

【VLDB2011勉強会】

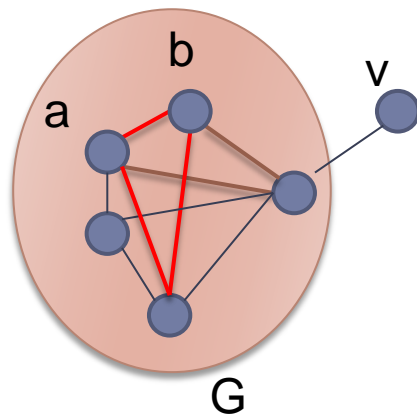
# Session 8: Graph Data

※3つあるGraphDataのうちのひとつ

担当：渡辺知恵美(お茶の水女子大学)

# On Triangulation-based Dense Neighborhood Graph Discovery

- ▶ Nan Wang, Jingbo Zhang, Kian-Lee Tan and Anthony K.H. Tung (National University of Singapore)
- ▶ 目的
  - ▶ グラフの中で密になっているグラフパターンを発見する
  - ▶ DN-Graph (Dense Neighborhood Graph)を定義
    - ▶ 隣り合う二つの点がいくつの隣接点を共有しているかに注目
    - ▶  $G$ のサブグラフ $G'$ が全ての隣接点のペアが $\lambda$ 個以上隣接点を共有しているとき、 $G'$ はDN-Graph  $G'(V', E', \lambda)$ である

