枝刈り最短路木を用いた
巨大グラフ上の高速な最短距離クエリ

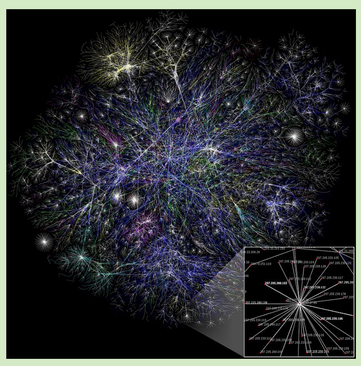
(SIGMOD'13, to appear)

秋葉拓哉(東大)・岩田陽一(東大)・吉田悠一(NII)

どんな研究？

(複雑)ネットワーク上の二点間の
最短距離を高速・省メモリで求める
複雑ネットワークの例:

- Facebookの友達関係
- TwitterのFollow関係
- ウェブページ間のリンク



どの経路が最短？
(出展: en.wikipedia.org/wiki/Complex_network)

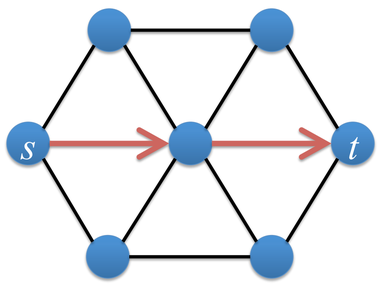
最短距離の計算の応用

- 友達検索 (共通の友人が多い人は上位へ)
- ウェブ検索のランキング (興味に近いページは上位へ)
- 複雑ネットワークの解析 (どの頂点が重要か)
- 代謝経路の解析 (タンパク質の合成経路)
- 地図中の経路探索

研究内容

問題設定

グラフ$G=(V, E)$が与えられ
1. 前処理により索引Iを構築
2. 索引Iを用いて距離$d(s,t)$を計算



スケーラビリティ
索引構築時間
索引サイズ

⇔

クエリ性能
クエリ処理時間

距離ラベリング

索引構築: 各頂点について以下の様な配列を前計算

頂点1:

頂点	1	4	5	7	10
距離	0	3	2	4	5

頂点1と10の距離は5

頂点2:

頂点	2	4	6	12
距離	0	1	5	3

頂点3:

頂点	3	2	4	6	7
距離	0	5	4	7	2

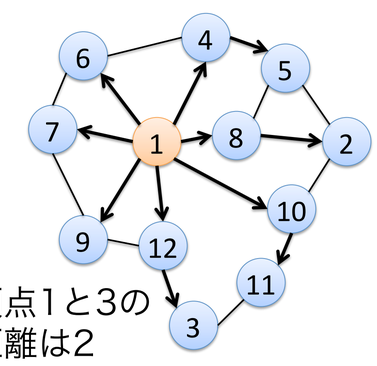
⋮

クエリ計算
1から3の距離は？
- 経路1-4-3: 3+4=7
- 経路1-7-3: 4+2=6

よって6と答える。

幅優先探索(BFS)による素朴な手法

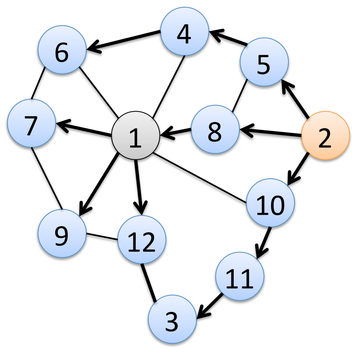
頂点1からBFS



頂点1と3の距離は2

全てのvに対して
$d(1, v)$を求める

頂点2からBFS



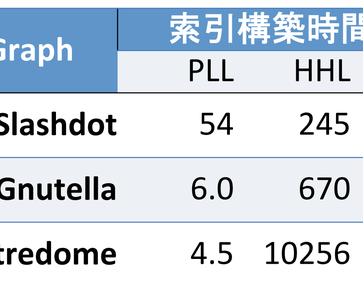
全てのvに対して
$d(2, v)$を求める

…

正しく距離が求まるが非効率的
索引構築時間$O(|V||E|)$, 索引サイズ$O(|V||E|)$, クエリ時間$O(|V|)$

BFSの枝刈り(本研究)

頂点2からBFS



頂点6までの距離 = 3 = $d(2,1)+d(1,6)$
- 6以降は全て1を経由するのが最短
- 6以降の探索は打ち切る

他の頂点からのBFSも同様に枝刈り
BFSの始点は次数(繋がっている枝数)順に選ぶ

実験結果

Graph	Network	$\|V\|$	$\|E\|$
Gnutella	Computer	63 K	148 K
Slashdot	Social	82 K	948 K
Notredome	Web	326 K	1.5 M
MetroSec	Computer	2.3 M	22 M
Holleywood	Social	1.1 M	114 M
Indochina	Web	7.4 M	194 M

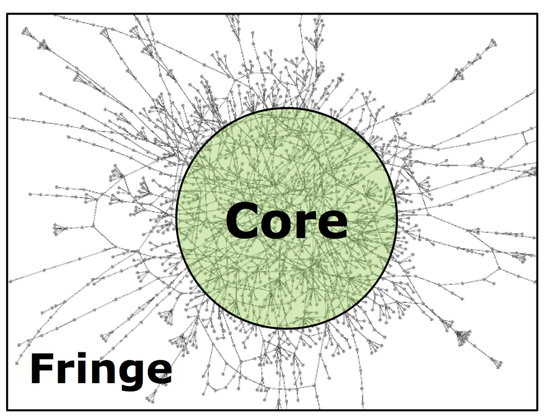
既存手法よりも
二桁大きい
ネットワークに対応

PLL [This work],
HHL [Abraham+'12],
TD [Akiba+'12]

Graph	索引構築時間 (s)			索引サイズ (MB)			クエリ時間 (μs)		
	PLL	HHL	TD	PLL	HHL	TD	PLL	HHL	TD
Slashdot	54	245	209	209	380	68	5.2	11	19
Gnutella	6.0	670	343	48	182	83	0.8	3.9	12
Notredome	4.5	10256	243	138	64	120	0.5	0.4	39
MetroSec	108	DNF	DNF	2.5 GB	-	-	0.7	-	-
Holleywood	15164	DNF	DNF	12 GB	-	-	15.6	-	-
Indochina	6068	DNF	DNF	22 GB	-	-	4.1	-	-

効率的な理由: 複雑ネットワーク = 中心部 + 周辺部

中心部: 殆どの最短パスが通る
⇒ 枝刈りに有利
周辺部: 木に似た構造
⇒ 本手法の効率性を理論的に証明



連絡先：吉田悠一／ 国立情報学研究所　情報学プリンシプル研究系　特任助教
TEL : 03-4212-2571　　Email : yyoshida@nii.ac.jp