約写像全体の集合
$\mathcal{W}$	ウェッジ全体の集合
$\mathcal{W}_{TRI}$	トライアングル型ウェッジ全体の集合
$\mathcal{W}_{PATH}$	パス型ウェッジ全体の集合
$loc(v)$	ノード v のローカルノード id (定義 6)
$blew((s,t))$	ウェッジ s, t 間の BLEW (定義 7)

定義 2 (グラフ要約)
グラフ $G(V, E)$，および要約 f が与えられたとき，次の 3 つの制約を満たす圧縮されたグラフ $f(G) = G'(S, P, \sigma)$ を G の要約グラフと呼ぶ．

（1） $S = \{f(v) | \forall v \in V\}$.
（2） $P = \{(s,t) | (\bigcup_{i \in F(s), j \in F(t), (i,j) \in E} (i,j) \neq \emptyset\}$.
（3） $\sigma = (s,t) \in P \mapsto |\bigcup_{i \in F(s), j \in F(t), (i,j) \in E} (i,j)|$.

定義 2 に示す 3 つの制約について順に説明する．制約 (1) は，スーパーノード集合が元ノード集合の f による像の集合で表されることを示す．制約 (2) は，スーパーノード s, t の元ノード集合対 $F(s), F(t)$ の間に一本でもエッジがある場合それらのスーパーノード間にスーパーエッジを張ることを示す．制約 (3) は，スーパーエッジの重みは $F(s), F(t)$ 間のエッジの本数とすることを示す．図 1 はグラフ要約の例である．図 1 において，18 ノードからなるノード集合 V は f によって S へと要約される．例えばスーパーエッジ (s_1, s_3) について，それぞれの元ノード集合の間には 2 本のエッジが存在するため，スーパーノード (s_1, s_3) が張られ，その重み $\sigma((s_1, s_3))$ は 2 となる．次に，グラフ要約問題について定義する．

問題定義 1 (グラフ要約問題)
グラフ $G(V, E)$ が与えられたとき，グラフ要約問題を以下に示す要約 f^* を見つける最小化問題として定義する．

$$f^* = \arg\min_{f \in \mathcal{F}} |P|.$$

ただし，$f(G) = G'(S, P, \sigma)$，$\mathcal{F}$ は要約写像全体の集合である．

問題定義 1 より，グラフ要約問題はスーパーエッジ数の最小化問題といえる．本来ならばグラフを保持するコストには S も含まれるが，一般的に $|S| \ll |P|$ であるためそのコストは無視できるほど小さい．なお，グラフ要約問題は NP 困難である [18].

3 提案手法

本稿ではグラフ要約問題を解く新たな手法を提案する．2 節で述べた通り，グラフ要約問題は NP 困難であるため，ヒューリスティックな解法を設計する．まず 3.1 節で提案手法の基本アイデアについて述べる．次に 3.2 節では独自のアプローチであるウェッジ分割法の詳細について述べ，3.3 節ではそのアルゴリズムについて概要を示す．しかし，ここで述べる手法には情報の欠損が含まれるため，3.4 節にて完全な情報を要約グラフ内に効率的に格納する最適化について説明する．

3.1 基本アイデア

本稿で提案するグラフ要約手法はグラフをウェッジと呼ばれる部分構造毎に切り分けることで要約を行う．3.2 節では，提案手法の基本的なアイデアと定理を示す．次に 3.3 節では，前節で示した定理を基に 3 つのアルゴリズムを提案する．最後に，3.4 節にて要約グラフの完全な情報を格納する新たなエッジ重みとして BLEW を導入し，情報の欠損が無い要約を実現する．

3.2 ウェッジ分割法

本節では新たなグラフ要約アプローチ：ウェッジ分割法について示す．まず，ウェッジを次のように定義する．

定義 3 (ウェッジ)
グラフ $G(V,E)$ に存在するウェッジとは，次の制約を満たす 3 ノード組 $a,b,c \in V$ のことを指す．

$$(a,b) \in E \wedge (b,c) \in E$$

また，$(a,c) \in E$ であるものをトライアングル型，そうでないものをパス型のウェッジと呼び，あらゆるウェッジは両者のいずれかに分類される．さらに，G 中のウェッジ全体の集合を $\mathcal{W}(G)$，トライアングル型ウェッジの集合を $\mathcal{W}_{TRI}(G)$，パス型ウェッジの集合を $\mathcal{W}_{PATH}(G)$ と表す．

ウェッジは実世界のグラフで頻繁に見られる構造であるため，様々なグラフ分析に用いられる他，グラフに存在するウェッジの数はグラフの傾向を示す特徴量としてもよく知られている．

次に，定義 3 を用いて，ウェッジ分割について定義する．

定義 4 (ウェッジ分割)
グラフ $G(V,E)$，ダミーノード集合 V_{DUM}，およびダミーエッジ集合 E_{DUM} に対して，次の (1) ～ (4) に示す制約を満たす要約 f_w を G のウェッジ分割と呼び，f_w による G の要約グラフ $G'(S,P,\sigma)$ をウェッジ分割グラフと呼ぶ．すなわち，

$$f_w(G(V \cup V_{DUM}, E \cup E_{DUM})) = G'(S,P,\sigma).$$

＜制約＞

（1） $S = \{f(v) | \forall v \in V \cup V_{DUM}\}$.

