READING:

Online Topic-aware Influence Maximization Queries

Cigdem Aslay[1] Nicola Barbieri[2] Francesco Bonchi[2] Ricardo Baeza-Yates[2]

[1]Web Research Group
University Pompeu Fabra
Barcelona, Spain
aslayci@acm.org

[2]Yahoo Labs
Barcelona, Spain
{barbieri,bonchi}@yahoo-inc.com
rbaeza@acm.org

from EDBT/ICDT 2014

SPEAKER: KAZUHIRO INABA

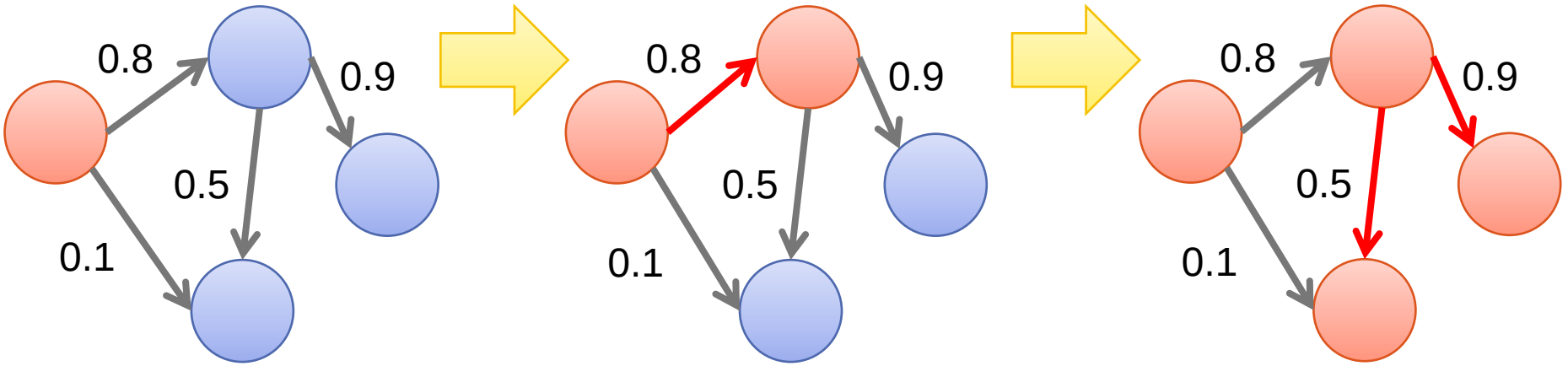
WHAT THE PAPER IS ABOUT?

- **Influence Maximization**
  - ソーシャルグラフ上を口コミで情報が伝搬していく。できるだけ大勢に情報を広める発信源 “seed set” を求めたい
- **Topic-aware**
  - 情報の伝わる確率が話題ごとに異なる
- **Online**
  - 次々と違う話題の情報が飛んでくるので、その都度、よい seed set を、あらかじめ用意しておいた前計算を使って一瞬で求める

INDEPENDENT CASCADE (IC) MODEL

[Kempe, Kleinberg, and Tardos; KDD 2003]

有向辺 u → v それぞれに
確率 $p_{u,v}$ が割り当てられている



u が情報を最初に知った直後のターン
（1回だけ）に隣接する v に確率 $p_{u,v}$ で
情報が伝わる

TOPIC-AWARE IC (TIC) MODEL

- Z 個の *Topic* がある。
- 伝わる情報 (論文中では *Item*) はZ次元ベクトル

  $\gamma = (\gamma^1, \gamma^2, \ldots, \gamma^Z)$

  where $0 \leqq \gamma^k \leqq 1,\ \Sigma\gamma^k = 1$
- 各辺にZ個の確率 $(p^1_{u,v}, \ldots, p^Z_{u,v})$ を割当て

確率 $\Sigma\ \gamma^k\ p^k_{u,v}$ で情報が伝わる

SOME NOTES ON

TOPIC-AWARE IC (TIC) MODEL

- **Item を1つに固定すると IC-model に一致**
- **実験では Z = 10. 映画レビューSNSで実験**
  - 例: Topic = {1:ホラー, 2:恋愛, 3:SF}
  - γ = (0.0, 0.8, 0.2) **(少しSF色のあるロマンス)**
  - $p_{u,v}$ = (0.9, 0.0, 0.4)

