修 士 論 文 の 和 文 要 旨

研究科・専攻	大学院情報理工学研究科　　情報・ネットワーク工学専攻　博士前期課程		
氏　　名	渡邊　裕貴	学籍番号	1631170
論 文 題 目	コミュニティ抽出手法を用いたソフトウェア構成要素の関係解析		

要　　旨

オープンソースソフトウェアの分野では, ソフトウェアをパッケージ単位で提供することがよく見られる. また, C言語のプログラムでは, 関数という形で処理をモジュール化することが一般的である. このようにソフトウェアを構成する要素を細分化することで, ユーザは必要な機能を得るにあたり最小限の要素を呼び出すだけで済むようになるが,それらの要素の数が多くなるにつれ, 互いがどのように関連しあっているかを理解することは困難になる.

こうした問題を解決するために, ソフトウェアの各要素の関係を表したネットワークに対してコミュニティ抽出手法を適用し, おおまかな構造を取り出す研究が近年行われてきた. しかし, 従来の手法では, コミュニティの構造をあらかじめ仮定しているために, 仮定と異なる特徴の構造を持つコミュニティの抽出をすることができない問題や, 抽出されたコミュニティ構造の妥当性を合理的な規準を用いて測ることができない問題があった.

本研究の目的は, ソフトウェアを構成する細かな要素同士の関係に対して, 確率モデルを用いたコミュニティ抽出手法を適用し, より真の分布に近い構造を推定することである.

本研究では, ネットワークを生成する真の分布に近い構造を推定するにあたって, 情報量規準を用いて確率モデルの選択を行う. 実際に, ソフトウェアパッケージの依存関係ネットワークと, 関数の呼び出し関係ネットワークという二つの異なる粒度の要素を持つネットワークに対して提案手法を適用し, その有効性を確かめる.

実験によって得られた結果から, ソフトウェアパッケージの依存関係ネットワーク, 関数の呼び出し関係ネットワークどちらからも, いくつかのコミュニティからなる構造を抽出することに成功した. また, パッケージ依存関係ネットワークでは, modularity最大化を用いた手法では検出できない, コミュニティ外部との間のエッジ密度が高くなるコミュニティが抽出されることが確認できた. エッジの向きを考慮したコミュニティ抽出, 他ノードとの接続数の少ないノードに対する正確なクラスタリングについては今後の課題である.

平成２９年度修士論文

コミュニティ抽出手法を用いた
ソフトウェア構成要素の関係解析

情報・ネットワーク工学専攻
コンピュータサイエンスコース

１６３１１７０ 渡邊　裕貴

主任指導教員　　寺田 実 准教授
　　指導教員　　岩崎 英哉 教授

提出日 ２０１８年 １月２９日

概要

目的

オープンソースソフトウェアの分野では, ソフトウェアをパッケージ単位で提供することがよく見られる. また, C 言語のプログラムでは, 関数という形で処理をモジュール化することが一般的である. このようにソフトウェアを構成する要素を細分化することで, ユーザは必要な機能を得るにあたり最小限の要素を呼び出すだけで済むようになるが, それらの要素の数が多くなるにつれ, 互いがどのように関連しあっているかを理解することは困難になる.

こうした問題を解決するために, ソフトウェアの各要素の関係を表したネットワークに対してコミュニティ抽出手法を適用し, おおまかな構造を取り出す研究が近年行われてきた. しかし, 従来の手法では, コミュニティの構造をあらかじめ仮定しているために, 仮定と異なる特徴の構造を持つコミュニティの抽出をすることができない問題や, 抽出されたコミュニティ構造の妥当性を合理的な規準を用いて測ることができない問題があった.

本研究の目的は, ソフトウェアを構成する細かな要素同士の関係に対して, 確率モデルを用いたコミュニティ抽出手法を適用し, より真の分布に近い構造を推定することである.

方法

本研究では, ネットワークを生成する真の分布に近い構造を推定するにあたって, 情報量規準を用いて確率モデルの選択を行う. 実際に, ソフトウェアパッケージの依存関係ネットワークと, 関数の呼び出し関係ネットワークという二つの異なる粒度の要素を持つネットワークに対して提案手法を適用し, その有効性を確かめる.

結論

実験によって得られた結果から, ソフトウェアパッケージの依存関係ネットワーク, 関数の呼び出し関係ネットワークどちらからも, いくつかのコミュニティからなる構造を抽出することに成功した. また, パッケージ依存関係ネットワークでは, modularity 最大化を用いた手法では検出できない, コミュニティ外部との間のエッジ密度が高くなるコミュニティが抽出されることが確認できた. エッジの向きを考慮したコミュニティ抽出, 他ノードとの接続数の少ないノードに対する正確なクラスタリングについては今後の課題である.

目次

図目次

表目次

第 1 章　序論

1.1　背景

オープンソースソフトウェアにおいては, ソフトウェアはいくつかのプログラムをまとめたパッケージ単位で提供されることがよく行われる. また, C 言語のように, 処理をモジュール化することができるプログラミング言語においては, プログラムを細かな機能ごとにサブルーチンとして分割する手法がよく用いられる. こうしたソフトウェアを構成する要素を細分化する手法により, ユーザーは既存の要素から必要な機能だけを取り出して組み合わせることで, 望んだ機能を得ることができる.

しかし, 多くのソフトウェアは膨大な数の要素により構成されるため, それらが互いにどのように関連しあっているかを理解することは困難である. 例えば, あるパッケージを改修した際に他のパッケージに与える影響や, 既存のパッケージと類似した機能を持つパッケージを新たに開発する際に必要となるパッケージを知るためには, 人間による試行錯誤が必要となると考えられる.

近年, 上記の課題を解決するための一つのアプローチとして, 複雑ネットワーク解析を用いる研究が提案されてきている. ソフトウェアを構成する要素の関係は, ネットワーク構造として表現することができる. ここで, ネットワークにおける各ノード (頂点) はそれぞれの要素を, 各エッジ (辺) は要素間のつながりを表す. このネットワークの構造を解析することにより, 各要素が互いにどのような形で関連しあっているかを知ることができる. 特に, ネットワーク構造において, 類似した性質を持つ頂点をグループ (以下では, コミュニティと呼ぶ) としてみつけることができれば, 一つのソフトウェアをなす膨大な数の構成要素が, 大まかにどのような内部構造を持つのかを捉えることが可能になる.

このように, あるネットワークから, つながりの構造上似た性質を持つノードをコミュニティとして分類する手法は, ネットワークのコミュニティ抽出 (もしくは, ネットワーククラスタリング) と呼ばれる. 与えられたネットワークからコミュニティを抽出するための具体的なアルゴリズムとしては, 様々な手法 [1][2] が提案されている. さらに, ソフトウェア工学の分野においても, コミュニティ抽出を利用した研究は数多く行われている. 例として, Java のクラス間の関係を表すネットワークからコミュニティ構造を抽出し, プログラムのモジュール化・リファクタリングの支援への応用を目指した研究 [3] や, ソースコードの各行の依存関係を表すネットワークに対しコミュニティ抽出を行い, 処理内容ごとに分類することを可能とした研究 [4], 関数のコールグラフからコミュニティを抽出し, 得られたコミュニティ構造の特徴からマルウェアを検出する研究 [5] などが行われている.

1.2　現状の問題

上述したように, ソフトウェアの構成要素同士の関係を表すネットワーク構造から, コミュニティ抽出を行うことにより, 様々な応用上有用な情報を獲得しようとする研究はいくつか行われているが, いずれもヒュー

リスティックな手法に基づくものであった. 例えば, 抽出されたコミュニティ構造において, コミュニティ内部でのつながりが多く, 異なるコミュニティ間でのつながりが少ない度合いを測る指標として modularity と呼ばれるものがあり, 多くの研究においてはこの値を最大化するようなコミュニティ構造の抽出を目指している. しかし, 与えられたネットワークが上記のような構造を持つか否かは事前には分からず, またこのような modularity 最大化に基づくアプローチの妥当性を定量的に評価することができる規準も存在しない. そのため, 合理的な規準を用いてコミュニティ抽出の結果の妥当性を測るための手法が必要であり, またそのような規準を適用可能なコミュニティ抽出法を提案することが望まれている.

1.3　目的

本研究の目的は, ソフトウェアパッケージの依存関係ネットワークと, 関数の呼び出し関係ネットワークという二つの異なる粒度の要素を持つネットワークに対して, 確率モデルを用いたコミュニティ抽出を行い, より真の分布に近い構造を推定することである. 確率モデルを用いたコミュニティ抽出手法では, ネットワークが未知の分布 (情報源) から発生したサンプルであると捉え, 真の情報源を推測することでコミュニティ構造を推定する. 本研究では, 得られたコミュニティ構造の妥当性を示すにあたって, 推測された分布と真の情報源との間の汎化誤差を情報量規準を用いて推測する. また, 推定されたコミュニティ構造が, どういった特徴をもつ構造であるかをいくつかの観点から調査する.

1.4　本論文の構成

本論文の構造を簡単に説明する. 本章では, 本研究の背景・目的について簡単に述べた. 第 2 章では, 本研究と特に関連が深い研究を紹介する. 第 3 章では, 本研究の提案する手法について述べ, 第 4 章では提案手法で用いた理論について詳細に記述する. 第 5 章では, 提案手法の有効性を示すにあたって行った実験の手法と結果・考察について述べる. 第 6 章では本研究のまとめと, 今後の課題について述べる.

第 2 章　関連研究

2.1　Analysis of the package dependency on Debian GNU/Linux

2.1.1　概要

Sousa ら [6] は, Debian GNU/Linux[*1]のパッケージマネージャである apt[*2]の提供するパッケージについて解析を行い, その依存関係が作るネットワークの特性を調査した.

著者らは, まずパッケージ全体の依存関係から, パッケージをノード, 依存するパッケージから依存されるパッケージへのつながりを有向エッジとして, ネットワークを構築した. その上で, 各ノードの出次数, 入次数の分布を計測し, ベキ乗則に従うことを確かめた.

また, 著者らは構築した依存関係ネットワークに対し, 有向ネットワーク向けに拡張されたクラスタ係数を測定し, そのネットワークがランダムなネットワークに比べ高いクラスタ性を持つことを確かめた. さらに, 依存関係ネットワークの各エッジに対して媒介中心性を測定し, その分布から, 極端に媒介中心性の高いボトルネックになるようなエッジは多くないことを発見した.

それに加えて, 著者らは有向ネットワークにおける modularity 最大化を依存関係ネットワークに適用し, ネットワークからコミュニティの抽出を行った. 具体的には, modularity 最大化問題を, 以下のように示されるハミルトニアンの最小化に置き換え, ポッツモデルを用いた手法 [7] により解く手法を用いた.

$$H = -\frac{1}{M}\sum_{ij}\left[a_{ij} - \frac{k_i^{in}k_j^{out}}{M}\right]\delta_{\sigma_i\sigma_j} \tag{2.1}$$

ここで, M はノード数, a_{ij} は入力ネットワークの隣接行列, k_i^{in} はノード k_i の入次数, k_j^{out} はノード k_j の出次数, $\delta_{\sigma_i\sigma_j}$ は i, j が同一コミュニティならば 1, そうでなければ 0 となる変数である. この手法により, 著者らはネットワークから X window に関するパッケージを含んだコミュニティや, libc, perl に関するパッケージを含んだコミュニティを発見した.

2.1.2　本研究との関連

この研究は, ソフトウェアパッケージの依存関係を表すネットワークからコミュニティを抽出するという点で本研究と類似するが, この研究がリポジトリに含まれるパッケージ全てを含んだネットワークに対してコミュニティ抽出を行っているのに対し, 本研究では単一のソフトウェアを構成するパッケージの関係について調べるため, 一つのソフトウェアをインストールする際に必要となるすべてのパッケージで構成されるネットワークに対してコミュニティ抽出を行うという点で異なる.

[*1] https://www.debian.org/

[*2] https://wiki.debian.org/Apt

また, この研究で用いている modularity 最大化問題で抽出できるコミュニティは, 同一コミュニティ間のエッジが密になり, そうでない所は疎になる特徴を持つ構造を持つことを前提としている一方, 本研究ではより一般的に複数の構造が混在するコミュニティ構造に対応している手法を用いている点でも異なる (詳細は 3.2 節参照).

2.2 Community Detection in Directed Weighted Function-call Networks

2.2.1 概要

Zhao ら [8] は, プログラミング言語の関数の呼び出し関係のグラフについて, 呼び出しの向きと, 呼び出しの回数によって表される重みを考慮し, クラスタリングをするアルゴリズムを提案した.

この研究では, Ahn らが提案したグラフに所属するリンク (エッジ) をクラスタリングする手法 [9] を拡張し, リンクの持つ向きと重みの情報を含めたクラスタリングを行えるようにする手法を紹介している. この手法では, リンクをコミュニティに分類したとき, その両端のノードも同時にそのコミュニティに所属すると考えるため, 一つのノードが複数のコミュニティに所属する可能性がある.

具体的な手法としては, 二つのリンク間の相関として,

$$C(e_{ik}, e_{jk}) = L(e_{ij}) \cdot S(e_{ik}, e_{jk}) \tag{2.2}$$

$$L(e_{ij}) = a_{ij}(1 - p_{ij}) + a_{ji}(1 - p_{ji}) \tag{2.3}$$

$$S(e_{ik}, e_{jk}) = \frac{\boldsymbol{a}_i \cdot \boldsymbol{a}_j}{|\boldsymbol{a}_i|^2 + |\boldsymbol{a}_j|^2 - \boldsymbol{a}_i \cdot \boldsymbol{a}_j} \tag{2.4}$$

$$p_{ij} = \frac{k_j^{out} k_i^{in}}{k_j^{out} k_i^{in} + k_i^{out} k_j^{in}} \tag{2.5}$$

で定義される $C(e_{ik}, e_{jk})$ を用いて, 階層化クラスタリングによりリンクをクラスタリングするようにしている. ここで, e_{ij} はノード i から j へ接続するリンクを, $\boldsymbol{a}$ はノード i とノード i, j ともに隣接するノードの間の重みを含んだベクトルを意味する.

この手法により, 著者らは, 静的解析に基づいた Lua1.0 の関数呼び出しグラフについてクラスタリングを行い, 既存の Girvan-Newman 法 [10] やスピングラスアルゴリズム [11] との比較を行っている. その結果, これら既存のアルゴリズムと同様のコミュニティを抽出することに成功し, なおかつ複数のコミュニティに所属するノードを発見することにも成功している.

2.2.2 本研究との関連

この研究は, 関数の呼び出し関係ネットワークからコミュニティを抽出するという点で本研究と類似するが, この研究では呼び出し関係の向きと, 呼び出した回数を考慮したリンクの相関度を元にネットワーク上のリンクをクラスタリングしているのに対して, 本研究ではリンクの向き, 及び重さは無視し, 各ノードをリンクの接続パターンの類似性からクラスタリングしているという点で異なる. リンクの向きを除くことで元となるデータの情報が一部失われることになるが, 後述するようにこれはノードへ入ってくるエッジとノードから出ていくエッジ両方の情報を見るためである.

2.3 Exploring community structure of spftware Call Graph and its applicatopns in class cohesion measurement

Qu ら [12] は, Java で書かれた大規模なオープンソースプログラムに対して, 関数の呼び出し関係グラフに対してコミュニティ抽出を行い, 抽出されたコミュニティ構造から, 各クラスが持つメソッドの結束力を示す指標を新たに提案した. 具体的な手法は以下の通り.

ソフトウェアに含まれるクラスが持つメソッドについて, 静的解析を行い, メソッドの呼び出し関係のネットワークを生成する. その中でもっとも規模の大きいネットワークを LCC(Largest weakly Connected Components) と定め, LCC にしきい値 t 以上の割合でメソッドが含まれるクラスの集合を $C_{LCC-Cohesion}$ とする. また, LCC に対してコミュニティ抽出を行った結果から, i 番目のコミュニティに含まれるメソッドのうちクラス $C \in C_{LCC-Cohesion}$ に含まれるメソッドの個数を n_i $(i = 1, 2, \cdots, N)$ とし, $n_{max} = \max\{n_i\}$ とする.

このとき, クラス C について, $MCC(C)$ を次のように定める.

$$MCC(C) = \begin{cases} 1 & \text{if } m = 1 \\ 0 & \text{if } n_{max} = 1 \text{ and } m \geq 2 \\ \frac{n_{max}}{m} & \text{otherwise} \end{cases} \tag{2.6}$$

また MCEC(C) を次のように定める.

$$MCEC(C) = \begin{cases} 1 & \text{if } m = 1 \\ 1 - (-\frac{1}{\ln N} \sum_{i=1}^{N} \frac{n_i}{m} \ln \frac{n_i}{m}) & \text{if } N \geq 2 \end{cases} \tag{2.7}$$

これらの指標は, 同一クラス内のメソッドの結束が強ければ 1 に近づき, そうでなければ 0 に近づくことを示す. 著者らは, 実際の Java で書かれたソフトウェアに対する計測結果から, 主成分分析を用いることで, 他の結束度を測る指標では得られなかった情報が MCC 及び $MCEC$ から得られることを示している. また, 障害を含んだソフトウェアのデータセットに対してこれらの指標を計測し, ロジスティック回帰を用いて各指標 がどの程度障害の予測に寄与しているかを検証し, MCC 及び $MCEC$ を既存の指標と組み合わせることで予測性能を向上させることができることを示している.

2.3.1 本研究との関連

この研究では, コミュニティ抽出を行うにあたって, コミュニティ内のエッジが密になり, それ以外が疎となるコミュニティ構造を仮定している. これは, より同一クラス内のメソッドのつながりが多いほうがクラス内部の結束性が高まるという考え方に基づいているからである. 本研究では, こういったコミュニティ構造に限定することなく, 関数のつながりからコミュニティ構造を取り出しているという点で異なる.

第 3 章　提案手法

3.1　関係ネットワークの構築

本研究では, コミュニティ抽出を行う対象のネットワークとして, ソフトウェアパッケージの依存関係ネットワークと, 関数の呼び出し関係ネットワークを選んだ. 以下, その詳細を述べる.

