ICDE2013勉強会
R29 Large Graph Indexing

塩川 浩昭@NTT

2013/06/29

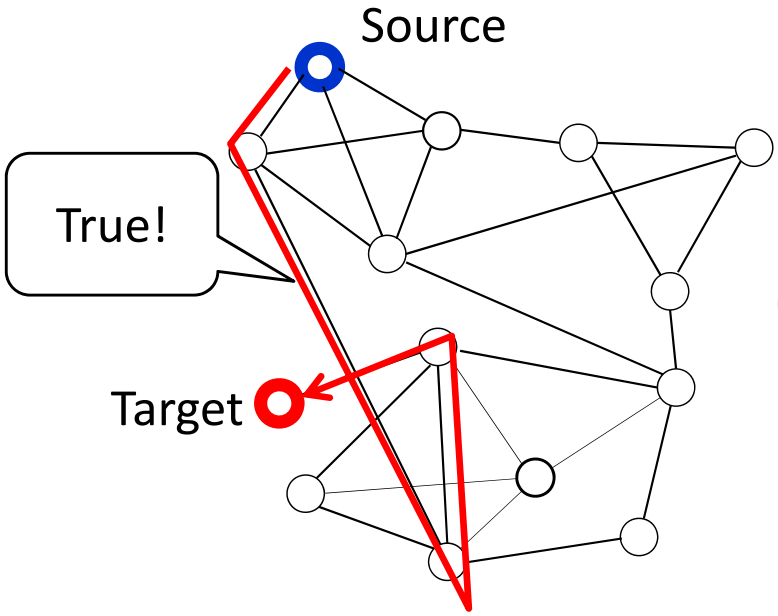
FERRARI: Flexible and Efficient Reachability Range Assignment for Graph Indexing

Stephan Seufert[1]
Avishek Anand[1]
Srikanta Bedathur[2]
Gerhard Weikum[1]

[1] Max Planck Institute for Informatics, Germany
[2] IIIT Delhi, India

背景

- 大規模なグラフ上でReachabilityを確認できるようにする



これまでのアプローチ
- 全域木を用いた索引構造を利用し高速化

↓

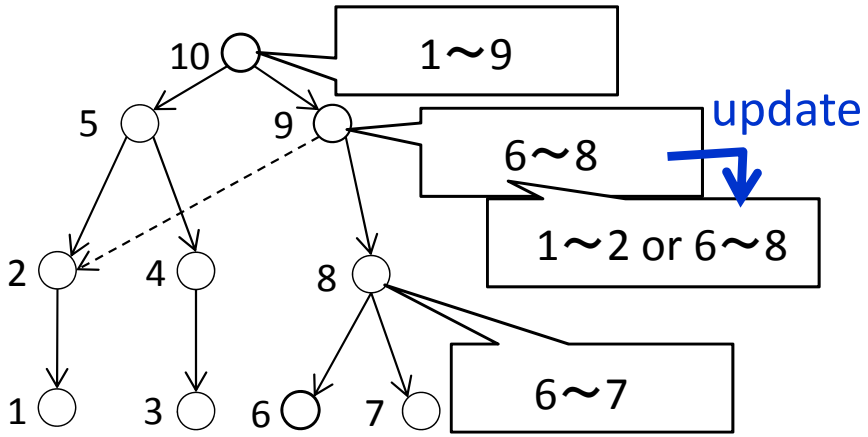
データの大規模化に伴い空間計算量に制約が発生

→空間計算量をコンパクトにしつつ，できるだけ高速化できる様な索引構造を構築

- Contributions
  - 空間計算量を適応的に調整可能な索引構造を提案**←本発表のトピック**
    - Exact reachability intervalとapproximate reachability intervalの組合せで実現
  - 効率的に索引構造を構築するアルゴリズムの提案
  - 索引サイズ制約下において，既存手法GRAILよりも高速に動作

Interval Indexingを用いたベースライン手法

- Exact Interval Indexing
- グラフから全域木を構築
- 後置順にノードIDを割り振り，各ノードの子ノードのノードIDの範囲を索引として保持
- DAGとなるエッジを追加してindexをupdate