(vさんはuさんの恋愛映画を見る目を全く信用していない)

確率 $\Sigma\ \gamma^k\ p^k_{u,v}$ で情報が伝わる

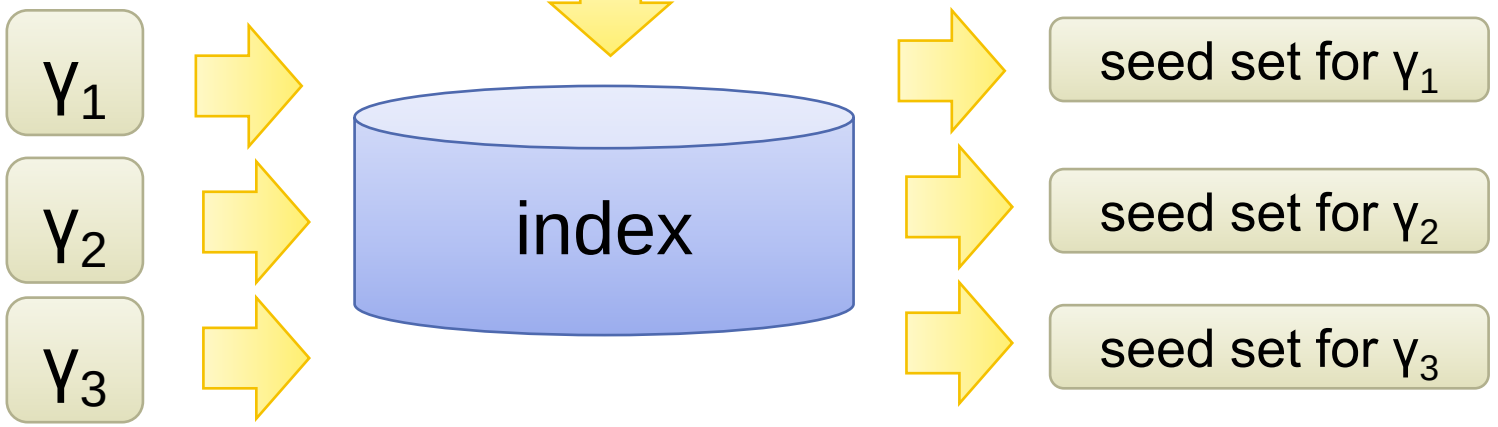
PROBLEM

前もって1つ与えられる入力: グラフ (V, E, p)

Graph

実験では |V|～30k, |E|～400k

あらかじめ index を 構築しておくことで



実験では |seed|≦50

リアルタイムに与えられる Item γ に対し 影響力の大きい seed set を即座に返したい

PROBLEM?

- **「Item を1つに固定すると IC-model に一致」**
  - ➔ Topic のことを忘れて、毎回 IC-model での影響最大化の既存アルゴリズムを実行すれば良いのでは？
  - ➔ 遅い。論文での比較対象は CELF++ [Goyal et al. WWW'11] で、数日かかる
  - ➔ [Ohsaka et al. AAAI'14] [Tang et al. SIGMOD'14] などと比べるとどうでしょうか