（2） $P = \{(s,t) | (\bigcup_{i \in s, j \in t, (i,j) \in E \cup E_{DUM}} (i,j)) \neq \emptyset\}$.

（3） $\sigma = (s,t) \in P \mapsto |\bigcup_{i \in s, j \in t, (i,j) \in E \cup E_{DUM}} (i,j)|$.

（4） $\forall s \in S, F_w(s) \in \mathcal{W}$.

ここで，$F_w(s)$ は s の元ノード集合である．

定義 4 は，f_w によって G が G' へと要約されることを示す．ただし，グラフの構造上ウェッジとして構成できず制約 (4) に違反する場合があるため，ダミーノード集合やダミーエッジ集合を用いてこれを補正する．また，制約 (1) ～ (3) は，定義 2 に対応する制約である．加えて，制約 (4) は，全てのスーパーノードがウェッジであることを示す．ウェッジ分割グラフはスーパーノードが全てウェッジになるように要約されたグラフであり，効率的な要約のためにはダミーノードやダミーエッジの数を減らすことが重要である．

本研究では，定義 2 に示したウェッジ分割を用いてグラフ要約問題を近似的に解く．2 節で述べたように，グラフ要約はスーパーエッジ数の最小化問題である．そこで，スーパーエッジの少ないグラフ要約に関して次の補題および定理を導入する．

補題 1
グラフ $G(V,E)$ とそのウェッジ分割 $f_w(G) = G'(S,P,\sigma)$ において，次の式が成り立つ．

$$3S_{TRI} + 2S_{PATH} + \sum_{p \in P} \sigma(p) = |E|.$$

ただし，S_{PATH}, S_{TRI} はそれぞれウェッジ $G[s] (s \in S)$ がトライアングル型である数とパス型である数，すなわち，$S_{TRI} = |S \cap \mathcal{W}_{TRI}|, S_{PATH} = |S \cap \mathcal{W}_{PATH}|$ である．

証明
E を次のような二つの集合 $E_1 = \bigcup_{s \in S} E(G[F_w(s)]), E_2 = E \setminus E_1$ に分割する．ここで E_1 はウェッジ内部のエッジの集合であり，E_2 はその他のエッジの集合である．E_1 について，$S = (S \cap \mathcal{W}_{TRI}) \cup (S \cap \mathcal{W}_{PATH})$ であり共通部分は無いため，

$$E_1 = \bigcup_{s \in S \cap \mathcal{W}_{TRI}} E(G[F_w(s)]) \cup \bigcup_{s \in S \cap \mathcal{W}_{PATH}} E(G[F_w(s)]).$$

このとき，定義 3 より，スーパーノード内のエッジ数はトライアングル型であれば 3，パス型であれば 2 であるため，

$$|E_1| = 3|S \cap \mathcal{W}_{TRI}| + 2|S \cap \mathcal{W}_{PATH}| = 3|S_{TRI}| + 2|S_{PATH}|.$$

E_2 について，E_2 は複数のスーパーノードに跨ったエッジであるため，定義 4 より $E_2 = \sum_{p \in P} \sigma(p)$ である．以上より，

$$|E| = |E_1| + |E_2| = 3|S_{TRI}| + 2|S_{PATH}| + \sum_{p \in P} \sigma(p).$$

となり，補題は成り立つ． □

定理 1
グラフ $G(V,E)$ のウェッジ分割問題の目的関数 $|P|$ について，$|P| = \frac{|E| - |S_{TRI}| - \frac{2}{3}|V|}{avg(\sigma)}$ が成り立つ．ここで，$avg(\sigma)$ はスーパーエッジの重みの平均値である．

証明
まず，スーパーエッジの重みの平均値は，$avg(\sigma) = \frac{\sum_{p \in P} \sigma(P)}{|P|}$ と表せる．補題 1 より，$3|S_{TRI}| + 2|S_{PATH}| + \sum_{p \in P} \sigma(p) = |E|$ であるため，次が成り立つ．

$$|P| = \frac{|E| - (3|S_{TRI}| + 2|S_{PATH}|)}{avg(\sigma)}.$$

また $|S| = |S_{TRI}| + |S_{PATH}|$ より，以下の式が成り立つ．

$$|P| = \frac{|E| - |S_{TRI}| - 2|S|}{avg(\sigma)}.$$

ウェッジ分割の定義より $|S| \approx \frac{1}{3}|V|$ であるため，

$$|P| = \frac{|E| - |S_{TRI}| - \frac{2}{3}|V|}{avg(\sigma)}.$$

が成り立ち，定理は成り立つ． □

定理 1 は，優れたウェッジ分割の基準を示す．目的関数 $|P|$ は

Algorithm 1 DEGREE

Input: グラフ $G(V,E)$;
Output: 要約グラフ $G'(S,P,\sigma)$;
1: $S \leftarrow \emptyset$;
2: $V_{DUM} \leftarrow \emptyset$;
3: $G \leftarrow$INIT_DEGREE (G);
4: **while true do**
5: **If** 全てのノードが選択済み
6: **break**;
7: $S \leftarrow S \cup$ SELECT_WEDGE_DEGREE (G,S);
8: S にダミーノードが追加されたならば V_{DUM}, E_{DUM} を更新;
9: **end while**
10: P,σ を計算;
11: **return** $G'(S,P,\sigma)$;

12: **function** INIT_DEGREE (G) **:**
13: V を次数昇順にソート;
14: **return** G;

