What Is a Network Community?
A Novel Quality Function and Detection Algorithms

宮内 敦史　**河瀬 康志**

東京工業大学

ERATO **感謝祭** Season III

河原林巨大グラフプロジェクト
Kawarabayashi Large Graph Project

CIKM 2015

The 24th ACM International Conference on Information and Knowledge Management

- **期間**: 10 **月** 19 **日から** 23 **日**
- **場所**: Melbourne Convention and Exhibition Centre

CIKM 2015 の採択率

3 つのトラックに分かれて論文を募集

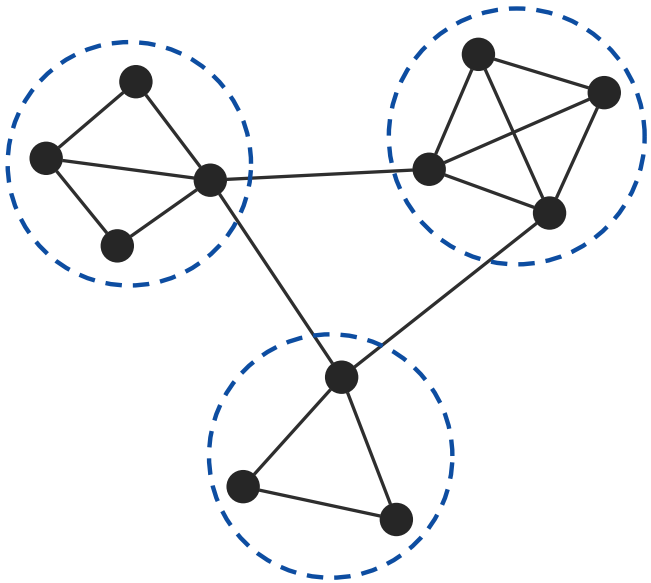
- Database
  - Long: $35/129 = 27\%$
  - Short: $6/37 = 16\%$
- Information Retrieval
  - Long: $43/171 = 25\%$
  - Short: $27/101 = 27\%$
- Knowledge Management
  - Long: $87/346 = 25\% \leftarrow$ **本研究**
  - Short: $36/138 = 26\%$

(Long paper: 10 **ページ**，Short paper: 4 **ページ**)

はじめに

コミュニティ検出

ネットワーク上の “まとまりらしい部分” を**コミュニティ**と呼ぶ



様々なネットワークに存在

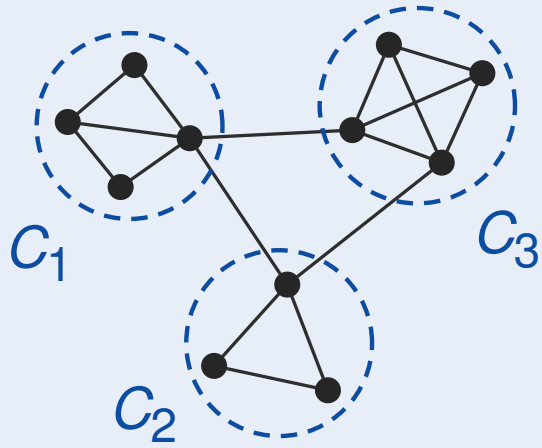
- ソーシャルネットワーク
- Web グラフ
- タンパク質ネットワーク
- etc.