OVERVIEW OF THE APPROACH

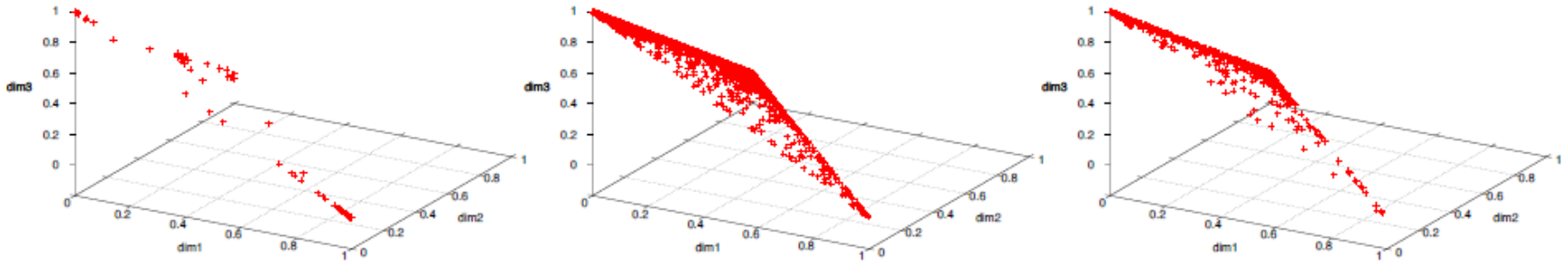
確率のついたグラフ (V, E, p) が与えられる

1. h 個の Item を代表点として選ぶ (実験では h = 1000)
2. 各代表点について、あらかじめ seed list を
   既存の IC-model 用オフラインアルゴリズムで
   計算しておく (実験では CELF++)

クエリ γ が与えられる

3. γ に近い代表点をいくつか (10個) 取り出す
4. それぞれの代表点でのSeed Listを巧く混ぜる

1. INDEX POINTS



(a) Distribution of TIC items (b) Distribution of sampled items from $Dirichlet(\boldsymbol{\alpha})$ (c) Distribution of index items

元のデータセットに
含まれている Item
(レビューされた映画)

から学習した
Dirichlet分布で
Itemを大量に
ランダム生成

したものを
KL-divergence で
K-means++ で
h 個にクラスタリング

the hyper-parameters $\boldsymbol{\alpha} = \{\alpha_1, \cdots, \alpha_Z\}$ which define the Dirichlet distribution that maximizes:

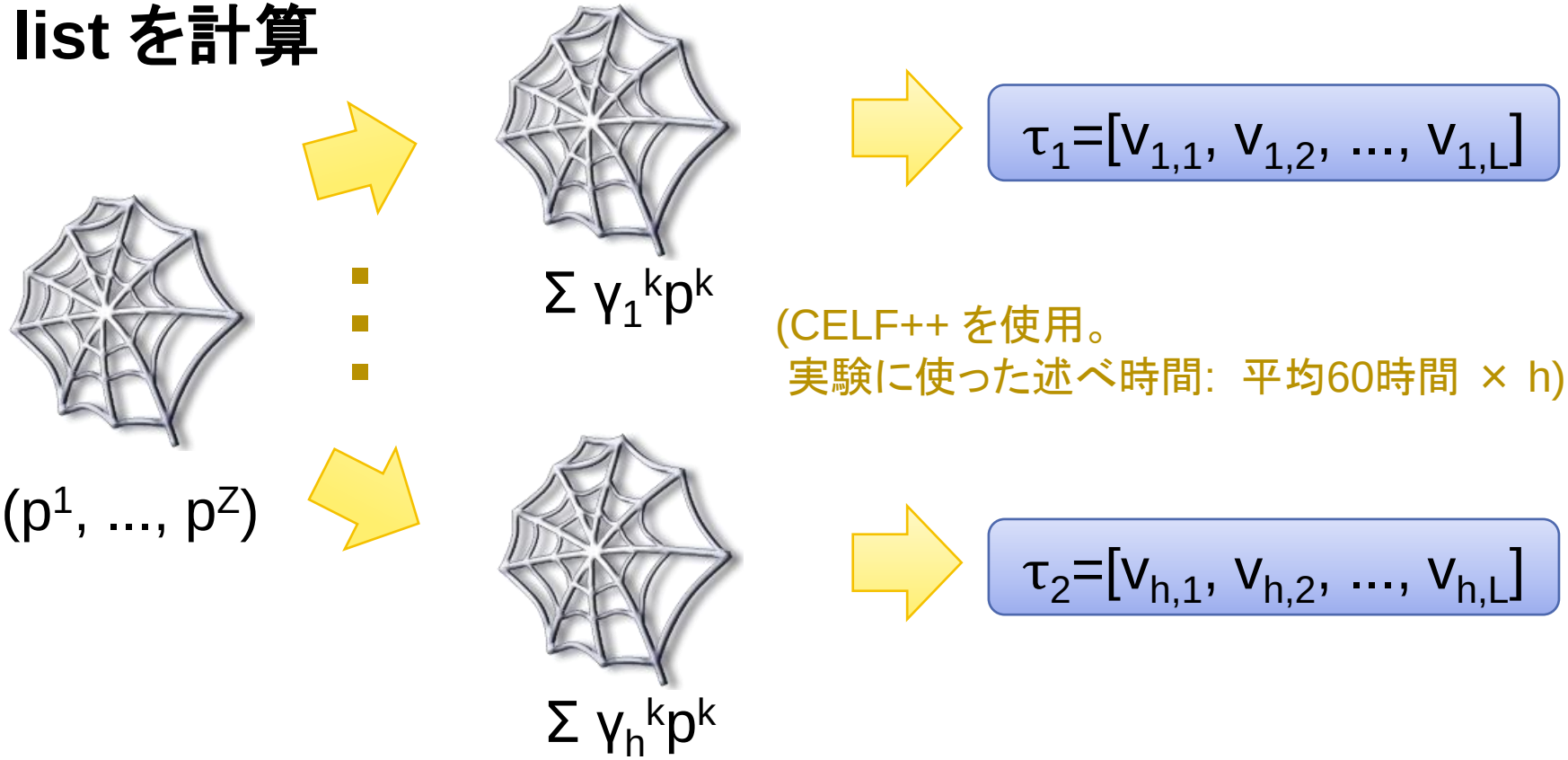
$$\prod_{i \in \mathcal{I}} P(\vec{\gamma}_i | \boldsymbol{\alpha}) = \prod_{i \in \mathcal{I}} \frac{\Gamma(\sum_z \alpha_z)}{\prod_z \Gamma(\alpha_z)} \prod_z (\gamma_i^z)^{\alpha_z - 1}.$$

