コミュニティ構造を制御する属性付きグラフ生成

前川　政司† 　佐々木勇和† 　George Fletcher†† 　鬼塚　真†

† 大阪大学大学院情報科学研究科　〒 565–0871 大阪府吹田市山田丘 1–5
†† Eindhoven University of Technology, 5612 AZ Eindhoven, Netherlands
E-mail: †{maekawa.seiji,sasaki,onizuka}@ist.osaka-u.ac.jp, ††g.h.l.fletcher@tue.nl

あらまし　実世界のグラフ持つ特徴を示す大規模な正解ラベル付きの人工グラフ生成は，グラフインデックスやマイニングタスクの評価に必要とされている．本稿では，コミュニティ構造を制御する属性付きグラフ生成器である acMark を提案する．acMark の利点は以下の三つである．(i) acMark はユーザが指定したグラフ特徴を持つグラフを生成する．(ii) ユーザはノードとクラスの間の接続割合を柔軟に制御することができる．(iii) グラフ生成はエッジ数に対して線形時間で実行可能である．acMark はこれらの利点を全て有する最初の生成器である．多様な実験を通して，acMark が効率的にユーザが制御可能なクラス構造を持つ属性付きグラフを生成できることを示した．

キーワード　属性付きグラフ, ベンチマーク, コミュニティ

1　はじめに

実世界のグラフデータでは多くの場合, ノードが属性を持っている（e.g., ウェブグラフのノードはテキスト情報を保持している）．属性付きグラフのための分析手法は広く研究されており，クラスタリング手法 [4], [21] やノード分類手法 [3], [12] などが挙げられる．

研究者たちは自身の設計した手法を評価するために, 大規模かつ多様なデータセットを必要としている．グラフデータのアーカイブとして広く知られている SNAP [15] において多くの実データが利用可能であるが，多くの場合コミュニティの正解（ノードのコミュニティへの割当，本稿では**ラベル**と呼ぶ）は付与されていない上に小規模なデータである．そのため，手法の評価のために十分な量のデータを集めることは困難である．

以上の議論から，実グラフが持つ多様な特性を示し，正解ラベルを持つ属性付きグラフを生成することは重要である．ウェブグラフやソーシャルグラフが持つ主要な構造の特徴として，スモールワールド性やべき乗則に従うノード次数分布がある．構造と属性の両方に関しては，属性付きグラフの基礎的な現象である **core/border** と **homophily/heterophily** に注目する．

Core/border. 最初に属性の観点から，似た属性を持つノードの集合として**クラス**を導入する．クラスの中には，core と border の二種類のノードが存在する [8]．Core ノードは所属するクラスのノードが持つ属性の平均と似た属性の値を持つノードである．一方，border ノードは複数のクラスの属性を混ぜ合わせた属性の値を持つノードである．これらの core/border の現象を捉えるために，

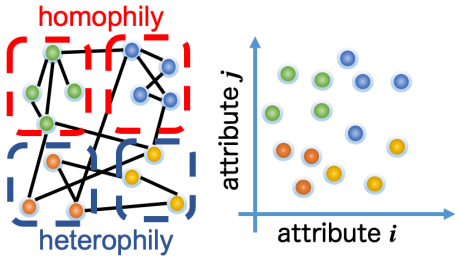


図 1: ノードの色はそのノードの所属クラスを意味する．各クラス内のノードは似た属性を持つ．赤枠，青枠はそれぞれ homophily プロパティを持つクラスに属するノードおよび heterophily プロパティを持つクラスに属するノードを示している．

各ノードが複数のクラスにいくぶんかの割合で所属する [1] と仮定し，その割合を**ノード-クラス所属プロポーション**と呼ぶ．Core ノードは所属するクラスに対して大きな値を持ち，border ノードは複数のクラスに対して大きな値を持つ．

Homophily/heterophily. 実グラフの現象として homophily はよく知られており，同じクラスに属するノードが互いに接続する可能性が高い傾向がある現象である．また，homophily がコミュニティ構造に影響を与えていることが知られている [16]．コミュニティによっては似ていない属性を持つノードを内包するものがあり，この現象は heterophily と呼ばれている．それは homophily の反対の概念であり，異なる属性を持つノードに接続する傾向があることを意味する．図 1 において homophily/heterophily の直感的な説明を示す．homophily/heterophily プロパティの両方をサポートするために，ノードとクラスの接

1：クラスラベルは各ノードが最も強く所属しているクラスを表す．

続割合を表す**ノード-クラス接続プロポーション**を導入する．それは homophily プロパティを持つクラスでは，クラス内のノードは互いに接続する傾向があるため，ノード-クラス所属プロポーションと一致し，heterophily プロパティを持つクラスでは反転した値を持つ．これらのノードレベルでのプロポーションは，core/border や homophily/heterophily などの現象を掴み，クラスレベルの接続割合よりも粒度の細かい表現を可能にする．

既存研究．グラフ生成器はノード-クラス所属プロポーションをサポートする必要があるが，多くの手法はクラス構造を考慮して設計されていない [1], [19]．コミュニティ推定のために設計された手法として LFR [13]，DC-SBM [10]，ANC [2], [14] が提案されている．しかし，これらの手法はクラス内のノードに対して全てが同じノード-クラス所属プロポーションを持つと仮定しているため，クラス内のノードの多様性を考慮することができない．近年，ディープラーニングベースの手法である GAE と GraphVAE [11], [20] や，GAN [9] ベース手法である NetGAN [5] が提案されている．これらの手法は入力グラフからのグラフ再生成を目的とするため，ユーザが柔軟にクラス構造を制御することができない．以上から，ユーザが制御可能なノード-クラス所属プロポーションをサポートした既存手法は存在しない．