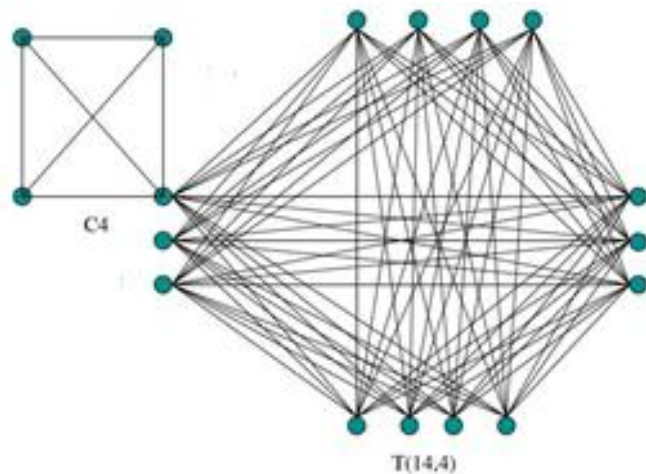


DN-Graphは $\lambda$ において最大のサブグラフ

$$\lambda(V' \cup \{v\}) < \lambda$$

$$\lambda(V' - \{v\}) < \lambda$$

# DN-Graphと他の密パターンとの比較

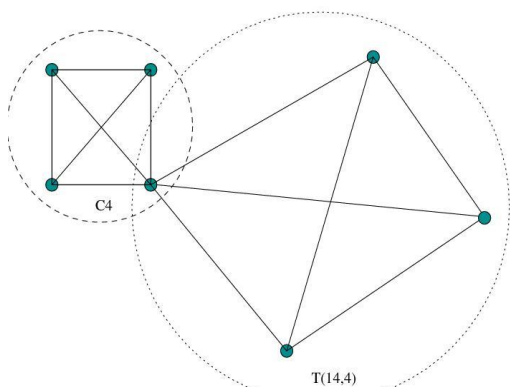


DN-graphはCliqueを一般化したものと考えられる

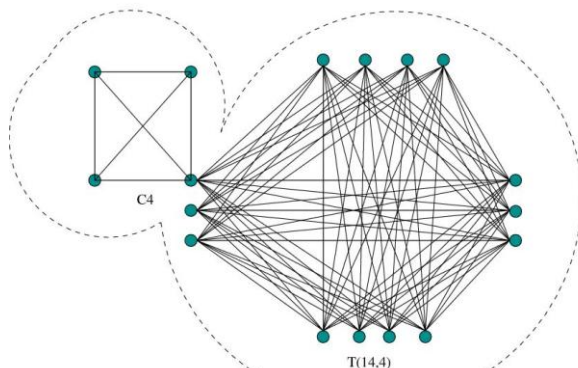
▶ Closed Clique

Quasi-clique

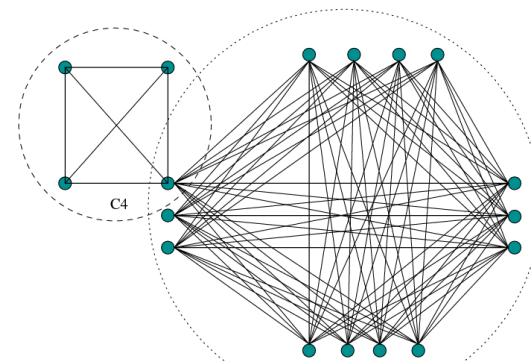
DN-Graph



クリーク部分しか抽出できない



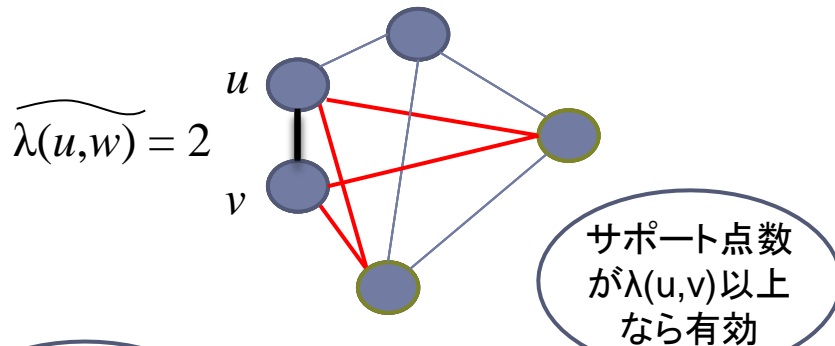
複数の密パターンがつながってしまう



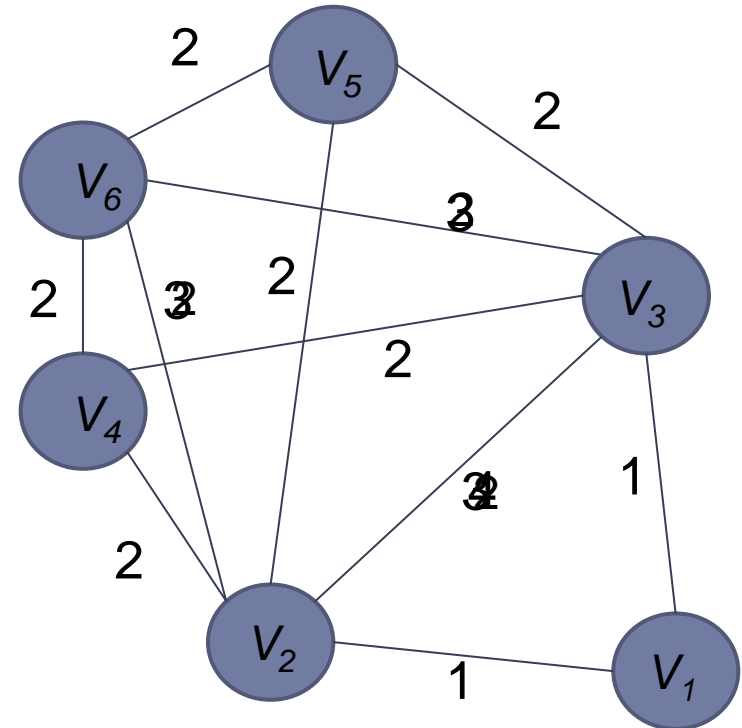
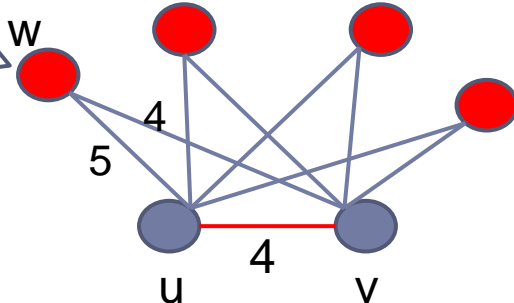
まとまった密パターンが取れる

# Triangulationを使ったDN-graphの抽出

- ▶ DN-Graph miningはNP-Complete
  - ▶ local neighborhood sizeのcountingがボトルネック
  - ▶ 三角形を利用して近似値を求める