3.1.1　パッケージ依存関係のネットワーク

本研究では, コミュニティ抽出を行う一つ目の対象として, GNU/Linux ディストリビューションの一つである ubuntu16.04[*1]で採用されているパッケージマネージャ apt によって提供されているパッケージから, パッケージ間の依存関係ネットワークを構築し, コミュニティ抽出を行った.

apt では, いくつかのプログラムの集まりを一つのパッケージとして扱っており, あるパッケージをインストールする際に, 別のパッケージの機能が必要になれば, そのパッケージも自動的にインストールする. パッケージマネージャのこのような機能を「依存関係の解決」と呼び, パッケージ A がパッケージ B を必要とした場合であれば, 「パッケージ A はパッケージ B に依存している」という. このような二つのパッケージ間の依存関係には, 複数の種類の関係が存在するが, 本研究では apt が標準で解決する次の 3 つの依存関係を取り扱う. ただし, 実験上でこれらの依存関係を区別することはせず, 全てパッケージ間のつながりとして等しく扱う.

1. Depends
   パッケージをインストールする際に, 同時にインストールされる必要があるパッケージに対する依存
2. Pre-Depends
   パッケージをインストールする際に, 先にインストールされている必要があるパッケージに対する依存
3. Recommends
   パッケージをインストールする際に, 必ずしも必要ではないが, 同時にインストールすることが推奨されるパッケージに対する依存

3.1.1.1　対象とするパッケージ

apt で提供されているパッケージのうち, 次の二つのパッケージの依存を満たすパッケージ群から構成されるネットワークを採用した.

1. chromium-browser[*2]

[*1] https://wiki.ubuntu.com/XenialXerus/ReleaseNotes
[*2] https://www.chromium.org/

オープンソースで開発されているウェブ・ブラウザ

2. fcitx-mozc[*3][*4]
   UNIX-like なオペレーティング・システムのためのインプットメソッドと日本語変換用エンジン

3.1.1.2　データセットの生成手法

はじめに, 元となるパッケージ (chromium 及び fcitx-mozc) から依存関係を再帰的にたどり, パッケージをノード, パッケージ間の依存をエッジとする有向エッジを持つネットワークを作成する. その後, 一度重みの無い無向ネットワークに変換してから隣接行列を作成する. これは, コミュニティ抽出において, 各パッケージに対して依存するパッケージ・依存されるパッケージ両方の情報を使うためである. なお, 依存関係の解析には debtree[*5]を使用する.

3.1.1.3　データセット生成の際の備考

パッケージの依存関係の中には, 依存を満たすパッケージの選択肢が複数ある場合がある. このような場合には, その中の一つのパッケージを一つ任意に選び, それを依存されるパッケージとした.

また, パッケージの中には, 実体を持たない仮想パッケージがある. あるパッケージが仮想パッケージに依存した場合, 実際には仮想パッケージでなく, 仮想パッケージの機能を満たす実体のあるパッケージに依存していると見なせる. 仮想パッケージの機能を満たす実パッケージは, 通常複数存在し得る. そこで, 仮想パッケージへの依存があった場合, 生成するデータセットでは, 仮想パッケージではなく仮想パッケージの機能を満たす実パッケージのどれか一つを任意に選び, そのパッケージに直接依存するようにした.

3.1.2　関数の呼び出し関係のネットワーク

本研究では, コミュニティ抽出を行う対象として, さらに C 言語で書かれたソフトウェアの関数呼び出し関係のネットワーク (コールグラフ) を採用した. これは, 関数をノードとして, 関数 A 内で別の関数 B が一度でも呼ばれれば, 関数 A, B の間にエッジがあるとみなすネットワークである. 本研究では, ソフトウェアを実際に実行したときの動作をトレースした結果でなく, ソースコードの状態から読み取れる, すなわち静的な解析結果から分かる関係を扱う.

対象とするソフトウェアは以下の通りである.

1. GNU sed バージョン 4.2
2. GNU less バージョン 487

コールグラフを作る手順としては, 解析対象のソースコードから gcc の中間出力である RTL(Register Transfer Language) を作り, それを egpyt[*6]に渡すことで作成した.

具体的には, C 言語のソースコードをコンパイルする際に,

```
$ gcc -o foo foo.c -fdump-rtl-expand
```

[*3] https://fcitx-im.org/wiki/Fcitx
[*4] https://github.com/google/mozc
[*5] https://collab-maint.alioth.debian.org/debtree/
[*6] https://www.gson.org/egypt/

とオプションを指定して, RTL が記述された.expand ファイルを生成させる. egypt は expand ファイルを解析して, その中に記述されている isin と呼ばれる関数を表現したオブジェクト[*7]と, isin で表現された関数を呼ぶ call_isin 命令から, 関数の静的な呼び出し関係を読み取って, dot 言語[*8]で出力する.

コールグラフについても, パッケージの依存関係ネットワークの時と同様, 一度有向なネットワークを作成してから, エッジの方向を無視した無向ネットワークへの変換を行った.

3.2　コミュニティ抽出

前述したように, コミュニティ抽出とは, ネットワークのノードを何らかの基準に基づいてクラスタリングを行う手法である. 代表的なコミュニティ抽出の手法として, Newman らによって提案された modularity と呼ばれる指標 [13] を最大化する手法がいくつか考案されている. これらの手法では, ネットワークがコミュニティ内部の接続が密であり, それ以外は疎であるという構造を持つと仮定し, ネットワークをそうした性質を持つコミュニティに分割することを目的としている. ネットワークとノードのコミュニティ分割が与えられた時, modulairty Q は次のように表される. ここで, i はコミュニティ番号を表す.

$$Q = \sum_i (e_{ii} - a_i^2) \tag{3.1}$$

ここで, e_{ii} はグラフの総エッジ数に対するコミュニティ i 内部のエッジ数の割合, a_i はグラフに含まれる全てのノードの出次数の合計 (グラフのエッジ数 $\times 2$ と等しい) に対するコミュニティ i 内部のノードの出次数の合計の割合を表す. この指標が大きくなるほど, コミュニティ内部のエッジの密度が高くなっていることが示せる.

この手法に対して, 本研究では, 与えられたネットワークからコミュニティを抽出するにあたって, 確率モデルを用いたギブス・サンプリングによるコミュニティ抽出手法を用いた. この手法では, ネットワークが何らかの確率モデルから生成されている一つのサンプルであると仮定し, そのモデルを推測することで, ネットワークを構成しているコミュニティ構造を推定するというものであり, 詳細は第 4 章で述べる. こうした確率モデルを用いた手法は, modularity を用いた手法に対して, (1) コミュニティの持つ構造をあらかじめ仮定しない, すなわちコミュニティ内部の接続が密であり, それ以外は疎であるという構造を仮定する必要がない (2) 情報量規準を用いて得られたコミュニティ構造を定量的に評価することができる という二つの点で優れているといえる.

[*7] https://gcc.gnu.org/onlinedocs/gccint/Insns.html#Insns
[*8] https://www.graphviz.org/doc/info/lang.html

第 4 章　理論

4.1　確率モデル

本研究では, ネットワークからコミュニティ構造を推定するにあたり, Wang らの手法 [14] で提案されている確率モデルを用いた. このモデルでは, コミュニティ r に所属する各ノードからノード j へ, 独立した θ_{rj} の確率で一本のエッジが存在すると仮定し, 混合ベルヌーイ分布からネットワークが生成すると考える. 以下, 確率モデルの詳細を示す.

ネットワークのノード数を N とし, 隣接行列を A(A は $N \times N$ 行列), コミュニティ数を C とおく. また, ノード i が所属するコミュニティを $g_i (1 \leq g_i \leq C)$ と定義する.

確率モデルを作るにあたって, 次のようにパラメータを設定する.

$$\begin{cases} \pi_r \ \ldots \ \text{各ノードがコミュニティ } r \text{ に所属する確率 } (\sum_{r=1}^{C} \pi_r = 1) \\ \theta_{rj} \ \ldots \ \text{コミュニティ } r \text{ に所属するノードからノード } j \text{ へ一本のエッジが存在する確率 } (0 < \theta_{rj} < 1) \end{cases}$$

$\boldsymbol{\pi} = \{\pi_r\}$ は一般に混合比と呼ばれるパラメータであり, すべてのノードについて同じ値をとる.

$\boldsymbol{\pi} = \{\pi_r\}$, $\boldsymbol{\theta} = \{\theta_{rj}\}$ が与えられたとき, $\boldsymbol{g} = \{g_i\}$ を生成する確率は,

$$\Pr(\boldsymbol{g}|\boldsymbol{\pi}, \boldsymbol{\theta}) = \prod_{i=1}^{N} \pi_{g_i} \tag{4.1}$$

また, A を生成する確率は,

$$\Pr(A|\boldsymbol{g}, \boldsymbol{\pi}, \boldsymbol{\theta}) = \prod_{i=1}^{N} \prod_{j=1}^{N} \theta_{g_i j}^{A_{ij}} (1 - \theta_{g_i j})^{1 - A_{ij}} \tag{4.2}$$

式 (4.1), (4.2) より,

$$\begin{aligned} \Pr(A, \boldsymbol{g}|\boldsymbol{\pi}, \boldsymbol{\theta}) &= \Pr(\boldsymbol{g}|\boldsymbol{\pi}, \boldsymbol{\theta}) \Pr(A|\boldsymbol{g}, \boldsymbol{\pi}, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} \left[\pi_{g_i} \prod_{j=1}^{N} \theta_{g_i j}^{A_{ij}} (1 - \theta_{g_i j})^{1 - A_{ij}} \right] \end{aligned} \tag{4.3}$$

であるから,

$$\begin{aligned} \Pr(A|\boldsymbol{\pi}, \boldsymbol{\theta}) &= \sum_{g_1=1}^{C} \cdots \sum_{g_n=1}^{C} \Pr(A, \boldsymbol{g}|\boldsymbol{\pi}, \boldsymbol{\theta}) \\ &= \prod_{i=1}^{N} \sum_{r=1}^{C} \pi_r \prod_{j=1}^{N} \theta_{rj}^{A_{ij}} (1 - \theta_{rj})^{1 - A_{ij}} \end{aligned} \tag{4.4}$$

さらに,

$$\Pr(A, g_i = r|\boldsymbol{\pi}, \boldsymbol{\theta}) = \left[\pi_r \prod_{j=1}^{N} \theta_{rj}^{A_{ij}} (1-\theta_{rj})^{1-A_{ij}}\right] \left[\prod_{\substack{i'=1 \\ i' \neq i}}^{N} \sum_{s=1}^{C} \pi_s \prod_{j=1}^{N} \theta_{sj}^{A_{i'j}} (1-\theta_{sj})^{1-A_{i'j}}\right] \tag{4.5}$$

式 (4.4), (4.5) から, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ が与えられたときにノード i がコミュニティ r に所属する確率は,

$$\begin{aligned} \Pr(g_i = r|\boldsymbol{\pi}, \boldsymbol{\theta}) &= \frac{\Pr(A, g_i = r|\boldsymbol{\pi}, \boldsymbol{\theta})}{\Pr(A|\boldsymbol{\pi}, \boldsymbol{\theta})} \\ &= \frac{\pi_r \prod_{j=1}^{N} \theta_{rj}^{A_{ij}} (1-\theta_{rj})^{1-A_{ij}}}{\sum_{s=1}^{C} \pi_s \prod_{j=1}^{N} \theta_{sj}^{A_{ij}} (1-\theta_{sj})^{1-A_{ij}}} \end{aligned} \tag{4.6}$$

[14] では, 観測された A に対して, 式 (4.4) で示される尤度が最大になるようなパラメータを EM 法 [15] により点推定する手法を提案している. その結果から, 各ノード i について式 (4.6) を計算し, 最も所属確率の大きいコミュニティにそれぞれ所属するものとしている.

4.2　ギブス・サンプリング

本研究では, 前述した確率モデルに対して, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ の事前分布を設定し, 得られた事後分布 $\Pr(\boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}|A, \boldsymbol{\alpha}, \boldsymbol{\beta})$ から $\boldsymbol{y}$, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ をサンプリングすることで, コミュニティの推定を行った. ただし, このサンプリングを直接行うことは難しいため, 以下のようにギブス・サンプリングを用いる.

はじめに, ハイパーパラメータ

$$\boldsymbol{\alpha} = \{\alpha_1, \alpha_2, \cdots, \alpha_r\} \tag{4.7}$$

$$\boldsymbol{\beta}^{(1)} = \{\beta_{rj}^{(1)}; r = 1, 2, \cdots, C, j = 1, 2, \cdots, N\} \tag{4.8}$$

$$\boldsymbol{\beta}^{(2)} = \{\beta_{rj}^{(2)}; r = 1, 2, \cdots, C, j = 1, 2, \cdots, N\} \tag{4.9}$$

を設定したとき, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ の確率分布を以下のように設定する.

$$\Pr(\boldsymbol{\pi}|\boldsymbol{\alpha}) = \frac{1}{Z_1} \prod_{r=1}^{C} \pi_r^{\alpha_r - 1} \tag{4.10}$$

$$\Pr(\boldsymbol{\theta}|\boldsymbol{\beta}^{(1)}, \boldsymbol{\beta}^{(2)}) = \frac{1}{Z_2} \prod_{r=1}^{C} \prod_{j=1}^{N} \theta_{rj}^{\beta_{rj}^{(1)} - 1} (1-\theta_{rj})^{\beta_{rj}^{(2)} - 1} \tag{4.11}$$

ここで, Z_1, Z_2 は正規化定数である.

ギブス・サンプリングをするに当たって, 次のように変数 $\boldsymbol{y} = \{y_{ir}\}$ の定義を行う.

$$y_{ir} = \delta(g_i, r) \tag{4.12}$$

ここで, δ はクロネッカーのデルタであり, i 番目のノードが所属するコミュニティ g_i が r ならば 1, そうでなければ 0 となるような変数である. ゆえに, $\boldsymbol{y} = \{y_{ir}\}$ は各 i ごとに一つの r のみで 1 になるという条件を満たす.

$\boldsymbol{y} = \{y_{ir}\}$ を用いて, 式 (4.1), (4.2) を書き換えると,

$$\Pr(\boldsymbol{y}|\boldsymbol{\pi}, \boldsymbol{\theta}) = \prod_{r=1}^{C} \exp\left(\ln \pi_r \sum_{i=1}^{N} y_{ir}\right) \tag{4.13}$$

$$\Pr(A|\boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}) = \prod_{r=1}^{C}\prod_{j=1}^{N} \exp\left(\sum_{i=1}^{N} y_{ir}\left\{A_{ij}\ln\theta_{rj} + (1-A_{ij})\ln(1-\theta_{rj})\right\}\right) \tag{4.14}$$

となる.

A, $\boldsymbol{y}$, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ が同時に発生する分布は,

$$\Pr(A, \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta}) = \Pr(A|\boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}) \Pr(\boldsymbol{y}|\boldsymbol{\pi}, \boldsymbol{\theta}) \Pr(\boldsymbol{\pi}|\boldsymbol{\alpha}) \Pr(\boldsymbol{\theta}|\boldsymbol{\beta}) \tag{4.15}$$

よって, A, $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ が与えられたとき,

$$\begin{aligned}
\Pr(\boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}|A, \boldsymbol{\alpha}, \boldsymbol{\beta}) =& \frac{\Pr(A, \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta})}{\Pr(A)} \\
\propto& \Pr(A, \boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}|\boldsymbol{\alpha}, \boldsymbol{\beta}) \\
=& \Pr(A|\boldsymbol{y}, \boldsymbol{\pi}, \boldsymbol{\theta}) \Pr(\boldsymbol{y}|\boldsymbol{\pi}, \boldsymbol{\theta}) \Pr(\boldsymbol{\pi}|\boldsymbol{\alpha}) \Pr(\boldsymbol{\theta}|\boldsymbol{\beta}) \\
=& \prod_{r=1}^{C} \exp\left(\ln \pi_r \sum_{i=1}^{N} y_{ir}\right) \\
& \times \prod_{r=1}^{C}\prod_{j=1}^{N} \exp\left(\sum_{i=1}^{N} y_{ir}\left\{A_{ij}\ln\theta_{rj} + (1-A_{ij})\ln(1-\theta_{rj})\right\}\right) \\
& \times \frac{1}{Z_1}\prod_{r=1}^{C} \pi_r^{\alpha_r - 1} \times \frac{1}{Z_2}\prod_{r=1}^{C}\prod_{j=1}^{N} \theta_{rj}^{\beta_{rj}^{(1)}-1}(1-\theta_{rj})^{\beta_{rj}^{(2)}-1} \\
\propto& \prod_{r=1}^{C} \exp\left(\left\{\alpha_r - 1 + \sum_{i=1}^{N} y_{ir}\right\}\ln \pi_r\right) \\
& \times \prod_{r=1}^{C}\prod_{j=1}^{N} \exp\left(\left\{\beta_{rj}^{(1)} - 1 + \sum_{i=1}^{N} A_{ij} y_{ir}\right\}\ln\theta_{rj}\right) \\
& \times \prod_{r=1}^{C}\prod_{j=1}^{N} \exp\left(\left\{\beta_{rj}^{(2)} - 1 + \sum_{i=1}^{N} (1-A_{ij}) y_{ir}\right\}\ln(1-\theta_{rj})\right)
\end{aligned} \tag{4.16}$$

この分布から $\boldsymbol{y}$, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ を直接サンプリングすることは難しいため, 代わりに以下のような繰り返しを行いサンプリングする.

A が与えられた時, $\boldsymbol{y}$ を前述した条件を満たすように初期化し, 次の (1), (2) を繰り返す.

(1)　$\boldsymbol{y} = \{y_{ir}\}$ をサンプリングする.

$\boldsymbol{\pi}$, $\boldsymbol{\theta}$ を固定したとき, $\boldsymbol{y}$ を生成する条件付き確率は,

$$\begin{aligned}\Pr(\boldsymbol{y}|\boldsymbol{\pi},\boldsymbol{\theta},A,\boldsymbol{\alpha},\boldsymbol{\beta}) =& \frac{\Pr(A,\boldsymbol{y}|\boldsymbol{\pi},\boldsymbol{\theta})}{\Pr(A)} \\ \propto& \prod_{r=1}^{C}\left[\exp\left(\ln\pi_r\sum_{i=1}^{N}y_{ir}\right)\right. \\ & \left.\prod_{j=1}^{N}\exp\left(\sum_{i=1}^{N}y_{ir}\left\{A_{ij}\ln\theta_{rj}+(1-A_{ij})\ln(1-\theta_{rj})\right\}\right)\right] \\ =& \prod_{r=1}^{C}\prod_{i=1}^{N}\exp\left(y_{ir}\left\{\ln\pi_r+\sum_{j=1}^{N}A_{ij}\ln\theta_{rj}+\sum_{j=1}^{N}(1-A_{ij})\ln(1-\theta_{rj})\right\}\right) \end{aligned} \tag{4.17}$$

各 i ごとに y_{ir} は多項分布で独立に発生できる.