ネットワーク解析における基本的かつ重要な操作

評価関数

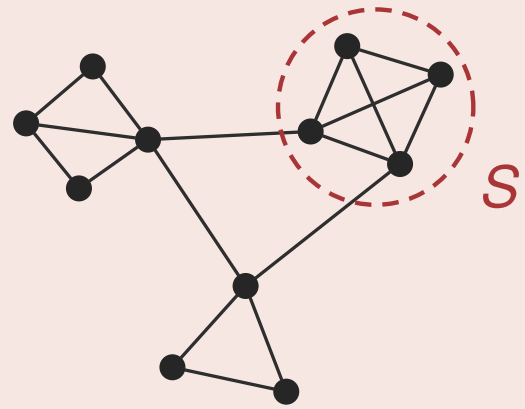
“コミュニティらしさ” を数値で返す関数

頂点集合の分割に対して



$f(C) = ???$

頂点部分集合に対して



$f(S) = ???$

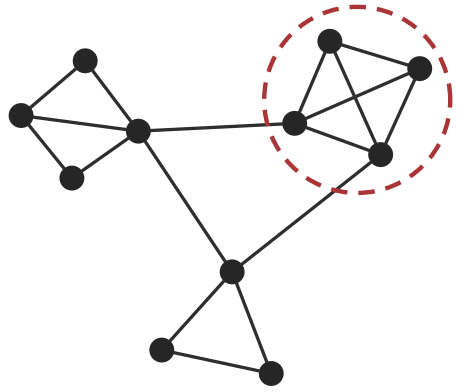
頂点部分集合に対する評価関数に着目

- 多くのコミュニティ検出法で用いられている
- クラスタリング結果の整理のためにも利用できる

評価関数

記号

- $e[S] := S$ **内の枝数**
- $\mathrm{cut}[S] := S$ **のカット枝の本数**
- $D[S] := S$ **内の頂点の次数和**



$e[S] = 6$
$\mathrm{cut}[S] = 2$
$D[S] = 14$

既存の評価関数

- $\mathrm{DS}(S) = \dfrac{2e[S]}{|S|}$ **平均次数**
- $\mathrm{OQC}(S) = e[S] - \alpha \binom{|S|}{2}$ OQC **関数** (Tsourakakis et al. '13)
- $\mathrm{COND}(S) = \dfrac{\mathrm{cut}[S]}{\min\{D[S], D[V \setminus S]\}}$ **コンダクタンス**

標準的な評価関数として定着しているものはない

本研究の成果

頂点部分集合に対する評価関数**コミュニチュード**を提案

- コミュニチュード最大化問題
- 線形時間ヒューリスティック
- 実験的評価

コミュニチュード

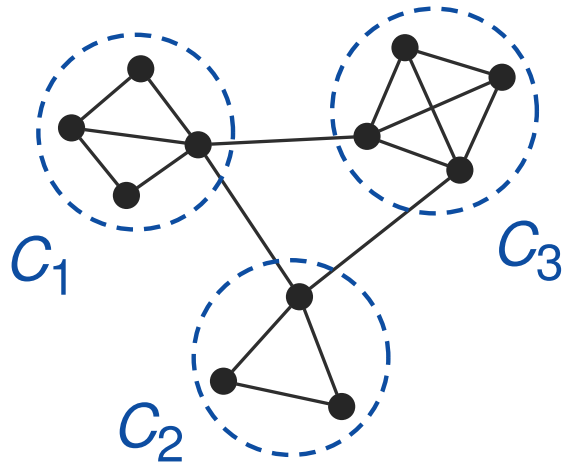
コミュニチュードの考え方

モジュラリティ(Newman & Girvan '04) **という評価関数を参考にする**

頂点集合の分割 $\mathcal{C} = \{C_1, \ldots, C_k\}$ **に対して**

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{e[C]}{m} - \left(\frac{D[C]}{2m} \right)^2 \right)$$

(m: **グラフの枝数**)



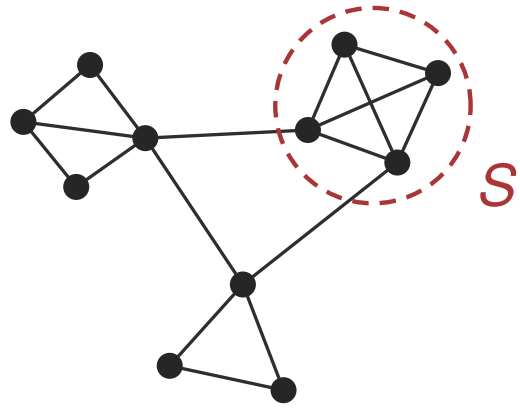
- **モジュラリティの値が高い** $\approx$ **コミュニティらしく分かれている**
- **頂点集合の分割に対する標準的な評価関数**

モジュラリティの "頂点部分集合バージョン" を作る

コミュニチュードの考え方

単純な関数はすでに存在 (Leskovec et al. '10)

$$Q(S) = \frac{e[S]}{m} - \left(\frac{D[S]}{2m}\right)^2$$



- **モジュラリティから一つの項を抜き出した**
- $S \subseteq V$ **内の枝の割合がランダムと比べてどれだけ大きいかを評価**

優れている点

$e[S]$ **だけでなく** $\mathrm{cut}[S]$ **の大きさも間接的に考慮**

問題点

多くの実ネットワークでは， $S \subseteq V$ **を** $|S| \approx |V|/2$ **まで大きくしていくと，関数値がほぼ単調に増加** (Leskovec et al. '10)