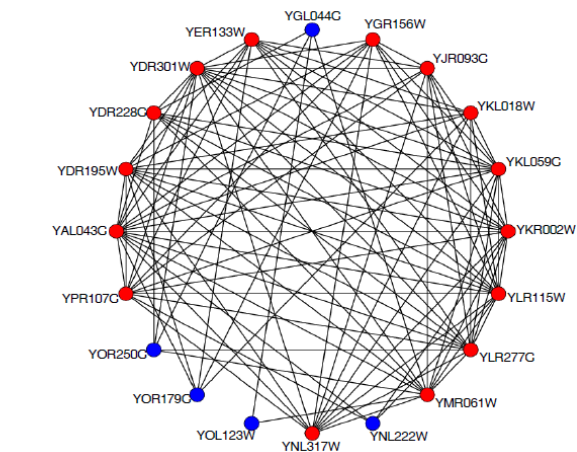
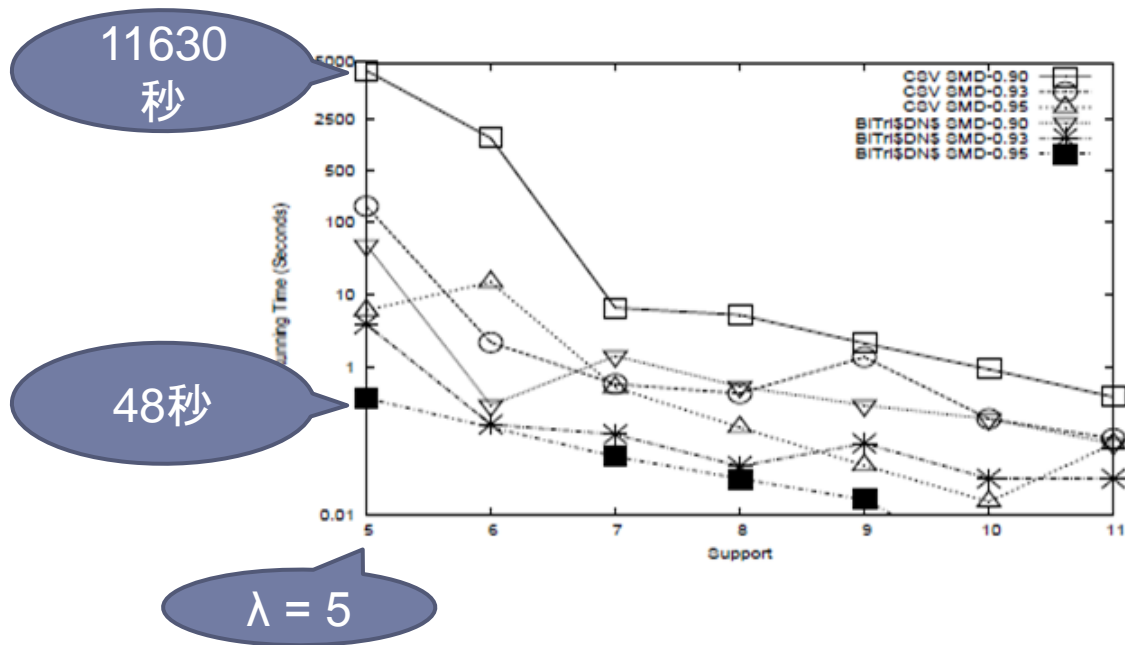


wは $\lambda(u,v)$ をサポートする



# 実験

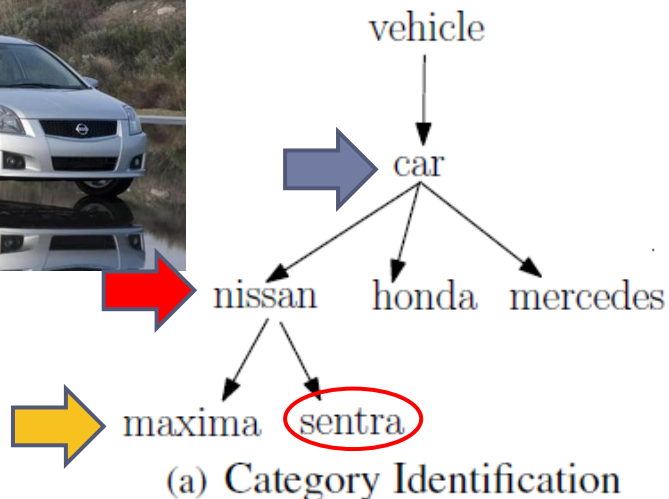
- ▶ CSV (Cohesive Subgraph mining and Visualization)と比較
  - ▶ closed cliqueを発見するアルゴリズム
- ▶ Stock Market Dataset



20-protein complexを検出  
(Protein-Protein Interaction dataより)

# Human Assisted Graph Search: It's Okay to Ask Questions

- ▶ A. Parameswaran, A. D. Sarma, H. Garcia-Molina, N. Polyzotis and J. Widom (Stanford, Yahoo!, UC Santa Cruz)
- ▶ 目的
  - ▶ クラウドソーシングを使ってグラフ検索をする
  - ▶ 数回、人間に「このノードはTarget Nodeか」を訪ね、Yes/Noで答えてもらいながらTarget Nodeにたどりつく



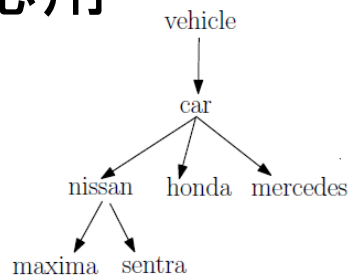
Q1. “ Is this a car?” → A1. “ Yes”