(2)　$\boldsymbol{\pi}$, $\boldsymbol{\theta}$ をサンプリングする.

$\boldsymbol{y}$ を固定したとき, $\boldsymbol{\pi}$, $\boldsymbol{\theta}$ を生成する条件付き確率は,

$$\begin{aligned}\Pr(\boldsymbol{\pi},\boldsymbol{\theta}|A,\boldsymbol{y},\boldsymbol{\alpha},\boldsymbol{\beta}) =& \prod_{r=1}^{C}\exp\left(\left\{\alpha_r-1+\sum_{i=1}^{N}y_{ir}\right\}\ln\pi_r\right) \\ & \times\prod_{r=1}^{C}\prod_{j=1}^{N}\exp\left(\left\{\beta_{rj}^{(1)}-1+\sum_{i=1}^{N}A_{ij}y_{ir}\right\}\ln\theta_{rj}\right) \\ & \times\prod_{r=1}^{C}\prod_{j=1}^{N}\exp\left(\left\{\beta_{rj}^{(2)}-1+\sum_{i=1}^{N}(1-A_{ij})y_{ir}\right\}\ln(1-\theta_{rj})\right) \\ =& \mathrm{Dirichlet}(\boldsymbol{\pi}|\hat{\boldsymbol{\alpha}})\mathrm{Beta}(\boldsymbol{\theta}|\hat{\boldsymbol{\beta}^{(1)}},\hat{\boldsymbol{\beta}^{(2)}}) \end{aligned} \tag{4.18}$$

であるから, $\boldsymbol{y}$ は $\hat{\boldsymbol{\alpha}}$ をパラメータとするディリクレ分布と, $\hat{\boldsymbol{\beta}^{(1)}}$, $\hat{\boldsymbol{\beta}^{(2)}}$ をパラメータとするベータ分布から生成することができる. ここで, $\hat{\boldsymbol{\alpha}}$, $\hat{\boldsymbol{\beta}^{(1)}}$, $\hat{\boldsymbol{\beta}^{(2)}}$ は,

$$\hat{\alpha_r} = \alpha_r + \sum_{i=1}^{N}y_{ir}\ \ (r=1,2,\cdots,C) \tag{4.19}$$

$$\hat{\beta_{rj}^{(1)}} = \beta_{rj}^{(1)} + \sum_{i=1}^{N}A_{ij}y_{ir}\ \ (r=1,2,\cdots,C;j=1,2,\cdots,N) \tag{4.20}$$

$$\hat{\beta_{rj}^{(2)}} = \beta_{rj}^{(2)} + \sum_{i=1}^{N}(1-A_{ij})y_{ir}\ \ (r=1,2,\cdots,C;j=1,2,\cdots,N) \tag{4.21}$$

と表される.

(1)(2) を繰り返してサンプリングする際, はじめの何回かはバーン・イン区間として捨て, それ以降のサンプリングされたデータを数回おきに記録する. K 回記録を行ったとすると,

$$\{\boldsymbol{y}^{(k)},\boldsymbol{\pi}^{(k)},\boldsymbol{\theta}^{(k)};k=1,2,\cdots,K\} \tag{4.22}$$

という K 組ずつのデータが得られる. これらが事後分布 $\Pr(\boldsymbol{y},\boldsymbol{\pi},\boldsymbol{\theta}|A,\boldsymbol{\alpha},\boldsymbol{\beta})$ から生成されたデータであるとする. この結果から, $y_{ir}=1$, すなわち i 番目の頂点が r 番目のコミュニティに属する確率は,

$$\frac{1}{K}\sum_{k=1}^{K}y_{ir}^{(k)} \tag{4.23}$$

であると表せる. コミュニティ抽出の結果としては, 各ノードについてこの確率が最も高くなるコミュニティに所属すると判断する.

4.3　ハイパーパラメータ・コミュニティ数の推定

ギブス・サンプリングを行うにあたり, 事前分布のハイパーパラメータ及びコミュニティ数をはじめに与える必要があるが, これらを設定するために以下のような手法を用いる.

4.3.1　次数分布に基づくハイパーパラメータの設定

混合ベルヌーイ分布を用いたコミュニティ抽出手法において, ネットワークの持つ次数分布の情報から事前分布のハイパーパラメータを設定する手法 [16] が提案されている. 以下, その詳細を記す.

$\boldsymbol{\theta} = \{\theta_{rj}\}$ の事前分布であるベータ分布は,

$$\mathrm{Beta}(x; a, b) = \frac{1}{\mathrm{B}(a, b)} x^{a-1}(1-x)^{b-1} \tag{4.24}$$

と表せる (B はベータ関数). また, その最頻値は,

$$\frac{a-1}{a+b-2} \ \ (a, b > 1) \tag{4.25}$$

であり, 平均値は

$$\frac{a}{a+b} \tag{4.26}$$

となる.

ここで, 事前分布に対してネットワークの次数分布の情報を取り入れることを考える. ネットワークの次数の分布の種類として, ランダムネットワークで見られるような釣鐘型, スケールフリーネットワークで見られるようなロングテール型がある. また, コミュニティ抽出を行う前では, 各ノードからノード j への接続確率は k_j/N (ここで, k_j はノード j の次数) であると予想される. 釣鐘型の形を取り, なおかつ最頻値が k_j/N となるようなベータ分布は,

$$\mathrm{Beta}\left(\theta_{rj}; 1 + \frac{k_j}{C}, 1 + \frac{N-k_j}{C}\right) \tag{4.27}$$

となる. ロングテール型の場合はベータ分布の最頻値が求まらないため, 次のように平均値が k_j/N となる分布を作る.

$$\mathrm{Beta}\left(\theta_{rj}; 1, \frac{N-k_j}{k_j}\right) \tag{4.28}$$

これらネットワークの既知の情報を取り入れた事前分布を使い, 学習を行う.

4.3.2　情報量規準を用いたモデル選択

本研究では, ギブス・サンプリングによって推測された確率モデルの選択を, 情報量規準 WAIC(Widely Applicative Information Criterion)[17] を用いて行う. WAIC は, 推定した分布 (予測分布) と真の分布の間の Kullback-Leibler 情報量 (汎化誤差) の近似値を表している. WAIC が小さくなるモデルほど, より真の情報源に近く, よいモデルであるといえる. WAIC は, 予測誤差 T と, 汎関数分散 V, サンプル数 n を用いて,

$$\mathrm{WAIC} = T + \frac{V}{n} \tag{4.29}$$

と表される. ここで, n 個のサンプル x_i $(i = 1, 2, \cdots, n)$ があり, 確率モデルを $p(x|w)$ (w はパラメータ), 予測分布を $p^*(x)$ としたとき,

$$T = -\frac{1}{n}\sum_{i=1}^{n} \ln p^*(x_i) \tag{4.30}$$

$$V = \sum_{i=1}^{n} \mathbb{E}_w\left[(\ln p(x_i|w))^2\right] - \mathbb{E}_w\left[\ln(p(x_i|w))\right]^2 \tag{4.31}$$

である. $\mathbb{E}_w[]$ は, パラメータ w の事後分布についての期待値を表している.

さて, 上述したギブス・サンプリングから得られたサンプルから WAIC を導出するには次のようにする. ギブス・サンプリングによって k 番目にサンプリングされたパラメータの組を, $w_k = (\boldsymbol{\pi}^{(k)}, \boldsymbol{\theta}^{(k)})$ とすると, K 組の確率モデルのサンプルは,

$$\Pr(A_l|w_k) = \sum_{r=1}^{C} \pi_r^{(k)} \prod_{j=1}^{N} \left(\theta_{rj}^{(k)}\right)^{A_{lj}} (1-\theta_{rj}^{(k)})^{1-A_{lj}} \quad (l = 1, 2, \cdots, N) \tag{4.32}$$

である.

よって, この結果から推定される予測分布 $p^*(A_l)$ は,

$$\begin{aligned} p^*(A_l) &= \mathbb{E}_w\left[\Pr(A_l|w_k)\right] \\ &= \frac{1}{K}\sum_{k=1}^{K} \Pr(A_l|w_k) \\ &= \frac{1}{K}\sum_{k=1}^{K}\sum_{r=1}^{C} \pi_r^{(k)} \prod_{j=1}^{N} \left(\theta_{rj}^{(k)}\right)^{A_{lj}} (1-\theta_{rj}^{(k)})^{1-A_{lj}} \end{aligned} \tag{4.33}$$

である.

ゆえに, WAIC は,

$$\begin{aligned} \mathrm{WAIC} = &-\frac{1}{N}\sum_{l=1}^{N} \ln p^*(A_l) \\ &+ \frac{1}{N}\sum_{l=1}^{N} \left\{\mathbb{E}_w\left[(\ln \Pr(A_l|w_k))^2\right] - \mathbb{E}_w\left[\ln \Pr(A_l|w_k)\right]^2\right\} \\ = &-\frac{1}{N}\sum_{l=1}^{N} \ln\left(\frac{1}{K}\sum_{k=1}^{K}\sum_{r=1}^{C} \pi_r^{(k)} \prod_{j=1}^{N} \left(\theta_{rj}^{(k)}\right)^{A_{lj}} (1-\theta_{rj}^{(k)})^{1-A_{lj}}\right) \\ &+ \frac{1}{N}\sum_{l=1}^{N}\left[\frac{1}{K}\sum_{k=1}^{K}\left\{\ln\left(\sum_{r=1}^{C} \pi_r^{(k)} \prod_{j=1}^{N} \left(\theta_{rj}^{(k)}\right)^{A_{lj}} (1-\theta_{rj}^{(k)})^{1-A_{lj}}\right)\right\}^2\right. \\ &\left. + \left\{\frac{1}{K}\sum_{k=1}^{K} \ln\left(\sum_{r=1}^{C} \pi_r^{(k)} \prod_{j=1}^{N} \left(\theta_{rj}^{(k)}\right)^{A_{lj}} (1-\theta_{rj}^{(k)})^{1-A_{lj}}\right)\right\}^2\right] \end{aligned} \tag{4.34}$$

と表すことができる.

表 4.1: chromium-browser の説明文のベクトル表現

version	browser	way	internet	web	open-source	users	chromium	project	chrome
1	2	1	1	2	2	1	1	1	1

4.4　パッケージの説明文を用いたコミュニティの性質評価

パッケージ依存関係ネットワークから抽出されたコミュニティ構造の性質を評価するにあたっては, 各パッケージに付属する説明文からパッケージ間の類似度を測ることで, 同一コミュニティに類似するパッケージが集まるかどうかを調べる.

パッケージの説明文としては, 開発者によって提供されている英語の説明文を用いた. 例として, chromium-browser のパッケージであれば,

```
Description: Chromium web browser, open-source version of Chrome
 An open-source browser project that aims to build a safer, faster, and more
 stable way for all Internet users to experience the web.
```

といった説明文が与えられている.

この説明文を用いて類似度を測るにあたり, はじめに各説明文を名詞の出現回数を含んだベクトル表現に変換する. 例えば, 上で示した chromium-browser のパッケージの説明文について, 出現した名詞とその出現回数をカウントすると, 表 4.1 のようになる.

このようにして, 全ての説明文について各単語の出現数を計測し, 各説明文中に出現した単語の和集合 $\boldsymbol{t} = \{t_j\}(j = 1, 2, \cdots, M; M$ は単語の総種類数) を生成する. 各説明文をベクトル $\boldsymbol{v}^{(i)}(i = 1, 2, \cdots, N)$ として表し, 単語 t_j の出現数を $v_j^{(i)}$ に記録する. また, $\boldsymbol{v}^{(k)}$ 全体の集合を V とする (N は全説明文の個数).

さらに, 各ベクトル $\boldsymbol{v}^{(i)}(i = 1, 2, \cdots, N)$ について, 各要素を tf-idf によって重み付けを行う. tf-idf は, 文書中に出現する各単語について, その文書中に頻出する・他と比べ特にその文書にのみ存在するという二つの観点から重み付けを行う指標である. ベクトル $\boldsymbol{v}^{(i)}$ の要素 $v_j^{(i)}$ について, tf-idf の値を計算するには, 次のようにする.

$$\text{tf-idf}_{i,j} = \text{tf}_{i,j} \cdot \text{idf}_j \tag{4.35}$$

$$\text{tf}_{i,j} = \frac{\boldsymbol{v}_j^{(i)}}{\sum_k \boldsymbol{v}_j^{(k)}} \tag{4.36}$$

$$\text{idf}_j = \ln \frac{|\boldsymbol{V}|}{|\{\boldsymbol{v}' | \boldsymbol{v}'_j \neq 0\}|} \tag{4.37}$$

ここで, $\text{tf}_{i,j}$ は, 全てのベクトル集合 $\boldsymbol{V}$ に含まれる単語 t_j の個数のうち, 何割が $\boldsymbol{v}^i$ に含まれているかを表し, idf_i は, 単語 t_j を一つでも含むベクトルの個数とすべてのベクトル集合 $\boldsymbol{V}$ の個数の比の対数を表します.

最終的に, 各パッケージの説明文の類似度は, 各説明文に対応する tf-idf によって重み付けを行ったベクトル間の cos 類似度を測って求める. すなわち, パッケージ A の説明文について tf-idf で重み付けを行ったベクトルを $A_{\text{tf-idf}}$, パッケージ B の説明文について tf-idf で重み付けを行ったベクトルを $B_{\text{tf-idf}}$ としたとき, パッケージ A とパッケージ B の類似度は

$$cos(A_{\text{tf-idf}}, B_{\text{tf-idf}}) = \frac{A_{\text{tf-idf}}}{|A_{\text{tf-idf}}|} \cdot \frac{B_{\text{tf-idf}}}{|B_{\text{tf-idf}}|} \tag{4.38}$$

と表す.

本研究では, 各コミュニティに含まれるパッケージの類似度を測る基準として, コミュニティに含まれるパッケージの総当たりのペアについて式 (4.38) を計算し, その平均をとった値を用いる. 比較する対象としては, 異なる二つのコミュニティ間で片方ずつからパッケージを選んで作る総当たりのペアについて式 (4.38) を計算し, その平均をとった値を使う.

また, それぞれのコミュニティに含まれるパッケージ全体の説明文を対象にして, 名詞の出現回数ベクトルを作成し, tf-idf による重み付けを行うことで, 各コミュニティの説明文が持つ特徴的な単語を調べた.

第 5 章　実験

5.1　人工データによる予備実験

前述した手法の有効性を確かめるために, 情報源が既知である人工データを使った予備実験を行った.

5.1.1　実験に用いたデータ

人工データのノード数を N, コミュニティ数を C とし, 以下のような手順でデータセットの作成を行った.

1. 各ノード i $(1 \leq i \leq N)$ について, 所属コミュニティ z_i $(1 \leq z_i \leq C)$ を次の多項分布から決定する.

$$z_i \sim \mathrm{Multi}(\boldsymbol{\phi}) \tag{5.1}$$

   ここで, パラメータ $\boldsymbol{\phi}$ は, $\boldsymbol{\phi} = \{1/C\}$ と設定している.
2. 各ノード i について, 他ノード j $(1 \leq j \leq N)$ への接続確率 ρ_{ij} を, 次のように決定する.

$$\rho_{ij} = \begin{cases} \mu & \text{if } z_i = z_j \\ 1-\mu & \text{otherwise} \end{cases} \tag{5.2}$$

   ここで, μ $(0 < \mu < 1)$ は値をあらかじめ決めておく定数である.
3. 人工データの隣接行列 A^{test} の各要素の値を, 次のベルヌーイ分布から決定する.

$$A_{ij}^{test} \sim \mathrm{Bernoulli}(\rho_{ij}) \tag{5.3}$$

2. において, μ の値を変えることにより様々な特徴を持つネットワークを生成できる. $\mu > 0.5$ であれば, 同一コミュニティ間の接続が密になり, それ以外は疎になる assortative な特徴を持つネットワークになる. 逆に $\mu < 0.5$ であれば, 二部グラフのような異なるコミュニティ間の接続が密になる disassortative な特徴を持つネットワークになる. $\mu = 0.5$ の場合, コミュニティ構造を持たない (設定したコミュニティ構造が意味をなさない) ランダムネットワークになる.

5.1.2　実験手順

以下のような手順で実験を行った.

1. $N = 100$, $C = 4$ として, $\mu = 0.8$, $\mu = 0.2$ の二通りの人工データを生成する.
2. 二つの人工データに対し, ハイパーパラメータ $\boldsymbol{\alpha}$, $\boldsymbol{\beta}^{(1)}$, $\boldsymbol{\beta}^{(2)}$ を一定値に固定し, コミュニティ数だけを変動させる条件でそれぞれコミュニティ抽出を行い, 与えたコミュニティ数ごとの WAIC の変化を観察する.

表 5.1: 人工データの情報

	ノード数	エッジ数
$\mu = 0.8$	100	3643
$\mu = 0.2$	100	6339

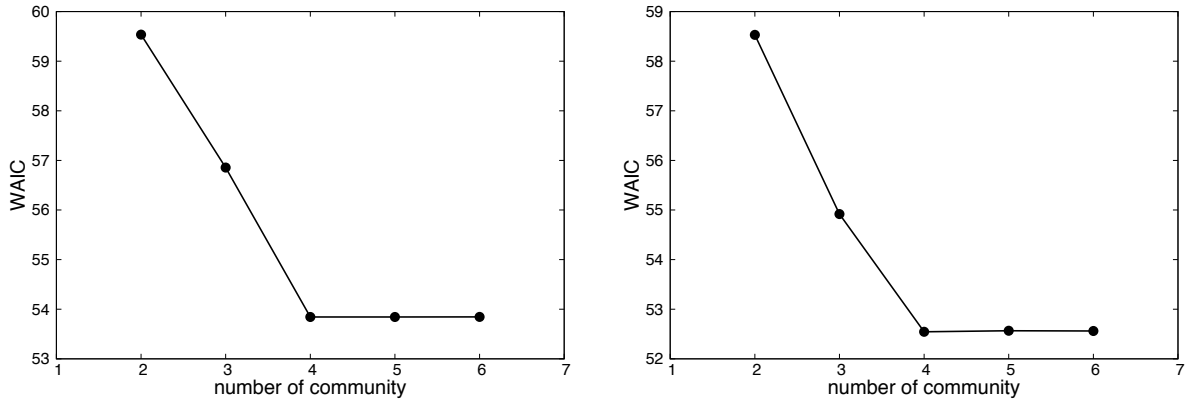

図 5.1: $\mu = 0.8$(左), $\mu = 0.2$(右) として生成したデータに対する WAIC の変化

5.1.3　実験結果

5.1.3.1　生成されたデータ

上述した手法で実際に生成された人工データの情報を, 表 5.1 に示す.