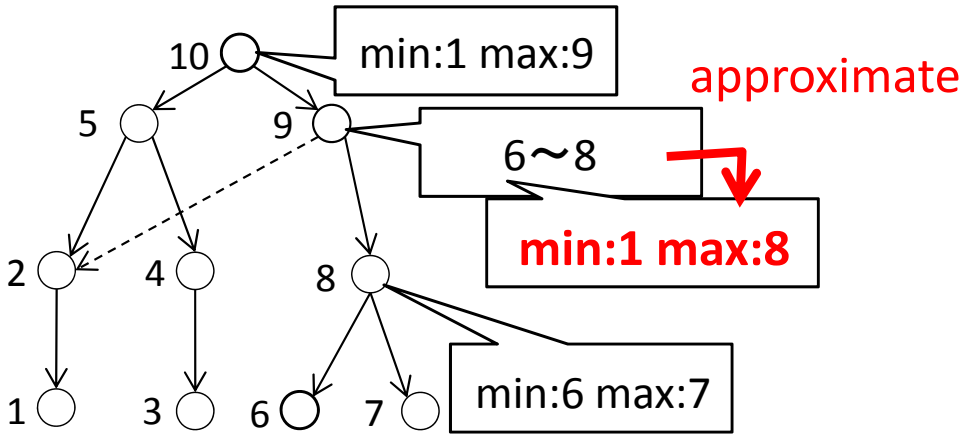


- Reachability queryがO(1)で解くことができる

  Source=9, Target=6の場合

  ノードID9の索引に6が含まれるためTrue
- 空間計算量が大きくなる

- Approximate Interval Indexing
- 事前にExact Interval Indexを構築
- 複数のInterval Indexを束ね，indexの最小値と最大値を新たなIndexとして保持



- 空間計算量を小さくできる
- False Positiveな解が生じるため速度が低下

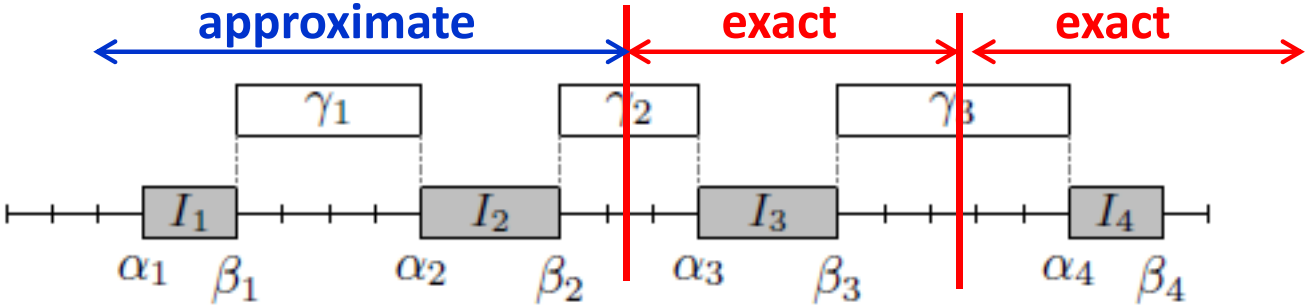
Source=9, Target=5の場合

ノードID9の索引に6が含まれるためTrueにみえるが，本当はFalse

提案手法：FERRARI

- Exact/Approximate interval indexを混合した索引構造を構築
- 事前に与えられた空間計算量の上限値を満たす範囲で，可能な限りExact interval indexを構築する
  - コストモデルによる最適化
    - $\Gamma_{k-1}{}^{*} := \arg \min_{G \subset \Gamma, |G| < k} \sum (1 - \eta_I{}') |I|$