Q2. “ Is this Nissan car?” → A2 “ Yes”

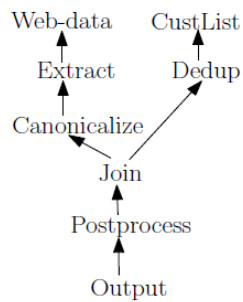
Q3. “ Is this Maxima?” → A3 “ No”

# HumanGS : Human Graph Search

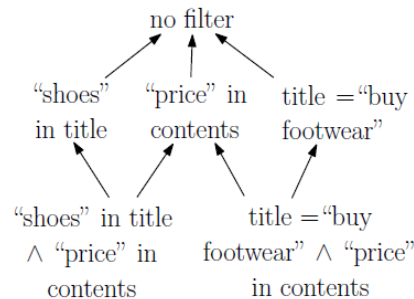
## ▶ 応用



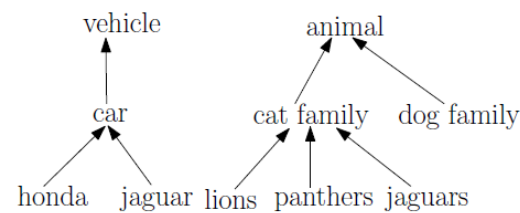
(a) Category Identification



(b) Workflow Provenance



(c) Filter Synthesis



(d) Interactive Search

## ▶ HumanGS問題の分類

- ▶ Dimension 1 : **Single** or Multi
  - ▶ Target Nodeが単一か複数か
- ▶ Dimension 2 : **Bounded** or Unlimited
  - ▶ 質問数が限られているか無制限か(クラウドソーシングのコスト上重要)
- ▶ Dimension 3 : Dag, **Downward-Forest** or Upward-Forest
  - ▶ 対応するグラフの種類

# Single-Bounded : Downward Forest

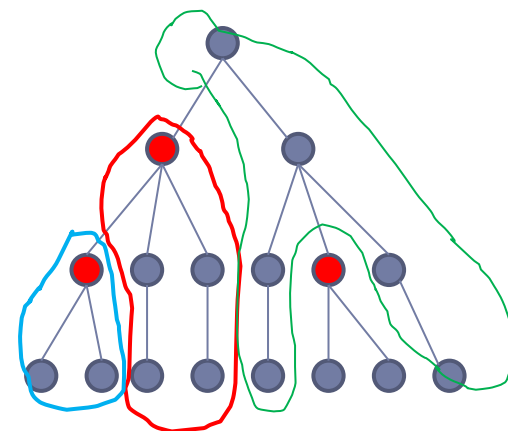
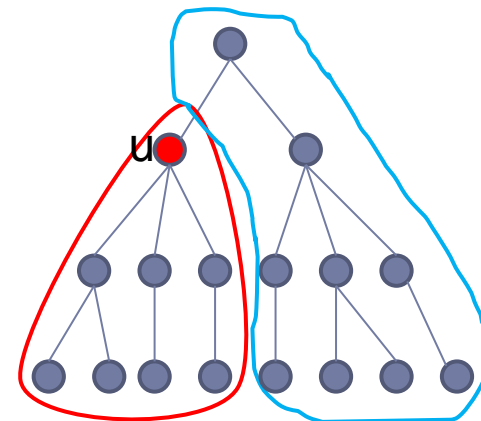
## ▶ 質問の答えによるノードの絞り込み

$$\text{cand}(\{u\}, U^*) = \begin{cases} V - \text{rset}(u) & q(\{u\}, U^*) = \text{NO} \\ V - \text{pset}(u) & q(\{u\}, U^*) = \text{YES} \wedge \text{Multi} \\ \text{rset}(u) & q(\{u\}, U^*) = \text{YES} \wedge \text{Single} \end{cases}$$

## ▶ 要求

- ▶ n回の質問でできる限り絞り込めるようにする
- ▶ n回の質問はあらかじめ用意しなければならない(クラウドソーシングのシステムの仕様上, 対話的に答えを変えられない)
- ▶ n回質問後の候補セット  $N = \{q_1, \dots, q_n\}$  の worst-case を最小限にする

## ▶ グラフの分割問題として考えることができる



# 実験結果

- ▶ DMOZ ( <http://dmoz.org> )を用いて実験
  - ▶ 手動カテゴリ付与型検索エンジン(カテゴリはDownward Forest)
- ▶ 比較
  - ▶ random : ランダムに選ぶ
  - ▶ general-first : 幅優先探索で質問していく