5.1.3.2　WAIC の値の変化

$\boldsymbol{\alpha} = \{1.0\}$, $\boldsymbol{\beta}^{(1)} = \{1.0\}$, $\boldsymbol{\beta}^{(2)} = \{1.0\}$ として, コミュニティ数を 2 から 6 まで変化させたときの, WAIC の変化を図 5.1 に示す. $\mu = 0.8, 0.2$ どちらの場合も $C = 4$ で WAIC が僅差で最小になった.

5.1.3.3　抽出されたコミュニティ構造

WAIC が最小となるコミュニティ数を与えたときに, コミュニティ抽出によって得られたコミュニティ構造を図 5.2 に示す. この図は, 縦軸を i 成分, 横軸を j 成分とし, 隣接行列 A_{ij}^{test} を分割されたコミュニティごとに並べ替えたものである. 行列の要素が 0 であるとき図中では白, 1 であるとき図中では黒で表現している. 以下に示す隣接行列の図も全て同様である. $\mu = 0.8, 0.2$ いずれの場合も $C = 5, 6$ を与えた時に空のコミュニティが抽出され, それらを除いた実質のコミュニティ数は 4 であった.

また, μ の値に関わらず, WAIC が最小となるコミュニティ数で抽出されたコミュニティは, 既知のコミュニティ構造である $\boldsymbol{z} = \{z_i\}$ と一致した.

5.1.4　予備実験に対する考察

図 5.1 から, $\mu = 0.8, 0.2$ いずれの場合であっても, $C = 2, 3, 4$ と真のコミュニティ数に近づくにつれ WAIC は小さくなっていき, 以降はほぼ変わらないという変化が見て取れる. これは, $C = 5, 6$ では抽出されたコミュ

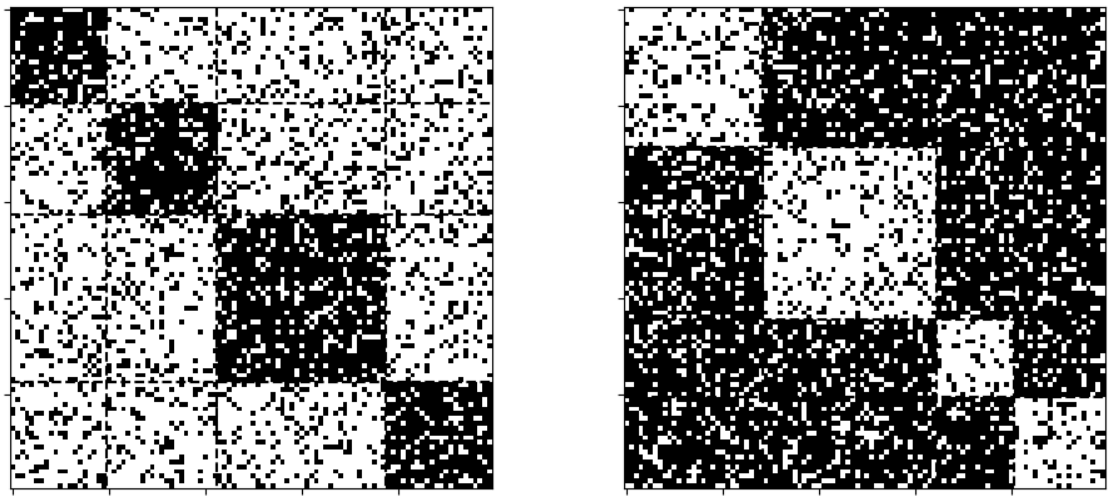

図 5.2: $\mu = 0.8$(左), $\mu = 0.2$(右) として生成したデータから抽出されたコミュニティ

ニティに空のコミュニティが含まれており, それらを除いたコミュニティ数は 4 から変わらないということが原因であると考えられる. こうした変化から, $C = 5, 6$ は過剰な分割数であり, 真のコミュニティ数は 4 であると推測できる. これは, 既知のコミュニティ数と一致する結果である.

また, コミュニティ抽出の際に, 推測されたコミュニティ数 4 を与えることで, 既知のコミュニティ構造 $\boldsymbol{z}$ を正しく推測でき, 手法の有効性を確かめらることができた.

5.2　パッケージ依存関係ネットワークに対するコミュニティ抽出実験

modularity 最大化手法の一つである Louvain 法 [18] と, 4 章で述べた理論 (提案手法) の二通りの方法で, chromium-browser, fcitx-mozc の二つのパッケージの依存関係ネットワークからコミュニティ抽出を行った. また, コミュニティ抽出結果を評価するにあたって, 依存関係とは別の情報として説明文を用いた評価を行った.

5.2.1　chromium-browser の依存関係ネットワークに対する実験

5.2.1.1　データセットの詳細

本実験で使用したデータセットは, ノード数が 353, エッジ数が 1229 の無向ネットワークであった.

また, 次数分布は図 5.3 のように, おおむねロングテール型を示した.

5.2.1.2　提案手法の実験の条件

$\boldsymbol{\beta^{(1)}}$ と $\boldsymbol{\beta^{(2)}}$ はロングテール型となるように値を設定したうえで, $\boldsymbol{\alpha}$ を $1.0, 3.0, 5.0, 7.0, 9.0, 11.0$, コミュニティ数 C を $6 \sim 10$ と変化させる条件のもとで提案手法のコミュニティ抽出を行い, WAIC の値の計測を行った.

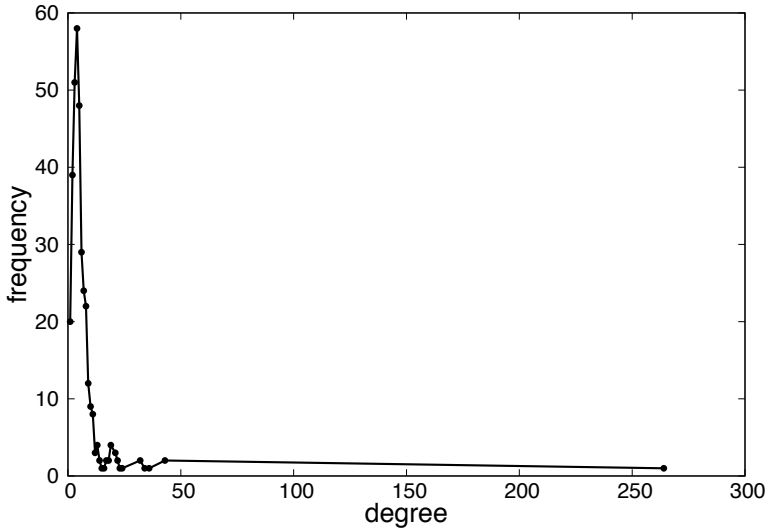

図 5.3: chromium-browser の依存関係ネットワークの次数分布

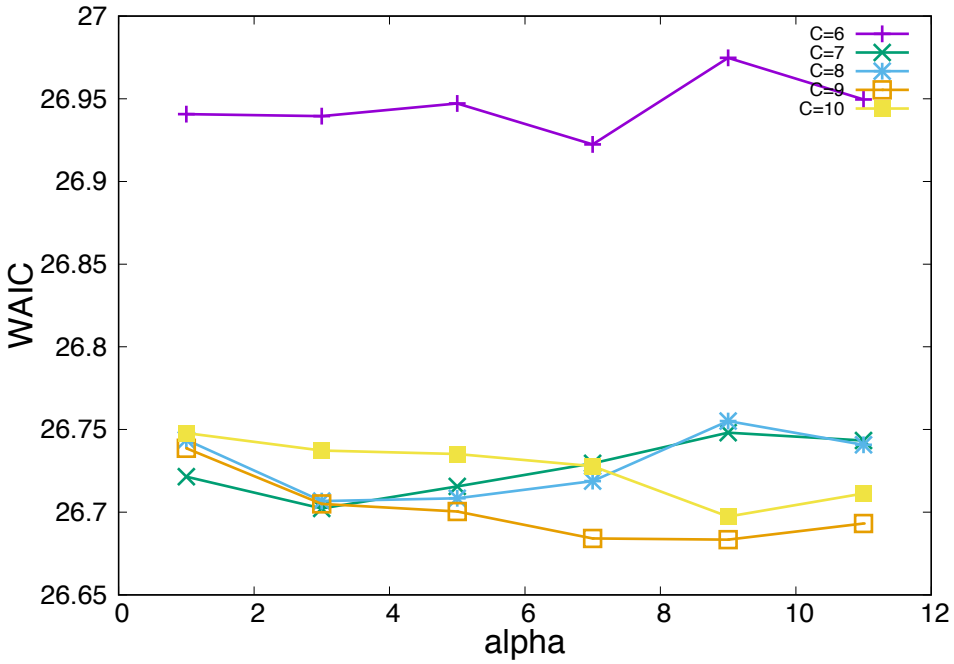

図 5.4: chromium-browser パッケージ依存関係ネットワークの WAIC の値の変化

5.2.1.3　WAIC の値の変化

提案手法において, ハイパーパラメータ $\boldsymbol{\alpha}$ 及びコミュニティ数を変えた時の WAIC の値の変化を図 5.4 に示す. $\boldsymbol{\alpha} = 9.0$, コミュニティ数 $C = 9$ の時に WAIC は最小になった.

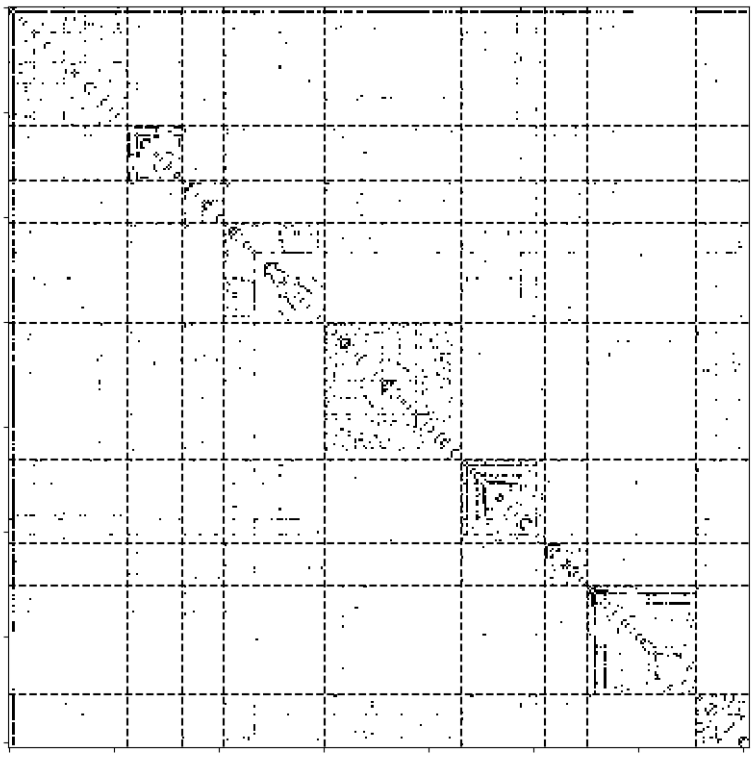

図 5.5: chromium-browser パッケージ依存関係ネットワークから抽出されたコミュニティ (Louvain 法)

5.2.1.4　抽出されたコミュニティ

Louvain 法で抽出されたコミュニティを, 図 5.5 に示す. また, 提案手法で WAIC が最小になった $\boldsymbol{\alpha} = 9.0$, $C = 9$ のときに抽出されたコミュニティを図 5.6 に示す. また, 提案手法の実験において, 各コミュニティに含まれていたパッケージの一覧を以下に示す. なお, 各コミュニティ番号は, 図 5.6 で示したコミュニティ番号と対応している.

- コミュニティ 1
  libc6, dpkg, multiarch-support, zlib1g, libxcb1, libx11-6, libglib2.0-0, libxext6, x11-xserver-utils, x11-utils, libgtk-3-0, libcairo2, libgl1-mesa-dri, chromium-browser, libegl1-mesa, libgl1-mesa-glx
- コミュニティ 2
  perl-base, libnet-dbus-perl, libxml-twig-perl, perl, libio-html-perl, libio-socket-ssl-perl, libnet-libidn-perl, netbase, libnet-ssleay-perl, libx11-protocol-perl, libfile-mimeinfo-perl, libhtml-tagset-perl, libfile-desktopentry-perl, libfile-basedir-perl, liburi-perl, libhtml-form-perl,

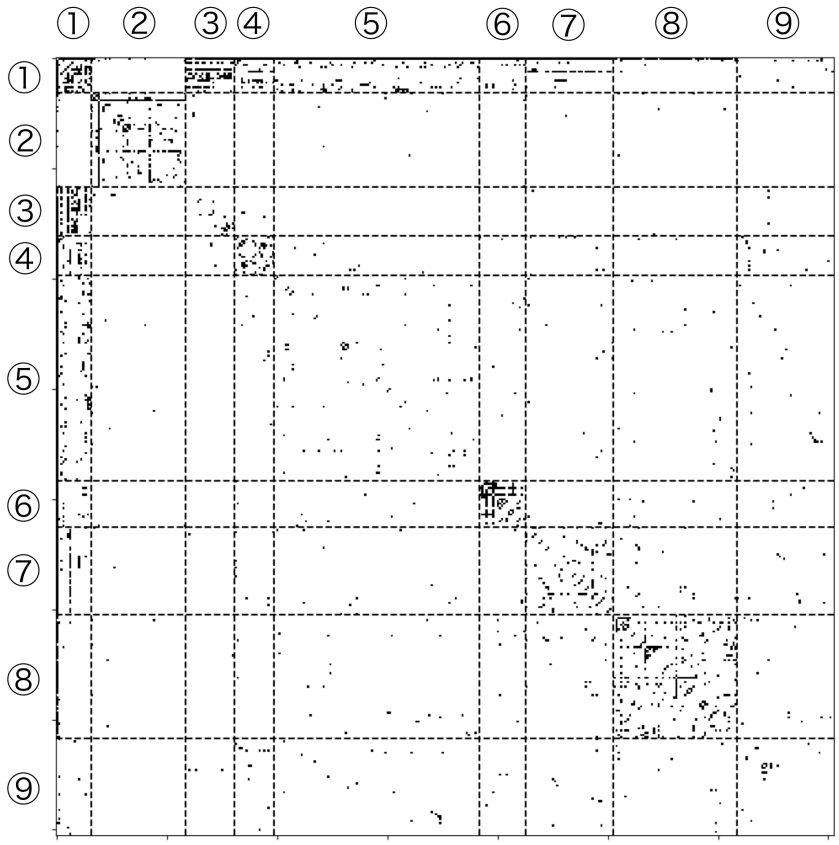

図 5.6: chromium-browser パッケージ依存関係ネットワークから抽出されたコミュニティ (提案手法)

libhttp-message-perl, libhtml-parser-perl, libauthen-sasl-perl, libwww-robotrules-perl, libtie-ixhash-perl, libhttp-date-perl, liblwp-mediatypes-perl, libencode-locale-perl, libxml-parser-perl, libxml-xpathengine-perl, libwww-perl, libtimedate-perl, ca-certificates, libipc-system-simple-perl, libnet-http-perl, libnet-smtp-ssl-perl, libhttp-negotiate-perl, libhttp-cookies-perl, libfont-afm-perl, libfile-listing-perl, libhttp-daemon-perl, libhtml-format-perl, libhtml-tree-perl, liblwp-protocol-https-perl, libmailtools-perl, rename, perl-modules-5.22

- コミュニティ 3
  libxcomposite1, libxtst6, libxxf86dga1, xdg-utils, libxss1, libxfixes3, libxcursor1, libxinerama1, libxdamage1, libxi6, libxrandr2, libxft2, libxrender1, libxpm4, libxv1, libxmuu1, libice6, libxaw7, libxt6, libxmu6, libxxf86vm1, libsm6
- コミュニティ 4
  libxml2, liblzma5, fontconfig, libfontconfig1, libfreetype6, libtiff5, libgtk-3-bin, librsvg2-common, libpango-1.0-0, libpangoft2-1.0-0, libcairo-gobject2, libgdk-pixbuf2.0-0, libpangocairo-1.0-0, libgd3, libpng12-0, libgphoto2-6, libharfbuzz0b, librsvg2-2
- コミュニティ 5

libwayland-cursor0, libwayland-client0, libgpm2, libglapi-mesa, cpp-5, libk5crypto3, libkrb5support0, libdrm-amdgpu1, libdrm2, libjbig0, libgdbm3, libxcb-glx0, libxkbcommon0, libp11-kit0, libusb-0.1-4, libcap-ng0, libmnl0, libdrm-nouveau2, libthai0, libdatrie1, libidn11, libxcb-xfixes0, libexpat1, libxcb-dri3-0, libreadline6, libattr1, libelf1, libtasn1-6, libepoxy0, libwayland-egl1-mesa, libcups2, libgssapi-krb5-2, libcomerr2, libkrb5-3, libltdl7, libxcb-sync1, libexif12, libavahi-common3, libvpx3, libsensors4, libdrm-intel1, libtxc-dxtn-s2tc0, libdrm-radeon1, libpixman-1-0, libjpeg-turbo8, chromium-codecs-ffmpeg-extra, libmpfr4, libgmp10, libkeyutils1, libpciaccess0, libtext-iconv-perl, libdns-export162, libisc-export160, libedit2, libatomic1, libxcb-present0, libx11-xcb1, libwayland-server0, libxshmfence1, libgbm1, libxcb-dri2-0, openssl, libtext-charwidth-perl, liblz4-1, libsepol1, libustr-1.0-1, libxcb-render0, libgraphite2-3, libxau6, libssl1.0.0, libxtables11, libnss3, libisl15, libasound2, gawk, bash, liblocale-gettext-perl, libxdmcp6, libbsd0, libgnutls30, libnspr4, libnettle6, libieee1284-3, libxcb-shm0, libfontenc1, libatm1, libhogweed4, e2fslibs, insserv, libmpc3, libsigsegv2, libss2, libxcb-shape0

- コミュニティ 6

  libicu55, libmircommon7, libboost-system1.58.0, libgcc1, libcapnp-0.5.3, libmircore1, libstdc++6, libboost-filesystem1.58.0, apt-utils, apt, libapt-pkg5.0, libapt-inst2.0, gpgv, libmirprotobuf3, libprotobuf-lite9v5, libmirclient9, libllvm4.0, libbz2-1.0, gnupg, libproxy1v5, libperl5.22