Approximateの索引範囲を小さくするようにk個の索引グループに分割



- 動的計画法もしくは，貪欲法を用いてコストモデルを最適化する

図は論文より引用

評価結果

索引構築時間

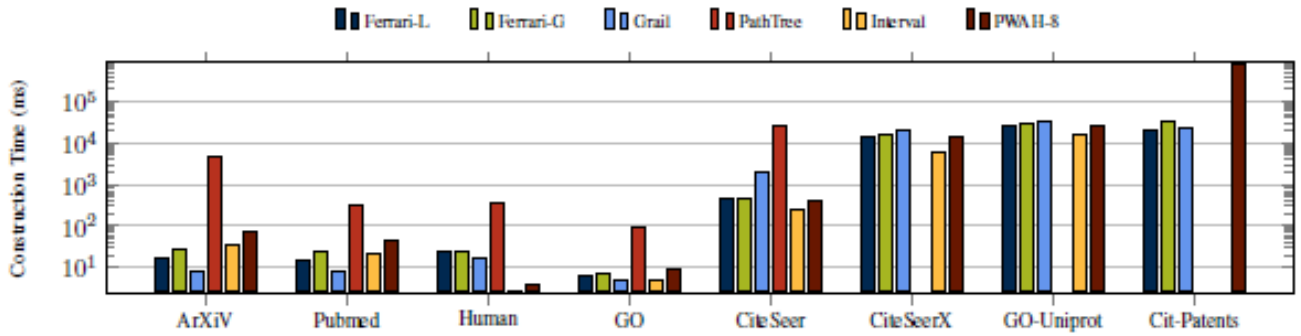


Fig. 1. Index Construction Time

索引サイズ

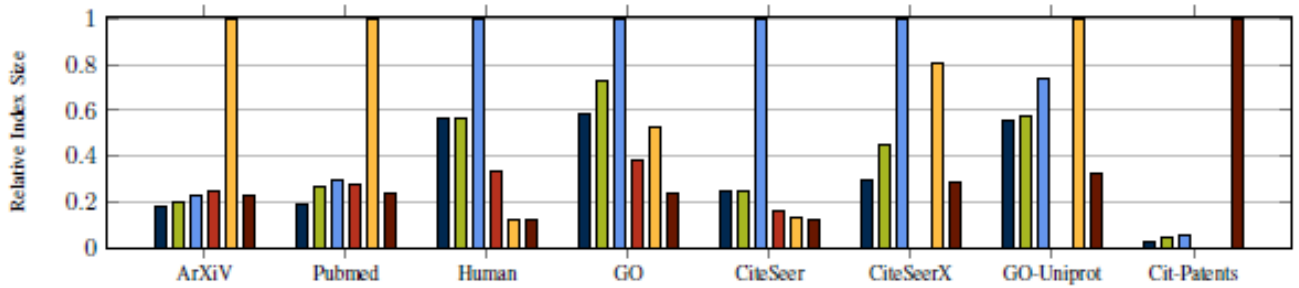


Fig. 2. Index Space Consumption

探索速度

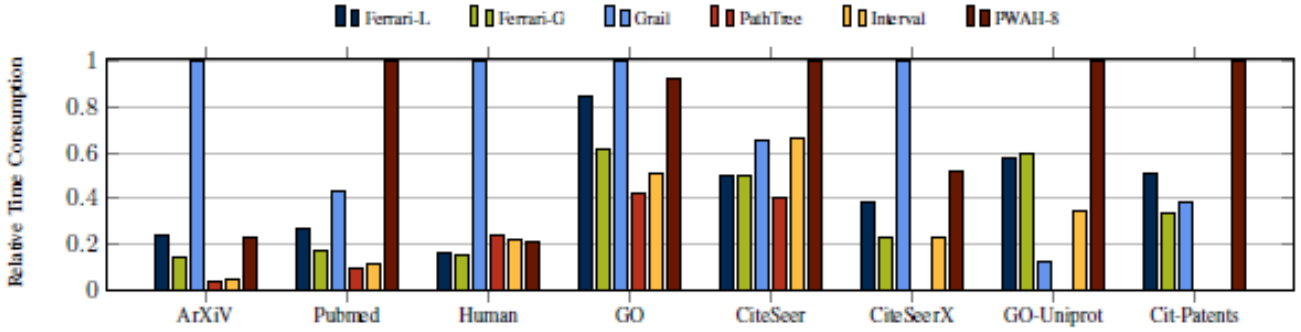


Fig. 3. Query Processing Times for 100k Random Queries

図は論文より引用

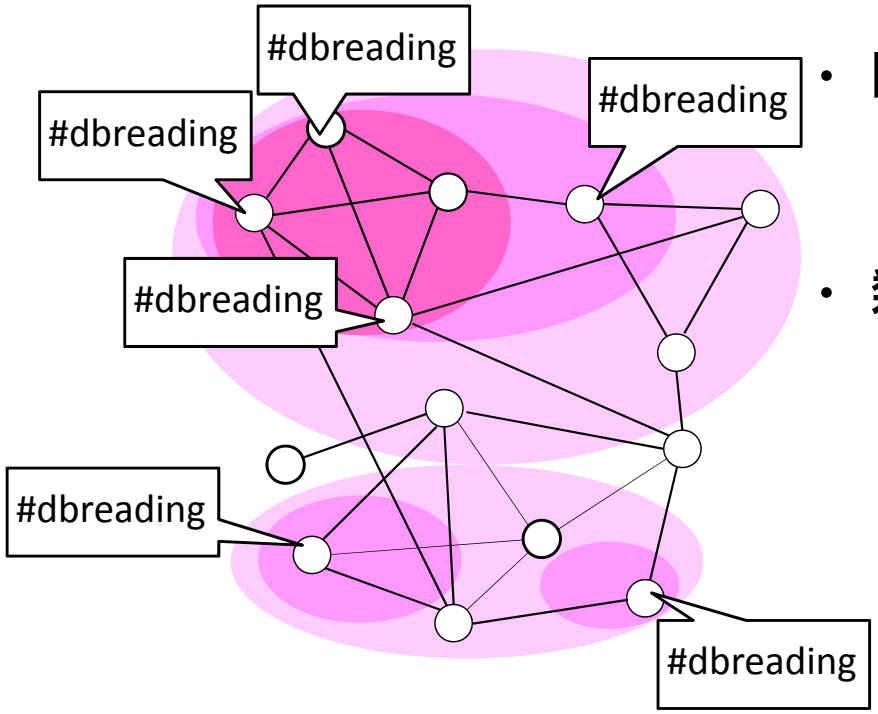
gIceberg: Towards Iceberg Analysis in Large Graphs

Nan Li[1]
Ziyu Guan[1]
Lijie Ren[1]
Jian Wu[2]
Jiawei Han[3]
Xifeng Yan[1]

[1] *Department of Computer Science, University of California at Santa Barbara*