$$D_{KL}(P \| Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

$$D_{KL}(\vec{\gamma}_i \| \vec{\gamma}_q)$$

2. PRECOMPUTE

Index Point H = $\{\gamma_1, \ldots, \gamma_h\}$ のそれぞれについて、IC model での既存手法で L (= 50) ノードの seed list を計算



(CELF++ を使用。
実験に使った述べ時間: 平均60時間 × h)

OVERVIEW OF THE APPROACH

確率のついたグラフ (V, E, p) が与えられる

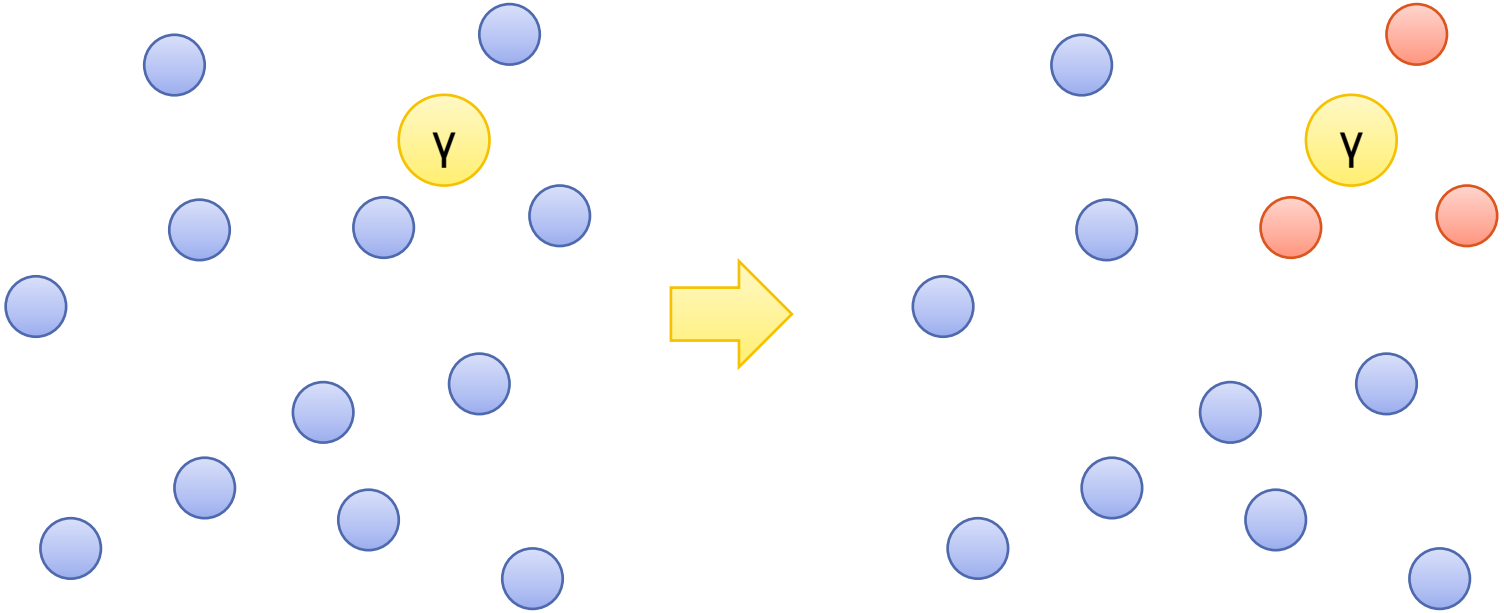
1. h 個の Item を代表点として選ぶ (実験では h = 1000)
2. 各代表点について、あらかじめ seed list を
   既存の IC-model 用オフラインアルゴリズムで
   計算しておく (実験では CELF++)