- コミュニティ 7

  libdbus-1-3, libffi6, libcroco3, at-spi2-core, libatspi2.0-0, libpcre3, libnih-dbus1, libnih1, libusb-1.0-0, libpolkit-backend-1-0, libpolkit-gobject-1-0, acl, dconf-gsettings-backend, libatk1.0-0, libjson-glib-1.0-0, librest-0.7-0, libcolord2, libatk-bridge2.0-0, shared-mime-info, liblcms2-2, xdg-user-dirs, libsoup-gnome2.4-1, libsoup2.4-1, libcgmanager0, libcolorhug2, libgusb2, libsqlite3-0, policykit-1, libpolkit-agent-1-0, systemd-shim, colord, libgudev-1.0-0, libsane, glib-networking-services, glib-networking, libgphoto2-port12, cgmanager, dconf-service, libdconf1, libavahi-client3

- コミュニティ 8

  libapparmor1, libpam-modules, debconf, libselinux1, libpam-modules-bin, libaudit1, libpam0g, libdb5.3, libncurses5, libtinfo5, libudev1, libsystemd0, sysvinit-utils, init-system-helpers, util-linux, libblkid1, libfdisk1, libncursesw5, libuuid1, libmount1, lsb-base, libsmartcols1, libacl1, libdevmapper1.02.1, dmsetup, libseccomp2, coreutils, libcap2, systemd, libkmod2, libpam-systemd, mount, dbus, libgcrypt20, libgpg-error0, libcryptsetup4, adduser, libcap2-bin, tar, psmisc, procps, initscripts, libprocps4, libpam-runtime, isc-dhcp-client, iproute2, debianutils, libpam-cap, libsemanage1, sed, ifupdown, uuid-runtime, passwd, e2fsprogs, sysv-rc, udev

- コミュニティ 9

  libsane-common, cpp, xml-core, update-motd, fontconfig-config, libjpeg8, debconf-i18n, xkb-data, fonts-dejavu-core, ucf, libthai-data, humanity-icon-theme, adwaita-icon-theme, hicolor-icon-theme, x11-common, ubuntu-mono, ubuntu-keyring, readline-common, libgtk-3-common, libatk1.0-data, libavahi-common-data, libjson-glib-1.0-common, krb5-locales, isc-dhcp-common, gcc-5-base, chromium-browser-l10n, libaudit-common, colord-data, glib-networking-common, libsemanage-common, gcc-6-base, libgphoto2-l10n, sgml-base,

表 5.2: 各コミュニティに出現した単語 (かっこ内の数字は tf-idf)

	tf-idf の高い単語 (上位 10 単語)
1	x (0.47) extension (0.24) tool (0.24) opengl (0.16) server(0.16)
2	perl (0.35) html (0.26) http (0.25) module (0.21) net(0.18)
3	git (0.40) extension (0.35) x (0.30) www.x.org (0.20) xorg (0.20)
4	fonts (0.41) pango (0.35) layout (0.26) font (0.26) text (0.22)
5	kerberos (0.24) xcb (0.22) protocol (0.22) interface (0.20) library (0.17)
6	packages (0.28) c++ (0.26) mir (0.23) data (0.20) proto(0.19)
7	d-bus (0.21) policykit (0.17) api (0.16) authentication (0.15) processes (0.15)
8	linux (0.24) utilities (0.22) device (0.17) system (0.16) kernel (0.16)
9	catalog (0.28) files (0.28) xml (0.22) sgml (0.22) data (0.17)

表 5.3: コミュニティの持つパッケージの cos 類似度の平均 (太字は行ごとの最高値)

	1	2	3	4	5	6	7	8	9
1	0.044	0.014	**0.057**	0.025	0.031	0.018	0.014	0.015	0.018
2	0.014	**0.043**	0.019	0.0095	0.011	0.013	0.0094	0.0079	0.0096
3	0.057	0.019	**0.17**	0.017	0.027	0.011	0.0098	0.0073	0.016
4	0.025	0.0095	0.017	**0.065**	0.012	0.012	0.012	0.0080	0.017
5	**0.031**	0.011	0.027	0.012	0.029	0.014	0.013	0.012	0.014
6	0.018	0.013	0.011	0.012	0.014	**0.052**	0.012	0.013	0.016
7	0.014	0.0094	0.0098	0.012	0.013	0.012	**0.031**	0.014	0.016
8	0.015	0.0079	0.0073	0.0080	0.012	0.013	0.014	**0.026**	0.014
9	0.018	0.0096	0.016	0.017	0.014	0.016	0.016	0.014	**0.023**

libnss3-nssdb, libasound2-data, base-files, bash-completion, libtext-wrapi18n-perl, dash, gsettings-desktop-schemas, libx11-data, libgdk-pixbuf2.0-common, sensible-utils, libglib2.0-data

5.2.1.5　抽出されたコミュニティの評価

提案手法によって得られた各コミュニティに含まれるパッケージ全体の説明文の名詞の出現回数ベクトルから, tf-idf の高い単語を抽出した結果は, 表 5.2 のようになった.

また, 各コミュニティの持つパッケージ間の cos 類似度の平均をコミュニティのペアごとに測定した結果が表 5.3 である.

5.2.2　fcitx-mozc の依存関係ネットワークに対する実験

5.2.2.1　データセットの詳細

本実験で使用したデータセットは, ノード数が 416, エッジ数が, 1652 の無向ネットワークであった.

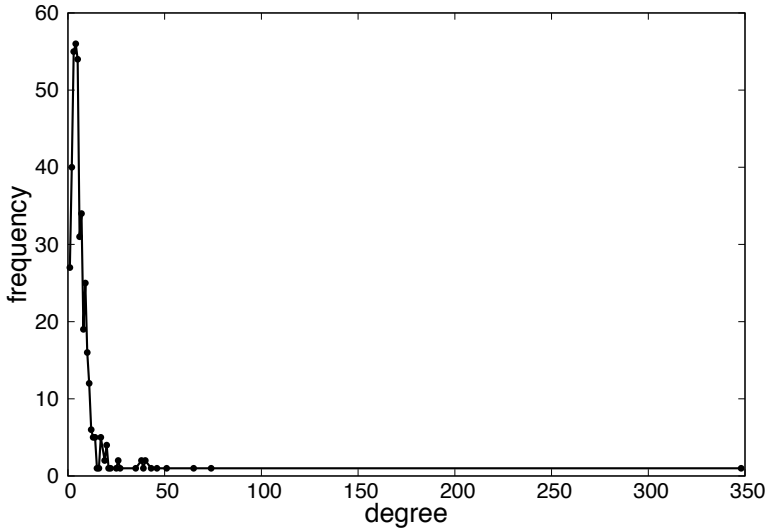

図 5.7: fcitx-mozc の依存関係ネットワークの次数分布

また, 次数分布は図 5.7 のようにおおむねロングテール型を示した.

5.2.2.2　提案手法の実験の条件

$\boldsymbol{\beta^{(1)}}$ と $\boldsymbol{\beta^{(2)}}$ はロングテール型となるように値を設定したうえで, $\boldsymbol{\alpha}$ を $1.0, 3.0, 5.0, 7.0, 9.0, 11.0$, コミュニティ数 C を $7 \sim 12$ と変化させる条件のもとでコミュニティ抽出を行い, WAIC の値の計測を行った.

5.2.2.3　WAIC の値の変化

提案手法において, ハイパーパラメータ $\boldsymbol{\alpha}$ 及びコミュニティ数を変えた時の WAIC の値の変化を図 5.8 に示す. $\boldsymbol{\alpha} = 9.0$, コミュニティ数 $C = 10$ の時に WAIC は最小になった.

5.2.2.4　抽出されたコミュニティ

Louvain 法で抽出されたコミュニティを, 図 5.9 に示す. また, 提案手法で WAIC が最小になった $\boldsymbol{\alpha} = 9.0$, $C = 10$ のときに抽出されたコミュニティを図 5.10 に示す. また, 提案手法で各コミュニティに含まれていたパッケージの一覧を以下に示す. なお, 各コミュニティ番号は, 図 5.10 で示したコミュニティの番号と対応している.

- コミュニティ 1

  multiarch-support, libc6, libxcb1, libstdc++6, libgcc1, zlib1g, libglib2.0-0, libx11-6, libcairo2, libxml2, libgtk-3-0, libjpeg8, libqtgui4, libqt5gui5, libegl1-mesa, libgtk2.0-0, libgl1-mesa-glx, libwebkit2gtk-4.0-37, libwebkit2gtk-4.0-37-gtk2, gstreamer1.0-plugins-good
- コミュニティ 2

  libqt5core5a, fcitx-frontend-qt5, libqt5dbus5, libfcitx-qt5-1, libqt5svg5, libqt5widgets5, libqt5network5
- コミュニティ 3

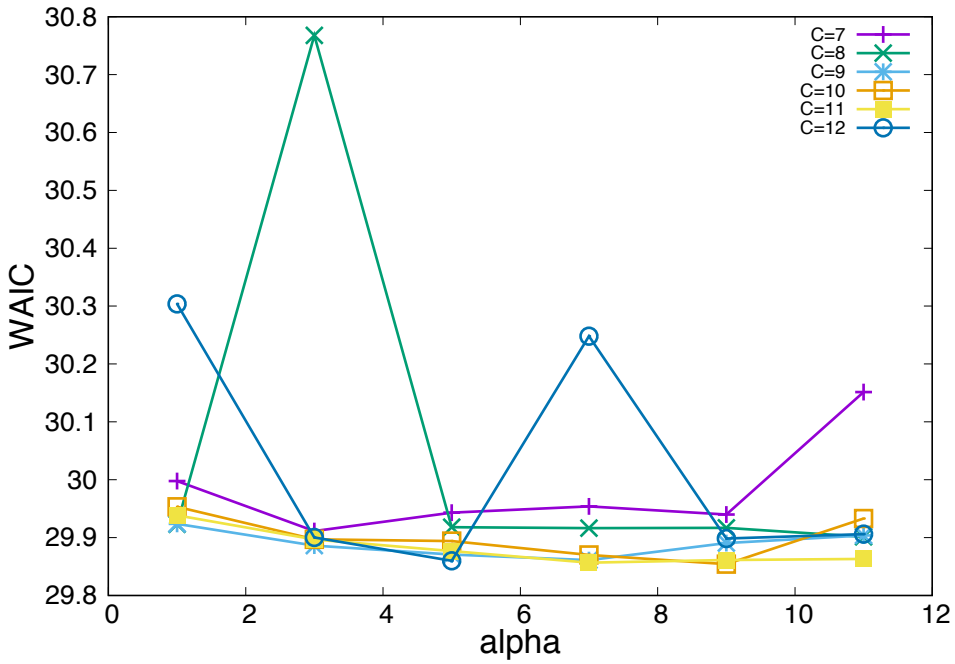

図 5.8: fcitx-mozc パッケージ依存関係ネットワークの WAIC の値の変化

libselinux1, dpkg, coreutils, libacl1, libncurses5, libtinfo5, adduser, udev, perl-base, libncursesw5, libcryptsetup4, libdevmapper1.02.1, libuuid1, libgpg-error0, lsb-base, libsemanage1, libaudit1, util-linux, libmount1, libsystemd0, sysvinit-utils, libudev1, libfdisk1, libpam0g, libblkid1, libsmartcols1, debconf, libpam-systemd, dbus, procps, libprocps4, initscripts, psmisc, libpam-runtime, libpam-modules, libpam-modules-bin, init-system-helpers, libcap2-bin, libpam-cap, libcap2, libkmod2, liblzma5, tar, passwd, dmsetup, libapparmor1, systemd, uuid-runtime, mount, debianutils, sysv-rc, e2fsprogs

- コミュニティ 4

  libsqlite3-0, libicu55, libgstreamer-plugins-base1.0-0, libgstreamer1.0-0, gstreamer1.0-plugins-base, libjavascriptcoregtk-4.0-18, libwayland-client0, libharfbuzz0b, libgcrypt20, libhyphen0, libharfbuzz-icu0, libwebp5, libxslt1.1, libwayland-egl1-mesa, libsoup2.4-1, libenchant1c2a, libwayland-server0, libsecret-1-0, libgeoclue0, libgstreamer-plugins-good1.0-0

- コミュニティ 5

  libwacom-bin, libwacom2, dconf-gsettings-backend, libsane, libgphoto2-6, libgphoto2-port12, libusb-1.0-0, libavahi-client3, libatk-bridge2.0-0, libffi6, xdg-user-dirs, libpcre3, libdconf1, systemd-shim, policykit-1, libpolkit-gobject-1-0, libpolkit-backend-1-0, libpolkit-agent-1-0, glib-networking-services, glib-networking, dconf-service, libgtk-3-bin, libjson-glib-1.0-0, librest-0.7-0, libcolord2, libsoup-gnome2.4-1, at-spi2-core, enchant, libgudev-1.0-0, libdbus-glib-1-2, libinput-bin, libgusb2, librsvg2-common, liblcms2-2, colord, libcroco3, libcolorhug2

- コミュニティ 6

  libxfixes3, fcitx-module-x11, libxinerama1, libpangocairo-1.0-0, libxrender1, libpango-1.0-0,

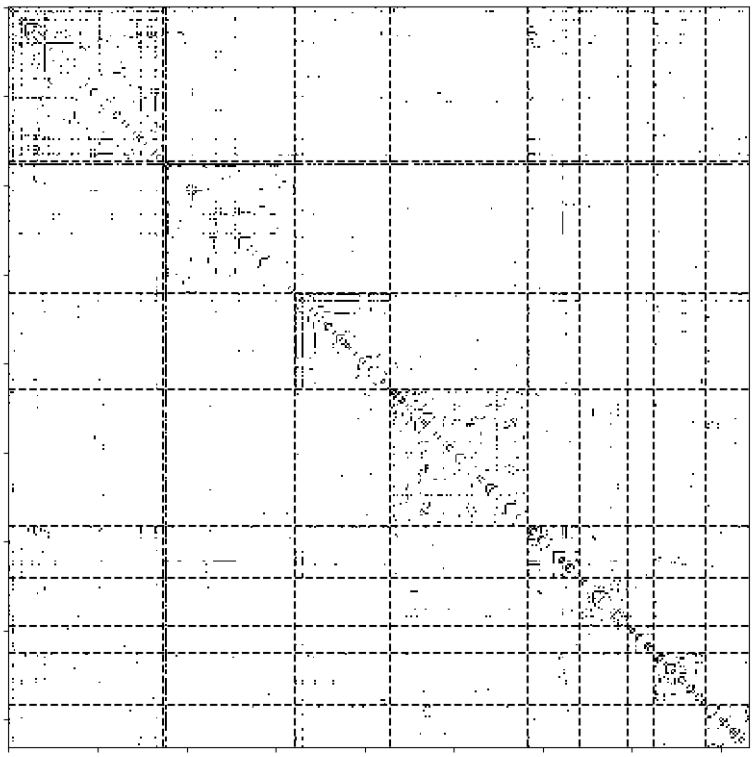

図 5.9: fcitx-mozc パッケージ依存関係ネットワークから抽出されたコミュニティ (Louvain 法)

libxcursor1, libtiff5, notification-daemon, libgdk-pixbuf2.0-0, libatk1.0-0, libatspi2.0-0, libgd3, libnotify4, libxext6, libxi6, libpng12-0, libcups2, libfreetype6, libfontconfig1, fontconfig, shared-mime-info, libcairo-gobject2, libxrandr2, libpangoft2-1.0-0, libxdamage1, libxcomposite1, libxv1, fcitx-ui-classic, zenity, gstreamer1.0-x, librsvg2-2, libxtst6

- コミュニティ 7

  sed, libattr1, libelf1, gsettings-desktop-schemas, libxkbfile1, libltdl7, ubuntu-keyring, gpgv, fcitx-config-common, qtcore4-l10n, libgraphite2-3, libieee1284-3, libavahi-common3, libsane-common, acl, libtheora0, libogg0, libkrb5support0, fontconfig-config, fonts-dejavu-core, ucf, libtext-wrapi18n-perl, libtext-charwidth-perl, libgphoto2-l10n, libexif12, libbz2-1.0, libpcre16-3, qttranslations5-l10n, iso-codes, liborc-0.4-0, libgpm2, libseccomp2, gettext-base, libasprintf0v5, libtext-iconv-perl, libtxc-dxtn-s2tc0, libdv4, libvorbisenc2, libvorbis0a, x11-common, libsemanage-common, libustr-1.0-1, libsepol1, libglib2.0-data, libhogweed4, libnettle6, libgmp10, libidn11, libdrm-amdgpu1, libdrm-radeon1, libsensors4, libdrm-intel1, libdrm-nouveau2, ca-certificates, openssl, liblocale-gettext-perl, glib-networking-common,

