コミュニティのハブ占有度とクラスタ係数を制御可能なグラフ生成機構

山口　寛人† 　小川　裕也† 　山崎　翔平† 　佐々木　勇和† 　鬼塚　真†

† 大阪大学情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5

E-mail: †{yamaguchi.hiroto,ogawa.yuya,yamasaki.shohei,sasaki,onizuka}@ist.osaka-u.ac.jp

あらまし　属性付きグラフのクラスタリングやノード分類に関する研究は広く行われている．これらの手法の評価にはコミュニティの正解ラベルが付与された属性付きグラフが必要である．しかし，コミュニティの正解ラベルが付与された属性付きグラフで公開されているものは数少ない．そのため，コミュニティの正解ラベルが付与された属性付きグラフ生成機構の研究が行われている．既存のグラフ生成機構では，acMark のようにノード次数やコミュニティサイズ，属性値について様々な分布を指定できるものが存在するが，コミュニティ内構造を制御する機能を有していない．本稿ではコミュニティ内構造を制御可能な属性付きグラフ生成機構を提案する．acMark を拡張し，コミュニティの最大ハブ占有度とクラスタ係数を制御することでコミュニティ内構造の制御を可能にする．実験では，コミュニティ内構造が制御できることを示し，実グラフにコミュニティ内構造を再現できることを示した．さらに，実行時間が acMark に比べて最大 1/1000 以下に高速化されていることを示した．

キーワード　グラフ生成, コミュニティ, 属性付きグラフ

1 はじめに

グラフはノード間の関係を示すデータ構造であり，幅広い領域で用いられている．例えば，ソーシャルネットワーク，タンパク質の結合，交通計画，コンピュータビジョン，遺伝子表現などが挙げられる．このようなグラフにおけるクラスタリングやノード分類，リンク予測などの解析は有用であり，研究者たちの注目を集めている．その中でもクラスタリングやノード分類はグラフ構造の理解や様々なアプリケーションへの応用に有用であり，ビッグデータ解析や人工知能など幅広い分野で利用されている．また，多くの場合，実世界のデータはノードが属性を有している．そのため，属性付きグラフを対象としたクラスタリング手法やノード分類手法が多く提案されている [1,2]．このような手法の研究の中で，研究者たちは自身の手法を評価するために多種多様なコミュニティの正解ラベルが付与された属性付きグラフを必要としている．しかし，現在コミュニティの正解ラベルが付与された属性付きグラフで公開されているものは数少なく，研究者たちが自身の手法の評価に必要なデータセットを十分に収集することが困難である．実際にグラフデータのアーカイブとして広く用いられている SNAP [3] では，コミュニティの正解ラベルが付与された属性付きグラフは公開されていない．そのため，実グラフの特徴を持ち，グラフサイズやグラフの特性を設定できるグラフ生成機構の需要は高い．

実グラフの特徴に関して，多くの場合，ノード次数とコミュニティサイズの分布がべき乗分布に従い [4]，属性値は正規分布やべき乗分布に従うことが知られている [5]．また，実グラフは様々なコミュニティ内の構造を有しており，これらはクラスタ係数（ Clustering coefficient ）と最大ハブ占有度（ Hub dominance ）の 2 つの指標から特徴付けることができる [6]．

既存の属性付きグラフ生成機構には，ノード次数やコミュニティサイズ，属性の値に対して分布を指定することで実グラフの特徴を有するグラフを生成する手法が存在する [7,8]．特に，acMark [8] はノード次数やコミュニティサイズ，属性の値の全てに対して，様々な分布を指定することが可能である．しかし，acMark には以下に示す 2 つの問題が存在する．1 つ目はコミュニティ内の構造を制御する機能を有していないということである．エッジ数やコミュニティサイズによりクラスタ係数と最大ハブ占有度を制御することが可能であるが，クラスタ係数と最大ハブ占有度が相互に影響してしまうため，これらを独立に制御することが困難である．2 つ目の問題は実行性能に関する問題である．acMark では，あるノード（ソースノード）からエッジを生成する対象となるノード（ターゲットノード）をランダムに選択し，そのターゲットノードとエッジ生成判定を行った上でエッジを生成する．しかし，ランダムに抽出されたノードがソースノードと異なるコミュニティに属する場合にはエッジ生成が失敗する確率が高い．さらに，コミュニティ数が増加するにつれてランダムに抽出されたノードがソースノードと異なるコミュニティに属する可能性が高くなるため，エッジ生成に失敗する確率が高くなる．したがって，指定された次数分布を満たすために長い実行時間が必要となる．

本稿では，acMark の拡張し，コミュニティ内の構造を制御する属性付きグラフ生成機構 CS-acMark を提案する．CS-acMark では，acMark のエッジ生成方法を変更・改善することでコミュニティ内の構造の制御を可能にするとともに高速化を実現する．CS-acMark では，クラスタ係数と最大ハブ占有度を独立に制御することで様々なコミュニティ内の構造を生成することが可能である．クラスタ係数と最大ハブ占有度を独立に制御するために，次数の低いノード群からターゲットノードを選択するというターゲットノードの制限とこの制限を受けないノードであるハブを導入する．これにより，最大ハブ占有度に影響を与えることなく，コミュニティのクラスタ係数を制御

することが可能となる．

実験では，まず CS-acMark がコミュニティ内の構造を制御できることを示した．次に，実グラフから得られるノード数とエッジ数，およびクラスタ数を用いて生成したグラフが実グラフのコミュニティ内の構造を再現できていることを示した．最後に，CS-acMark が acMark に比べて実行時間が最大 1/1000 以下に高速化されていることを示した．

本稿の構成は以下の通りである．2 章で事前準備，5 章で関連研究について説明する．3 章では，提案手法について述べ，4 章で実験結果を示し，考察を行う．6 章で結論を述べる．