15: **function** SELECT_WEDGE_DEGREE (G,S)**:**
16: $v,w \leftarrow$ 新規ダミーノード;
17: $a \leftarrow V$ の最も先頭に近い未選択のノード;
18: $b \leftarrow N(a)$ 中の最も次数の小さい未選択のノード;
19: **If** $b =$ **null**
20: **return** $\{\{a,v,w\}\}$;
21: $c \leftarrow N(b)$ 中の最も次数の小さい未選択のノード;
22: **If** $c =$ **null**
23: **return** $\{\{a,b,v\}\}$;
24: **return** $\{\{a,b,c\}\}$;

$|V|$，$|E|$，$|S_{TRI}|$，$avg(\sigma)$ によって表され，その中でも $|V|$，$|E|$ は入力変数である．すなわち，グラフ要約の圧縮率を上げるにはトライアングル型ウェッジの数とスーパーエッジ重みの平均値を増やせば良いことがわかる．

3.3 アルゴリズム

本節では提案手法がウェッジ分割を行うアルゴリズムについて説明する．まず素朴なアルゴリズムとして，ノード集合を次数順にソートして，順にスーパーノードを構築する手法である DEGREE を説明する．次にスーパーエッジ密度を上げ圧縮率を高めた手法である CORE を示し，最後にその両者を統合した HYBRID を説明する．

3.3.1 DEGREE

DEGREE の概要: DEGREE のアルゴリズムを Algorithm 1 に示す．DEGREE はまず初期化関数 INIT_DEGREE(G) を呼び出し，ノードを次数昇順に並び替える (3 行目)．以降は，ウェッジ選択関数 SELECT_WEDGE_DEGREE(G,S) を呼び出し，最も次数の小さいノードから作られるウェッジを一つ抽出し，S に追加する (7 行目)．この処理を全てのノードが選択されるまで行う．DEGREE は次数の小さい部分から優先的に確定させるため，ウェッジを構築できずにダミーノードを使う回数が減るという長所を持つが，スーパーエッジの数が多くなるという短所がある．最後に，DEGREE の時間計算量について述べる．

定理 2
DEGREE の平均時間計算量は，$O(|V|(log|V|+d_{avg}^2)$ である．ただし，d_{avg} はグラフ G の平均次数である．

証明
INIT_DEGREE では，ノード集合のソートに時間計算量 $O(|V|log|V|)$ を要する．SELECT_WEDGE_DEGREE は全体で $O(|V|)$ 回実行され，一回の実行に時間計算量 $O(d_{avg}^2)$ を要する．よって，命題は成り立つ． □

Algorithm 2 CORE

Input: グラフ $G(V,E)$;
Output: 要約グラフ $G'(S,P,\sigma)$;
1: $S \leftarrow \emptyset$;
2: $V_{DUM} \leftarrow \emptyset$;
3: $(\Delta, k_{max}) \leftarrow$INIT_CORE (G);
4: **while true do**
5: **If** 全てのノードが選択済み
6: **break**;
7: $t \leftarrow k_{max}$
8: $(S',t) \leftarrow$ SELECT_WEDGE_CORE (G,S,Δ,t);
9: $S \leftarrow S \cup S'$;
10: S にダミーノードが追加されたならば V_{DUM}, E_{DUM} を更新;
11: **end while**
12: P,σ を計算;
13: **return** $G'(S,P,\sigma)$;

14: **function** INIT_CORE (G)**:**
15: 配列 $\Delta \leftarrow \emptyset$
16: **For** $k = 2,3,...$
17: $\Delta(k) \leftarrow$G 中の k-core;
18: **If** $\Delta(k) =$ **null**
19: **return** $(\Delta, k-1)$;

20: **function** SELECT_WEDGE_CORE (G,S,Δ,t)**:**
21: **If** $|\Delta(t)| < 6$;
22: $t--$;
23: **If** $t \neq 1$
24: $d \leftarrow \Delta(t)$ から 6 ノードをß取り出す;
25: $\Delta(t) \leftarrow \Delta(t) \setminus d$;
26: $(s_1,s_2) \leftarrow d$ を 2 つのウェッジに分割;
27: **return** $(\{s_1,s_2\},t)$;
28: **If** 残ったグラフにウェッジ s がある
29: **return** $(\{s\},t)$;
30: $s \leftarrow$ 新規ダミーノードを用いてウェッジを作成;
31: **return** $(\{s\},t)$;

3.3.2 CORE

次に，スーパーエッジの密度を高めることに主眼を置いた手法である CORE を説明する． 3.2 で述べた通り，圧縮率の高いウェッジ分割を行うには，トライアングル型を多く作り，各スーパーエッジの重みを高くする必要がある．言い換えると，任意のスーパーノード対は元のグラフにおいてエッジが密なスーパーノード対とエッジの接続されないスーパーノード対に二極化されることが望ましい．そこで，密な部分グラフを抽出するために k-core と呼ばれる密部分グラフのモデルを導入する．

定義 5 (*k*-core)
グラフ $G(V,E)$ および整数 k について，ノード集合 C が k-core であるとは，$G[C]$ における全てのノードの次数が k 以上である，すなわち，$\forall v \in C, deg_{G[C]}(v) \geqq k$ を満たすことをいう．

k-core は k が大きいほど密度が大きくなる傾向にある．本研究では k-core の抽出はバケットソートを用いた時間計算量 $O(|E|+|V|)$ のアルゴリズム [19] を利用する．