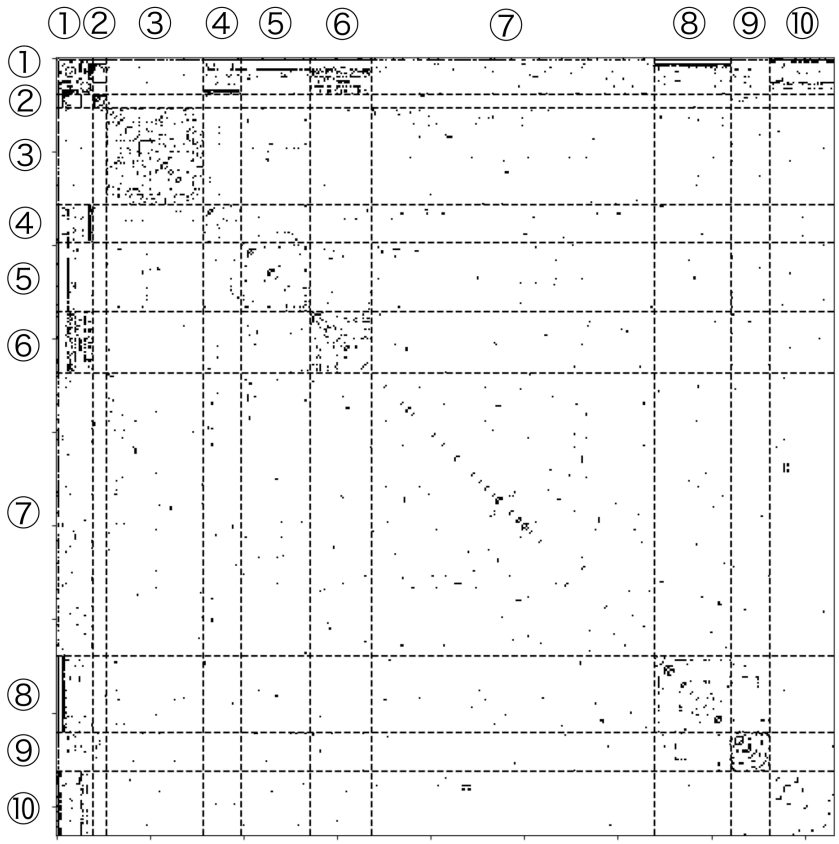

図 5.10: fcitx-mozc パッケージ依存関係ネットワークから抽出されたコミュニティ (提案手法)

libedit2, libbsd0, e2fslibs, gnupg, libusb-0.1-4, libreadline6, libunistring0, libaa1, libslang2, libgssapi-krb5-2, libkrb5-3, libk5crypto3, libcomerr2, libcap-ng0, libdb5.3, update-motd, whiptail, libpopt0, libnewt0.52, libgettextpo0, humanity-icon-theme, hicolor-icon-theme, adwaita-icon-theme, aspell, libaspell15, aspell-en, dictionaries-common, xml-core, sgml-base, libwayland-cursor0, libgtk-3-common, libepoxy0, libiec61883-0, libraw1394-11, liblz4-1, libxpm4, libvpx3, readline-common, libpciaccess0, libavahi-common-data, libjson-glib-1.0-common, libp11-kit0, krb5-locales, libkeyutils1, libsamplerate0, libsecret-common, fcitx-module-lua, libtasn1-6, libssl1.0.0, libpixman-1-0, libwacom-common, im-config, libflac8, libwavpack1, libpresage-data, libgnutls30, liblua5.2-0, debconf-i18n, libjbig0, libthai0, insserv, libaudit-common, hunspell-en-us, libcdparanoia0, ubuntu-mono, tegaki-zinnia-japanese, mozc-data, libv4l-0, libv4lconvert0, xkb-data, libevdev2, libx11-data, libgtk2.0-common, libss2, libopus0, libvisual-0.4-0, libgdk-pixbuf2.0-common, libjpeg-turbo8, libspeex1, libatk1.0-data, libdatrie1, mysql-common, libgtk2.0-bin, emacsen-common, sensible-utils, libfribidi0, zenity-common, libavc1394-0, libshout3, libtag1v5, gcc-5-base, libthai-data, colord-data, gcc-6-base

- コミュニティ 8

表 5.4: 各コミュニティに出現した単語 (かっこ内の数字は tf-idf)

	tf-idf の高い単語 (上位 10 単語)
1	content (0.27) web (0.25) webkit (0.21) gtk+ (0.17) markup (0.17)
2	qt (0.47) qt5 (0.34) module (0.25) widgets (0.20) chinese (0.19)
3	utilities (0.23) linux (0.22) system (0.21) device (0.18) pam (0.16)
4	wayland (0.28) gstreamer (0.27) engine (0.23) anything (0.22) processing (0.21)
5	policykit (0.19) processes (0.17) color (0.15) service (0.15) api (0.14)
6	extension (0.31) fonts (0.30) x (0.28) pango (0.24) font (0.20)
7	kerberos (0.24) files (0.20) gnu (0.17) data (0.15) runtime (0.13)
8	qt (0.43) c++ (0.22) packages (0.17) mozc (0.15) gui (0.15)
9	gtk+ (0.30) input (0.27) method (0.26) mapping (0.25) fcitx (0.24)
10	protocol (0.38) xcb (0.35) x (0.34) binding (0.24) interface (0.24)

libmirprotobuf3, libprotobuf-lite9v5, libqtcore4, presage, libpresage1v5, libmirclient9, libmircore1, libmircommon7, libboost-system1.58.0, libcapnp-0.5.3, libboost-filesystem1.58.0, fcitx-frontend-qt4, qt-at-spi, libqt4-declarative, libqtdbus4, libqt4-xml, qdbus, libtinyxml2.6.2v5, libllvm4.0, libqt4-sql, libqt4-sql-mysql, libproxy1v5, libapt-pkg5.0, apt, libzinnia0v5, libhunspell-1.3-0, libqt4-xmlpatterns, libqt4-network, libapt-inst2.0, libqt4-script, libmysqlclient20, apt-utils, mozc-server, libprotobuf9v5, fcitx-mozc, mozc-utils-gui, libcaca0, libjack-jackd2-0, libtag1v5-vanilla, qtchooser, libfcitx-qt0

- コミュニティ 9

  libdbus-1-3, fcitx-modules, fcitx-config-gtk, libfcitx-gclient0, libfcitx-config4, libfcitx-core0, libfcitx-utils0, libxkbcommon0, fcitx-frontend-all, fcitx-frontend-gtk3, fcitx-frontend-gtk2, libnih-dbus1, libnih1, fcitx-module-dbus, libqt4-dbus, cgmanager, fcitx-bin, fcitx-data, libcgmanager0, fcitx-module-kimpanel, fcitx

- コミュニティ 10

  libxcb-keysyms1, libxcb-sync1, libxcb-icccm4, libice6, libsm6, libaudio2, libmng2, libgl1-mesa-dri, libglapi-mesa, libdrm2, libexpat1, libxcb-dri3-0, libxcb-xkb1, libxau6, libxt6, libxcb-shm0, libxcb-shape0, libgbm1, libxcb-render-util0, libxcb-render0, libxcb-dri2-0, libxshmfence1, libxcb-util1, libxcb-present0, libxdmcp6, libxcb-xfixes0, libxxf86vm1, libx11-xcb1, libinput10, libmtdev1, libxcb-randr0, libxcb-image0, libxcb-glx0, libxkbcommon-x11-0

5.2.2.5　抽出されたコミュニティの評価

chromium-browser と同様, 提案手法から得られた各コミュニティに含まれるパッケージ全体の説明文の名詞の出現回数ベクトルから, tf-idf の高い単語を抽出した. 結果は, 表 5.4 のようになった.

また, 各コミュニティの持つパッケージ間の cos 類似度の平均をコミュニティのペアごとに測定した結果が表 5.5 である.

表 5.5: コミュニティの持つパッケージの cos 類似度の平均 (太字は行ごとの最高値)

	1	2	3	4	5	6	7	8	9	10
1	0.035	**0.066**	0.0094	0.025	0.015	0.023	0.017	0.031	0.036	0.044
2	0.066	**0.29**	0.010	0.011	0.018	0.025	0.018	0.12	0.12	0.027
3	0.0094	0.010	**0.029**	0.0068	0.013	0.0083	0.012	0.011	0.020	0.011
4	0.025	0.011	0.0068	**0.052**	0.013	0.016	0.012	0.015	0.015	0.019
5	0.015	0.018	0.013	0.013	**0.029**	0.013	0.014	0.014	0.023	0.014
6	0.023	0.025	0.0083	0.016	0.013	**0.056**	0.014	0.016	0.047	0.053
7	0.017	0.018	0.012	0.012	0.014	0.014	0.016	0.016	**0.022**	0.018
8	0.032	0.12	0.011	0.015	0.014	0.016	0.016	**0.064**	0.062	0.017
9	0.036	0.12	0.020	0.015	0.023	0.047	0.022	0.062	**0.29**	0.044
10	0.044	0.027	0.011	0.019	0.014	0.053	0.018	0.017	0.044	**0.16**

5.2.3　パッケージ依存関係ネットワークに対するコミュニティ抽出実験に対する考察

はじめに, 図 5.5 と, 図 5.9 で示した modularity 最大化によって抽出されたコミュニティ構造と, 図 5.6 と, 図 5.10 で示した, 提案手法で抽出されたコミュニティ構造を比較する. どちらのネットワークでも, modularity 最大化・提案手法ともにコミュニティ内部のエッジ密度が高くなり, それ以外は密度が低くなるコミュニティが抽出された. 例えば, chromium-browser のネットワークのコミュニティ 2 は, このような性質を持つ典型的なコミュニティである. このコミュニティは, 表 5.2 でも示されているように, perl 関連のパッケージを多く含んだコミュニティである. すなわち, perl 関連のパッケージは, お互いに繋がりあうことが多い反面, 別のパッケージとはそれほど多くつながらないことが分かる. また, fcitx-mozc のネットワークのコミュニティ 3 も同様に コミュニティ内部の繋がりが密になる特徴を持つが, こちらは libsystemd0[*1]や, coreutils[*2]など, GNU/Linux を動かす上で根幹となるパッケージを含んでいるコミュニティである. 表 5.4 に utilities や linux, system といった単語があることからも, このコミュニティの特徴が読み取れる.

一方, 提案手法でのみ, 図 5.6, 図 5.10 どちらでもコミュニティ内部との繋がりがいくらかあり, 外部とも多くの繋がりを持つコミュニティが見られた. 特に, どちらの図でもコミュニティ 1 は, 多くのコミュニティとの間にも繋がりをもつコミュニティになっている. 両者は libc6[*3]や, libx11-7[*4]など, 他のパッケージから呼ばれることの多い基幹となるパッケージを共通して含んでおり, それがコミュニティの特徴に結びついていると考えられる.

表 5.3 や, 表 5.5 を見ると, おおむね対角線上の要素, すなわち同一コミュニティ間のパッケージの平均距離が近くなっていることが観察できるが, 一方で別のコミュニティとの距離が近くなるコミュニティも存在する. これは, 両方のネットワークのコミュニティ 1 のように, 外部とのつながりが多いことが原因である場合と, chromium-browser のネットワークのコミュニティ 5 や, fcitx-mozc のコミュニティ 7 のように, 他のノードとの繋がりを多く持たないノードが一つのコミュニティに集まっていることが原因である場合がある.

[*1] https://packages.debian.org/ja/sid/libsystemd0
[*2] https://packages.debian.org/ja/sid/coreutils
[*3] https://packages.debian.org/ja/sid/libc6
[*4] https://packages.debian.org/ja/sid/libx11-6

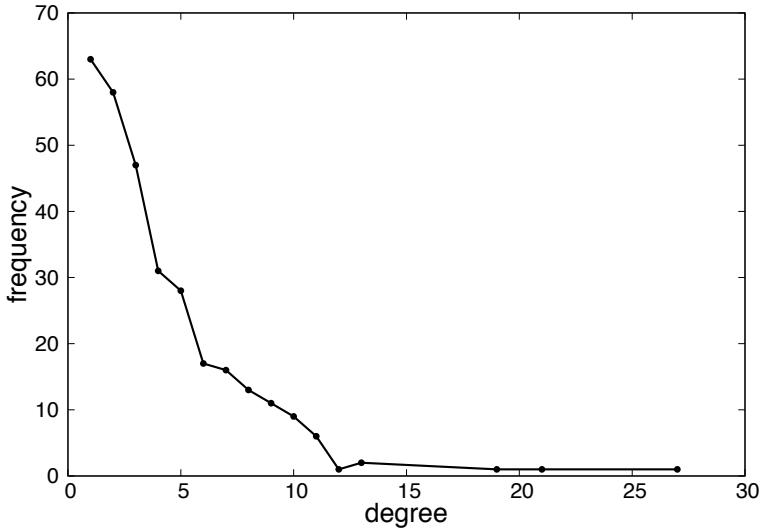

図 5.11: sed の関数呼び出しネットワークの次数分布

5.3　関数呼び出し関係ネットワークに対するコミュニティ抽出実験

5.3.1　関数定義場所を用いたコミュニティの評価

関数呼び出し関係ネットワークに対しても, Louvain 法, 提案手法の二通りのコミュニティ抽出の実験を行った. また, 提案手法で抽出されたコミュニティの評価として, それぞれのコミュニティに含まれている関数がどのファイルで定義されているかを調べ, コミュニティとファイルの二つの分割の間にある関係を調べた. また, 各関数が static 関数 (そのファイルの内部のスコープでのみ有効な関数) であるか否かについても調べた.

5.3.2　sed の関数呼び出しネットワークに対する実験

本実験で使用したデータセットは, ノード数が 305, エッジ数が 631 の無向ネットワークであった.
また, 次数分布は図 5.11 のようにおおむねロングテール型を示した.

5.3.2.1　提案手法の実験の条件

$\boldsymbol{\beta^{(1)}}$ と $\boldsymbol{\beta^{(2)}}$ はロングテール型となるように値を設定したうえで, $\boldsymbol{\alpha}$ を $4.0, 5.0, 6.0, \cdots, 11.0$, コミュニティ数 C を $4 \sim 9$ と変化させる条件のもとでコミュニティ抽出を行い, WAIC の値の計測を行った.

5.3.2.2　WAIC の値の変化

提案手法において, ハイパーパラメータ $\boldsymbol{\alpha}$ 及びコミュニティ数を変えた時の WAIC の値の変化を図 5.12 に示す. $\boldsymbol{\alpha} = 4.0$, コミュニティ数 $C = 6$ の時に WAIC は最小になった.

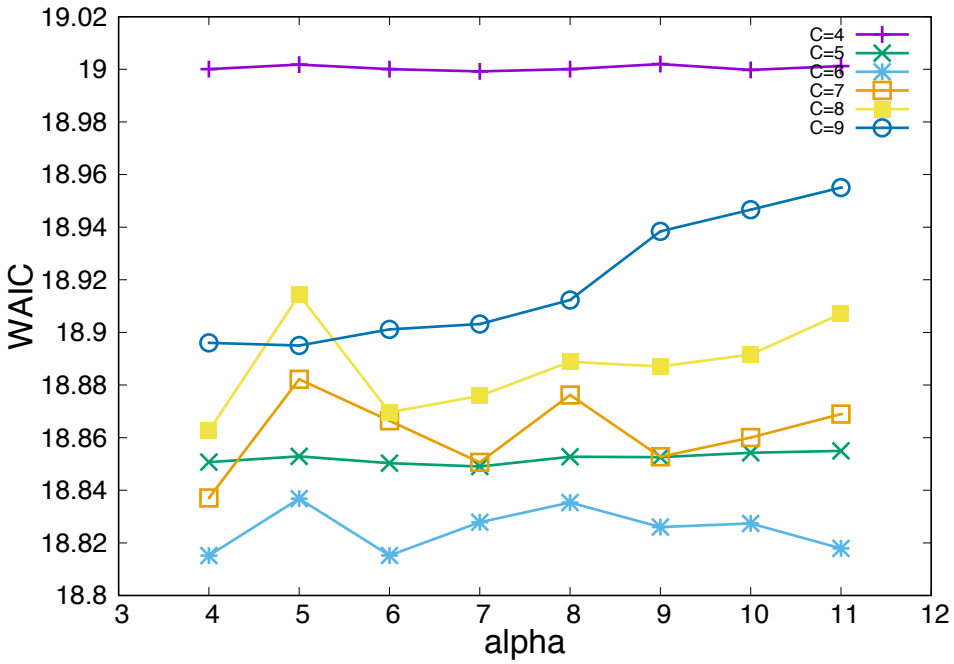

図 5.12: sed の関数呼び出しネットワークの WAIC の値の変化

5.3.2.3　抽出されたコミュニティ

Louvain 法で抽出されたコミュニティを, 図 5.13 に示す. また, 提案手法で WAIC が最小になった $\boldsymbol{\alpha} = 8.0$, $C = 6$ のときに抽出されたコミュニティを図 5.14 に示す. また, 各コミュニティに含まれていた関数の一覧を以下に示す. なお, 各コミュニティ番号は, 図 5.14 で示したコミュニティ番号と対応している.

- コミュニティ 1

  ck_fopen, ck_fclose, panic, register_open_file, do_ck_fclose, ck_realloc, resize_line, closedown, ck_rename, read_always_fail, test_eof, last_file_with_data_p, open_next_file, flush_output, ck_fflush, utils_fp_name, line_append, str_append, follow_symlink, read_file_line, ck_fdopen, ck_mkstemp, release_append_queue, line_copy, do_list, output_missing_newline, ck_fwrite, ck_fread, ck_getline, process_files, execute_program, line_init, read_pattern_space, do_subst, line_reset, output_line, line_exchange, dump_append_queue, match_address_p, next_append_slot, match_regex, append_replacement, str_append_modified, match_an_address_p, main, finish_program, usage
- コミュニティ 2

  merge_state_array, sift_states_backward, re_acquire_state, re_node_set_init_1, build_sifted_states, update_cur_sifted_state, check_arrival_add_next_nodes, check_node_accept_bytes, re_node_set_insert, check_node_accept, re_node_set_merge, create_ci_newstate, calc_state_hash, re_node_set_compare, free_state, check_arrival, search_cur_bkref_entry, find_subexp_node, re_node_set_init_copy, re_node_set_contains, check_arrival_expand_ecl, expand_bkref_cache, register_state, re_node_set_alloc, calc_eclosure_iter, create_cd_newstate, re_node_set_remove_at,

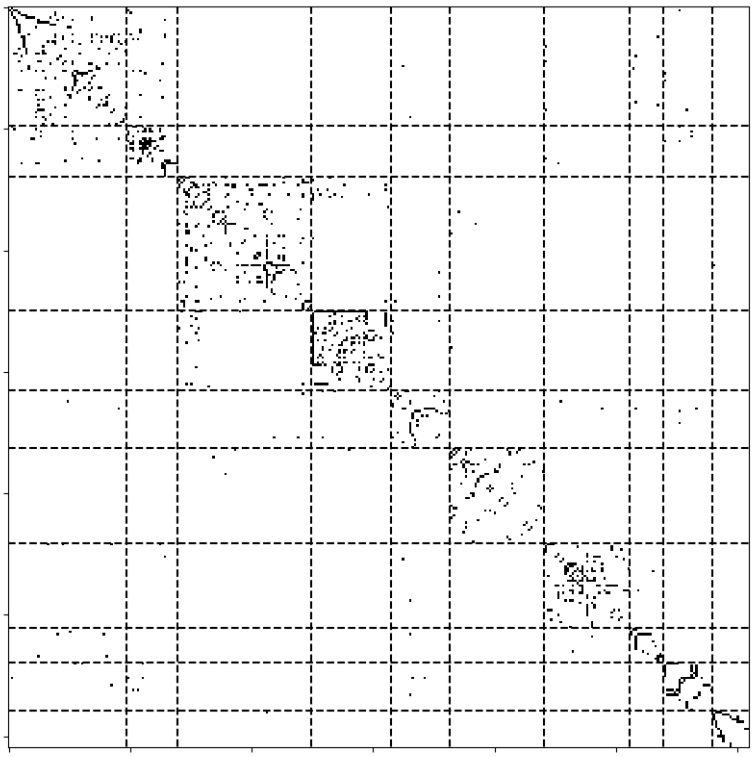

図 5.13: sed の関数呼び出しネットワークから抽出されたコミュニティ (Louvain 法)

proceed_next_node, push_fail_stack, duplicate_node_closure, check_arrival_expand_ecl_sub, add_epsilon_src_nodes, re_node_set_add_intersect, create_initial_state, check_dst_limits, sift_states_iter_mb, check_subexp_limits, sift_states_bkref, sub_epsilon_src_nodes, build_trtable, group_nodes_into_DFAstates, bitset_merge, bitset_empty