2 事前準備

属性付きグラフは, ノード集合 $V = \{v_1, v_2, ..., v_n\}$，エッジ集合 $E = \{v_i, v_j\} \subseteq [V] \times [V]$, ノードの属性行列 $A \in \mathbb{R}^{n \times d}$ からなり，$G = (V, E, A)$ と表すことができる．また属性グラフにおける各コミュニティを C と表し，V, E, A のサブセットにより構成されるものとする．本章ではグラフのコミュニティ内の構造の特徴を示す指標であるクラスタ係数と最大ハブ占有度を紹介する．またこの 2 つの指標を用いたコミュニティ内の構造の分類について説明する．最後に本稿で拡張を行う acMark について説明する．

2.1 クラスタ係数（ Clustering coefficient ）

クラスタ係数は, あるノード v_i の隣接ノード同士が隣接している割合を示す指標であり，グラフの推移性を示す指標である [9]．ノード v_i の次数を k_i, ノード v_i を頂点とする三角形が Δ_i 個存在するとするとクラスタ係数は以下の式で表される．

$$\mathrm{CCF}(v_i) = \frac{2\Delta_i}{k_i(k_i - 1)} \tag{1}$$

同じコミュニティ C 内のノードのクラスタ係数の平均をコミュニティのクラスタ係数と定義し，以下の式で求められるものとする．

$$\mathrm{CCF}(C) = \frac{1}{n} \sum_{v_i \in C} \mathrm{CCF}(v_i) \tag{2}$$

このコミュニティのクラスタ係数は値が大きいほど，コミュニティ内の三角形構造の密度が高いことを示している．本稿におけるクラスタ係数および CCF はこのコミュニティのクラスタ係数を表すものとする．

2.2 最大ハブ占有度（ Hub dominance ）

最大ハブ占有度はコミュニティ C 内の内部エッジのみを考えた場合に最大次数を持つノードとエッジを持っているコミュニティ内のノードの割合を示す指標であり，ハブがコミュニティに与えている影響度を示す指標である [10]．コミュニティ C のノード数を n_C，コミュニティ C 内のノード v_i が持つコミュニティ内のエッジ数を $d_{int}(v_i)$ とすると，最大ハブ占有度は以下の式で表せる．

$$\mathrm{Hub\_dom}(C) = \frac{\max_{v_i \in C} d_{int}(v_i)}{n_C - 1} \tag{3}$$

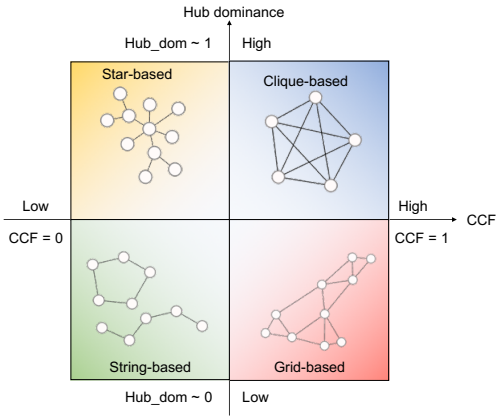


図 1: 2 つの構造的指標に基づいたコミュニティ内の構造の分類 [6]

2.3 コミュニティ内の構造の分類

コミュニティ内の構造はクラスタ係数と最大ハブ占有度から 4 種類に分類することができる [6]．この分類を 図 1 に示す．コミュニティ内の構造は，Star-based，Clique-based，String-based，Grid-based の 4 種類に分類できる．クラスタ係数の値が大きいコミュニティは，Clique-based や Grid-based のようなコミュニティ内に三角形の構造を多く持つコミュニティとなる．また，最大ハブ占有度の値が大きいコミュニティは Star-based や Clique-based のような，コミュニティ内で最大次数を持つノードがコミュニティ内の大部分のノードとエッジを持つ構造を有するコミュニティとなる．このように，コミュニティ内の構造はクラスタ係数と最大ハブ占有度から特徴付けることができ，この 2 つの指標を制御することにより，コミュニティ内の構造を制御することができる．ただし，実グラフにおいて Grid-based な構造を持つコミュニティは数少ないことが確認されている [6]．

2.4 acMark

Maekawa らによって提案された acMark [8] は，ノード次数やコミュニティサイズ，属性値の分布を任意に指定してグラフを生成することができる属性付きグラフ汎用生成機構である．コミュニティ割当とエッジ生成，および属性生成の 3 ステップでグラフが生成される．属性の値は，同じコミュニティ内のノード同士が近い属性値を持つように生成される．また，属性ごとに分布を指定できるため多種多様なグラフ生成が可能である．しかし，acMark には以下に示す 2 つの問題が存在する．1 つ目はコミュニティ内の構造を制御する機能は有していないということである．グラフ全体のエッジ数やコミュニティサイズによりクラスタ係数と最大ハブ占有度を制御することができるが，クラスタ係数と最大ハブ占有度が相互に影響してしまうため，これらを独立に制御することが困難である．2 つ目の問題は実行性能に関する問題である．acMark では，ソースノードからエッジを生成する対象となるターゲットノードをランダムに選択し，そのターゲットノードとエッジ生成判定を行った上でエッジを生成する．しかし，ランダムに選択されたノードがソースノードと異なるコミュニティに属する場合にはエッジ生

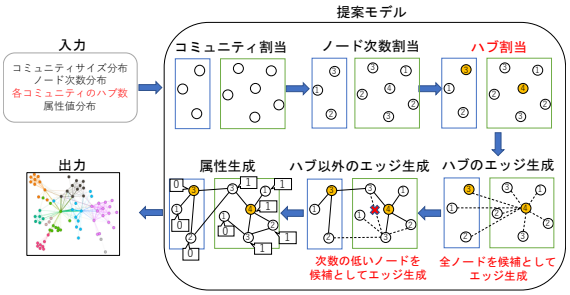


図 2: CS-acMark の概要図.