Leskovec **の関数** $Q(S)$ **を適切に修正する**

コミュニチュードの考え方

枝の生成過程

次数分布に従って，頂点集合 V 上に N 本の枝をランダムに生成

枝を 1 本生成するとき，その枝が $S \subseteq V$ 内に生成される確率は

$$p = \left(\frac{D[S]}{2m} \right)^2$$

$S \subseteq V$ 内に生成される枝の割合が従う確率分布を観察

X: **上記の過程で $S \subseteq V$ 内に生成される枝の本数を表す確率変数**

- $X \sim B(N, p)$
- **中心極限定理より，** $X \sim \mathcal{N}(Np, Np(1-p))$
- **よって，** $X/N \sim \mathcal{N}(p, p(1-p)/N)$

$S \subseteq V$ 内の実際の枝の割合がランダムと比べてどれだけ大きいか？

コミュニチュード

コミュニチュードは，分散も考慮して評価

$$\mathrm{com}(S) = \frac{\frac{e[S]}{m} - \left(\frac{D[S]}{2m}\right)^2}{\sqrt{\left(\frac{D[S]}{2m}\right)^2 \left(1 - \left(\frac{D[S]}{2m}\right)^2\right)}}$$

Leskovec **の関数** $Q(S)$

標準偏差

- $S \subseteq V$ **内の実際の枝の割合** $\frac{e[S]}{m}$ **の** Z **値を計算**
- $\mathrm{com}(S)$ **が高いほど，** $S \subseteq V$ **はコミュニティらしい**

コミュニチュードは，重み付きグラフに対しても同様に定義できる

コミュニチュード最大化

問題

- **入力**: $G = (V, E)$
- **出力**: $S \subseteq V$ that maximizes com(S)

応用を考慮して，制約付き問題も定義

サイズ制約付き

- **入力**: $G = (V, E)$ & $k_{\min}, k_{\max} \in \mathbb{Z}_{>0}$
- **出力**: $S \subseteq V$ that maximizes com(S) under $k_{\min} \leq |S| \leq k_{\max}$

クエリノード付き

- **入力**: $G = (V, E)$ & $Q \subseteq V$
- **出力**: $S \subseteq V$ that maximizes com(S) under $Q \subseteq S$

アルゴリズム

線形時間ヒューリスティック

1st Phase: Greedy peeling + 2nd Phase: Local search

1st Phase: Greedy peeling

1: **次数最小の頂点を逐次除去し，頂点部分集合の列を得る**
2: **そのなかで目的関数値が最大の $S \subseteq V$ を出力**

コミュニティ検出や密グラフ抽出において有用

	精度保証	実行時間
平均次数	1/2 (Charikar '00)	$O(m+n)$
OQC **関数**	**加法的** (Tsourakakis et al. '13)	$O(m+n)$

(n: **グラフの頂点数**)

コミュニチュード最大化問題に対しても，実行時間は $O(m+n)$

線形時間ヒューリスティック

Greedy peeling で得た解 $S \subseteq V$ を改善

2nd Phase: Local search

1: すべての頂点をランダムに並べ，それを順に見ていく
各頂点 v について，目的関数値が改善されるならば，
v を S 内へ移動 (あるいは S 外へ移動)

2: 以下のどちらかを満たすまで，上記の操作を繰り返す
   (i) 局所最適解が得られる
   (ii) 繰り返し回数が $T_{\max}$ を超える

3: 全体の操作を $R_{\max}$ 回繰り返し，最良解を出力

実行時間は $O((m+n) \cdot T_{\max} \cdot R_{\max})$

計算機実験では，$T_{\max} = \infty$, $R_{\max} = 10$ と設定

制約付き問題に対するアルゴリズム

線形時間ヒューリスティックを改造

サイズ制約付き

1st Phase: **制約 $k_{\min} \leq |S| \leq k_{\max}$ を満たす初期解 $S \subseteq V$ を得る**

2nd Phase: **サイズ制約を破る移動はスキップ**

クエリノード付き

1st Phase: Greedy peeling **は行わず，クエリノードを初期解に設定**

2nd Phase: **クエリノードに関する移動はスキップ**

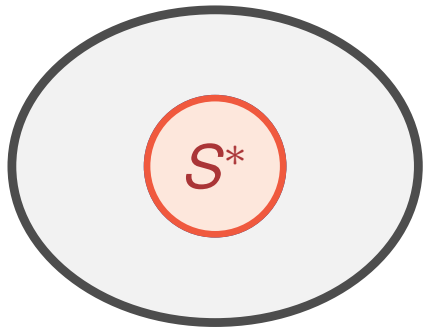
計算機実験

単一コミュニティモデル

コミュニティが 1 個埋め込まれている疎なグラフ

Parameters:

- n: **頂点数** $\leftarrow 1000$
- c: **コミュニティサイズ** $\leftarrow 50$ or 100
- p_{in}: **コミュニティ内の枝の生起確率** $\leftarrow 0.2$ or 0.4
- p_{out}: **コミュニティ外の枝の生起確率** $\leftarrow 0.01, \ldots$



$G = (V, E)$

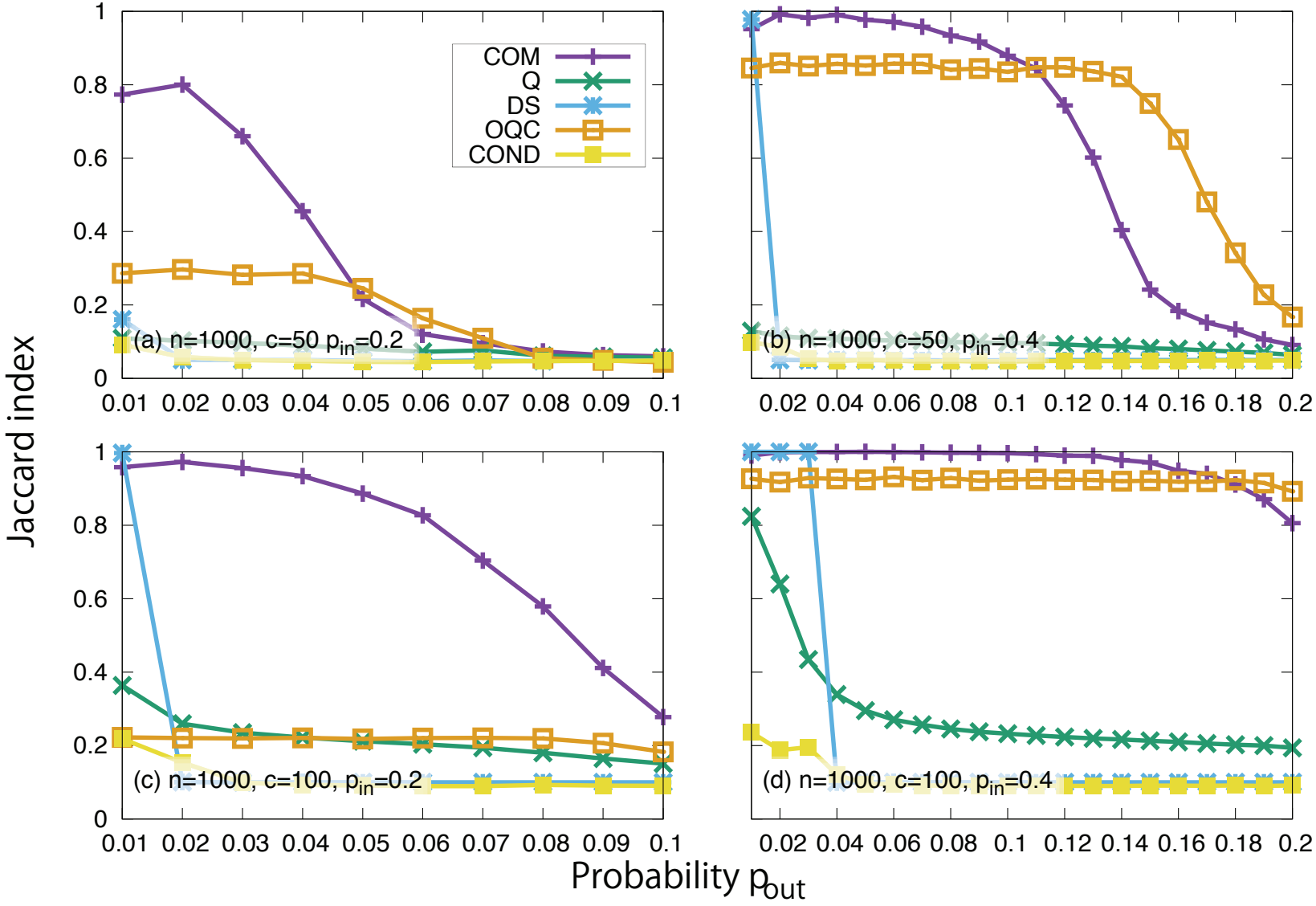
Algorithms:

各評価関数を目的関数として，Greedy peeling + Local search

Evaluation: Jaccard **係数**

$$J(S, S^*) = \frac{|S \cap S^*|}{|S \cup S^*|}$$

単一コミュニティモデル

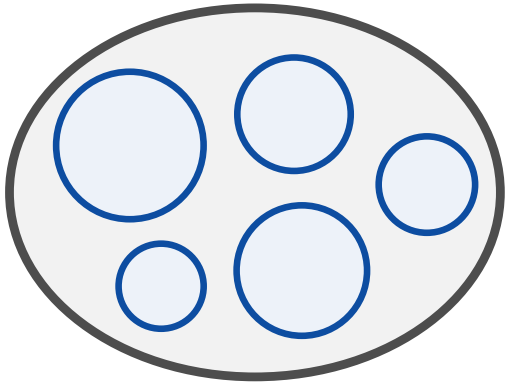


- ほとんどの場合で，COM が最も正確に S^* を検出
- OQC も健闘しているが，パラメータ p_{in} に強く依存

LFR ベンチマーク

サイズが異なる複数のコミュニティをもつグラフ

Parameters:



$G = (V, E)$

- n: **頂点数** $\leftarrow$ 1000 or 5000
- γ: **次数分布のべき指数** $\leftarrow -2$
- β: **コミュニティサイズの分布のべき指数** $\leftarrow -1$
- d: **平均次数** $\leftarrow 20$
- $d_{\max}$: **最大次数** $\leftarrow 50$
- $(c_{\min}, c_{\max})$: **コミュニティサイズの下界と上界** $\leftarrow (10, 50)$ or $(20, 100)$
- μ: **混合パラメータ** $\leftarrow 0.05, \ldots$

Algorithms:

各評価関数を目的関数として，クエリノードに対して Local search

Evaluation:

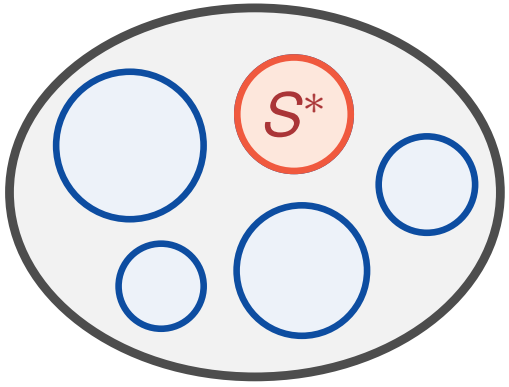
$$J(S, S^*) = \frac{|S \cap S^*|}{|S \cup S^*|}$$

LFR ベンチマーク

サイズが異なる複数のコミュニティをもつグラフ

Parameters:

- n: **頂点数** $\leftarrow$ 1000 or 5000
- γ: **次数分布のべき指数** $\leftarrow -2$
- β: **コミュニティサイズの分布のべき指数** $\leftarrow -1$
- d: **平均次数** $\leftarrow 20$
- $d_{\max}$: **最大次数** $\leftarrow 50$
- $(c_{\min}, c_{\max})$: **コミュニティサイズの下界と上界** $\leftarrow (10, 50)$ or $(20, 100)$
- μ: **混合パラメータ** $\leftarrow 0.05, \ldots$