CORE の概要: CORE のアルゴリズムを Algorithm 2 に示す．CORE は，まず初期化関数 INIT_CORE を呼び出し，予め全ての k に対する k-core を列挙する (3 行目)．以降は，ウェッジ選択関数 SELECT_WEDGE_CORE を呼び出し，その戻り値である

Algorithm 3 HYBRID

```
Input: グラフ G(V, E);
Output: 要約グラフ G'(S, P, σ);
 1: S ← ∅;
 2: V_DUM ← ∅;
 3: G ←INIT_DEGREE (G);
 4: (Δ, k_max) ←INIT_CORE (G);
 5: while true do
 6:     If 全てのノードが選択済み
 7:         break;
 8:     t ← k_max
 9:     S ← S∪ SELECT_WEDGE_DEGREE (G, S);
10:     (S', t) ← SELECT_WEDGE_CORE (G, S, Δ, t);
11:     S ← S ∪ S';
12:     S にダミーノードが追加されたならば V_DUM, E_DUM を更新;
13: end while
14: P, σ を計算;
15: return G'(S, P, σ);
```

ウェッジを S に追加する (8 行目)．SELECT_WEDGE_CORE では，k が大きな k-core に属するウェッジから順に取り出す (24〜26 行目)．残りのノードが少なくなり，ウェッジが作成できなくなれば，ダミーノードを用いてウェッジを作成する (30 行目)．CORE では密度の高い 6 ノードを取り出すことによって，それらの間に構築されるスーパーエッジの密度が高くなるよう要約を行う．CORE はスーパーエッジの本数が減るという長所を持つが，要約の終盤にどのウェッジにも属さなくなった余りのノードが発生してしまい，ダミーノードが多くなるという短所がある．最後に，CORE の時間計算量について述べる．

定理 3

CORE の平均時間計算量は，$O(|E| + |V|)$ である．

証明

INIT_CORE では，k-core の列挙に時間計算量 $O(|E| + |V|)$ を要する．SELECT_WEDGE_CORE では，全てのノードは一回ずつ処理されるため，時間計算量は $O(|V|)$ である．よって，CORE の平均時間計算量は $O(|E| + |V|)$ である． □

3.3.3 HYBRID

前述した DEGREE および CORE は互いに正反対の長所と短所を持つ．それぞれの手法を統合した HYBRID を提案する．

HYBRID の概要：HYBRID のアルゴリズムを Algorithm 3 に示す．HYBRID は，CORE における INIT_CORE と SELECT_WEDGE_CORE の直前にそれぞれ INIT_DEGREE と SELECT_WEDGE_DEGREE を実行することにより，ダミーノードの原因になりやすい次数の小さなノードを早期にウェッジとして保存し，同時に密なスーパーエッジを構築することを目指す．最後に，HYBRID の時間計算量について述べる．

定理 4

HYBRID の平均時間計算量は，$O(|E| + |V|log|V|)$ である．

証明

INIT_DEGREE では，ノード集合のソートに時間計算量 $O(|V|log|V|)$ を要する．INIT_CORE では，k-core の列挙に時間計算量 $O(|E| + |V|)$ を要する．その後各ノードは 1 度ずつ処理されるため，SELECT_WEDGE_DEGREE および SELECT_WEDGE_CORE は $O(|V|)$ 回呼び出される．SELECT_WEDGE_DEGREE は隣接ノード集合を走査するが，SELECT_WEDGE_DEGREE は次数の小さい部分に対してのみ実行される．実世界のグラフの次数分布は冪乗則に従う [20] ため，この計算量は無視できるほど小さい．以上のことから，HYBRID の平均時間計算量は $O(|E| + |V|log|V|)$ である． □

HYBRID は，DEGREE と CORE が持つ圧縮率に関する長所を併せ持つだけでなく，DEGREE が多くの実行時間を要する部分を補い高速な要約を実現する．

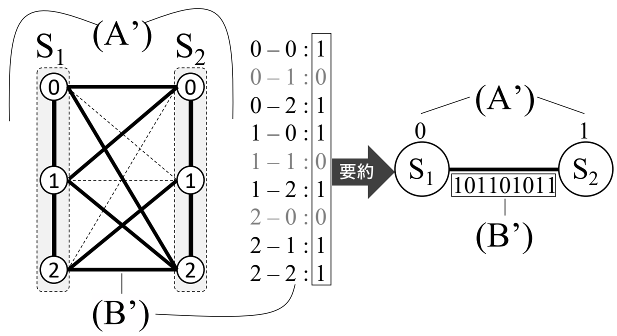

図 2 BLEW のイメージ図

3.3.4 各アルゴリズムの特徴

表 2 に 3 つのアルゴリズムの特徴を示す．DEGREE はダミーノードの生成原因となりやすい次数の小さいノードからウェッジを構築するため，ダミーノードが少ない．これに対し，エッジの密度が高い部分をまとめることができないため，スーパーエッジの数は多い．また，DEGREE の時間計算量は 3 つの中で最も大きい．逆に，CORE は k-core に着目してウェッジを作るためスーパーエッジが少ないが，次数の小さいノードが多く残ってしまうため多くのダミーノードを必要とする．また，CORE の時間計算量は 3 つの中で最も小さい．これら二つのアルゴリズムに共通して言えることは，処理の前半には効率良くウェッジを構築し，後半には非効率になってしまうということである．HYBRID は，双方の処理を交互に実行することで両者の長所を併せ持ったアルゴリズムである．HYBRID の時間計算量は CORE と比較して若干大きいが，DEGREE と比較して小さい．具体的には $|V|d_{avg} \approx |E|$ であるため，DEGREE の時間計算量は $O(|E|d_{avg} + |V|log|V|)$ となる．