変数	説明
$\boldsymbol{S} \in \mathbb{R}^{n \times n}$	隣接行列
$\boldsymbol{X} \in \mathbb{R}^{n \times d}$	属性行列
$\boldsymbol{U} \in \mathbb{R}^{n \times k_1}$	構造のためのクラスタ割合
$\boldsymbol{V} \in \mathbb{R}^{d \times k_2}$	属性のためのクラスタ割合
$\boldsymbol{H} \in \mathbb{R}^{k_1 \times k_2}$	クラスタ転移行列
$\boldsymbol{\theta} \in \mathbb{R}^{n}$	ノード次数リスト
$\boldsymbol{\chi} \in \mathbb{R}^{k_1}$	コミュニティサイズリスト
$\boldsymbol{C} \in \mathbb{N}^{n}$	コミュニティ割当リスト
$\boldsymbol{M} \in \mathbb{N}^{*}$	エッジ生成のための候補ノードリスト
$\boldsymbol{L} \in \mathbb{N}^{n}$	コミュニティのハブ数リスト

表 1: 変数の定義

成が失敗する確率が高い．さらに，コミュニティ数が増加するにつれてランダムに選択されたノードがソースノードと異なるコミュニティに属する可能性が高くなるため，エッジ生成に失敗する確率が高くなる．したがって，指定された次数分布を満たすために長い実行時間が必要となる．

3 提 案 手 法

我々は acMark を拡張し，コミュニティ内の構造を制御可能な属性付きグラフ生成機構 CS-acMark を提案する．CS-acMark では acMark の機能に加えて，コミュニティのクラスタ係数と最大ハブ占有度を独立に制御するために，次数の低いノード群からエッジの接続先となるターゲットノードを選択するというターゲットノードの制限とこの制限を受けないノードであるハブを導入する．ターゲットノードの制限を導入することにより，クラスタ係数は各コミュニティのハブ数によって制御することが可能となり，最大ハブ占有度はエッジ数やコミュニティサイズによって制御することが可能となる．CS-acMark の概要図を図 2 に示す．CS-acMark では，はじめにノードにコミュニティ割当を行う．次に，エッジ生成を行う．エッジ生成ステップはノード次数割当，ハブ割当，ハブのエッジ生成，ハブ以外のノードのエッジ生成の順に行われる．最後に属性を生成し，グラフを出力する．コミュニティ割当とノード次数割当，属性生成は acMark と同様の方法で行われ，入力の分布に従ったものが生成される．表 1 に本章で用いる変数とその定義を示し，表 2 に提案するグラフ生成機構の入力とその説明を示す．

入力	説明
$n \in \mathbb{N}$	ノード数
$m \in \mathbb{N}$	エッジ数
$d \in \mathbb{N}$	属性数
$k_1 \in \mathbb{N}$	構造のためのクラスタ数
$k_2 \in \mathbb{N}$	属性のためのクラスタ数
$CR \in \mathbb{N}$	ターゲットノード制限率
$L \in \mathbb{N}^{n}$	各コミュニティのハブ数リスト
$\alpha \in \mathbb{R}$	クラスタ間と内でのエッジ数の割合を示すパラメータ
$\beta \in \mathbb{R}$	属性クラスタプロポーションのためのディリクレ分布のパラメータ
$\gamma \in \mathbb{R}$	クラスタ転移行列のためのディリクレ分布のパラメータ
$\phi_d, \phi_c, \phi_V, \phi_H, \in \mathbb{R}$	べき乗分布で用いるパラメータ
$\delta_d, \delta_c, \delta_V, \delta_H, \in \mathbb{R}$	一様分布の値域
$\sigma_d, \sigma_c, \sigma_V, \sigma_H, \in \mathbb{R}$	一様分布の値域
$att_{ber} \in \mathbb{R}$	離散値をとる属性の割合
$att_{pow} \in \mathbb{R}$	べき乗分布に従う属性の割合
$att_{uni} \in \mathbb{R}$	一様分布に従う属性の割合
$att_{nor} \in \mathbb{R}$	正規分布に従う属性の割合
ϕ_{att_min},$\phi_{att\_max} \in \mathbb{R}$	属性値のためのべき乗分布のパラメータ
$\delta_{att\_nor} \in \mathbb{R}$	属性値のための一様分布の値域
σ_{att_min},$\sigma_{att\_max} \in \mathbb{R}$	属性値のための正規分布の分散

表 2: 入力とその説明．ϕ, δ, σ の添字の意味を以下に示す．d はノード次数，C はクラスタサイズ，V は属性のためのクラスタプロポーション，H は構造と属性のクラスタ間の転移行列を示す．

3.1 クラスタ係数の制御

クラスタ係数の制御のためにターゲットノードの制限を導入する．ターゲットノードの制限とは，あるノードからエッジを生成する際に，次数の低いノードからターゲットノードの制限率（CR）で定められた範囲内のノード群をターゲットノードの候補ノード群とし，この中から抽出したノードとエッジを生成するという方法である．acMark のようにコミュニティ内のエッジがほぼランダムで生成される場合，次数の高いノード同士は確率的にエッジが生成される可能性が高く，次数の高いノード間で三角形を生成することが多い． したがって，ターゲットノードの制限により，次数の高いノードが次数の低いノードとエッジを生成することでクラスタ係数を抑制することが可能である．さらに，コミュニティごとにクラスタ係数の抑制を調節するために，各コミュニティにハブを割り当てる． 各コミュニティのハブにはターゲットノードの制限を受けずにエッジ生成を行う．これにより，各コミュニティに割り当てるハブの個数によりコミュニティのクラスタ係数を制御することが可能となる．

3.2 最大ハブ占有度の制御

CS-acMark では以下の 2 つの方法によって，最大ハブ占有度を制御する．1 つ目は，エッジ数による制御である．エッジ数を増加させるとグラフ全体の次数が増加するため，コミュニティ内の最大次数も増加し，最大ハブ占有度の値が上昇する．2

Algorithm 1 Graph generation

Require: $n, m, d, k_1, k_2, att_{bar}, att_{pow}, att_{uni}, att_{nor}$
Ensure: $\boldsymbol{S}$,$\boldsymbol{X}$,$\boldsymbol{C}$