$G = (V, E)$

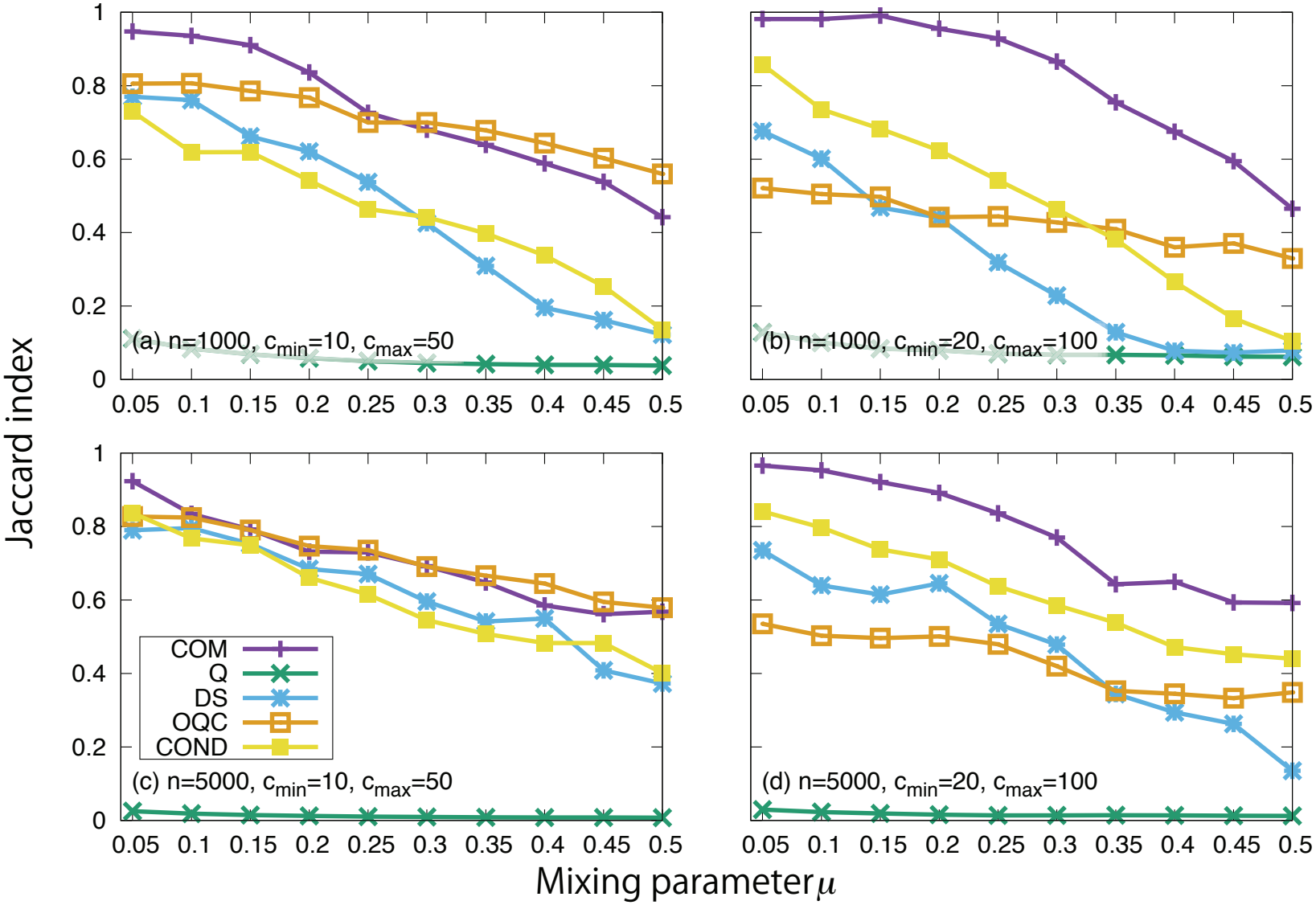
Algorithms:

各評価関数を目的関数として，クエリノードに対して Local search

Evaluation:

$$J(S, S^*) = \frac{|S \cap S^*|}{|S \cup S^*|}$$

LFR ベンチマーク



- コミュニティが小さいとき，COM と OQC が最良の検出性能
- コミュニティが大きいとき，COM が最も正確に S^* を検出

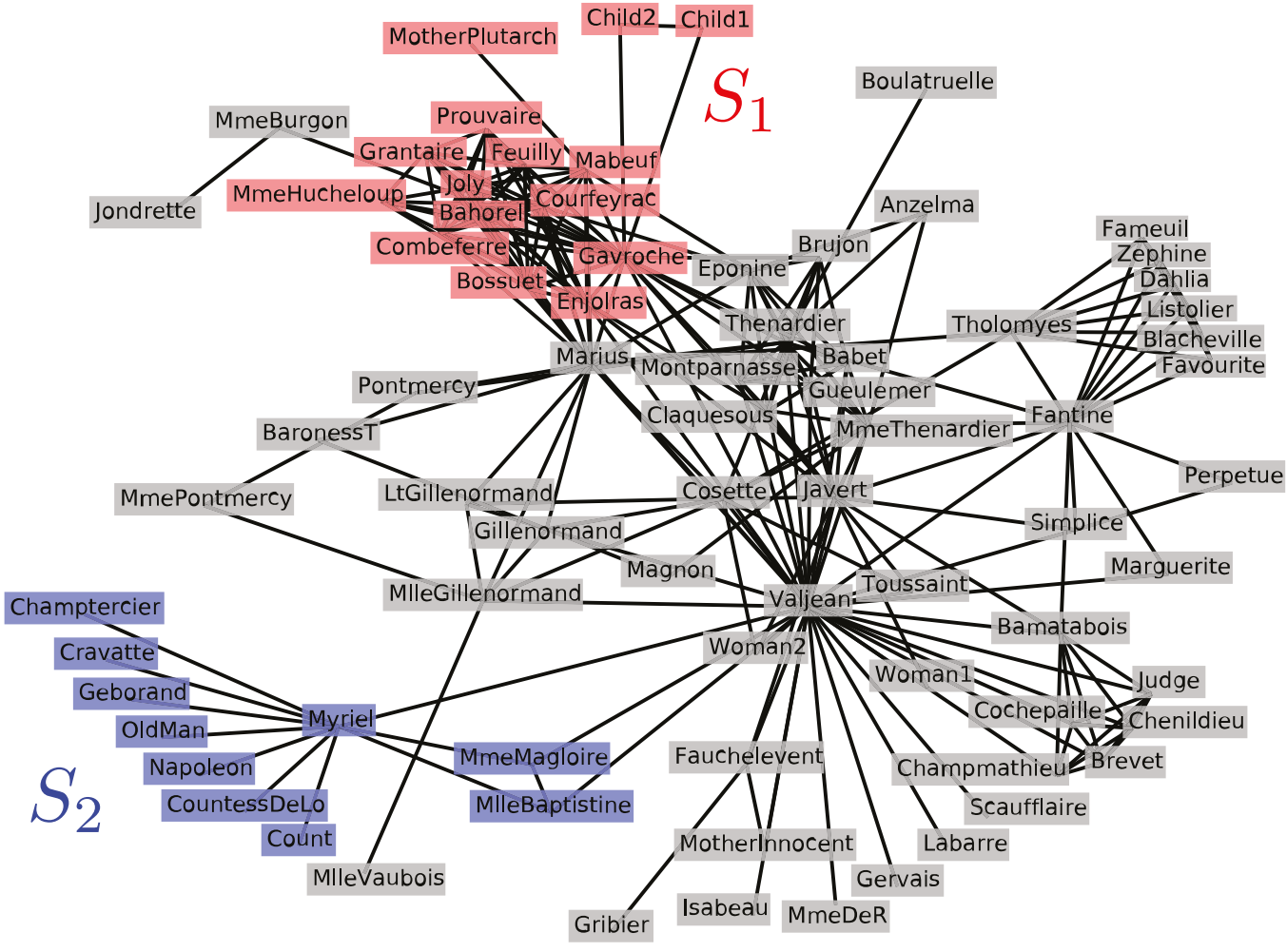
実ネットワーク

Algorithm:
コミュニチュードを目的関数として，Greedy peeling + Local search

Network	com(S)	$\|S\|$	$e[S]$	cut$[S]$
Lesmis	0.578	15	64	23
Polbooks	0.579	41	176	20
Football	0.665	9	36	25
Polblogs	0.515	500	6,904	1,333
AS-22july06	0.569	7,277	12,572	4,510
web-Stanford	0.865	5,749	227,449	11,142
DBLP	0.902	494	15,389	2,765
web-Google	0.939	184	3,172	408
AS-Skitter	0.729	157,216	2,570,493	265,258
LiveJournal	0.957	9,323	1,003,448	27,817

- **ほとんどの場合で** $e[S] \gg \mathrm{cut}[S]$ **(特に** com(S) **が高いときに顕著)**
- 400 **万頂点のネットワークに対しても** 20 **分程度**

Les Misérables ネットワーク



S_1: “ABC の友”　　S_2: 第 1 章の登場人物

おわりに

まとめと今後の課題

頂点部分集合に対する評価関数コミュニチュードを提案

- コミュニチュード最大化問題
- 線形時間ヒューリスティック
- 実験的評価

今後の課題

- 計算困難性の解析
- コミュニチュードを用いた実ネットワークの解析