クエリ γ が与えられる

3. γ に近い代表点をいくつか (10個) 取り出す
4. それぞれの代表点でのSeed Listを巧く混ぜる

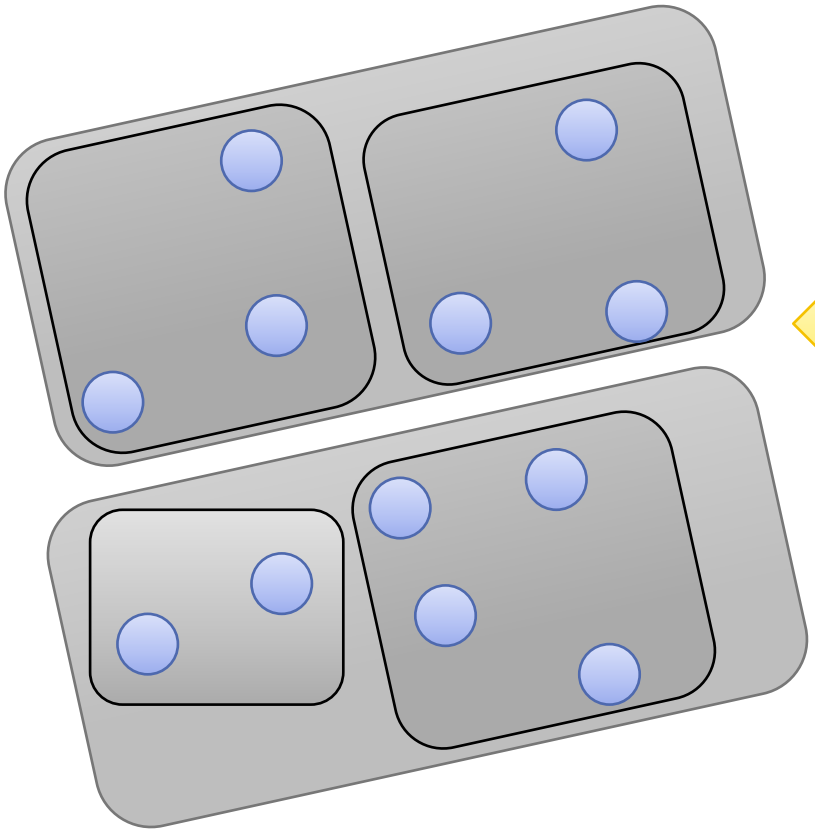
3. SEARCH IN INDEX

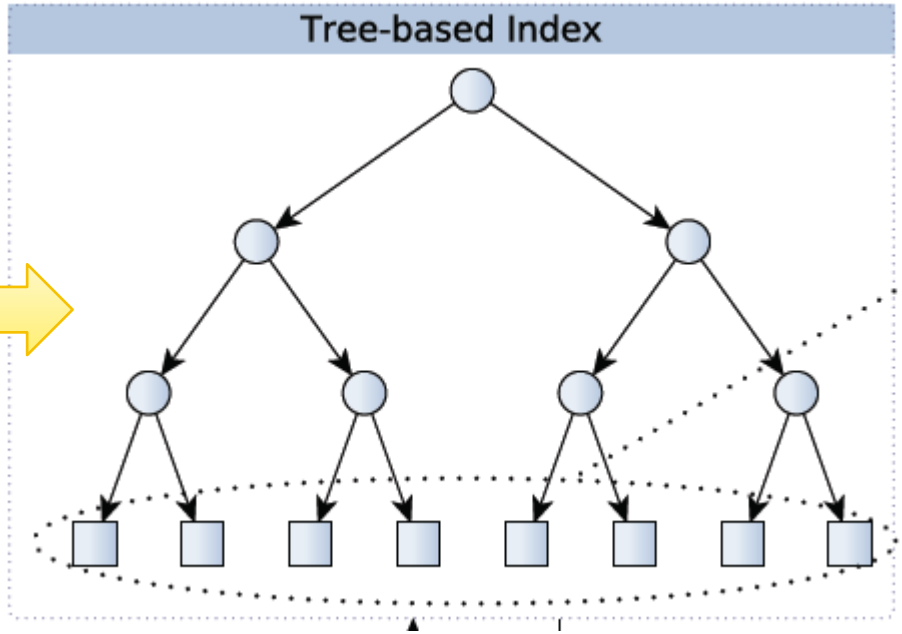
- 新しくアイテム γ が与えられたときに、Index Point 集合 H から γ に近い点をいくつか取り出す



3. SEARCH IN INDEX: HOW

Index集合を、KL-divergenceで定義されたballで繰り返し分割して、tree状に保持
(“Bregman ball tree”)





3. SEARCH IN INDEX: HEURISTICS
(How to search 10 neighbors from the tree index?)