取り組む課題．既存手法ではサポートできないノードレベルの制御を可能にするために，私たちは新しいグラフ生成器，acMark を提案する．ノード-クラス所属プロポーションをサポートしたグラフ生成器の実現には技術的課題が二つある：1）生成されたグラフの質，2）グラフ生成の効率．さらに，これら二つの課題は相互に依存関係があるため，グラフ生成器はその関係をバランスするように設計されなければならない．

貢献点．acMark は上記の二つの課題を，属性付きグラフの多様な特性による制約を表現するための潜在変数（ノード-クラス所属/接続プロポーション）を導入することと，ヒューリスティックな効率的グラフ生成アプローチを取ることで解決する．ノード-クラス所属/接続プロポーションは潜在変数で表すことができ，それらの潜在変数を用いて表現された制約を解くことによって，ユーザが指定したクラス構造に従うエッジを生成可能にする．効率性については，満たすべき制約の優先付けおよびサンプリング手法の活用によって，エッジ数に対して線形で動作することを可能にした．これらによって，acMark は高精度かつ効率的なグラフ生成を実現する．

2 事前準備

クラスラベルを持つ属性付きグラフは $G = (\boldsymbol{S}, \boldsymbol{X}, \boldsymbol{C})$ と表すことができ，隣接行列 $\boldsymbol{S} \in \{0,1\}^{n\times n}$，属性行列 $\boldsymbol{X} \in \mathbb{R}^{n\times d}$，クラスラベル $\boldsymbol{C} \in \{1,\ldots,k\}^n$ から構成される．ここでの n, d, k はそれぞれノード数，属性数，クラス数を表す．

この章では，グラフ生成器がサポートすべき特徴であるグラフ特徴とクラス特徴について述べる．その後，取り組む問題の厳密な定義を与え，最後にグラフ生成のために解く必要がある技術的な課題について説明する．

2.1 グラフ特徴

グラフ生成器がサポートするべき実グラフが持つ典型的な二つの特徴である構造特徴と属性特徴を導入する．

構造特徴．ソーシャルグラフが持つ広く知られた構造特徴として，ノード次数分布がべき乗分布従うことが挙げられる [17]．そのため，グラフ生成器はノード次数において多様な分布を選択可能であるべきである．

属性特徴．属性の値には離散値と連続値の二つのタイプが考えられる．一般に連続値では，正規分布とべき乗分布に従うことが知られている．このことから，グラフ生成器は属性値の分布についても多様な分布を選択可能であるべきである．

2.2 クラス特徴

1 章で示したようにグラフ生成器は core/border および homophily/heterophily の現象を表現するためにノード-クラス所属/接続プロポーションをサポートするべきである．ノード-クラス所属/接続プロポーションを捉えるために，入力パラメータとして三つの基本的な統計量であるクラス接続平均，クラス接続分散，クラスサイズ分布を導入する．これら三つの統計量は，潜在変数であるノード-クラス所属/接続プロポーションを生成するために利用される．

クラス接続平均．homophily/heterophily の現象を表現するために，クラス間の接続割合を表すクラス接続平均 $\boldsymbol{M} \in \mathbb{R}^{k\times k}$ を導入する．各要素 $\boldsymbol{M}_{l_1,l_2}$ はクラス l_1 のノードからクラス l_2 のノードへの接続割合の平均を表す．$\boldsymbol{M}$ の定式化を以下に示す：

$$\boldsymbol{M}_{l_1 l_2} = \frac{1}{|\Omega_{l_1}|} \sum_{i\in\Omega_{l_1}} \Big(\sum_{j\in\Omega_{l_2}} \boldsymbol{S}_{ij} \Big/ \sum_{j\in N} \boldsymbol{S}_{ij} \Big). \tag{1}$$

クラス接続平均は，バイナリ表現である homophily/heterophily のより一般的な表現である．対角要素はクラス内の接続割合を意味している．

クラス接続分散．core/border の現象を表現するために，クラス間の接続割合の分散を表すクラス接続分散 $\boldsymbol{D} \in \mathbb{R}^{k\times k}$ を導入する．クラス接続分散はクラス間のノード-クラス所属プロポーションの多様性を意味する．各要素 $\boldsymbol{D}_{l_1,l_2}$ はクラス l_1 のノードからクラス l_2 のノードへの接続割合の分散を表し，以下のように定式化される：

$$\boldsymbol{D}_{l_1 l_2} = \sqrt{\frac{1}{|\Omega_{l_1}|}\sum_{i\in\Omega_{l_1}}(\sum_{j\in\Omega_{l_2}}\boldsymbol{S}_{ij}/\sum_{j\in N}\boldsymbol{S}_{ij} - \boldsymbol{M}_{l_1 l_2})^2}. \tag{2}$$

クラス接続分散は，バイナリ表現である core/border のより一般的な表現である．

クラスサイズ分布． クラス接続平均および分散は詳細なクラスの特徴を捉えているが，クラスサイズの情報を含んでいないため，クラスサイズ分布を導入する．多くの実グラフでは，クラスサイズ分布はべき乗分布で近似できることが知られている [6], [18].