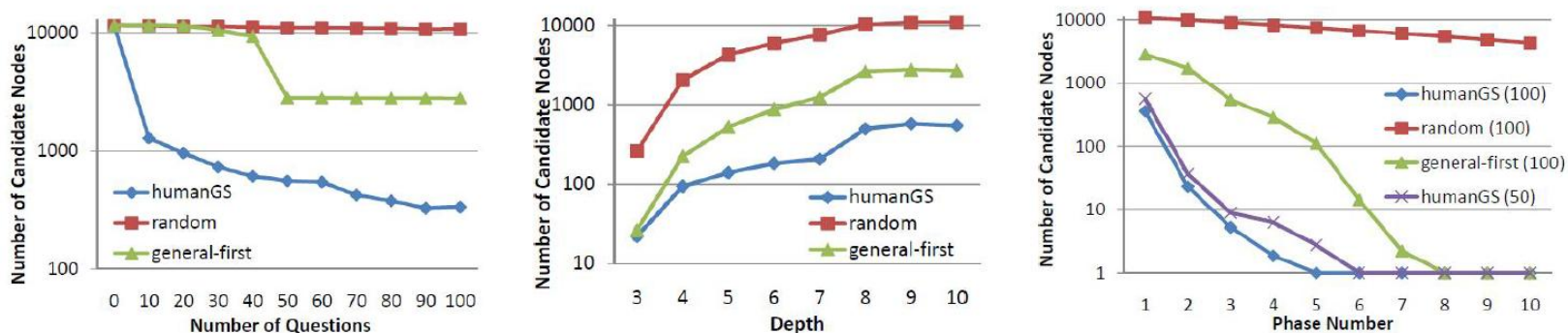


Figure 4: Experiments on (a) varying number of questions asked (b) varying the size of the tree (c) varying the number of phases

# On Querying Historical Evolving Graph Sequences

- ▶ C. Ren, E. Lo, B. Kao, X. Zhu and R. Cheng (Univ. of Hong Kong, Hong Kong Polytechnic Univ.)
- ▶ 目的
  - ▶ Historical Evolving Graph Sequence (EGS)における問合せを効率的に行う
  - ▶ 論文では最短経路探索を対象に述べている

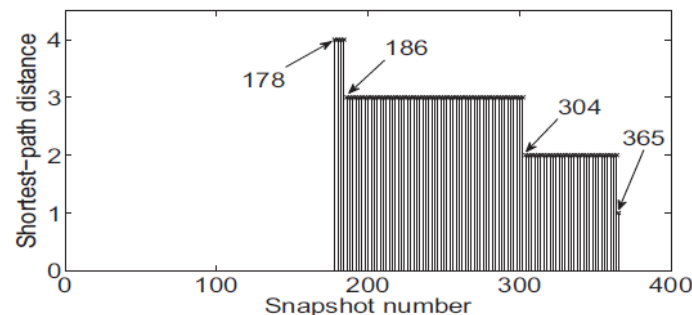
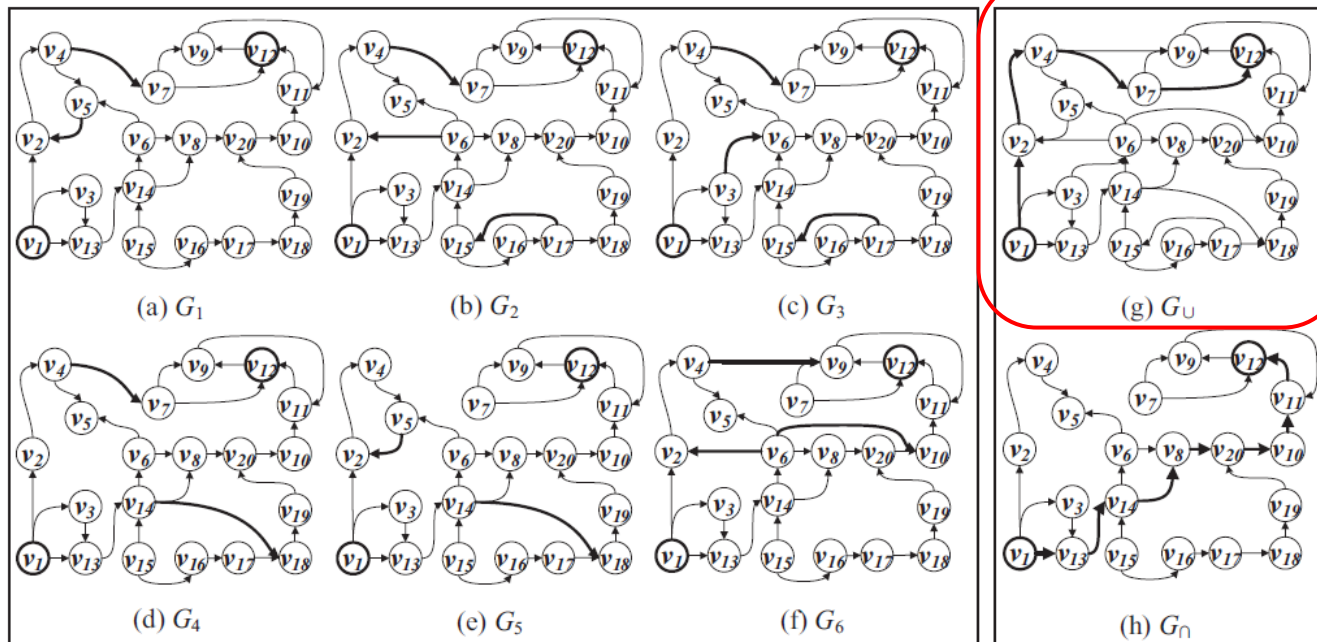