- **Approximate K-NN:**
  - leafを5個訪れたら探索打ち切り
- **Anderson-Darling test:**
  - 含まれているIndex Pointに対しクエリ γ を統計的に検定し、妥当ならそこで探索打ち切り
- **Selection:**
  - Rankを混ぜるときにKL-divergenceを正規化した重み付きで混ぜるので、重み0.5%
- **INFLEX (提案手法): 3つすべて**

3. SEARCH IN INDEX: HEURISTICS
(How to search 10 neighbors from the tree index?)

Seed list の質 (0 = CELF++と一致)

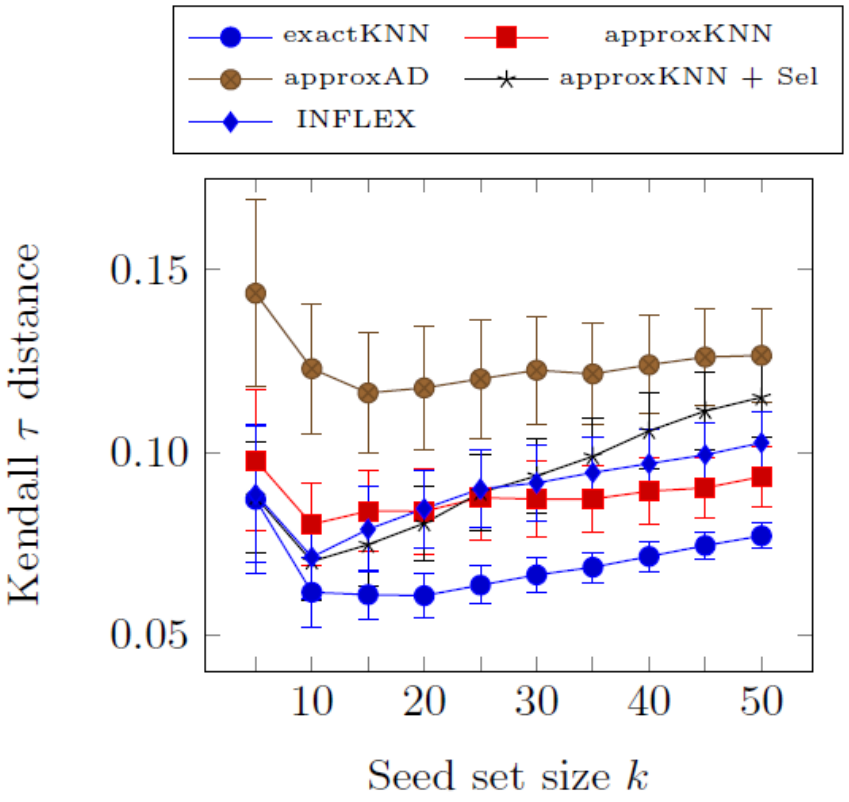


Figure 6: Accuracy comparison.