2.3 問題定義

本稿では，二つの実践的なシナリオを提供する．一つ目はグラフ特徴とクラス特徴を入力として，柔軟に特性が制御可能であるグラフを生成する．二つ目はグラフの隣接行列 $\boldsymbol{S}'$ とクラスラベル $\boldsymbol{C}'$ および属性特徴を入力として，所与のグラフと似た特性を持つグラフを生成する．いずれのシナリオにおいても，acMark は入力された特性に類似した特性を持つ人工グラフを効率的に生成することを目的とする．

2.4 技術的な課題

上記の問題を解くために，二つの主要な課題に取り組む．一つ目の課題は，グラフ特徴とクラス特徴からくる複数の構造的な制約を満たすようにエッジを生成しなければならないことである．これらの制約とグラフ生成問題の関係を明らかにするために，構造部分のロスを定義する：

$$L_{\text{topology}} = L_{\text{graph feature}} + L_{\text{class feature}} \tag{3}$$

ここでの $L_{\text{graph feature}}$ はグラフ特徴と生成グラフの間のロスであり，$L_{\text{class feature}}$ はクラス特徴と生成グラフの間のロスである．ノード-クラス所属/接続プロポーションを表現する潜在変数を活用することによって，$L_{\text{class feature}}$ は二つの部分に分けることができる：

$$L_{\text{class feature}} = \underbrace{L_{\text{edge precision}}}_{\text{生成グラフ-潜在変数間}} + \underbrace{L_{\text{mean}} + L_{\text{deviation}} + L_{\text{class size}}}_{\text{潜在変数-クラス特徴間}} \tag{4}$$

ここでの $L_{\text{edge precision}}$ は潜在変数から計算されるエッジの存在確率に基づいて，正確にエッジが生成されているかを表すロスであり，L_{mean} はユーザが指定したクラス接続平均と潜在変数から推定されるクラス接続平均の差を表すロスであり，$L_{\text{deviation}}$ はユーザが指定したクラス接続分散と潜在変数から推定されるクラス接続分散の差を表すロスであり，$L_{\text{class\_size}}$ はユーザが指定したクラスサイズ分布と生成されたグラフのクラスサイズ分布の差を表すロスである．

二つ目の課題は，効率的に大規模なグラフを生成することである．問題は二つあり，一つ目は所与のノード次数分布を満たすようなグラフが存在するかどうかの決定は NP-complete であることである [1]．二つ目はノード-クラス所属/接続プロポーションから全てのノードペアについてのエッジの存在確率を推定するためには $O(kn^2)$ の大きな計算量が必要であることである．一つ目の問題を解決するために，ヒューリスティックにノード次数を割り当てたのちにエッジを作成する効率的なアルゴリズムを設計する．二つ目の問題については，エッジ数に対して線形でアルゴリズムが動作するための効果的なサンプリング手法を導入する．

3 提案手法

この章では，acMark の設計について説明する．まず初めに 3.1 において潜在変数を導入し，それらによって式 (4) で示したロスを設計する．また acMark によるグラフ生成の概要も示す（一つ目のシナリオ）．次に 2.3 章で示したグラフ生成の二つ目のシナリオについて 3.2 章で説明する．最後に acMark の時間および空間計算量を 3.3 章で分析する．

3.1 生成モデル

acMark の基本的なアイデアは，潜在変数と呼ばれるデータ構造を用いてノードレベルのクラス情報を掴み，潜在変数からグラフを生成することである (図 2 を参照).

3.1.1 潜在変数

acMark は三つの潜在変数：ノード-クラス所属プロポーション $\boldsymbol{U} \in \mathbb{R}^{n\times k}$，ノード-クラス接続プロポーション $\boldsymbol{U}' \in \mathbb{R}^{n\times k}$ および，属性-クラスプロポーション $\boldsymbol{V} \in \mathbb{R}^{d\times k}$ を用いることでグラフを出力する．これらの潜在変数は実グラフのクラス特徴を捉えるための主要な構成要素である．

三つの潜在変数を以下のように定義する：

[Definition 31]（ノード-クラス所属プロポーション $\boldsymbol{U}$）$\boldsymbol{U} \in \mathbb{R}^{n\times k}$ はノードからクラスへの所属に関する射影を意味する．もし i 行目の j 列目の要素が 1 に近ければ，ノード i がクラス j に所属する傾向を意味する．

[Definition 32]（ノード-クラス接続プロポーション $\boldsymbol{U}'$）$\boldsymbol{U}' \in \mathbb{R}^{n\times k}$ はノードからクラスへの接続に関する射影を意味する．もし i 行目の j 列目の要素が 1 に近ければ，ノード i がクラス j に含まれるノードに接続する傾向を意味する．

[Definition 33]（属性-クラスプロポーション $\boldsymbol{V}$） $\boldsymbol{V} \in \mathbb{R}^{d\times k}$ は属性からクラスへの関係の強さの射影を意味す

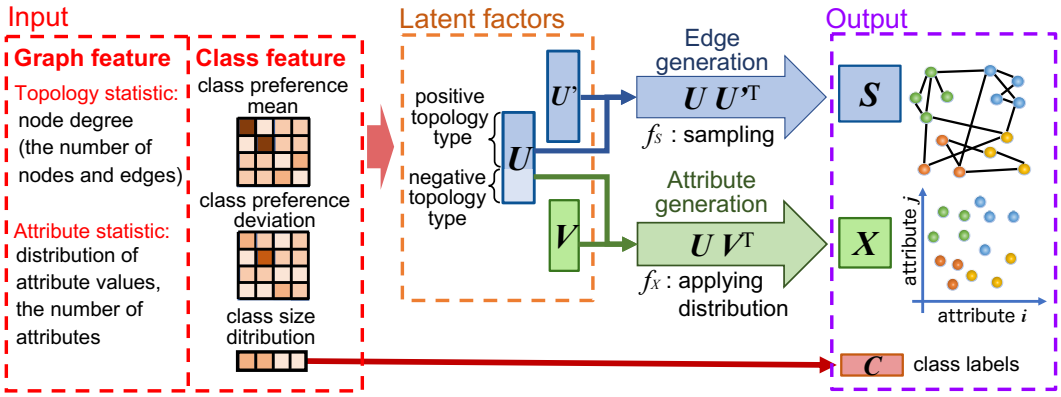


図 2: 提案手法の概要図．acMark はまず潜在変数である $\boldsymbol{U}, \boldsymbol{U}', \boldsymbol{V}$ を生成したのちに，隣接行列 $\boldsymbol{S}$ と属性行列 $\boldsymbol{X}$ を生成する．クラスラベル $\boldsymbol{C}$ はクラスサイズ分布に基づいて生成される．