Latent factors generation phase
$\boldsymbol{U}$ = latent_factor_generation($c, n, \alpha, \boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\sigma}$)
$d_{ber}, d_{pow}, d_{uni}, d_{nor} \Leftarrow int(d \times att_{ber}), int(d \times att_{pow}), int(d \times att_{uni}), int(d \times att_{nor}))$
$\boldsymbol{V}$ = latent_factor_generation($V, d_{ber} + d_{pow} + d_{uni} + d_{nor}, \beta, \boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\sigma}$)
$\boldsymbol{H}$ = latent_factor_generation($H, k1, \gamma, \boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\sigma}$)
# Graph generation phase
for $i = 0$ *to* n **do**
  $\boldsymbol{C}_i$ = argmax($\boldsymbol{U}_i$)
end for
$\boldsymbol{S}$ = adjacency_matrix_generation($\boldsymbol{U}$)
$\boldsymbol{X}$ = attribute_matrix_generation($\boldsymbol{U}, \boldsymbol{H}, \boldsymbol{V}, d - (d_{ber} + d_{pow} + d_{uni} + d_{nor}), \boldsymbol{\omega}$)

Algorithm 2 latent_factor_ generation($type, size, \tau, \boldsymbol{\phi}, \boldsymbol{\delta}, \boldsymbol{\sigma}$)

$\boldsymbol{\chi} = chooseFrom(power\,low(\phi_{type}), uniform(\delta_{type}), normal(\sigma_{type}))$
$\xi = Dirichlet(\alpha\boldsymbol{\chi}/\boldsymbol{\Sigma\chi})$
for $i = 1$ *to* $size$ **do**
  $\boldsymbol{W}_i = mutionomial(\xi)$
end for
return $\boldsymbol{W}$

つ目は，コミュニティサイズによる制御である．コミュニティサイズを大きく設定することで，コミュニティ内で最大次数を本ノードとエッジを持つノードの割合が減少するため，最大ハブ占有度の値を下げることができる．

3.3 CS-acMark アルゴリズム

CS-acMark の全体像を Algorithm 1 に示し，各種分布の生成方法を Algorithm 2， エッジ生成方法と属性生成方法を Algorithm 3,4 と Algorithm 5 に示す．

まず，入力の分布に従って Algorithm 1 の 2 – 7 行目で構造と属性のクラスタ割合，$\boldsymbol{U}, \boldsymbol{V}$ とクラスタ転移行列 $\boldsymbol{H}$ を生成する．この時，Algorithm 2 で $\boldsymbol{\chi}$ はユーザーの指定した分布に基づいて生成され (1 行目)，各ノードについて $\alpha\boldsymbol{\chi}$ に基づいたディリクレ分布から多項分布が導かれる (2 行目)．ここでの α はコミュニティ間のエッジの割合を調整するパラメータである．コミュニティ割当 $\boldsymbol{C}$ はクラスタ割合 $\boldsymbol{U}$ から得られる．

次に Algorithm 3，4 に示すエッジ生成方法について説明する．まず，コミュニティごとに指定された数だけ次数の大きいノードからハブに割り当てる．(Algorithm 4) その後，次数の大きいノードから順にエッジを生成する．ハブは全てのノードを候補ノード群とし，この候補ノード群から抽出したノードをターゲットノードとしてエッジを生成する．一方で， ハブ以外のノードは次数の低いノードからターゲットノードの制限率（Candidate Range）で指定された範囲内のノード群を候補ノード群とし，この候補ノード群から抽出したノードをターゲットノードとしてエッジ生成を行う．(8-12 行目) この時ソースノード i と候補ノード群のノード j のクラスタ割合の内積 $< \boldsymbol{U}_i, \boldsymbol{U}_j >$ にポアソン分布を適応し，その値にノード j の次数を乗算した値を重みとして抽出されたノードをターゲットノードとしてエッジを生成する．(14-19 行目) これは，エッジ生成に preferential attachment の考え方を導入している [11]．また例外として，ハブが 1 つでそのハブの次数が自身の属するコミュニティのノード数より大きい場合には，ハブを中心とする完全な star-based 構造 ((CCF，Hub_dom) = (0, 1)) のコミュニティとなるようにエッジを生成する．(4-7 行目) 以上のような方法を用いて割り当られた次数を満たすまでエッジを生成する．

Algorithm 3 *adjacency matrix generation*($\boldsymbol{U}$)

$\boldsymbol{\theta} = power\,low(\phi_d)$
$\boldsymbol{\theta} = descending\,order(\boldsymbol{\theta})$
hub_assignment($\boldsymbol{L}$)
for $i = 1$ *to* n **do**
  if i in hub_nodes and $\boldsymbol{L}_{\boldsymbol{C_i}} == 1$ and $\boldsymbol{\theta} > community\,size(C_i)$ **then**
    star_community(C_i)
  end if
  if $\boldsymbol{L} > 10$ or i in hub_nodes **then**
    $\boldsymbol{M} = \{i, i+1, ..., n\}$
  else
    $\boldsymbol{M} = \{int(i + (n-i) \times (1 - CR)), int(i + (n-i) \times (1 - CR)) + 1, ..., n\}$
  end if
  $counter = 0$
  while $\sum_h^n S_{hj} > \theta_j$ **do**
    $t = chooseFrom(\boldsymbol{M}, weights = \boldsymbol{\theta}(1 - exp(- < \boldsymbol{U}_i, \boldsymbol{U}_* >)))$
    $S_{it} = 1$
    $S_{ti} = S_{it}$
    $counter = counter + 1$
  end while
end for
return $\boldsymbol{S}$

Algorithm 4 *hub_assignment*($\boldsymbol{L}$)

$hub\_nodes = \{\}$
$hub\_count = [0] * k_1$
for $i = 1$ to n **do**
  if $hub\_count_{C_i} < L_{C_i}$ **then**
    $hub\_nodes = hub\_nodes \cup i$
    $hub\_count_{C_i} = hub\_count_{C_i} + 1$
  end if
end for

最後に属性生成に Algorithm 5 で示す属性生成方法について説明する．属性行列 $\boldsymbol{X}$ は Algorithm 1 で生成された行列 $\boldsymbol{U}, \boldsymbol{H}, \boldsymbol{V}$ から生成される．$\boldsymbol{H}$ は $\boldsymbol{U}$ と $\boldsymbol{V}$ の間の転移行列であり，$\boldsymbol{X}$ は $\boldsymbol{UHV}^\top$ で表される．(Algorithm 5)