表 2 アルゴリズムの特徴

アルゴリズム	ダミーノード	スーパーエッジ	時間計算量
DEGREE	少ない	多い	$O(\|V\|(log\|V\| + d_{avg}^2)$
CORE	多い	少ない	$O(\|E\| + \|V\|)$
HYBRID	少ない	少ない	$O(\|E\| + \|V\|log\|V\|)$

3.4 BLEW: Binary Local Edge Weight

3.3 節にて述べた提案手法は，スーパーエッジの内部情報として自身が内包するエッジ数のみを持つため，情報の欠損が生じる場合がある．具体的には，各スーパーノードの内部およびスーパーエッジが正確にどのように接続しているかわからない．そのため本稿では，圧縮効率を同等に保ちつつ，完全な情報を格納する拡張手法である **BLEW** (Binary Local Edge Weight) について説明する．

3.4.1 ローカルノード id

本節では，元のノード id を辿らずにスーパーノード内部の構造を特定するために，ローカルノード id を導入する．

定義 6 (ローカルノード id)
ウェッジ s 中の次数が 2 であるノード $v \in F_w(s)$ を 1 つ選択し，v 以外の 2 ノードを $u, w \in F_w(s)$ とする．このとき，$(loc(u), loc(v), loc(w)) = (0, 1, 2)$ を s のローカルノード id と呼ぶ．また，ローカルノード id からグローバルノード id への写像を $loc_s^{-1} : \{0, 1, 2\} \rightarrow F_w(s)$ で表す．

ここで，一般にローカルノード id の決め方は複数存在するが，差異は無いため適当な方法でひとつに決定する．ローカルノード id を用いると，各ウェッジは $(0, 1)$ 間および $(1, 2)$ 間に必ずエッジが存在することがわかり，さらにトライアングル型であれば $(0, 2)$ 間にも存在する．そのため，各スーパーノードにパス型であるかトライアングル型であるかの 1 bit の情報を格納することでスーパーノード内部のエッジを完全に保存できる．

3.4.2 BLEW

本節では，スーパーエッジの内部の情報，具体的には，ウェッジ内のどのノードとどのノードが接続しているのかという情報を格納する手法を説明する．まず，スーパーエッジの重み σ に置き換わる新たなエッジの重みである BLEW を定義する．

定義 7 (BLEW)
ウェッジ s, t について，その間に張られるスーパーエッジの重み $blew : P \rightarrow \mathbb{R}$ は，次の式で表される．

$$blew((s,t)) = \sum_{i,j \in \{0,1,2\}, (loc_s^{-1}(i), loc_t^{-1}(j)) \in E} 2^{3i+j}.$$

blew は，2 つのウェッジ間にある最大 9 エッジの有無をローカルノード id を用いて 9 bit の 2 進数に変換した値である．図 2 は BLEW の例である．2 つのウェッジ s_1 と s_2 はローカルノード id が割り振られており，s_1 はパス型，s_2 はトライアングル型である．ローカルノード id の導入によりそれぞれのウェッジは $(0, 1), (1, 2)$ にエッジが存在することが明らかである．ここで，図 2 (A') に示されたエッジ (各 1 bit) がそれぞれのウェッジがパス型かトライアングル型かを決定する．また BLEW により s_1, s_2 間のエッジ集合を (B') に示すビット列で表現することが可能である．このビット列は厳密に 9 bit からなるため，$2^9 = 512$ 通りの全てのエッジ情報を格納できる．本研究はスーパーノードの形をウェッジに固定することによりスーパーエッジの整数表現を可能にした．提案手法は，グラフ $G(V, E)$ をウェッジ分割 $f_w(G) = G'(S, P, blew)$ により要約する．

4 評価実験

本節では提案手法の圧縮率と速度，およびダミーノード・ダミーエッジの生成数を実験的に評価する．

4.1 実験設定

データセット: 実験に用いたデータセットを表 3 に示す．本稿では SNAP [1] に公開されている 4 つのデータセットを用いる．表の d_{ave}, T はそれぞれグラフの平均次数とトライアングルの数を示す．

比較手法: 本稿では 3.3 節に提案した 3 つの提案手法を以下の最新のグラフ要約手法と比較する．

表 3 実験に用いるデータセットの概要

	$\|V\|$	$\|E\|$	$\|d_{ave}\|$	$\|T\|$	Type
DBLP	317K	1.05M	8.7	2.22M	co-authorship
Amazon	403K	2.44M	6.1	667K	co-parchased
YouTube	1.13M	2.99M	2.6	3.06M	social
LiveJournal	4.00M	34.7M	8.7	177M	social