Figure 1: Distance between two users in FACEBOOK friendship graph over a 365-day EGS.

# 枝狩りによる処理の高速化

- ▶ 前処理: グラフ  $G_1, \dots, G_n$  をクラスタリングする
- ▶ クラスタ毎に, クラスタ内の snapshot に対して問合せ処理を行う

$G_U$ : 全スナップショットの合成



$G_\cap$ : スナップショットの共通部分

# 枝狩りルール

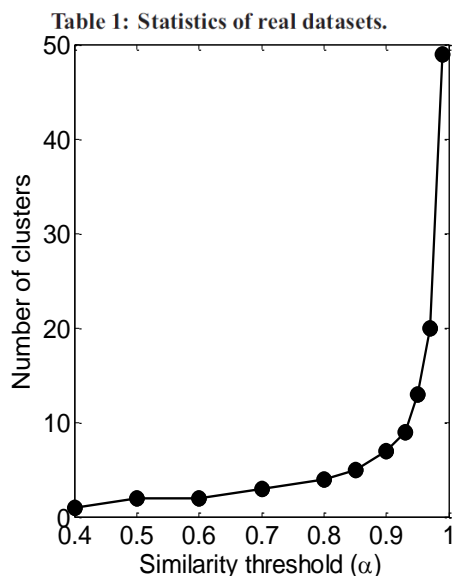
---

- ▶ 想定:  $G_1, \dots, G_n$  のノード  $u, v$  間の最短距離を求める
  - ▶  $\tilde{P}_n(u, v)$ :  $G_n$  における最短距離経路,  $\delta_n(u, v)$  はその長さ
- ▶ Lemma1:  $G_U$  において  $u, v$  が到達不能 ( $\delta_U(u, v) = \infty$ ) な場合
  - ▶  $G_1, \dots, G_n$  でも到達不能
- ▶ Lemma2:  $\delta_U(u, v) = \delta_n(u, v)$  のとき
  - ▶  $G_1, \dots, G_n$  の全ての最短距離経路は  $\tilde{P}_n(u, v)$
- ▶ Lemma3:  $\tilde{P}_U(u, v)$  が  $G_i$  に存在する
  - ▶  $G_i$  の最短距離経路は  $\tilde{P}_U(u, v)$
- ▶ Lemma4:  $G_i$  にあって  $G_n$  にない辺  $e \in \Delta(G_i, G_n)$  に対してその点を通る最短経路  $\Gamma(e)$  の全てが  $\delta_n(u, v)$  より大きい
  - ▶  $G_i$  の最短距離経路は  $\tilde{P}_n(u, v)$
- ▶ Lemma5:  $e \in \Delta(G_i, G_n)$  を通る最短経路  $\Gamma(e)$  の最小値が  $\Gamma^*(e)$ 
  - ▶  $G_i$  の最短距離は  $\Gamma^*(e)$  となり, その経路は  $e' \in \Delta^*(G_i, G_n)$  を通る経路  $\tilde{P}(u, p) \parallel e' \parallel \tilde{P}(q, v)$