る．もし i 行目の j 列目の要素が 1 に近ければ，属性 i がクラス j に対して強い相関があることを意味する．

3. 1. 2　クラス特徴に関するロス

エッジ生成にクラス特徴を組み込むために，ロス $L_{\text{class feature}}$（式 (4)）を設計する．$L_{\text{class feature}}$ は以下の二つのロスとして表現できる：1) 生成グラフ-潜在変数間ロス，2) 潜在変数-クラス特徴間ロス．

生成グラフ-潜在変数間ロス．私たちは潜在変数 $\boldsymbol{U}$，$\boldsymbol{U}'$ を用いることによってエッジを生成するように acMark を設計する．$\boldsymbol{U}$ と $\boldsymbol{U}'$ の定義より，$\boldsymbol{U}$ と $\boldsymbol{U}'^{\top}$ の乗算によってエッジの存在確率を計算することができる．その理由は，$\boldsymbol{U}\boldsymbol{U}'^{\top}$ はノードからクラスへの所属の強さ（$\boldsymbol{U}$）とクラスからノードへの接続の傾向（$\boldsymbol{U}'^{\top}$）の二つの射影の組み合わせであるためである．このエッジ確率を用いて，エッジ生成の正確性のロスを定式化する：

$$L_{\text{edge precision}} = \|\boldsymbol{S} - \boldsymbol{U}\boldsymbol{U}'^{\top}\|^2 \tag{5}$$

1 章で述べたように，homophily は似た属性を持つノードが互いに接続する現象であり，heterophily はその逆の現象である．homophily/heterophily の両方をサポートするために，$\boldsymbol{U}$ と $\boldsymbol{U}'$ を以下のように設計する．homophily プロパティのクラスでは二つの潜在変数は同じ値を持ち，heterophily プロパティのクラスでは二つの潜在変数は反転した値を持つ．正確な定義のために homophily プロパティのクラスを**ポジティブトポロジー**と呼び，ランダム接続よりクラス l 内エッジが多いとき（$\boldsymbol{M}_{ll} \geqq 1/k$），そのクラスはポジティブトポロジークラスとする．その逆の場合，すなわち heterophily プロパティのクラスを**ネガティブトポロジー**と呼ぶ．上記の議論から $\boldsymbol{U}'$ を定式化する：

$$\boldsymbol{U}'_{i.} = \begin{cases} \boldsymbol{U}_{i.} & (i \in \Omega_l \text{ and } l \in type_p) \\ inverse(\boldsymbol{U}_{i.}) & (i \in \Omega_l \text{ and } l \in type_n) \end{cases} \tag{6}$$

ここでの $type_p$，$type_n$ はそれぞれポジティブとネガティブトポロジークラスの集合である．また inverse 関数は以下のように定式化できる：

$$inverse(\boldsymbol{U}_{i.}) = \begin{cases} 1 - \boldsymbol{U}_{ih} & (h = l) \\ (1 - \boldsymbol{U}_{ih}) \frac{\boldsymbol{U}_{il}}{\sum_{j \neq l}^{k} (1 - \boldsymbol{U}_{ij})} & (h \neq l) \end{cases} \tag{7}$$

ここでのクラス l はノード i が属するクラスである．また，所属するクラス以外の要素を正規化することで，$\boldsymbol{U}'$ の各行の和を 1 にする．

潜在変数-クラス特徴間ロス．次に潜在変数が正確にクラス特徴を捉えられるように潜在変数とクラス特徴の間のロスを設計する．まず，ユーザが指定したクラス接続平均と潜在変数から推定するクラス接続平均の間の差を減らすために以下のようにロスを定式化する：

$$L_{\text{mean}} = \sum_{l}^{k} \|\boldsymbol{M}_{l.} - \underbrace{\frac{1}{|\Omega_l|} \sum_{i \in \Omega_l} \boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.}}_{\text{推定接続平均}} \|_F^2 \tag{8}$$

ここでの $*$ は要素ごとの積を意味する．また $\boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.}$ は推定値の近似計算であり，近似を用いる理由は全てのノードペアの間の確率を計算するには $O(kn^2)$ の高コストを要するからである．次にユーザが指定したクラス接続分散と潜在変数から推定するクラス接続分散の間の差を減らすために以下のようにロスを定式化する：

$$L_{\text{deviation}} = \sum_{l}^{k} \|\boldsymbol{D}_{l.} - \underbrace{\sqrt{\frac{1}{|\Omega_l|} \sum_{i \in \Omega_l} (\boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.} - \overline{\boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.}})^2}}_{\text{推定接続分散}} \|_F^2 \tag{9}$$

ここでの $\overline{\boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.}}$ は $\boldsymbol{U}_{i.} * \boldsymbol{U}'_{i.}$ の平均を表していて，ノード i は Ω_l に含まれるノードである．式 (8) と同様の理由で近似を用いて計算する．最後に推定クラスサイズと生成されたグラフのクラスサイズの間のロス $L_{\text{class size}}$ を以下のように定式化する：

$$L_{\text{class size}} = \sum_{l=1}^{k} (class\_size[l] - |\Omega_l|/n)^2 \tag{10}$$