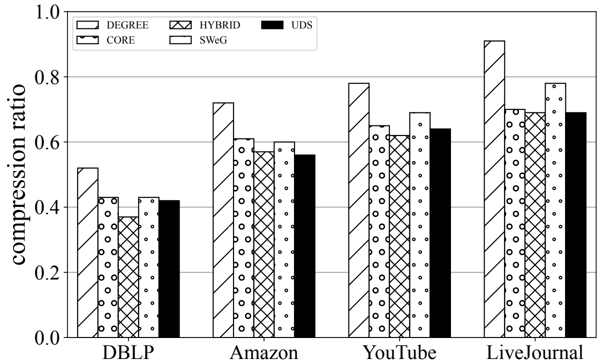

図 3 圧縮率の実験結果．縦軸は $|P|/|E|$ であり，小さいほど良い．

- **SWeG [12]**：補正セット法における最新手法である．情報の欠損は無いが分散環境を前提に設計されており，単一のマシンでは圧縮率が低い．
- **UDS [11]**：効用制限法における最新手法である．圧縮率は高いが情報の欠損が有る．本稿では論文中の設定の内最も情報の欠損が少ないものとして，Utility の閾値を 0.9 とした．

実験環境: 全ての実験は Xeon(R) Gold 6246R (3.4GHz) 350GB RAM を搭載した Linux サーバにて行い，C++ (gcc 8.2.0) による実装と，コンパイルオプション -O3 を用いて実行した．

4.2 圧縮率の評価

図 3 に圧縮率の実験結果を示す．図 3 より，HYBRID は既存手法より良い圧縮率であると言える．最も大きく差が開いたデータは DBLP (5 ポイント) であり，提案手法は三角形の多いデータと相性が良いことがわかる．また，他のデータセットにおいても圧縮率の高い既存手法である UDS と同程度の圧縮率を示しており，ウェッジに着眼することで効果的にグラフサイズを削減できていることが確認できる．1.1 節で述べたとおり，UDS は情報欠損が生じる手法である．これに対して，提案手法は BLEW の導入により情報欠損を生じさせずに UDS と同程度の圧縮率を達成しているという点からも高い優位性がある．

4.3 速度の評価

図 4 に速度の実験結果を示す．図 4 より，HYBRID は既存手法と比較して高速であると言える．最も大きく差が開いたデータは LiveJournal であり，約 2.5 倍の高速化を実現した．また CORE は 3 つの提案手法の中で最も高速である．これは表 2 にて示した理論的な計算量と同様の振る舞いであり，事前に *k*-core を列挙した後にウェッジを検出する方策が高速性の観点から優れていることを示す．ところで，YouTube のデータセットは 3 つ提案手法間で実行時間の差が非常に小さい．これは YouTube のデータセットは密度が非常に小さい，すなわちグラフ全体が疎であることにより，ウェッジの構築順序による影響が小さくなったことに起因すると考えられる．

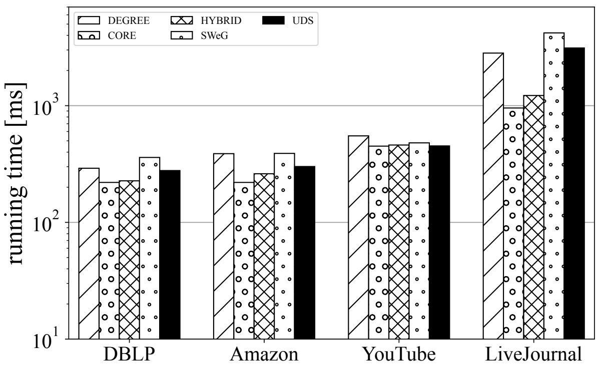

図 4　実行時間の実験結果.

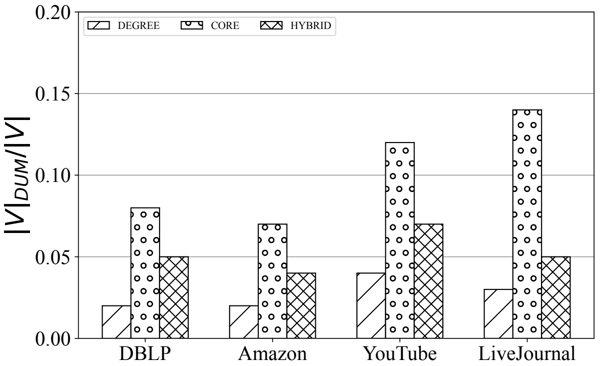

図 5　ダミーノードの割合.

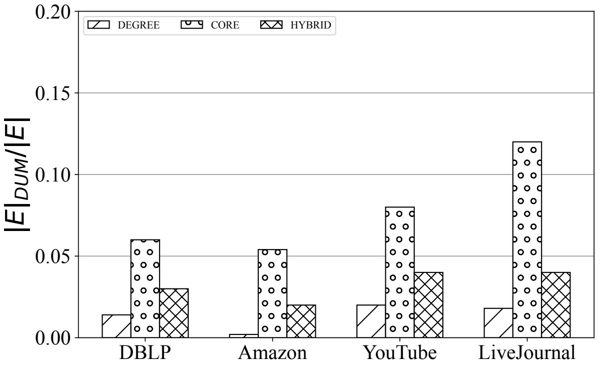

図 6　ダミーエッジの割合.