Algorithm 5 attribute_matrix_generation($\boldsymbol{U}, \boldsymbol{H}, \boldsymbol{V}, d_{ran}, \boldsymbol{\omega}$

1: $\boldsymbol{X} = f_X(\boldsymbol{U}, \boldsymbol{H}, \boldsymbol{V})$(e.g. $(powerlaw, uniform, normal)(\boldsymbol{UHV}^\top)$ for numerical values, and $Bernoulli(\boldsymbol{UHV}^\top)$ for categorical values)
2: **return** $\boldsymbol{X}$

3.4 属性生成

属性生成は acMark と同様の方法を用いる．acMark では，属性の値の分布に複数の分布を指定することが可能であり，連続値と離散地の両方を属性の値に用いることが可能である．

$att_{ber}, att_{pow}, att_{uni}, att_{nor}$ によって，それぞれの分布に従う属性の数を調整することができる．離散値をとる属性数は $d \times att_{ber}$ で表され，これらの属性に対して，$\boldsymbol{X}$ の対応する列にポアソン分布を適応する．連続値をとる値についても属性数は離散値の場合と同様に $att_{pow}, att_{uni}, att_{nor}$ を用いて表され，対応する $\boldsymbol{X}$ の列に対してべき乗分布，一様分布，正規分布を適応する．また，クラスタ情報から独立した属性を生成するために $(1 - att_{ber} - att_{pow} - att_{uni} - att_{nor})$ の割合でランダムな属性が生成される．これら属性はランダムに従う分布が選択される．ϕ_{att_min},ϕ_{att_max},δ_{att},σ_{att_min},ϕ_{att_max} はランダムな属性を生成するために必要な分布のパラメータである．各属性は最小値が 0，最大値が 1 となるように正規化される．

3.5 計算量

CS-acMark の空間計算量と時間計算量について以下に示す．

3.5.1 空間計算量

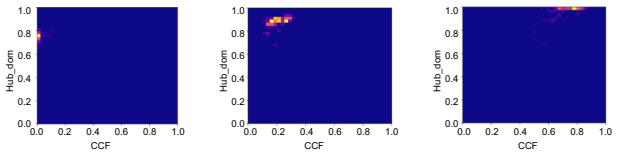
CS-acMark で用いられる変数の中で，最もサイズの大きい変数は隣接行列 $\boldsymbol{S}$ であるが，これは隣接リストを用いて表すことができるため，実際にはエッジ数だけ要素を保持すれば良い．その他の行列のサイズは表 1 に示す通りであるため，空間計算量は $O(m + nd)$ となる．

3.5.2 時間計算量

ここでは，各種分布の生成とエッジ生成，属性生成について考える．各種分布の生成では，それぞれの行列のサイズから $O_(nk_1 + dk_2 + k_1k_2)$ となる．次にエッジ生成では，まず，それぞれのノードに対してハブに割り当てられるかの判定が行われる．その後，各ノードのエッジ生成においてクラスタプロポーションからエッジ生成確率が計算され，そのノードに割り当てられた次数分だけエッジが生成される．したがって，エッジ生成の時間計算量は $O(nmk_1)$ である．最後に，属性生成では，$k = min(k_1, k_2)$ とすると，属性行列 $\boldsymbol{X}$ の計算のために必要な計算量は $O(ndk)$ である．よって，合計の計算量は $O(nmk_1 + ndk)$ である．acMark の実行時間は $O(mk_1r + ndk)$ であるが，指定された次数分布を満たすためにはエッジ生成ステップの反復回数 r を非常に大きくする必要があるため acMark と比べて実行時間は短縮される．

4 実験

本実験では，ターゲットノードの制限とハブ割当を導入することにより，クラスタ係数を制御できることを示す．また，エッジ数とコミュニティサイズによってそれぞれどのような最大ハブ占有度の制御が可能であるかを確認する．さらに，CS-acMark が実グラフのコミュニティ内構造を再現できることを示す．最後に CS-acMark の実行時間について acMark との比較を行う．コミュニティ内構造の評価には最大ハブ占有度とクラスタ係数の 2 次元ヒートマップを用いる．このヒートマップでは，点がコミュニティのクラスタ係数と最大ハブ占有度を表しており，同じ値を持つコミュニティの数が多いほど明るい点で表される．



(a) CR = 0.75　**(b)** CR = 0.9　**(c)** acMark

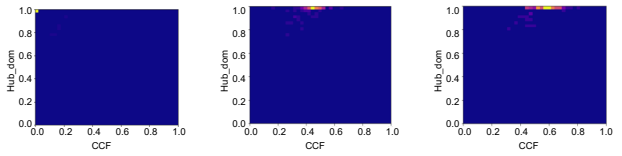
図 3: 全てのノードに対してターゲットノードの制限を適応した場合のコミュニティ内構造の特徴

4.1 ターゲットノードの制限によるクラスタ係数の抑制

ここでは，ターゲットノードの制限によってクラスタ係数が抑制されていることを確認する．本実験では，$n = 1000, m = 25000, k_1 = 5$，コミュニティサイズの分布を正規分布としてグラフを 10 回生成し，計 50 個のコミュニティについてコミュニティ内構造の特徴を示した．CR = 0.75, 0.9 として全ノードにターゲットノードの制限を適応して生成したグラフと acMark で生成したグラフのコミュニティ内構造の特徴を図 3 に示す．これらの図からエッジ生成時にターゲットノードの制限を加えることによりコミュニティのクラスタ係数が抑制されていることが確認できる．また，全てのノードに対してターゲットノードの制限を適応した場合，コミュニティ内で最大次数を持つノードにもターゲットノードの制限が適応されるため，最大ハブ占有度が平均して設定された CR の値に近づく．