ここでの $class\_size$ はユーザに指定されたクラスサイズ分布を示し，$|\Omega_l|/n$ はクラス l のクラスサイズ比率を表す．

3.1.3 グラフ特徴に関するロス

グラフ特徴は構造特徴と属性特徴の二つで表されるが，属性については 3.1.5 で述べるため，ここでは構造部分に注目する．グラフ生成器にはユーザが指定したノード次数をサポートすることが求められるため，生成グラフの質はユーザが指定したノード次数と生成グラフのノード次数の差として以下のように計算される：

$$L_{\text{graph feature}} = \frac{1}{n}\sum_{i=1}^{n}|\frac{\boldsymbol{\theta}_i - \boldsymbol{\theta}'_i}{\boldsymbol{\theta}_i}| \tag{11}$$

ここでの $\boldsymbol{\theta}$ は期待されるノード次数を表し，$\boldsymbol{\theta}'$ は実際に生成されたグラフのノード次数を表す．ロスの計算のために *Mean Absolute Percentage Error* (MAPE) を採用した．MAPE を利用する理由は，高次数ノードも低次数ノードも同等に扱うことができるためにである．他の指標としては mean squared error なども使用可能である．

3.1.4 エッジ生成

acMark では隣接行列 $\boldsymbol{S}$ は $\boldsymbol{U}\boldsymbol{U}'^{\top}$ を用いて生成される (式 (5) 参照)．潜在変数によってユーザが指定したクラス特徴をエッジ生成に組み込む．隣接行列の生成には以下の二つの課題がある：1) 所与のノード次数を持つグラフを存在するかを決定することは NP-complete な問題ある，2) $\boldsymbol{U}\boldsymbol{U}'^{\top}$ によるエッジ存在確率の計算コストは $O(kn^2)$ である．さらに，これらの問題はノード次数とエッジの存在を介して相互に依存している（式 (11)，(5)）．このことから，acMark はヒューリスティックにノード次数を割り当ててエッジを生成する効率的なアルゴリズムを採用する．またエッジ確率の高計算量を克服するために，逆関数法 [7] をアプローチに組み込む．

エッジ生成のアプローチ． アプローチのアイデアは二つあり，一つ目は高次数ノードから順にエッジを生成することであり，二つ目は逆関数法を活用し，エッジの存在確率の計算を高速化することである．一つ目について，高次数ノードからエッジ生成を始めることにより，エッジ生成の終盤に高次数ノードが残ることを避けることができる．これによって，高次数ノードが十分な数のエッジを持つことが可能になる．二つ目のアイデアは，$\boldsymbol{U}_{i.} * \boldsymbol{U}'_{j.}$ によって計算されるノード i とノード j の間のエッジ確率を $\boldsymbol{U}$ に基づく *Class selection from source node* ステップと $\boldsymbol{U}'^{\top}$ に基づく *Target node selection from class* ステップに分解して解釈することである [2]．行列積の計算を二つの確率的な選択ステップに変形する解釈によって，逆関数法をこのアプローチに組み込むことができる．まず，二つの選択ステップについて述べた後に，逆関数法の利用について説明する．

Class selection from source node ステップでは，各ノード i に対して $\boldsymbol{U}_{i.}$ に基づいて，ノード次数分だけクラス $Z \subset \{1,\ldots,k\}^{\boldsymbol{\theta}_i}$ を選択する．その後，*Target node selection from class* ステップにおいて，各クラス $l \in Z$ に対して $\boldsymbol{U}'^{\top}_{l}$ に基づいて，ノードを選択する．選択されたノードとノード i の間にエッジを生成する [3]．*Target node selection from class* を各ソースノードに対して実行するために，このアプローチは高コストである $O(kn^2)$ を必要としてしまう．

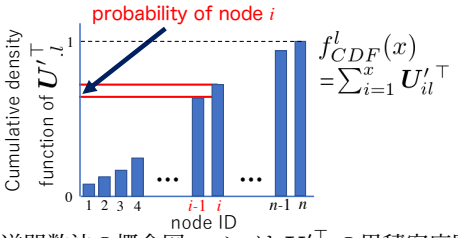


図 3: 逆関数法の概念図．バーは $\boldsymbol{U}'^{\top}_{.l}$ の累積密度関数を表し，ここでの l はクラスを表す．

逆関数法． アプローチの高速化のために単調増加な密度のための逆関数法を組み込む．これによって，*Target node selection from class* ステップの時間計算量を $O(1)$ にすることを実現する．まず各クラスごとに $\boldsymbol{U}'^{\top}$ の確率密度関数（CDF）を計算する．図 3 は $\boldsymbol{U}'$ のある列を正規化することによる確率密度関数を可視化する．横軸はノードを表し，縦軸はある列（クラス）の CDF を意味し，f_{CDF} はクラスからノードが選択される確率の CDF を表す．ノード $i-1$ からノード i までの幅はノード i を選択する確率を示し，$f_{CDF}(i) - f_{CDF}(i-1)$ で表現される．次に CDF の逆関数 $f^{-1}_{CDF}(x)$ を生成する．それは $f_{CDF}(i-1) \leqq x < f_{CDF}(i)$ であるときノード i のノード ID を返す関数である．つまり，この逆関数は 0 から 1 の乱数値を選択することで確率に基づいたノード ID を返すことを意味する．高速アクセスのために，乱数の幅を小さなステップに分割することにより，CDF の逆関数と似た結果を返すリストを生成する．このリストによって $f_{CDF}(i-1)$ と $f_{CDF}(i)$ の間の値を得たときにノード i のノード ID を得ることができ，その実行時間は $O(1)$ とすることができる．