4.4　ダミーノード・ダミーエッジの割合

図 5，図 6 にダミーノードとダミーエッジの生成についての実験結果を示す．図 5，図 6 より，DEGREE が最もダミーノードの生成数が最も少ない．またダミーエッジはダミーノードと通常のノードの間に張られるエッジであるため，同様に生成が抑えられている．その一方で CORE のダミーエッジの生成割合は最大で 12%であり，これによる総エッジ数の増加が CORE の圧縮率を低減させていると言える．CORE は次数の大きなノードから優先的にウェッジを構築するために，次数の少ないノードが余ってしまうため，ダミーノードを多量に生成してしまう．これらの手法を統合した HYBRID は DEGREE と CORE のウェッジ分割を交互に実行する．これにより，DEGREE の得意とするダミーノードの低減を行い，最大で 4%程度のダミーエッジに抑えている．これは CORE の得意とする密度の高いウェッジの分割による恩恵と比較すると，非常に小さな損害である．

5　関 連 研 究

グラフ要約: グラフ要約はこれまで静的グラフ [12, 13, 21–25] や動的グラフ [26–29] に対して幅広く研究されてきた．これらのグラフ要約手法は大別して，可逆圧縮であり情報の欠損が無い手法 (Lossless), および不可逆圧縮であり情報の欠損が発生する手法 (Lossy) の二つがあり，本研究は Lossless な手法である．Lossless な手法の代表的な既存手法として，SAGS [25], SWeG [12] がある．SAGS [25] は，補正セット法に基づき Locality sensitive hashing と呼ばれるハッシュアルゴリズムを利用して効率よくマージ可能なスーパーノードを選択する．これにより，従来計算コストが大きかった Savings 関数と呼ばれるどのノードを優先的にマージすべきかを示す関数の計算を高速化した．しかしながら，SAGS はグラフの圧縮率が低いため，実用的な性能では無い．SWeG [12] は，補正セット法による要約のアイデアを分散コンピューティングにより並列化した．これにより，従来より飛躍的に高速なグラフの要約が可能になったが，併せて提案されたシングルスレッドマシンでの速度は未だ十分でなく，実用的でない．また SWeG は併せて Lossy な手法も提案しているが，既に提案されていた UDS [11] より圧縮率が低く，本稿では比較対象としなかった．Lossy な手法の代表的な既存研究として，UDS [11] がある．UDS は効用制限法に基づき，各ノードに重要度を定義し，要約グラフに保持される重要度が閾値を下回らないよう情報を削りながら要約する．UDS はグラフの重要な部分が残るように要約する点，また閾値の調整により圧縮率と情報の精度のトレードオフを制御できる点で優れている．しかしながら，閾値を情報の欠損が小さくなるように近づける程，他の Lossless な研究と比較して圧縮性能が低くなるため，情報の精度が求められる場面では有効ではない．これまでに提案されたグラフ要約技術についての詳細は，Liu らによる総説論文 [30] が詳しい．

集約を用いるその他のグラフアルゴリズム: ノードの集約を行うグラフ要約に類似する技術分野に，グラフクラスタリングがある．グラフ要約がグラフ構造を効率的に保持し，空間消費量の削減や要約グラフ中のノードへの参照の効率化を目的とするのに対して，グラフクラスタリングはノードを何らかの指標によって意味のあるまとまりとして出力することで，直接的に知識を抽出することに主眼を置く．3. 3 節で用いた *k*-core もその一つの指標であり，その他にも様々な指標 [31, 32] とそれを用いたクラスタリング手法 [33, 34] が提案されている．

6　ま　と　め

本稿では，ウェッジに基づくグラフ要約手法について提案した．我々は新たなグラフ要約アプローチであるウェッジ分割法を定義し，ウェッジ分割に基づく要約アルゴリズムを設計した．さらには，効率的に元のグラフの完全な情報を格納する BLEW を導入した．実験より，提案手法は既存手法と比較して圧縮率・速度ともに優れており，圧縮率は最大 5 ポイント，速度は最大 2.5 倍向上した．今後の課題として，*k*-core 以外の指標による分割や，要約グラフ上で動作する高速なアプリケーションの実装などが挙げられる．

謝　　辞

本研究の一部はJST さきがけ（JPMJPR2033），JST 次世代研究者挑戦的研究プログラム (JPMJSP2124) ならびに科研費 若手研究 (18K18057) の助成を受けたものである．

文　　献

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

[2] Ryan Rossi and Nesreen Ahmed. The Network Data Repository with Interactive Graph Analytics and Visualization. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[3] Hiroaki Shiokawa and Makoto Onizuka. Scalable Graph Clustering and Its Applications. In Reda Alhajj and Jon Rokne, editors, *Encyclopedia of Social Network Analysis and Mining*, pages 1–10. Springer New York, 2017.

[4] Etsuji Tomita and Tomokazu Seki. An Efficient Branch-and-bound Algorithm for Finding a Maximum Clique. In *International Conference on Discrete Mathematics and Theoretical Computer Science*, pages 278–289. Springer, 2003.

[5] X. Huang and L. Lakshmanan. Attribute-Driven Community Search. *PVLDB*, 10(9):949–960, 2017.

[6] Y. Zhou, H. Cheng, and J. X. Yu. Graph Clustering Based on Structural/Attribute Similarities. *PVLDB*, 2(1):718–729, August 2009.

[7] Jinkun Lin, Shaowei Cai, Chuan Luo, and Kaile Su. A Reduction Based Method for Coloring Very Large Graphs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 517–523. AAAI Press, 2017.

[8] Yi Zhou, Shan Hu, Mingyu Xiao, and Zhang-Hua Fu. Improving Maximum k-Plex Solver via Second-Order Reduction and Graph Color Bounding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12453–12460, 2021.