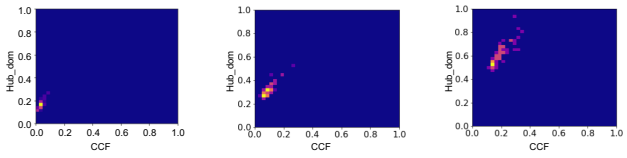
4.2 ハブ数によるクラスタ係数の制御

ここでは，各コミュニティにハブを割り当て，ハブにはターゲットノードの制限を適応せずにグラフを生成する．これにより，コミュニティのハブ数を用いてクラスタ係数を制御できることを示す．各コミュニティのハブ数を全て同じ値に設定し，その他のパラメータは 4.1 節で用いた値に加えて CR = 0.75 としてグラフを生成した．図 4 に生成したグラフのコミュニティ内構造の特徴を示す．これらの図ではコミュニティのハブ数が増加するにつれてクラスタ係数が増加している．したがって，コミュニティのハブ数により，クラスタ係数が制御できることが確認できる．また，各コミュニティに割り当てられるハブは全てのノードに対してエッジを持つことができるため，コミュニティ内で最大次数を持つノードがコミュニティ内の全てのノードとエッジを持つことが可能である．したがって，各コミュニティに 1 つ以上のハブを割り当てることにより最大ハブ占有度は CR の値に近づくということがなくなる．



(a) $L_i = 1$ (b) $L_i = 5$ (c) $L_i = 10$

図 4: 各コミュニティに一定数のハブを割り当てた場合のコミュニティ内構造の特徴



(a) $m = 5000$ (b) $m = 10000$ (c) $m = 20000$

図 5: エッジ数による最大ハブ占有度の制御

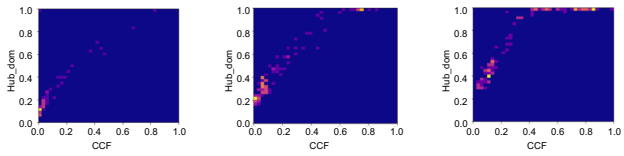
4.3 最大ハブ占有度の制御

ここでは，エッジ数とコミュニティサイズによってそれぞれどのような最大ハブ占有度の制御が可能であるかを示す．エッジ数を $m = 5000, 10000, 20000$，各コミュニティのハブ数を 5，その他のパラメータは 4.2 節に用いた値に設定して生成したグラフを図 5 に示す．これらの図に示すようにエッジ数による最大ハブ占有度の制御では，全コミュニティの最大ハブ占有度の平均を調節することができる．この図では，最大ハブ占有度に依存してクラスタ係数も増加しているが，これは全てのグラフを各コミュニティのハブ数 $L_i = 5$ で生成しているためである．コミュニティのハブ数が多くなるほど，acMark のエッジ生成方法に近づくため，最大ハブ占有度とクラスタ係数に強い依存関係が生じることになる．各コミュニティのハブ数を調節することで最大ハブ占有度のみを増加させることは可能である．

コミュニティサイズによる最大ハブ占有度の制御では，各コミュニティの最大ハブ占有度の値にばらつきを持たせることが可能となる．先ほどの実験設定からコミュニティサイズの分布をべき乗分布に変更し，グラフを生成する．生成されたグラフのコミュニティ内の構造の特徴を図 6 に示す．図 6 では，図 5 に比べてコミュニティ内の構造にばらつきが生まれている．これは，コミュニティサイズの分布をべき乗分布に設定したことでコミュニティサイズにばらつきが生じたためである．コミュニティサイズが小さいコミュニティは，コミュニティ内のエッジ密度が高くなるため，図 5 のコミュニティに比べて最大ハブ占有度とクラスタ係数が増加する．反対にコミュニティサイズが小さいコミュニティは，コミュニティ内のエッジの密度が低くなるため，最大ハブ占有度とクラスタ係数は図 5 のコミュニティに比べて減少する．

4.4 実グラフのコミュニティ内構造の再現

ここでは，実グラフから得られるノード数とエッジ数，コミュニティ数を入力として，実グラフのコミュニティ内構造が再現できることを示す．本節では，4.4.1 項で本実験で用いるデータセットの説明を行い，4.4.2 項で CS-acMark で生成したグラフが実グラフの次数分布とコミュニティ内構造の特徴を再現できていることを示す．また，コミュニティサイズが 10 以下のものは 1 つのエッジがコミュニティ内の構造に与える影響が大きいため考えないものとする．



(a) $m = 5000$ (b) $m = 10000$ (c) $m = 20000$

図 6: コミュニティサイズによるコミュニティ内構造の制御

4.4.1 データセット

本実験で用いるデータセットに関する説明を以下に示す．

データセット名	ノード数	エッジ数	コミュニティ数
email-Eu-core network	1005	16706	42
youtube	11348	81440	58

表 3: データセット

- email-Eu-core[1] network：ヨーロッパの研究機関における電子メールの送受信関係を表したグラフである．コミュニティはユーザーが所属する部署によって割り当てられる．元は有向グラフであるが，今回は無向グラフとして扱った．
- youtube：youtube[2] におけるユーザーの友好関係を表したグラフである．このデータセットはマルチラベルのデータセットであるため，louvain 法 [12] によるクラスタリング結果を用いてコミュニティ内の構造の分析を行った．本実験ではグラフからノードをサンプリングし，ノード数を 1/100 にしたグラフを再現する．表 3 にはサンプリング結果の値を記載する．

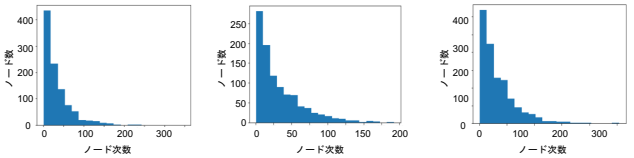
4.4.2 実グラフの再現

acMark と CS-acMark で生成したグラフと実グラフの次数分布とコミュニティ内の構造を図 7,8,9,10 に示す．

図 7,8 では，acMark と CS-acMark のどちらも次数分布とコミュニティ内の構造の両方を再現できていることが確認できる．CS-acMark のノード次数分布の生成方法は acMark と同じであり，ノード次数分布をべき乗分布に指定することでどちらの手法においても email-Eu-core network の次数分布を再現することができた．また，コミュニティ内構造に関しては email-Eu-core network が最大ハブ占有度とクラスタ係数に依存関係があるコミュニティ内構造であったため，acMark のコミュニティサイズの分布によるコミュニティ内構造の制御でもコミュニティ内の構造を再現できた．