3.1.5 属 性 生 成

属性行列 $\boldsymbol{X}$ は $\boldsymbol{U}\boldsymbol{V}^{\top}$ に基づいて生成され，この行列積はノードからクラスへの所属の強さの射影（$\boldsymbol{U}$）とクラスから属性への相関の強さの射影（$\boldsymbol{V}^{\top}$）の組合せとして考えることができる．クラス構造と属性値の分布である属性特徴の両方をサポートするために，1）同じクラスに属するノードが似た属性を共有するように，$\boldsymbol{U}, \boldsymbol{V}^{\top}$ の積によって属性の基底ベクトルを得て，2）ユーザが指定した分布（ベルヌーイ，正規，べき乗分布）に属性値が

2：無向グラフを考えているが，説明を容易にするために source node と target node という表現を用いている．

3：ノード i とノード j の間のエッジ確率は $\sum_{l=1}^{k}\boldsymbol{U}_{il}\boldsymbol{U}_{jl}$ によって計算され，これは $(\boldsymbol{U}\boldsymbol{U}'^{\top})_{ij}$ に一致する．

従うように，分布を属性の基底ベクトルに適用する．

以上によって，acMark はユーザが制御可能なクラス構造を持つ隣接行列 $\boldsymbol{S}$ と属性行列 $\boldsymbol{X}$ を生成可能である．またノードのクラスラベル $\boldsymbol{C}$ はクラスサイズ分布から生成し，出力する．

3.2 入力グラフからのパラメータ抽出

2.3 章で述べたように，二つ目のシナリオではクラスラベル付きの入力グラフからパラメータを抽出し，似た特徴を持つグラフを再生成する．acMark はパラメータ抽出のための関数を用いて，構造特徴とクラス特徴（ノード次数，クラス接続平均，クラス接続分散，クラスサイズ分布）を得る．抽出したパラメータを用いて，一つ目のシナリオと同じ方法でグラフを生成する．

このパラメータ抽出関数によって，acMark はユーザが入力したグラフと似たグラフを容易に生成できる．加えて，任意のグラフサイズにも対応が可能であり，ユーザは入力グラフによってクラス特徴を指定した上でノード数やエッジ数を変更することができる．

3.3 計　算　量

提案手法の時間および空間計算量について議論する．実グラフはほとんどの場合において，疎な隣接行列を持つ [17] . そのような場合，ノード次数の平均 θ_{Avg} を定数として考えることができ，$m \propto n$ とすることができる．そのため，m は n^2 に比べて非常に小さい値と考えることができる．

時間計算量. 計算量の大きい隣接行列，属性行列の生成について，それぞれの時間計算量を分析する．隣接行列の生成は *Class selection from source node* と *Target node selection from class* からなる．前者のステップでは，各ノードに対してノード次数分だけクラスを選択するため，$O(kn\theta_{Avg})$ のコストを要する．後者のステップでは，乱数生成とリストアクセスなのでコストは $O(1)$ である．よって，隣接行列生成の計算量は $O(mkr)$ と表すことができる．ここでは，$m = n\theta_{Avg}$ であり，r は隣接行列生成での反復を表す定数である．最後に，属性行列の生成においては，$\boldsymbol{U}\boldsymbol{V}^\top$ の行列積の計算のために $O(dkn)$ を要する．以上の議論から，合計の計算量は $O(mkr + dkn)$ である．

空間計算量. 隣接行列 $\boldsymbol{S}$ については，隣接リストを用いた表現によって空間計算量は $O(m)$ である．また，逆関数法のためのリストのサイズは $O(kn)$ である．最後に，属性行列 $\boldsymbol{X}$ のサイズは $O(nd)$ である．したがって，空間計算量は $O(m + kn + dn)$ である．

4 実　　験

本章では，acMark が高精度なグラフを効率的に生成可能であることを示すために，以下の質問に答える実験を行う．**Q1**：生成されたグラフはユーザが指定したグラフ特徴を満たすか？(4.1 章)，**Q2**：acMark はノードとクラス間の接続割合を柔軟に制御可能か？(4.2 章)，**Q3**：acMark はどれだけよくスケールするか？(4.3 章)．

表 1: 割当てノード次数と実際のノード次数との間の MAPE.

m	2^{16}	2^{17}	2^{18}	2^{19}	2^{20}
Error	3.56e–3	7.20e–4	3.04e–4	2.17e–4	1.73e–4

本実験では，LFR[4] と DC-SBM[5] を主要な比較手法とする．その理由は，LFR と DC-SBM が生成するグラフはクラスラベルを持ち，かつクラスレベルの接続プロポーションは制御可能であり，構造の面で acMark が生成できるグラフと似ているからである．

acMark は Python3 によって実装されており，実験は Intel(R) Xeon(R) CPU E5-2640 v4 @ 2.40GHz with 1TB memory を用いた．また全ての実験は単一スレッドかつ単一コアで実施した．

4.1 Q1：生成されたグラフはユーザが指定したグラフ特徴を満たすか？

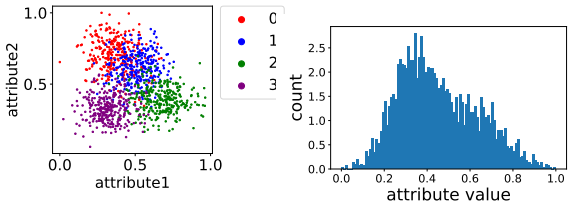
ここではユーザが入力したノード次数と属性の分布に従うように acMark がグラフを生成するかを検証する．最初に，式 (11) で示したグラフ特徴のロスである割り当てノード次数と実際に生成されたノード次数の間の MAPE を計算する．エッジ数 m を $\{2^{16}, 2^{17}, 2^{18}, 2^{19}, 2^{20}\}$ の範囲で変化させる．パラメータ k, r, n はそれぞれ $5, 50, m/32$ と設定し，$\boldsymbol{M}$ の対角要素を 0.4 とし，その他の要素は 0.15 とし，5 回平均を結果として利用する．表 1 において，ノード次数の MAPE を示す．この結果から acMark によって生成されたグラフはほとんど与えられたノード次数に従うことが確認できる．次に図 4 において，属性の分布とヒストグラムを示す．属性の分布には正規分布を設定する．図 4a は二つの属性の 2-D プロットであり，図 4b は一つの属性のヒストグラムを示している．属性の分布が正規分布に従うので，与えられた分布に基づいて acMark が属性を生成できることが示された．まとめとして，この実験ではユーザが指定したグラフ特徴に従うグラフを acMark が生成できることを示した．