# 実験結果

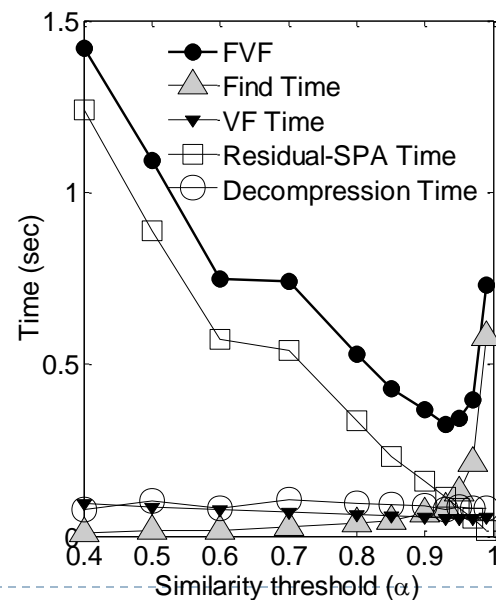
- ▶ Baseline (全てのスナップショット  $G_i$ において最短経路探索を行う)と比較

Dataset	FBFRIEND	YOUTUBE	WIKIPEDIA	FBWALL
Graph Type	undirected	undirected	directed	directed
Vertex Information	user	user	article	user
Edge Information	friendship	friendship	hyperlink	message
Number of Snapshots	365	203	365	365
Snapshot Frequency	daily	daily	daily	daily
$ V $ of First Snapshot	26,249	1,004,777	1,352,623	18,859
$ V $ of Last Snapshot	61,096	3,223,643	1,870,709	42,859
$ E $ of First Snapshot	251,251	8,782,672	19,956,191	90,694
$ E $ of Last Snapshot	905,552	37,048,190	39,953,145	188,869
Average $ges$	99.824%	99.644%	99.905%	99.752%



500 queries

$EGS$	Average Query Time		Speedup
	Baseline	FVF	
FBFRIEND	3.10s	0.34s	9.1
YOUTUBE	52.68s	9.73s	5.4
WIKIPEDIA	104.04s	13.46s	7.7



# Mining Top-K Large Structural Patterns in a Massive Network

---

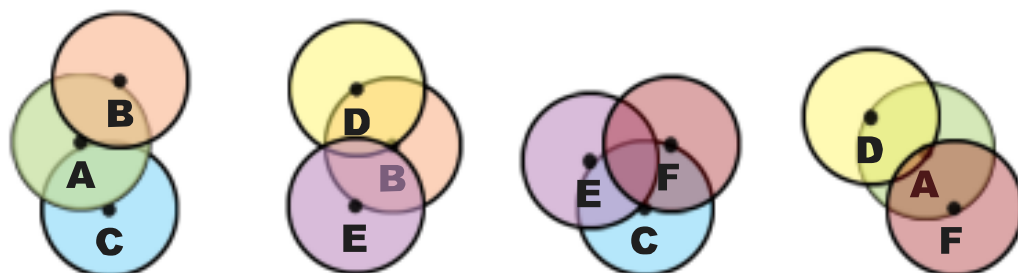
- ▶ Feida Zhu (Singapore Management University), Qiang Qu (Peking University), David Lo (Singapore Management University), Xifeng Yan (UCSB), Jiawei Han (UIUC), Philip Yu (UIC)
- ▶ 目的
  - ▶ 一つの大きなラベル付きグラフから頻出サブグラフを求める
- ▶ 特徴
  - ▶ 頻出サブグラフのうちサイズの大きなものをK個求める
  - ▶ 最大距離が  $r$  の小さな頻出サブグラフを  $r$ -spiderとし,  $r$ -spiderの組合せでサイズの大きな頻出サブグラフを発見する

# r-spiderを用いた大きな頻出サブグラフの探索

- ▶ 従来の頻出サブグラフ検出手法では、頻出パターンを1ノードずつ増やしながら頻度をチェック

6つの10-spider (距離が10の頻出サブグラフ) A~Fで大きなサブグラフを求める事を考える

※ spider間に平均20%の重複があるとする



これらのパターンのサイズ:  $10 * 3 * 80\% = 24$

従来手法

$$24 \times 4 = 96 \text{ steps}$$

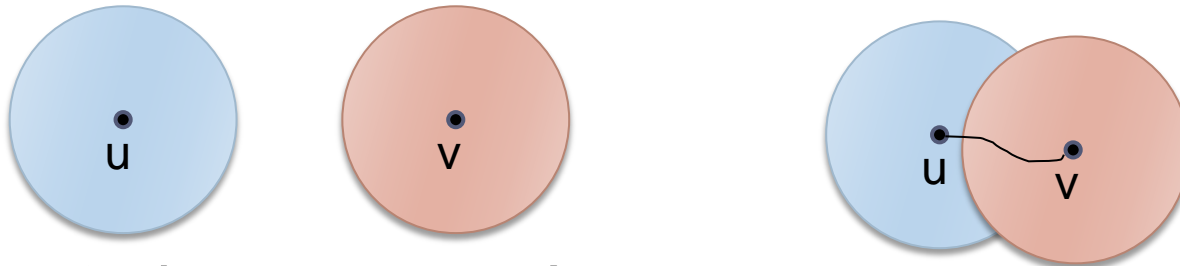
SpiderMine

- 10-spiderを作る  
 $10 \times 6 = 60 \text{ steps}$
  - 組み合わせる  
 $3 \times 4 = 12 \text{ steps}$
- 合計: 72 steps