- コミュニティ 3

  prune_impossible_nodes, check_halt_state_context, re_node_set_init_union, re_string_context_at, bitset_contain, get_subexp, get_subexp_sub, extend_buffers, clean_state_log_if_needed, re_acquire_state_context, build_wcs_upper_buffer, build_upper_buffer, re_string_translate_buffer, re_string_realloc_buffers, build_wcs_buffer, re_search_internal, re_compile_internal, re_string_construct, check_matching, re_string_reconstruct, re_string_allocate, transit_state_bkref, check_subexp_matching_top, merge_state_with_log, transit_state_mb, re_string_construct_common, transit_state, find_recover_state

- コミュニティ 4

  sift_ctx_init, check_halt_node_context, link_nfa_nodes, re_node_set_init_2, initialize_mbcs,

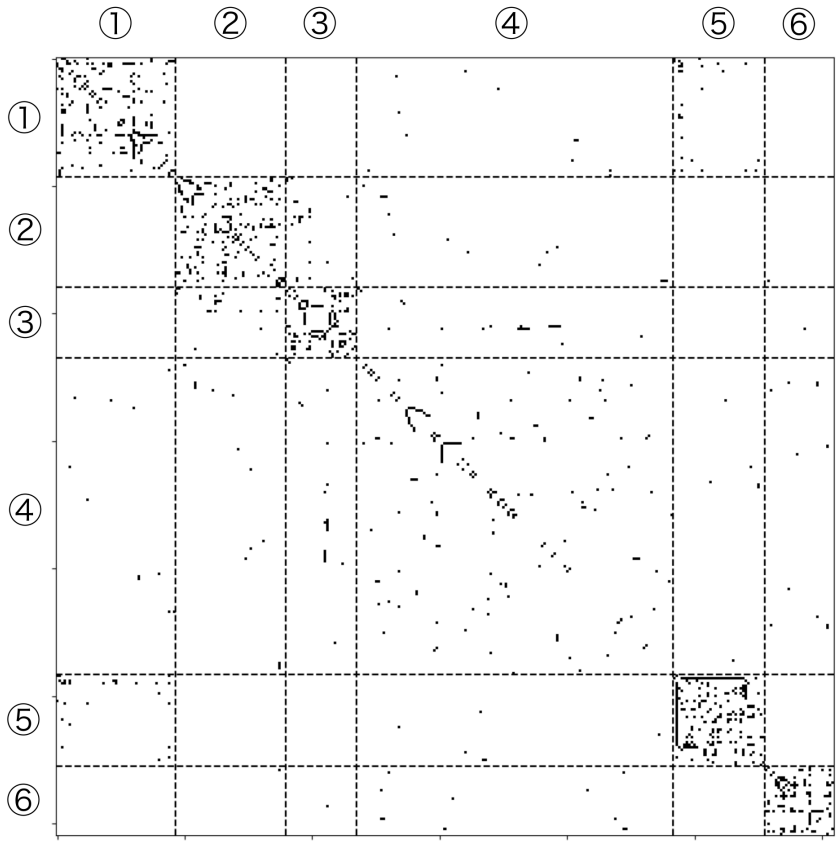

図 5.14: sed の関数呼び出しネットワークから抽出されたコミュニティ (提案手法)

locale_charset, get_charset_aliases, xrealloc, xalloc_die, re_string_wchar_at, re_string_char_size_at, re_string_elem_size_at, free_dfa_content, calc_inveclosure, re_node_set_insert_last, get_backup_file_name, xmalloc, match_ctx_add_sublast, match_ctx_add_entry, fmt, get_paragraph, put_paragraph, fmt_paragraph, get_line, same_para, copy_rest, put_line, base_cost, line_cost, quote_n, quotearg_n_style, quotearg_n_options, quoting_options_from_style, analyze, postorder, calc_eclosure, calc_next, lower_subexps, preorder, calc_first, optimize_subexps, re_dfa_add_node, resize_buffer, duplicate_node, quotearg_colon_mem, quotearg_char_mem, set_char_quoting, flush_paragraph, convert_number, rpl_re_compile_pattern, rpl_re_set_syntax, re_search_2_stub, re_search_stub, rpl_re_compile_fastmap, re_copy_regs, copy_acl, quote, qcopy_acl, qset_acl, quotearg_alloc_mem, xcharalloc, quotearg_buffer_restyled, gettext_quote, rpl_regcomp, re_string_destruct, free_workarea_compile, optimize_utf8, init_dfa, re_compile_fastmap_iter, put_word, put_space, check_dst_limits_calc_pos, quotearg_style_mem, quotearg_n_style_mem, search_duplicated_node, rpl_regexec, match_ctx_free, match_ctx_clean, set_regs, match_ctx_init, match_ctx_add_subtop, check_dst_limits_calc_pos_1, xstrdup, xmemdup, rpl_re_search, rewind_read_files, reset_addresses, x2nrealloc, set_acl, chmod_or_fchmod, re_string_peek_byte_case,

check_punctuation, get_space, quotearg_alloc, rpl_re_match_2, rpl_regfree, xcalloc, xzalloc, quotearg_char, contact, rpl_re_search_2, parse_bracket_symbol, quotearg_mem, quotearg_n_mem, free_tree, mark_opt_subexp, add_buffer, re_string_fetch_byte_case, pop_fail_stack, free_fail_stack_return, update_regs, re_string_skip_chars, quotearg, quotearg_n, quotearg_style, rpl_re_match, quotearg_colon, clone_quoting_options, new_replacement, bitset_copy, bitset_clear, bitset_set_all, x2realloc, quotearg_buffer

- コミュニティ 5
  compile_file, compile_program, ck_free, ck_malloc, size_buffer, free_buffer, ck_memdup, bad_prog, get_buffer, release_label, get_openfile, ck_strdup, read_filename, compile_regex, normalize_text, inchar, read_label, savchar, in_nonblank, next_cmd_entry, bad_command, mark_subst_opts, in_integer, setup_replacement, match_slash, setup_label, compile_address, read_text, check_final_program, init_buffer, add1_buffer, add_then_next, compile_string, compile_regex_1, snarf_char_class, brlen
- コミュニティ 6
  build_equiv_class, bitset_set, free_token, free_charset, parse_sub_exp, parse_reg_exp, fetch_token, create_tree, parse_branch, peek_token, create_token_tree, parse_expression, lower_subexp, build_range_exp, build_charclass, parse, duplicate_tree, parse_bracket_exp, bitset_mask, bitset_not, parse_bracket_element, peek_token_bracket, build_collating_symbol, init_word_char, parse_dup_op, build_charclass_op, fetch_number

5.3.2.4　sed におけるコミュニティと関数の定義場所の関係

前述したように, 提案手法によって抽出されたコミュニティに含まれる関数の定義場所についてまとめた結果を表 5.6 に示す.

5.3.3　less の関数呼び出しネットワークに対する実験

本実験で使用したデータセットは, ノード数が 455, エッジ数が 1220 の無向ネットワークであった.
また, 次数分布は以下の図 5.15 のようにおおむねロングテール型を示した.

5.3.3.1　提案手法の実験の条件

$\boldsymbol{\beta^{(1)}}$ と $\boldsymbol{\beta^{(2)}}$ はロングテール型となるように値を設定したうえで, $\boldsymbol{\alpha}$ を $4.0, 5.0, 6.0, \cdots, 11.0$, コミュニティ数 C を $6 \sim 11$ と変化させる条件のもとでコミュニティ抽出を行い, WAIC の値の計測を行った.

5.3.3.2　WAIC の値の変化

提案手法において, ハイパーパラメータ $\boldsymbol{\alpha}$ 及びコミュニティ数を変えた時の WAIC の値の変化を図 5.16 に示す. $\boldsymbol{\alpha} = 7.0$, コミュニティ数 $C = 8$ の時に WAIC は最小になった.

5.3.3.3　抽出されたコミュニティ

Louvain 法で抽出されたコミュニティを, 図 5.17 に示す. また, 提案手法で WAIC が最小になった $\boldsymbol{\alpha} = 7.0$, $C = 8$ のときに抽出されたコミュニティを図 5.18 に示す. また, 各コミュニティに含まれていた関数の一覧を

表 5.6: 各コミュニティに出現した関数の定義場所 (括弧内の数字は static 関数の個数)

	コミュニティ番号					
	1	2	3	4	5	6
utils.c	15(3)	0	0	2(1)	9	0
xalloc.h	0	0	0	2(2)	0	0
regcomp.c	0	3(3)	1(1)	25(20)	0	24(24)
quotearg.c	0	0	0	21(4)	0	0
quote.c	0	0	0	2	0	0
regex_internal.h	0	2(2)	1(1)	6(6)	0	3(3)
localcharset.c	0	0	0	2(1)	0	0
regexp.c	1	0	0	0	2(1)	0
fmt.c	0	0	0	15(14)	0	0
regexec.c	0	23(23)	14(14)	22(17)	0	0
execute.c	28(27)	0	0	2(2)	0	0
mbcs.c	0	0	0	1	1	0
xmalloc.c	0	0	0	7	0	0
set-mode-acl.c	0	0	0	3	0	0
regex_internal.c	0	15(15)	12(12)	7(7)	0	0
compile.c	1	0	0	3(2)	24(18)	0
sed.c	2(1)	0	0	1(1)	0	0
copy-acl.c	0	0	0	2(1)	0	0
xalloc-die.c	0	0	0	1	0	0

以下に示す. なお, 各コミュニティ番号は, 図 5.18 で示したコミュニティ番号と対応している.

- コミュニティ 1

  error, quit, less_printf, putchr, putstr, flush, clear_bot, at_exit, at_enter, ierror, edit_ifile, use_logfile, unsave_ifile, query, save_curr_ifile, reedit_ifile, raw_mode, deinit, edit, getchr, psignals, init, iread, exec_mca, multi_search, edit_list, lsystem, cmd_exec, edit_next, edit_istep, winch, commands, prompt, init_signals, pipe_data, opt_t, main, cat_file
- コミュニティ 2

  prev_pattern, pos_clear, get_scrpos, squish_check, lower_left, hist_pattern, ch_getflags, clr_hilite, lastmark, hilite_screen, prep_hilite, position, repaint_hilite, back_line, is_filtered, is_filtering, hlist_find, gomark, jump_loc, adjsline, back, home, add_back_pos, forw_line, next_unfiltered, clear, forw, prev_unfiltered, repaint, eof_bell, put_line, search, search_pos, make_display, jump_forw, empty_screen, empty_lines, backward, goto_line, rrshift, undo_search, eof_displayed, get_quit_at_eof, forw_loop, forward, clear_attn, entire_file_displayed
- コミュニティ 3

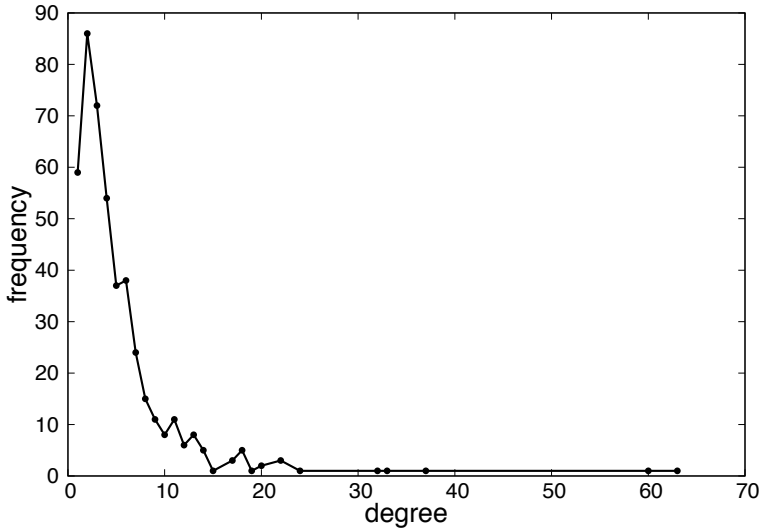

図 5.15: less の関数呼び出しネットワークの次数分布

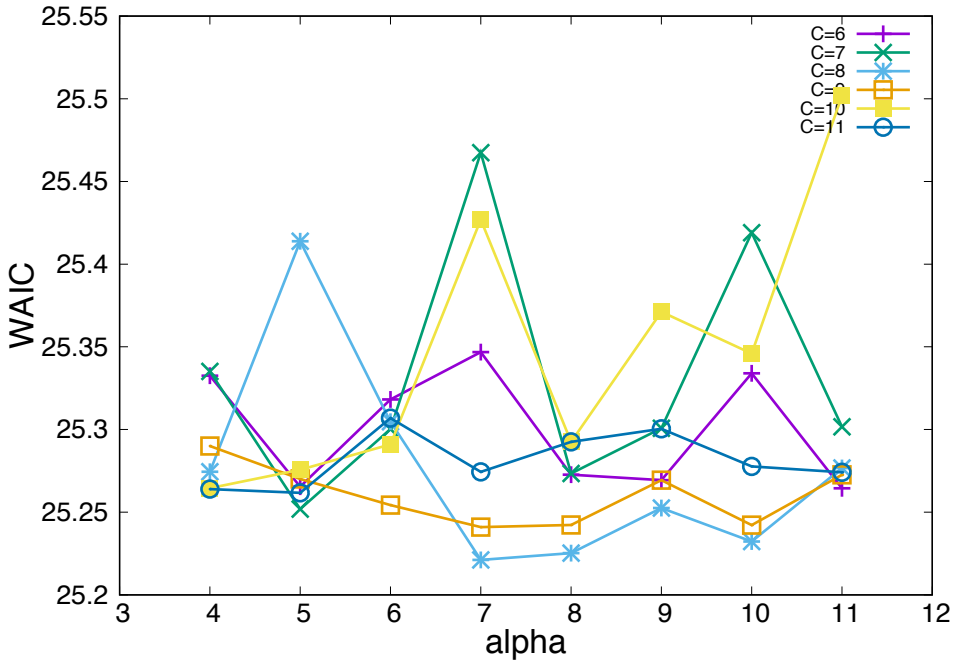

図 5.16: sed の関数呼び出しネットワークの WAIC の値の変化

delimit_word, cmd_right, cmd_step_right, cmd_lshift, clear_eol, cmd_updown, bell, cmd_home, beep, vbell, cmd_step_left, putbs, cmd_complete, cmd_erase, cmd_istr, next_compl, cmd_repaint, cmd_left, cmd_ichar, cmd_char, cmd_edit, editchar, cmd_kill, cmd_werase, cmd_wdelete, cmd_delete, cmd_rshift, u_interrupt

- コミュニティ 4

  readfd, ecalloc, set_pattern, compile_pattern, fcomplete, lglob, shell_unquote, lgetenv,

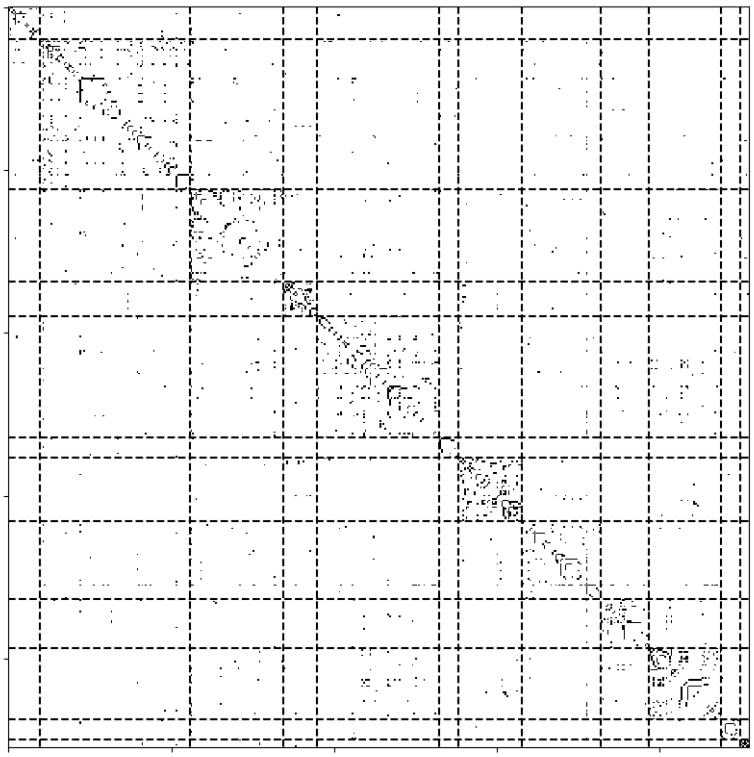

図 5.17: less の関数呼び出しネットワークから抽出されたコミュニティ (Louvain 法)

get_meta_escape, fexpand, metachars, shell_quote, shellcmd, get_term, pos_init, scrsize, ltgetnum, cmd_decode, ltget_env, get_ifile, new_ifile, save, num_error, get_return, set_charset, icharset, ichardef, is_dir, bad_file, open_altfile, cmd_addhist, errno_message, close_file, ungetcc, close_altfile, num_pct_s, ch_ungetchar, save_cmdhist, init_option, scan_option, optstring, skipsp, init_charset, init_textlist, init_compl, cleantags, compile_pattern2, histfile_name, metachar, shell_coption, opt_j, opt__T, opt_o, dirfile, homefile, opt__P, opt_p, maketagent, make_tempname, add_hometable, findtag, findgtag, findctag, opt_shift, edit_stdin, init_line, init_prompt

- コミュニティ 5

  ungetsc, mca_char, mca_search_char, mca_opt_char, len_cmdbuf, mca_search, mca_opt_nonfirst_char, is_erase_char, cmd_reset, get_cmdbuf, toggle_option, start_mca, mca_opt_toggle, clear_cmd

- コミュニティ 6

  find_pos, back_raw_line, ch_seek, add_lnum, forw_raw_line, ch_back_get, ch_tell, expand_linebuf, ch_forw_get, buffered, ch_length, find_linenum, getmark, ch_end_seek, ch_get, pipe_mark,