[2] *College of Computer Science & Technology, Zhejiang University*
[3] *Department of Computer Science, University of Illinois at Urbana-Champaign*

背景

- グラフ上で特定の属性におけるIceberg検出を実現する

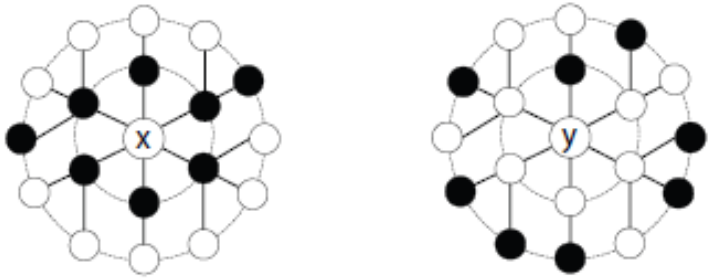


- **同一の属性qが局所的に集中した部分グラフを検出**

  時系列データ処理や，多次元データ分析における
  スパイク検出と同じようなことをグラフでやりたいという話

- **類似研究：Yanらによる研究[ICDE'10]**

  hホップまでのノードが持つ属性qの値をSUM/AVGする



(a) x's 2-hop neighborhood　(b) y's 2-hop neighborhoo d

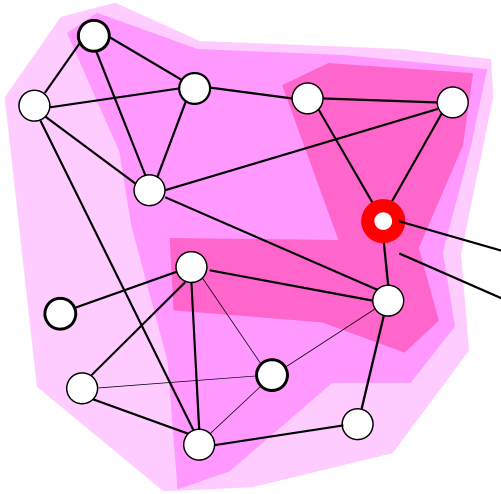
距離をIcebergに反映することが難しい

- Contributions
  - 新しいコンセプトGraph Icebergという課題を提起
  - gIcebergと呼ばれる手法を提案し，高い検出精度とスケーラビリティを達成
  - gIcebergにおいて2つの処理最適化手法を提案