# r-spiderを使った同型判定

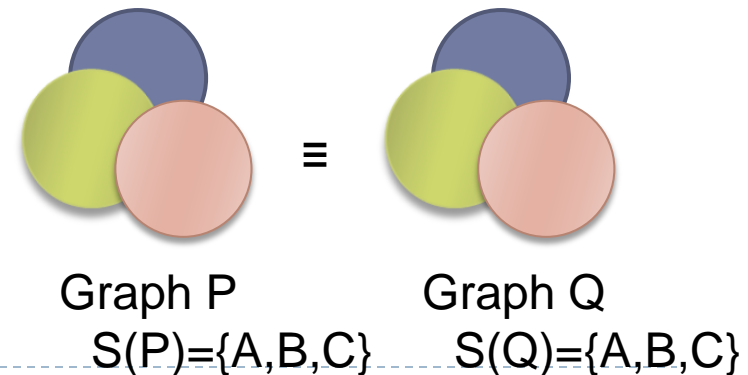
## ▶ 基本的な増やし方

- ▶ あるspiderの基点  $u$  から 距離  $r$  以内に他のspiderの基点  $v$  が位置する spiderパターンがあれば, そのspiderを合成する
- ▶ その時 組合せに用いた spiderを記録しておく



## ▶ spiderを使った同型判定

- ▶ 二つのグラフ  $P$  と  $Q$  はその中に含まれている spider集合  $S(P)$  と  $S(Q)$  が一致していたら同型である可能性が高い



# 実験結果

## ▶ 関連研究

### ▶ SUBDUE, SEuS, MoSSと比較

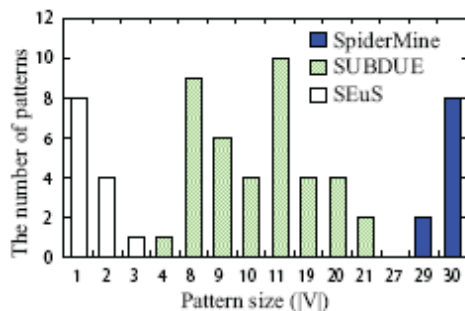


Figure 4: GID 1.

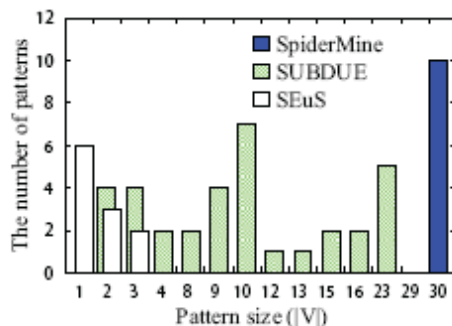
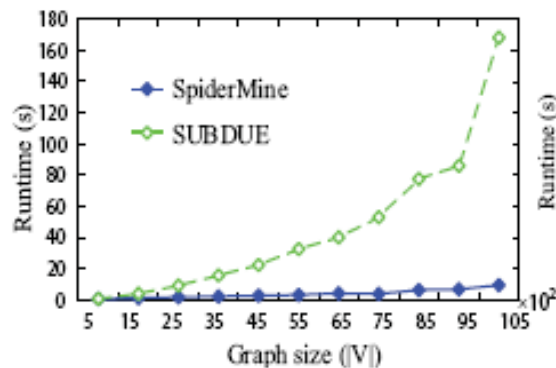
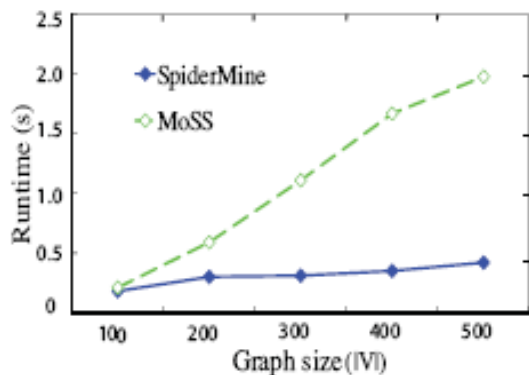


Figure 5: GID 2.

SUBDUEやSEuSは頻度の高い小さなパターンを発見するのに対し、SpiderMineは大きなパターンを見つけることができる

## 実行時間の比較



グラフのサイズに対してSpiderMineは安定した処理速度を保証できる