[9] Jian Gao, Jiejiang Chen, Minghao Yin, Rong Chen, and Yiyuan Wang. An Exact Algorithm for Maximum k-Plexes in Massive Graphs. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 1449–1455. International Joint Conferences on Artificial Intelligence Organization, 7 2018.

[10] Matthew C Schmidt, Nagiza F Samatova, Kevin Thomas, and Byung-Hoon Park. A Scalable, Parallel Algorithm for Maximal Clique Enumeration. *Journal of Parallel and Distributed Computing*, 69(4):417–428, 2009.

[11] K Ashwin Kumar and Petros Efstathopoulos. Utility-driven Graph summarization. *Proceedings of the VLDB Endowment*, 12(4):335–347, 2018.

[12] Kijung Shin, Amol Ghoting, Myunghwan Kim, and Hema Raghavan. SWeG: Lossless and Lossy Summarization of Web-Scale Graphs. In *The World Wide Web Conference*, WWW '19, page 1679–1690. Association for Computing Machinery, 2019.

[13] Antonio Maccioni and Daniel J. Abadi. Scalable Pattern Matching over Compressed Graphs via Dedensification. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1755–1764, New York, NY, USA, 2016. Association for Computing Machinery.

[14] Wenfei Fan, Jianzhong Li, Xin Wang, and Yinghui Wu. Query Preserving Graph Compression. In *Proceedings of the 2012 ACM SIGMOD International Conference on Management of Data*, SIGMOD '12, page 157–168, New York, NY, USA, 2012. Association for Computing Machinery.

[15] Hossein Maserrat and Jian Pei. Neighbor Query Friendly Compression of Social Networks. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 533–542, New York, NY, USA, 2010. Association for Computing Machinery.

[16] Neil Shah, Danai Koutra, Lisa Jin, Tianmin Zou, Brian Gallagher, and Christos Faloutsos. On Summarizing Large-Scale Dynamic Graphs. *IEEE Data Eng. Bull.*, 40:75–88, 2017.

[17] Zeqian Shen, Kwan-Liu Ma, and T. Eliassi-Rad. Visual Analysis of Large Heterogeneous Social Networks by Semantic and Structural Abstraction. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1427–1439, 2006.

[18] Yuanyuan Tian, Richard A. Hankins, and Jignesh M. Patel. Efficient Aggregation for Graph Summarization. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, page 567–580. Association for Computing Machinery, 2008.

[19] Vladimir Batagelj and Matjaz Zaversnik. An O(m) Algorithm for Cores Decomposition of Networks. *CoRR*, cs.DS/0310049, 2003.

[20] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On Power-Law Relationships of the Internet Topology. *SIGCOMM Comput. Commun. Rev.*, 29(4):251–262, 08 1999.

[21] Cody Dunne and Ben Shneiderman. Motif Simplification: Improving Network Visualization Readability with Fan, Connector, and Clique Glyphs. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 3247–3256, New York, NY, USA, 2013. Association for Computing Machinery.

[22] Danai Koutra, U Kang, Jilles Vreeken, and Christos Faloutsos. VOG: Summarizing and Understanding Large Graphs. In *Proceedings of the 2014 SIAM International Conference on Data Mining (SDM)*, pages 91–99, 2014.

[23] Cheng-Te Li and Shou-De Lin. Egocentric Information Abstraction for Heterogeneous Social Networks. In *2009 International Conference on Advances in Social Network Analysis and Mining*, pages 255–260. IEEE, 2009.

[24] Saket Navlakha, Rajeev Rastogi, and Nisheeth Shrivastava. Graph Summarization with Bounded Error. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pages 419–432, 2008.

[25] Kifayat Ullah Khan, Waqas Nawaz, and Young-Koo Lee. Set-based Approximate Approach for Lossless Graph Summarization. *Computing*, 97(12):1185–1207, 2015.

[26] Bijaya Adhikari, Yao Zhang, Aditya Bharadwaj, and B. Aditya Prakash. Condensing Temporal Networks using Propagation. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM)*, pages 417–425, 2017.

[27] Yu-Ru Lin, Hari Sundaram, and Aisling Kelliher. Summarization of Social Activity over Time: People, Actions and Concepts in Dynamic Networks. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, page 1379–1380, New York, NY, USA, 2008. Association for Computing Machinery.

[28] Sriram Raghavan and Hector Garcia-Molina. Representing Web Graphs. In *Proceedings 19th International Conference on Data Engineering (Cat. No. 03CH37405)*, pages 405–416. IEEE, 2003.

[29] Qi Song, Yinghui Wu, Peng Lin, Luna Xin Dong, and Hui Sun. Mining Summaries for Knowledge Graph Search. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1887–1900, 2018.

[30] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph Summarization Methods and Applications: A Survey. *ACM Comput. Surv.*, 51(3), jun 2018.

[31] Jonathan Cohen. Trusses: Cohesive Subgraphs for Social Network Analysis. *National Security Agency Technical Report*, 16(3.1), 2008.

[32] Mark EJ Newman. Fast Algorithm for Detecting Community Structure in Networks. *Physical Review E*, 69(6):066133, 2004.

[33] Ian McCulloh and Onur Savas. k-Truss Network Community Detection. In *2020 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 590–593, 2020.

[34] 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)*, pages 4597–4604, 2019.