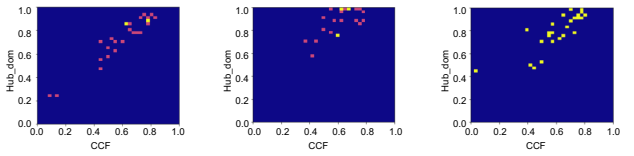
次に，図 9 に着目すると，次数分布は acMark と CS-acMark のどちらも再現できているとは言えない．acMark と CS-acMark はどちらも同じ方法でノード次数分布を生成している．ユーザーが指定した分布で 0 ～ 1 の範囲の乱数を生成し，これをノード次数としてユーザーが指定したエッジ数を満たすよう

1 : https://snap.stanford.edu/data/email-Eu-core.html
2 : https://snap.stanford.edu/data/com-Youtube.html



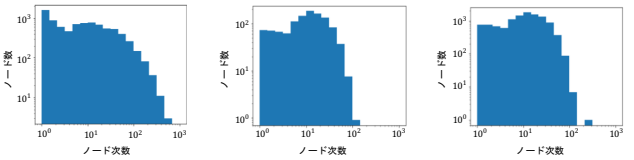
(a) email-Eu-core **(b)** acMark **(c)** CS-acMark

図 7: email-Eu-core network の次数分布の再現



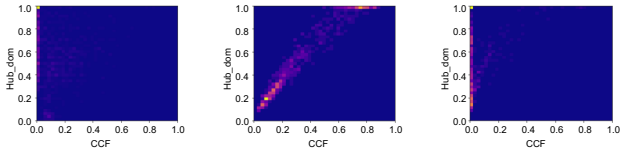
(a) email-Eu-core **(b)** acMark **(c)** CS-acMark

図 8: email-Eu-core network のコミュニティ内構造の再現



(a) youtube **(b)** acMark **(c)** CS-acMark

図 9: youtube の次数分布の再現



(a) youtube **(b)** acMark **(c)** CS-acMark

図 10: youtube のコミュニティ内構造の再現

に正規化した上でノードに割り当てる．しかし，ユーザーが指定したべき乗分布の傾きが大きい場合，1 に近い値が生成される確率は非常に小さくなるため，乱数生成において 1 に近い値が生成されず，正しい分布を再現することができない．このようなことが原因となり，youtube のグラフのような非常に傾きの大きいべき乗分布に従う次数分布は再現できなかった．一方で，図 10 から，CS-acMark は acMark に比べて，実グラフに類似したコミュニティ内の構造を再現できていることが確認できる．youtube のコミュニティは最大ハブ占有度が高く，クラスタ係数が低い構造となっている．acMark のエッジ生成方法ではこのようなコミュニティ内の構造を再現できない．一方で，最大ハブ占有度とクラスタ係数を独立に制御できる CS-acMark ではコミュニティ内の構造を再現することができている．

4.5 実行時間

ここでは，CS-acMark の実行時間について acMark と比較し，分析を行う．図 11 には，ノード数を $n = 10000$ に固定し，エッジ数を変更した場合の実行時間を示す．また，図 12 には，ノード数を $n = m/10$ とし，エッジ数を変化させた場合の実行時間を示す．さらに，図 13 には，ノード数を $n = 1000$，エッジ数を $m = 10000$ に固定し，コミュニティ数を変化させた場合の実行時間を示す．各設定で 3 回ずつ実行時間を測定し，その平均値を結果として記載する．特に記述のないパラメータに関しては以下の値を用いている: $d = 1, k = 20, k_2 = 10, \alpha = 0.1, \phi_d = 3, \phi_c = 3, CR = 0.75, L_i = 5$．なお，実装は全て python3.6 で行った．使用した PC のスペックを以下に示す．メモリ : 15.6 GB，プロセッサ : Intel® Core™ i7 – 6700 CPU @ 3.40GHz × 8.

図 11 では，全体的に CS-acMark がより短い実行時間でグラフを生成できていることが確認できる．また， エッジ数が多くなるにつれて実行時間の差が大きくなっている．これは，acMark がエッジ生成を行う際にランダムで選択したノードとエッジ生成判定を行ってエッジを生成するため，グラフの密度が高くなるにつれてランダムに選択されたノードと既にエッジを生成している確率が高くなり，エッジ生成に失敗しやすくなるためである．一方，CS-acMark はエッジ数を $m = 100000$ としても約 400$[s]$ の実行時間でグラフを生成できており，ノード数を一定にした場合には実行時間がエッジ数に対して線形に増加している．事前にソースノードとその他の各ノードとのエッジ生成確率を計算し，重複することのないように抽出されたノードと確定でエッジを生成するため，実行時間がエッジ数に対して線形に増加する．よって，事前にエッジ生成確率を生成することが実行時間の高速化に有用であることを示している．

次に，図 12 に着目すると，CS-acMark はグラフサイズによらず常に acMark より短い実行時間でグラフを生成できていることが確認できる．CS-acMark の実行時間はエッジ数に対して 2 次関数的に実行時間が増加している．本実験では $n = m/10$ としており，CS-acMark はエッジの生成確率の計算に $O(nmk_1)$ の計算量を必要とするため，グラフサイズが大きくなると m^2 に依存して実行時間が増加している．これは，大規模グラフを生成するためには非常に長い実行時間を必要とすることを示しており，今後エッジ生成確率の計算方法に改善の余地がある．