4.2 Q2：acMark はノードとクラス間の接続割合を柔軟に制御可能か？

ノードとクラス間の接続プロポーションを acMark が

4：https://networkx.github.io/documentation/networkx-2.1/reference/algorithms/generated/networkx.algorithms.community.community_generators.LFR_benchmark_graph.html

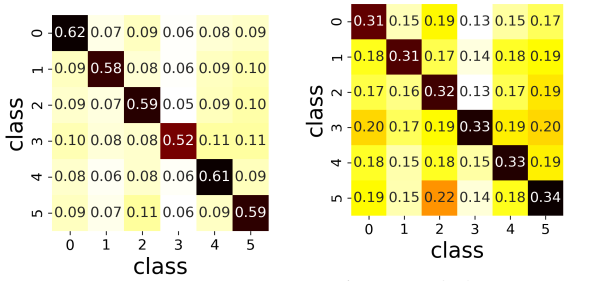
5：https://networkx.org/documentation/stb/reference/generated/networkx.generators.community.stochastic_block_model.html



(a) 生成された属性の分布．色はノードが属するクラスを意味する．

(b) 生成された属性（attribute1）のヒストグラム．

図 4: 属性特徴．属性生成のパラメータである α，σ，ω はそれぞれ 0.1, 0.0, 0.2 となるように設定する．

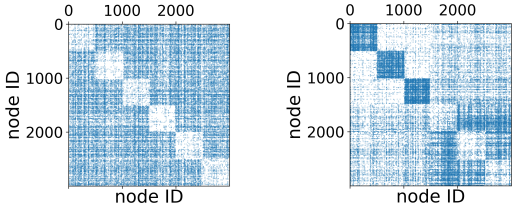


(a) 各セルは生成グラフのクラス接続平均 $\boldsymbol{M}^{gen}$ を表す．対角要素は 0.6 および他の要素は 0.08 に設定する．

(b) 各セルは生成グラフのクラス接続分散 $\boldsymbol{D}^{gen}$ を表す．対角要素はそれぞれ $[0.2, 0.2, 0.25, 0.25, 0.3, 0.3]$ とし，他の要素は 0.05 に設定する．

図 5: クラス接続平均とクラス接続分散の可視化．パラメータは $n = 10000$, $m = 100000$, $k = 6$ とする．

サポートすることを示すために，図 5 は生成グラフのクラス接続平均とクラス接続分散のヒートマップを示す．ここでは，$\boldsymbol{M}^{gen}$ と $\boldsymbol{D}^{gen}$ を生成グラフのクラス接続平均とクラス接続分散とする．まず，クラスレベル接続プロポーションを acMark が制御可能であることを示す．$\boldsymbol{M}$ の全ての対角要素を 0.6 とする．図 5a は，$\boldsymbol{M}^{gen}$ の対角要素が 0.6 に近い値であることを示す．$\boldsymbol{M}^{gen}$ の対角要素が 0.6 に完全に一致しない理由は，エッジ生成ステップが確率的な手続きに基づいているからである．この図から acMark がポジティブトポロジークラスをサポートすることを確認できる．図 5b は，ノードとクラスの間の接続プロポーションの分散を acMark が制御可能かを示す．$\boldsymbol{D}$ の対角要素をそれぞれ $[0.2, 0.2, 0.25, 0.25, 0.3, 0.3]$ とし，その他の要素は 0.05 と設定する．図 5b から，$\boldsymbol{D}^{gen}_{00}$ と $\boldsymbol{D}^{gen}_{11}$ は $\boldsymbol{D}^{gen}_{44}$ と $\boldsymbol{D}^{gen}_{55}$ 比べて小さい値であることがわかる．$\boldsymbol{D}^{gen}$ の対角要素が設定した値に完全に一致しない理由は，acMark はノード次数やクラス接続平均の制約を優先的に満たすからである．この結果は，優先度が低いとはいえ acMark が接続プロポーションの分散をサポートすることを示している．まとめとして，この実験では acMark がクラス接続平均とクラス接続分散を用いて，ノードとクラス間の接続プロポーションをサポートすることを示した．



(a) ネガティブトポロジークラス．$\boldsymbol{M}$ の全ての対角要素は 0.05 に設定する．

(b) ポジティブとネガティブトポロジークラスの両方を含むグラフ．

図 6: 隣接行列の可視化．パラメータは $n = 3000$, $m = 30000$, $k = 6$ と設定する．

表 2: 実行時間とメモリ消費量．TO と OOM はそれぞれ 36 時間以内に終了しなかった場合とアウトオブメモリを意味する．

	m	2^{21}	2^{24}	2^{27}	2^{30}
time [sec]	acMark	1.52e2	1.24e3	1.16e4	1.12e5
	LFR	2.17e2	1.50e4	TO	TO
	DC-SBM	9.60e3	OOM	OOM	OOM
memory [MiB]	acMark	1.00e3	7.80e3	5.33e4	4.88e5
	LFR	1.40e3	7.45e3	TO	TO
	DC-SBM	6.37e5	OOM	OOM	OOM