図は論文より引用

提案手法：gIceberg

- Given： q (問合せ対象属性), θ (Iceberg判定の閾値)
- Personalized PageRankを用いた重要度計算



Preference distributionで与えたノード集合に対する
各ノードの重要度や影響力を計算する手法

$$\boldsymbol{p} = (1-c)\boldsymbol{M}\boldsymbol{p} + c\boldsymbol{s}$$

Preference node

Importance of nodes

$\boldsymbol{M}$：Transition matrix
c：Restart probability
$\boldsymbol{p}$：Importance of nodes for $\boldsymbol{s}$
$\boldsymbol{s}$：Preference distribution

- 各ノードの問合せqに対するスコアの計算

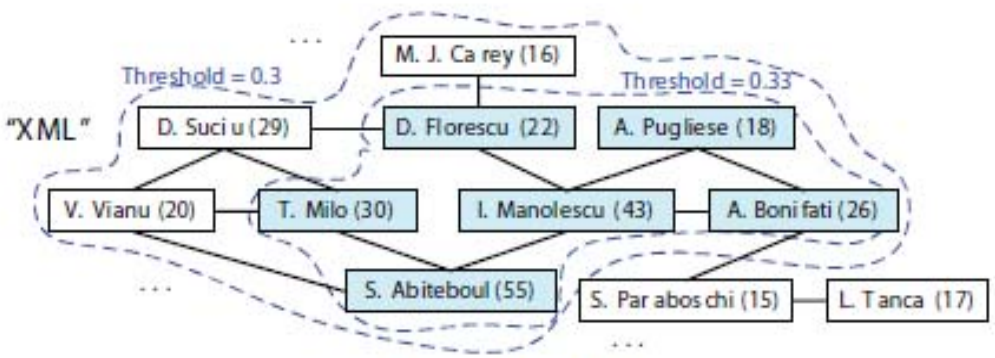
  Personalized PageRankで求めた各ノードの重要度を加算し，ノードvのスコアとする

$$P_q(v) = \sum_{x|x\in V, q\in L(x)} p_v(x)$$ → **$P_q(v) \geq \theta$となるノードvをIcebergとして検出**

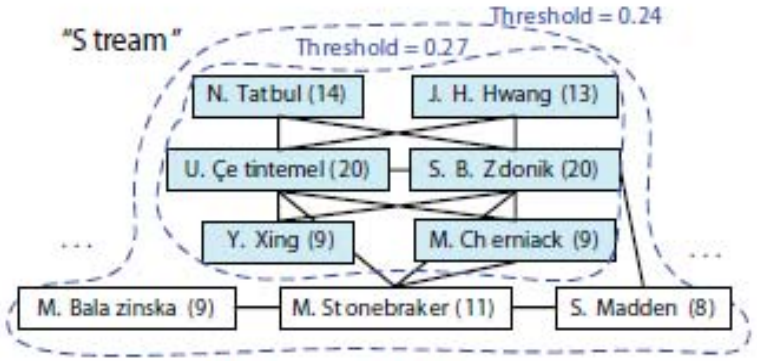
- 計算量削減のため，2つのApproximationを提案
  - Forward Aggregation(FA) & Backward Aggregation(BA)（詳細は論文参照）