最後に，図 13 に着目すると，acMark はコミュニティ数が増加するにつれて大幅に実行時間が増加している．これは，コミュニティ数が増加すると，acMark のエッジ生成ステップにおいて，ランダムに選択したノードがソースノードと同じコミュニティに属している確率が減少し，エッジ生成に失敗する確率が増加するためである．この場合，指定された分布に従うまでに必要なエッジ生成判定の回数が増加し，実行時間が大幅に増加する．一方，CS-acMark はコミュニティ数に依存せず，ほぼ一定の実行時間を保っている．これは，エッジ生成確率を事前に計算してソースノードの次数分だけ確定的にエッジを生成するためである．これにより，コミュニティ数に依存せず，エッジ生成にかかる時間を一定に保つことができる．したがって，エッジ生成確率を事前に計算しておくことがコミュニティ数に対するスケーラビリティに貢献していることを示している．

5 関連研究

既存のグラフ生成機構では，CS-acMark と同様にノード次数

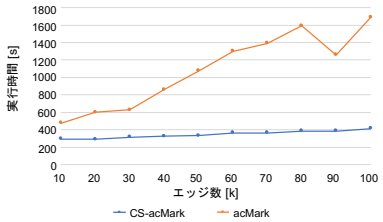


図 11: エッジ数による実行時間の変化

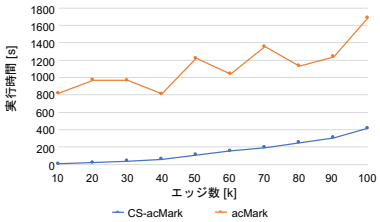


図 12: グラフサイズによる実行時間の変化

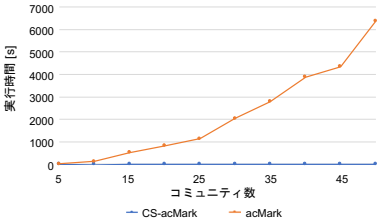


図 13: コミュニティ数による実行時間の変化

やコミュニティサイズ，属性値に対してユーザーが分布を指定可能であるものが存在する [7,8,13]．この中でも，LFR [13] によって生成された人工グラフはクラスタリング手法の評価のためのデータセットとして広く用いられている．しかし，LFR ではノードの属性を生成することができない．一方で，ANC [7] では正規分布に従うようなノードの属性値を生成することが可能である．また，ノード次数の分布はべき乗分布に従うようにエッジが生成される．しかし，ANC ではコミュニティサイズの分布を指定することができない．これらの問題を解決した手法が acMark [8] である．acMark ではノード次数や コミュニティサイズ，属性値に対して様々な分布をユーザーが指定可能である．しかし，これらの既存手法は全てコミュニティ内構造を制御する機能を有していないという問題がある．

また，既存のグラフ生成機構にはグラフ構造のみを エッジ数のオーダーで高速にグラフを生成できる手法が存在する [11,14]．Barabási-Albert モデル [11] ではエッジ数のオーダーでグラフを生成することが可能である．逐次ネットワークにノードが追加され，新しく追加されたノードは既存のノードの中から次数に比例する確率でノードを指定された数だけ選択し，エッジを生成する．このエッジ生成方法は preferential attachment と呼ばれており，この方法を用いて生成されたグラフはスケールフリー性を持つことが知られている．この手法をさらに高速化したグラフ生成機構として，ROLL [14] というグラフ生成機構がある．この手法では，Barabási-Albert モデルが大幅に高速化されており，約 10 億ノード，60 億エッジのグラフを約 1 時間で生成することが可能である．

6 終わりに

本稿ではコミュニティ内の構造を制御可能な属性付きグラフ生成機構である CS-acMark を提案した． CS-acMark は acMark の機能に加え，ハブ割当とターゲットノードの制限を導入することで，コミュニティのクラスタ係数と最大ハブ占有度を独立に制御することを可能にした．また，エッジ生成確率を事前に計算し，抽出したノードと確定的にエッジを生成することで acMark と比べて大幅な実行時間の短縮を実現した．また，今後の課題として，大規模なグラフの生成やコミュニティ間のエッジ生成の制御などが挙げられる．

文　献

[1] Zhichao Huang, Yunming Ye, Xutao Li, Feng Liu, and Huajie Chen. Joint weighted nonnegative matrix factorization for mining attributed graphs. In *Proceeding of the Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 368–380, 2017.

[2] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[3] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection at http://snap. stanford. edu/data, 2014.

[4] Mark EJ Newman. Networks: An introduction. *Oxford University Press*, 2010.

[5] MEJ Newman. Power laws, pareto distributions and zipf's law. *The Informa UK Limited*, Vol. 46, No. 5, p. 323–351, 2005.

[6] Vinh-Loc Dao, Cécile Bothorel, and Philippe Lenca. An empirical characterization of community structures in complex networks using a bivariate map of quality metrics. *arXiv preprint arXiv:1806.01386*.

[7] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R Zaïane. Generating attributed networks with communities. *The Public Library of Science*, Vol. 10, No. 4, p. e0122777, 2015.

[8] Seiji Maekawa, Jianpeng Zhang, George Fletcher, and Makoto Onizuka. General generator for attributed graphs with community structure. In *proceeding of the ECML/PKDD Graph Embedding and Mining Workshop*, pp. 1–5, 2019.

[9] A. Barrat, M. Barthélemy, R. Pastor-Satorras, and A. Vespignani. The architecture of complex weighted networks. *The National Academy of Sciences*, Vol. 101, No. 11, pp. 3747–3752, 2004.

[10] Andrea Lancichinetti, Jari Saramäki, Mikko Kivelä, and Santo Fortunato. Characterizing the community structure of complex networks. *arXiv preprint arXiv:1005.4376*.

[11] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *The American Association for the Advancement of Science*, Vol. 286, No. 5439, pp. 509–512, 1999.

[12] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *The Journal of Statistical Mechanics: Theory and Experiment*, Vol. 2008, No. 10, p. P10008, 2008.

[13] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *arXiv preprint arXiv:0805.4770v4*, Vol. 78, .

[14] Ali Hadian, Sadegh Nobari, Behrooz Minaei-Bidgoli, and Qiang Qu. Roll: Fast in-memory generation of gigantic scale-free networks. In *Proceedings of the 2016 International Conference on Management of Data*, p. 1829–1842. Association for Computing Machinery, 2016.