次に，acMark がネガティブトポロジーと混合したトポロジータイプをサポートすることをそれぞれ示す．図 6 は生成グラフの隣接行列を表す．隣接行列では縦軸と横軸にそれぞれ対応するノード ID の間にエッジがある場合を青いドットで示す．この実験ではクラス数は 6 とし，図中の対角線上にブロックとしてクラスが観測される．図 6a は全てのクラスがネガティブトポロジーである場合を示す．$\boldsymbol{M}$ の全ての対角要素をランダムのときより低い 0.05 に設定する．図より，ノードは明らかに他のクラスに属するノードと接続する傾向があることがわかる．さらに複雑な場合として，ポジティブとネガティブの両方のクラスを含む一つのグラフを図 6b に示す．クラス $0, 1, 2$ はポジティブトポロジーなので，それらに属するノードは内部で密に接続している．クラス $3, 4, 5$ に属するノードは他のクラスに属するノードに接続する傾向があることが確認される．以上のことから，acMark がノードとクラス間の接続プロポーションを柔軟に制御できることを確かめた．

4.3 Q3：acMark はどれだけよくスケールするか？

acMark のスケーラビリティを調査するために，エッジ数を変化させることで実行時間とメモリ消費量を示す．実験ではエッジ数を $\{2^{21}, 2^{24}, 2^{27}, 2^{30}\}$ の範囲で変化させる．パラメータ k, r, n はそれぞれ $5, 50, m/32$ とし，$\boldsymbol{M}$ の対角要素は 0.6 とし，他の要素は 0.1 とする．acMark

との比較対象として，既存手法の中では最も acMark と似たグラフを生成する LFR と DC-SBM を用いる．

時間計算量．まず，acMark，LFR および DC-SBM の実行時間の比較を行う．表 2 において，エッジ生成に要する実行時間を示し，acMark がエッジ数に対して線形に動作することと，10 億（2^{30}）エッジを 32 時間で生成可能であることを示している．LFR と DC-SBM と比較して，acMark が大幅に高速であることが確認できる．これは acMark が逆関数法によって効率的にエッジを生成している一方，LFR はグラフサイズの増加に伴いエッジ生成にしばしば失敗し，DC-SBM の時間計算量が $O(n^2)$ であるからである．これらの結果より，acMark がクラスラベルを持った大規模グラフを効率的に生成することを確認した．

空間計算量．次にメモリ消費量について分析を行う．表 2 はグラフ生成に使用したメモリ消費量を示す．acMark がエッジ数に線形にスケールすることがわかる．acMark と LFR は似たメモリ消費量である．DC-SBM ではエッジ数に対して 2 乗に比例してメモリ消費量が増加することを観測した．

まとめとして，これらの実験では acMark は時間および空間計算量の両方でエッジ数に対して線形にスケールすることを検証した．さらに最先端の既存手法を大幅に上回ることを示し，他の手法では実践的には生成不可能な大規模グラフを生成することを示した．

5 おわりに

本稿では，クラスラベルを持った属性付きグラフ生成器である acMark を提案した．実験を通して acMark の四つの主要な特性を検証した．1) 生成されたグラフはユーザが指定したグラフ特徴（ノード次数分布と属性値の分布）に従う，2) acMark はユーザが指定したノードとクラスの間の接続割合をサポートする，3) acMark はエッジ数に対して線形にスケールし，10 億エッジを持つグラフが生成可能であることを示した，以上のことから，acMark がユーザが指定したクラス構造を持つ大規模な属性付きグラフを生成できることを示した．

謝　辞

本研究は JSPS 科研費 JP20H00583 の助成を受けたものです。

文　献

[1] Guillaume Bagan, Angela Bonifati, Radu Ciucanu, George Fletcher, Aurélien Lemay, and Nicky Advokaat. gMark: Schema-driven generation of graphs and queries. *IEEE TKDE*, 2017.

[2] Oualid Benyahia, Christine Largeron, Baptiste Jeudy, and Osmar R Zaïane. Dancer: Dynamic attributed network with community structure generator. In *Proceedings of ECML PKDD*, 2016.

[3] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. Node classification in social networks. In *Social network data analytics*. Springer, 2011.

[4] Aleksandar Bojchevski and Stephan Günnemann. Bayesian Robust Attributed Graph Clustering: Joint Learning of Partial Anomalies and Group Structure. In *Proceedings of AAAI*, 2018.

[5] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.

[6] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. Finding community structure in very large networks. *Physical review E*, 2004.

[7] Luc Devroye. Nonuniform random variate generation. *Handbooks in operations research and management science*, 2006.

[8] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of KDD*, 1996.

[9] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in NIPS*, 2014.

[10] Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 2011.

[11] Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[12] P Krishna Kumar, Paul Langton, and Wolfgang Gatterbauer. Factorized graph representations for semi-supervised learning from sparse data. *arXiv preprint*, 2020.

[13] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 2008.

[14] Christine Largeron, Pierre-Nicolas Mougel, Reihaneh Rabbany, and Osmar R Zaïane. Generating attributed networks with communities. *PloS one*, 2015.

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

[16] Miller McPherson, Lynn Smith-Lovin, and James M Cook. Birds of a feather: Homophily in social networks. *Annual review of sociology*, 2001.

[17] Mark EJ Newman. *Networks: An Introduction*. Oxford University Press, 2010.

[18] Gergely Palla, Imre Derényi, Illés Farkas, and Tamás Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *nature*, 2005.

[19] Himchan Park and Min-Soo Kim. TrillionG: A trillion-scale synthetic graph generator using a recursive vector model. In *Proceedings of SIGMOD*, 2017.

[20] Martin Simonovsky and Nikos Komodakis. Graphvae: Towards generation of small graphs using variational autoencoders. In *Proceedings of ICANN*. Springer, 2018.

[21] Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. In *Proceedings of IJCAI*, 2019.