評価結果

- Case study：DBLPの共著ネットワーク

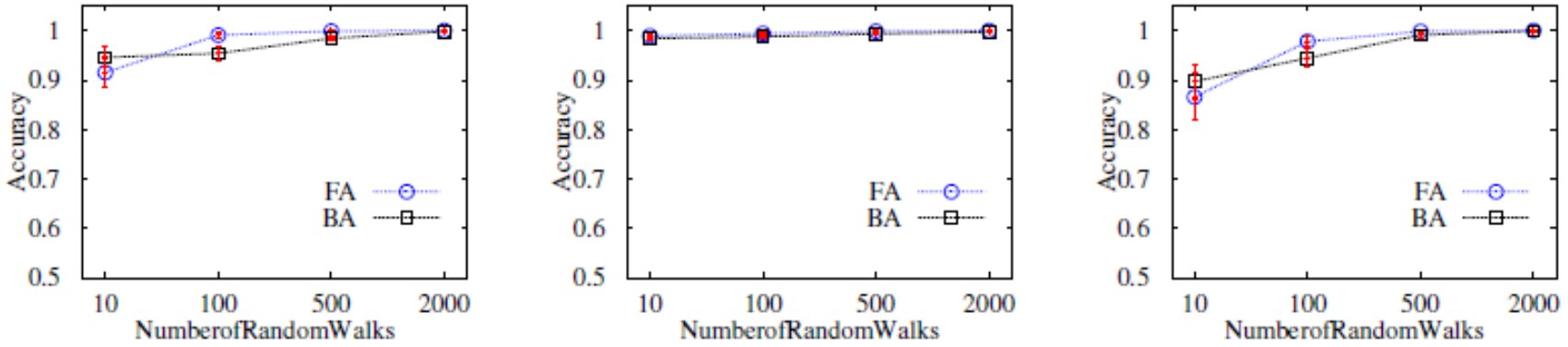


(a) Keyword: "XML"



(b) Keyword: "Stream"

- Real world datasetsに対しても正確性の高い結果を出している



(a) *Customer Subgraph* (b) *DBLP 2010* (c) *R-MAT 3K*

Fig. 9. Random Walk-Based Aggregation Accuracy

図は論文より引用

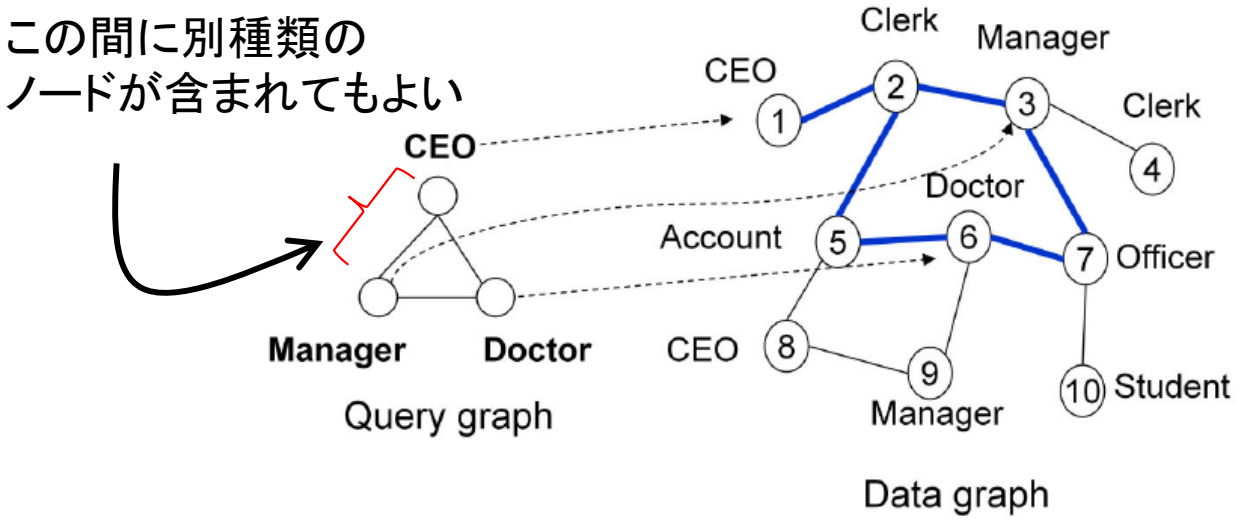
Top-k Graph Pattern Matching over Large Graph

Jiefeng Cheng[1,2]
Xianggang Zeng[1]
Jeffrey Xu Yu[3]

[1] Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China
[2] Shenzhen Key Laboratory of High Performance Data Mining, Shenzhen, 518055, China
[3] The Chinese University of Hong Kong, Hong Kong, China