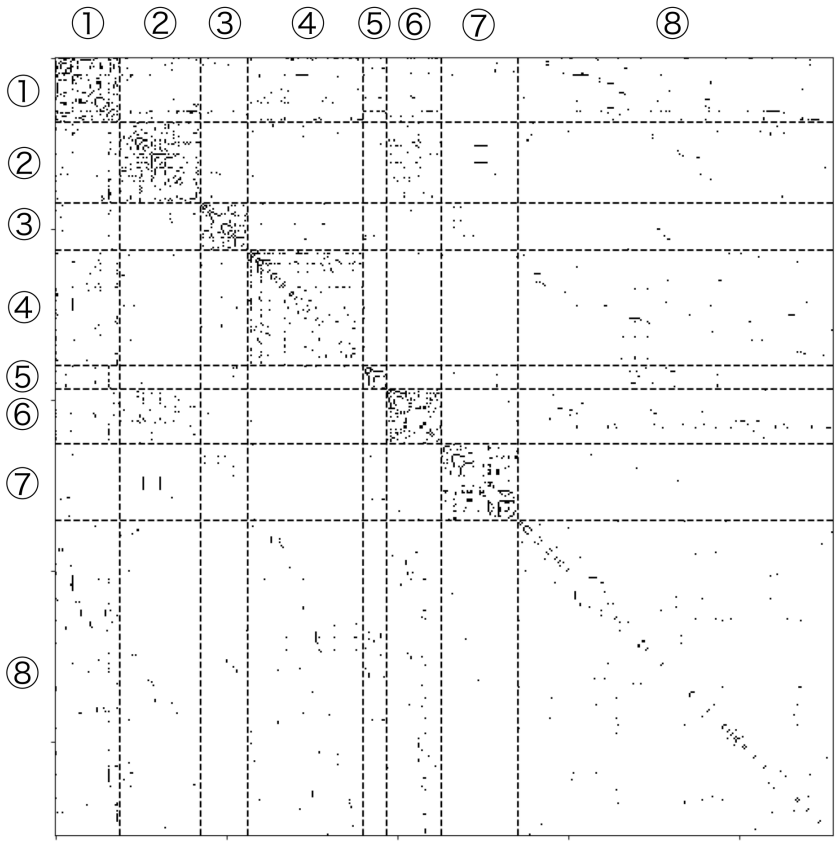

図 5.18: less の関数呼び出しネットワークから抽出されたコミュニティ (提案手法)

match_brac, plinenum, search_range, currline, pr_expand, cond, protochar, set_attnpos, jump_percent, jump_line_loc, jump_forw_buffered, ch_end_buffer_seek, ctagsearch, jump_back, ch_beg_seek, curr_byte

- コミュニティ 7

  is_ucase, prutfchar, control_char, is_ubin_char, put_wchar, is_in_table, bin_file, step_char, is_ansi_middle, binary_char, utf_bin_count, cmd_step_common, get_wchar, utf_len, prchar, is_combining_char, is_wide_char, is_composing_char, is_utf8_well_formed, set_status_col, prewind, is_hilited, pdone, pshift_all, pappend, pflushmbc, null_line, do_append, pshift, is_ansi_end, flush_mbc_buf, at_switch, cmd_putstr, pwidth, attr_ewidth, is_ascii_char, backc, store_prchar, store_tab, store_char, attr_swidth, is_at_equiv, apply_at_specials, in_ansi_esc_seq, cvt_text

- コミュニティ 8

  add_cmd_table, expand_special_keys, special_key_str, set_filter_pattern, clear_pattern, clr_filter, uncompile_pattern, clr_hlist, is_null_pattern, tmodes, ltgetflag, cheaper, ltgetstr, cost, find_ifile, link_ifile, iprint_linenum, iprint_int, linenumtoa, inttoa, calcgap, unlink_ifile, incr_index, get_time, longish, abort_long, longloopmessage, ilocale, opt_i, chg_caseless, markpos, getumark, clr_linenum,

ch_init, get_filename, held_ifile, del_ifile, end_logfile, opened, set_open, get_pos, set_filestate, seekable, get_filestate, init_hashtbl, ch_flush, close_getchr, hold_ifile, getoff_ifile, unmark, ch_close, store_pos, next_ifile, edit_inext, edit_iprev, prev_ifile, ap_char, ap_str, add_ecmd_table, stop, calc_shift_count, calc_jump_sline, line_left, clear_eol_bot, intread, findopt, flip_triple, opt_desc, getnum, propt, findopt_name, nostring, sprefix, is_optchar, cmd_int, getfraction, setbinfmt, badmark, chg_hilite, wait_message, ch_addbuf, forw_textlist, back_textlist, edit_index, get_index, in_mca, cmd_accept, ecmd_decode, getcc, opt_has_param, opt_prompt, mca_opt_first_char, ch_delbufs, is_hilited_range, onscreen, nopendopt, wherechar, add_fcmd_table, opt_quote, opt_query, copy_hist, write_mlist, write_mlist_header, ap_int, add_line, get_back_scroll, gline, filesize, skipcond, edit_first, edit_last, edit_prev, reopen_curr_ifile, read_cmdhist2, seek_filesize, ap_linenum, set_mlist, fcmd_decode, cmd_search, new_lesskey, gint, add_var_table, hlist_getnode, hlist_getstorage, ap_pos, postoa, create_hilites, add_hilite, hlist_rotate_right, hlist_rotate_left, cvt_alloc_chpos, hilite_line, match_pattern, match, eq_message, opt_x, sync_logfile, percent_pos, muldiv, get_cvt_ops, cvt_length, cmd_lastpattern, opt__O, tagsearch, gtagsearch, check_winch, nexttag, clear_buffers, dispversion, prevtag, ntags, setmark, get_swindow, nextgtag, ch_set_eof, pr_string, prevgtag, ap_quest, histfile_modified, make_file_private, mlist_size, read_cmdhist, opt_k, lesskey, old_lesskey, init_cmds, edit_tagfile, gettagtype, addhist_init, init_cmdhist, opt__V, nifile, isoptpending, init_mark, open_getchr, init_search, init_pattern, inc_costcount, getentry, add_forw_pos, percentage, last_component, curr_tag, valid_pattern

5.3.3.4　less におけるコミュニティと関数の定義場所の関係

sed の時の同様, 提案手法によって抽出されたコミュニティに含まれる関数の定義場所についてまとめた結果を表 5.7 に示す.

5.3.4　関数呼び出し関係ネットワークに対するコミュニティ抽出実験の考察

はじめに, Louvain 法で抽出したコミュニティ構造である図 5.13 および図 5.17 と, 提案手法で抽出したコミュニティ構造を表した図 5.14, および図 5.18 を比較すると, パッケージの依存関係ネットワークとは対照的に, 提案手法によって極端に他のコミュニティとのつながりを持つコミュニティが抽出されないことが見て取れる.

提案手法によって抽出された sed のコミュニティ構造を見てみると, コミュニティ 4 を除いて, おおむねコミュニティ内部の繋がりが密になり, それ以外が疎となる性質をもつコミュニティによって構成されていることが分かる. コミュニティ 1 は, プログラム全体のエントリポイントである main 関数を含んだコミュニティであり, 5.6 と対応づけて観察すると, sed.c (main 関数が定義されている) をはじめとして, execute.c, utils.c で定義されている関数の大部分がこのコミュニティに所属していることが分かる. コミュニティ 2, 3 は, 二つのコミュニティ内部の繋がりが密になるコミュニティであるが, どちらも regcomp.c や, regexec.c, regex_internal.c など, 正規表現に関連する処理を集めたファイルで定義されている関数を含んだ, 関連性の強いコミュニティであると分かる. コミュニティ 4 は, 他のノードへの繋がりをあまり多く持たないノードを含んだコミュニティであり, さまざまなファイルで定義されている関数を万遍なく含んでいるが, quotearg.c や,

fmt.c のように, 定義されている関数が全てこのコミュニティに含まれているファイルもある. コミュニティ 5 は, compile.c で定義されている関数の大部分を含んだコミュニティであり, コミュニティ 6 は, regcomp.c で定義されいてる関数の大部分を含んだコミュニティとなっている.

提案手法で抽出された less のコミュニティ構造についても, 8 番目のコミュニティを除いて, おおむねコミュニティ内部の繋がりが密になる性質をもつコミュニティによって構成されているが, sed に対してコミュニティ間のつながりが多い傾向にあることが分かる. 5.7 と対応づけると, sed に比べて, 特にあるファイルで定義されている関数の大部分を含んでいるといえるコミュニティの割合は少なく, コミュニティ 6 が filename.c で定義されている関数の大部分を, コミュニティ 7 が line.c で定義されている関数の大部分を含んでいるほか, 様々なファイルで定義されている関数を万遍なく含んだコミュニティ 8 で, prompto.c, option.c, ifile.c で定義されている大部分の関数を含んでいる程度であった.

表 5.7: 各コミュニティに出現した関数の定義場所 (括弧内の数字は static 関数の個数)

	コミュニティ番号							
	1	2	3	4	5	6	7	8
prompt.c	0	0	0	1	0	4(3)	0	11(8)
optfunc.c	1	0	0	6	0	0	0	10
lsystem.c	2	0	0	0	0	1	0	0
opttbl.c	0	0	0	1	0	0	0	3(1)
cvt.c	0	0	0	0	0	0	1	2
option.c	0	1	0	3(2)	1	0	0	10(3)
edit.c	10(1)	0	0	3(1)	0	0	0	9(2)
screen.c	6	4	5(1)	4(2)	0	0	3	11(7)
main.c	2	0	0	3	0	0	0	1
command.c	5(4)	2(2)	0	1	9(8)	0	0	5(2)
position.c	0	7	0	1	0	0	0	2
decode.c	0	0	1	3(1)	0	0	0	13(7)
search.c	0	15(4)	0	1(1)	0	1(1)	2(1)	17(11)
output.c	7(1)	1	0	1	0	0	0	5(2)
charset.c	0	0	0	4(3)	0	0	15(1)	2(2)
ch.c	0	1	0	1	0	10(1)	0	10(3)
pattern.c	0	0	0	2(1)	0	0	0	5(1)
line.c	0	1	0	1	0	4(1)	21(11)	1
forwback.c	0	8(1)	0	0	0	0	0	1
tags.c	0	0	0	5(3)	0	1(1)	0	11(4)
brac.c	0	0	0	0	0	1	0	0
linenum.c	0	0	0	0	0	4	0	5(4)
ttyin.c	1	0	0	0	0	0	0	2
os.c	1	0	0	1	0	0	0	5(1)
mark.c	0	2	0	0	0	1(1)	0	6(1)
cmdbuf.c	0	0	21(20)	5(3)	4	0	2(1)	14(9)
input.c	0	2	0	0	0	1	0	0
filename.c	0	0	0	18(6)	0	0	1	3(1)
ifile.c	0	0	0	2(1)	0	0	0	19(4)
signal.c	3	0	1(1)	0	0	0	0	1(1)
jump.c	0	3	0	0	0	4	0	0

第 6 章　結論

6.1　まとめ

本研究では, ソフトウェアパッケージの依存関係ネットワーク・関数の呼び出し関係ネットワークという二つの異なる粒度から, 確率モデルを用いてコミュニティ抽出を行い, 情報量規準によって真の分布に近い分布を推定し, その有効性を確かめた.

5 章での実験から, パッケージの依存関係ネットワーク・関数の呼び出し関係ネットワークどちらにおいても, 情報量規準を用いたコミュニティ数の選択をすることによって, 与えられたネットワークがコミュニティ構造をもつことを確かめることができた.

また, パッケージ依存関係のネットワークにおいては, 提案手法を用いることで, modularity 最大化では得られない同一コミュニティ間のエッジ密度が高くなるコミュニティと, そうでないコミュニティが混在する構造を発見することができた. 一方で, 関数の呼び出し関係ネットワークにおいては, 提案手法を用いても, 極端に他のコミュニティとのつながりを持つコミュニティが抽出されないことが確認された.

6.1.1　パッケージの依存関係ネットワークに対するコミュニティ抽出

パッケージの依存関係ネットワークにおいては, 提案手法を適用した結果から, 同一コミュニティ間のエッジ密度が高くなるコミュニティと, 色々なコミュニティとの間につながりを持つコミュニティが発見された (5.2.1.4 節, 5.2.2.4 節での結果を参照). 色々なコミュニティとの間につながりをもつコミュニティは, libc のような基本的なライブラリのパッケージを含むことが分かった. また, 他のノードとの接続数が少ないノードの集まったコミュニティも抽出された.

5.2.1.5 節, 5.2.2.5 節で示したコミュニティに対する評価からは, 同一コミュニティ間のパッケージは, パッケージの説明文から測った距離も近くなる傾向にあることが確かめられた.

6.1.2　関数の呼び出し関係ネットワークに対するコミュニティ抽出

関数の呼び出し関係ネットワークでは, 5.3.2.3 節, 5.3.3.3 節で示した結果から, おおむね同一コミュニティ間のエッジ密度が高くなるようなコミュニティを持つが, それとは別に, 他との接続の少ないノードの集まるコミュニティが抽出された.

また, 5.3.2.4 節, 5.3.3.4 節で示された結果から, sed のネットワークではコミュニティとコミュニティ内部の関数が定義された場所にある程度の関係性がみられたものの, less のネットワークでは関係性を確かめることができなかった. これは, 抽出されたコミュニティの多くが, less と比較して同一コミュニティ内部のエッジ密度が極端に高くなるような構造でなかったことが原因であると考えられる.

6.2　今後の課題

今後の課題としては, ネットワークの向きを考慮したコミュニティ抽出を行うことである. 現状では, 各ノードについて自分に向かって接続しているノードとの間のエッジ, 自分から他のノードに向かって接続しているエッジ両方の情報を使うために, もともとあったエッジの向きの情報を考慮していない. エッジの向きを考慮してコミュニティ抽出するには, 別のデータの表現法や, 別のクラスタリング手法を用いる必要があると考えられる.

また, パッケージの依存関係ネットワーク・関数の呼び出し関係ネットワークの両方において, 他のノードとの接続数の少ない, いわゆる「葉」と呼ばれるノードが集まるコミュニティが生成された. 接続パターンを用いた確率モデルによる手法においては, こうした接続数の少ないノードは, ノードの持つ特徴的な接続の情報が不足してしまっているために, 互いに区別のつかないノードとみなされてしまう場合が多い. こうしたノードをより適切にクラスタリングするためには, 各ノードの接続パターン以外の特徴を取り入れたクラスタリングを適用することが必要であると考えられる.

謝辞

本研究は，電気通信大学大学院 情報理工学研究科 情報・通信工学専攻の寺田研究室において，寺田 実准教授のご指導のもと行われました.

寺田 実准教授には, 本稿を執筆するにあたり, 数多くのアドバイスと, 有用なコメントを頂きましたことを感謝いたします.

岩崎 英哉教授には, 本論文の査読をしていただき, 大変お世話になりました.

また, 安部 文紀さん, 本田 裕人さん, 山本 愛美さん, 佐々木 透さん, 藤本 明優さん, 村松 啓寛さん, 飯尾 直樹さん, 前田 喜洋さん, 三谷 将大さん, 毛利 勇摩さんには, 3 年間の研究室での生活を通じ, 研究に関することだけでなく, 合宿などの研究以外の場でもお世話になりました. 心より感謝申し上げます.

参考文献

[1] Santo Fortunato. Community detection in graphs. *Physics Reports*, Vol. 486, No. 3, pp. 75–174, 2010.

[2] Fragkiskos Malliaros and Michali Vazirgiannis. Clustering and community detection in directed networks: A survey. *Physics Reports*, Vol. 533, pp. 95–142, 2013.

[3] Jens Dietrich, Vyacheslav Yakovlev, Catherine McCartin, Graham Jenson, and Manfred Duchrow. Cluster analysis of java dependency graphs. In *Proceedings of the 4th ACM Symposium on Software Visualization*, SoftVis '08, pp. 91–94. ACM, 2008.

[4] J. Hamilton and S. Danicic. Dependence communities in source code. In *2012 28th IEEE International Conference on Software Maintenance (ICSM)*, pp. 579–582, Sep. 2012.

[5] Y. Du, J. Wang, and Q. Li. An android malware detection approach using community structures of weighted function call graphs. *IEEE Access*, Vol. 5, pp. 17478–17486, 2017.

[6] Orahcio Felício de Sousa, Marcio Argollo, and Thadeu Penna. Analysis of the package dependency on debian gnu/linux. *Journal of Computational Interdisciplinary Sciences*, pp. 127–133, Jan. 2009.

[7] Jörg Reichardt and Stefan Bornholdt. Detecting fuzzy community structures in complex networks with a potts model. *Phys. Rev. Lett.*, Vol. 93, p. 218701, Nov. 2004.

[8] Zhengxu Zhao, Yang Guo, and Weihu Zhao. Community detection in directed weighted function-call network. *International Journal of Automation and Control Engineerin*, Vol. 4, pp. 9–13, Jan. 2015.

[9] Yong-Yeol Ahn, James P. Bagrow, and Sune Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, Vol. 466, No. 761-764, Jun. 2010.

[10] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 99, No. 12, pp. 7821–7826, Jun. 2002.

[11] Filippo Radicchi, Claudio Castellano, Federico Cecconi, Vittorio Loreto, and Domenico Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, Vol. 101, No. 9, pp. 2658–2663, 2004.

[12] Yu Qu, Xiaohong Guan, Qinghua Zheng, Ting Liu, Lidan Wang, Yuqiao Hou, and Zijiang Yang. Exploring community structure of software call graph and its applications in class cohesion measurement. *Journal of Systems and Software*, Vol. 108, pp. 193–210, Oct. 2015.

[13] Mark E.J. Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, Vol. 69, p. 026113, Mar. 2004.

[14] Jiao Wang and C-H Lai. Detecting groups of similar components in complex networks. *New J. Phys.*,

Vol. 10, No. 12, p. 123023, 2008.

[15] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, Vol. 39, No. 1, 1977.

[16] 原田尚幸, 竹内一郎, 中野良平. 次数分布に基づく事前情報を用いた複雑ネットワークのクラスタリング. 信学技報, Vol. 108, No. 372, pp. 1–6, 2008.

[17] Sumio Watanabe. Asymptotic equivalence of bayes cross validation and widely applicable information criterion in singular learning theory. *Journal of Machine Learning Research*, Vol. 11, pp. 3571–3594, 2010.

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