背景

- 大規模なグラフから特定のグラフパタンを発見する
  - クエリ中のノード間に別のノードが含まれることを許容したcyclic graphが対象



解1 {1, 3, 6}
解2 {8, 3, 6}
解3 {8, 9, 6}
⋮

解候補が大量に出現

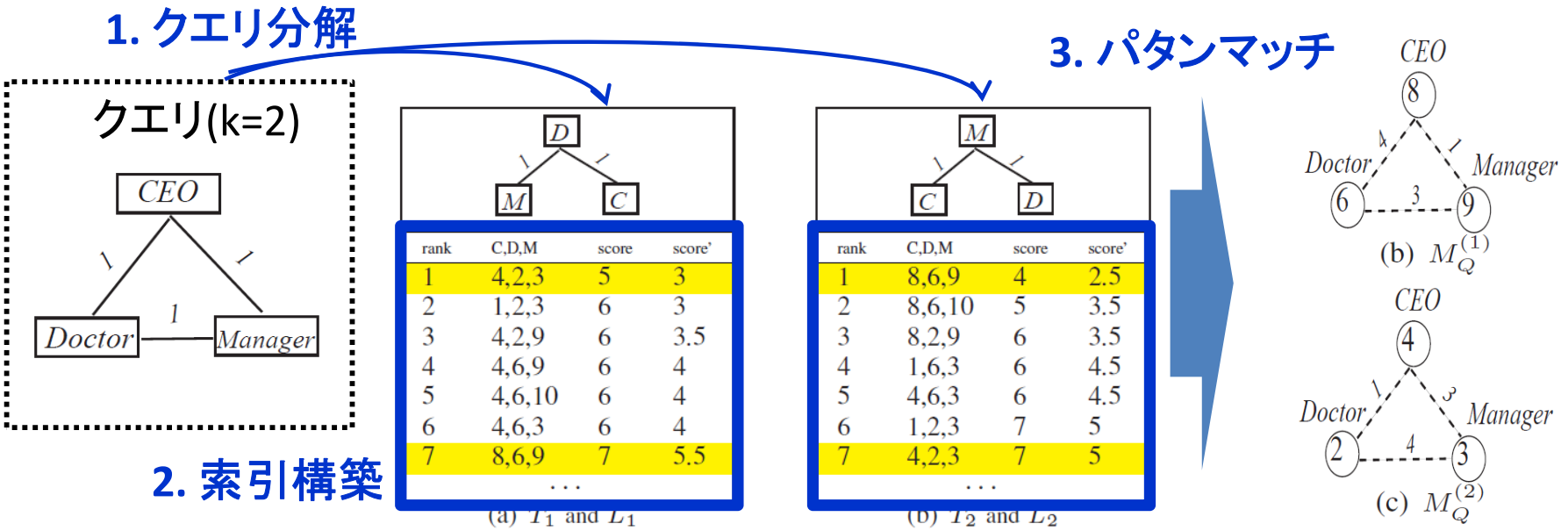
解の持つ重みの小さい順に
Top-*k*個のみ取得したい

- Contributions
  - Cyclic graph をクエリとしたtop-k graph pattern matching(kGPM)に着眼
  - 複数の全域木からなる索引の多次元表現を提案**←本発表のトピック**
  - コストモデルと問合せ最適化手法を提案

図は論文より引用

Gouらによるtwig-pattern matching[SIGMOD'08]を用いたベースライン手法

1. クエリを2つ以上の全域木に分割
2. 各全域木にマッチする最小全域木をグラフから探索し索引を構築
3. 各索引をjoinして，クエリにマッチするパタンをk個探索

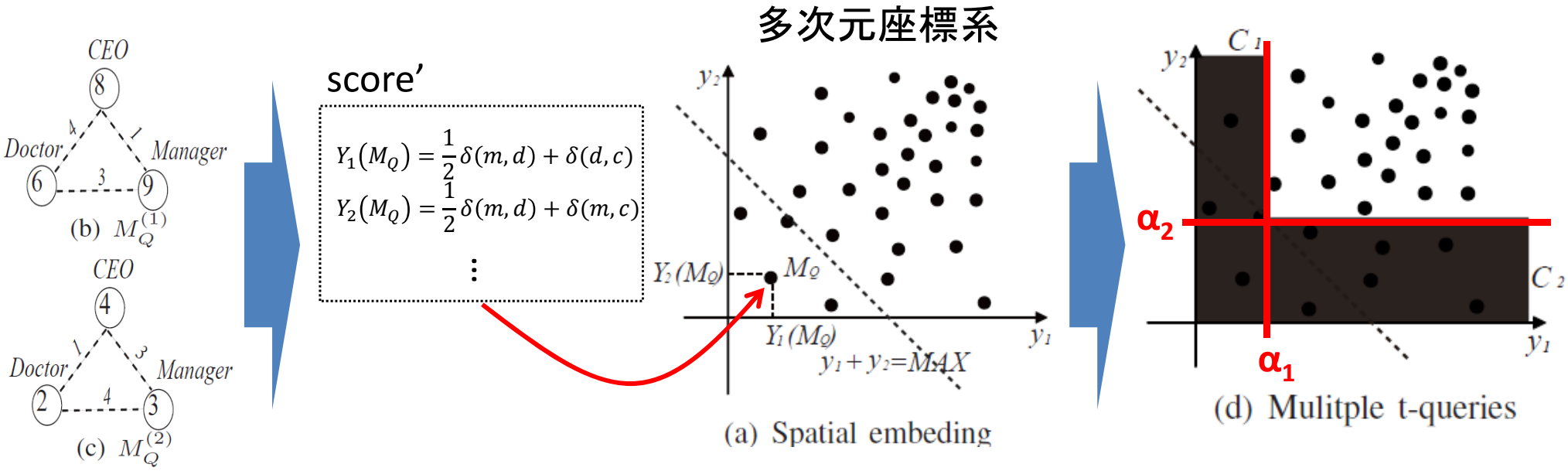


- 計算量を削減するための課題
  1. 複数の索引の中から，最適解を持つ索引の組合せをどう見つけるか？
  2. 与えられた索引の中からいかにして効率よく解を探索するか？

図は論文より引用

提案手法(多次元表現・問合せ最適化)

- 索引を多次元座標系を用いた問合せ最適化
  - 解候補に対する各索引の寄与成分score'を計算し，座標系に解候補を射影
  - 各軸に$\alpha_1 + \alpha_2 = MAX$（*MAX*: k番目の解のscore'合計値）となるしきい値αを与えて，解を絞込み探索
  - 各軸のαはコストモデルにラグランジュの未定乗数法を適用し算出（詳細は論文参照）



図は論文より引用

評価結果

- 適切な数の索引構造を選択し，処理時間の最適化に成功

k	#t-lists in plan		optimization time(ms)		execution time(ms)				
	Greedy	Exaustive	Greedy	Exaustive	1 t-lists	2 t-lists	3 t-lists	4 t-lists	5 t-lists
$k = 5$	2	2	610	9,251	4,936	10,856	10,836	11,776	13,464
$k = 50$	3	3	680	9,294	30,4714	20,424	12,641	12,351	14,265
$k = 500$	4	4	614	9,294	307,919	101,236	20,061	14,804	17,889
$k = 1,000$	4	4	618	9,286	310,020	171,466	25,896	16,902	20,958

- いずれのクエリについても最も早く処理を実現している

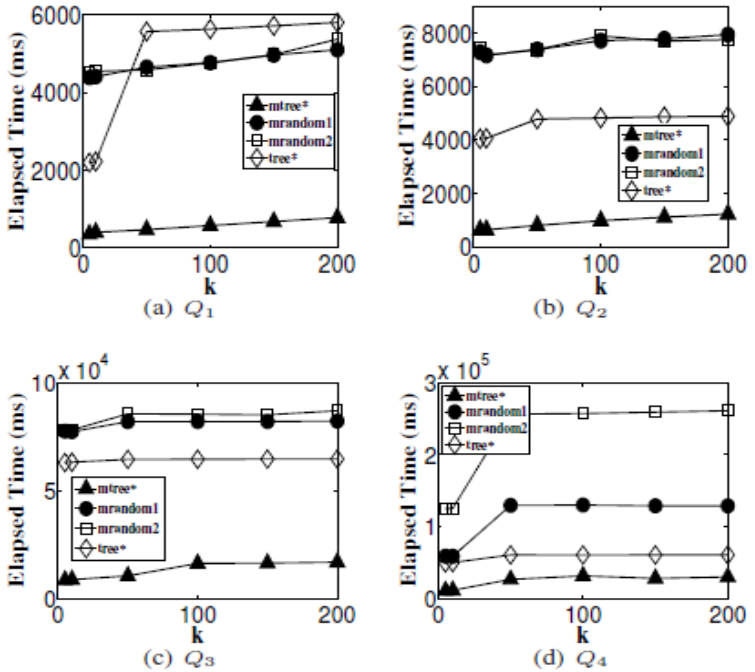


Fig. 10. Performance of Our Cost-based Optimization-Real Dataset

図は論文より引用