計算時間

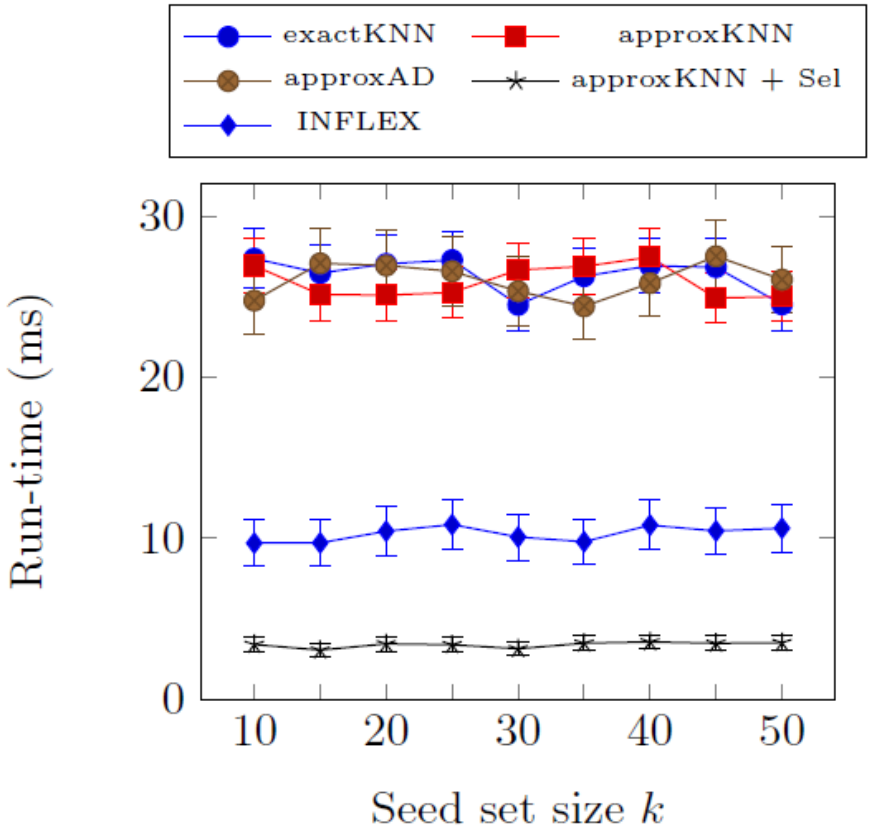


Figure 7: Run-time comparison.

4. RANK AGGREGATION

- h個の Index point から得られたノードのランキング (0≦w≦1 の重み付き) を混ぜる
  - (W_1, [$V_{1,1}$, $V_{1,2}$, …, $V_{1,L}$])
  - …
  - (W_h, [$V_{h,1}$, $V_{h,2}$, …, $V_{h,L}$])

$$W(\vec{\gamma}_i, \vec{\gamma}_q) = \frac{e^{KL_{max}} - e^{D_{KL}(\vec{\gamma}_i \| \vec{\gamma}_q)}}{1 - e^{-KL_{max}}}$$

4. RANK AGGREGATION

Weighted Copeland score

$$\mathrm{Copeland}^{w}(v) = \Sigma_{v'} \Sigma_{j} \{ w_j \mid \tau_j(v) < \tau_j(v') \}$$

where $\tau_j(v)$:= Index Point j の seed list での v の順位の高い順に並べる

Table 1: Kendall-τ distance between the seed sets produced by aggregation algorithms and the ground truth computed by standard offline influence maximization computation.

Seed Set size k	*Borda*	*Borda*^{w}	*Copeland*	*Copeland*^{w}
5	0.100	0.096	0.104	**0.087**
10	0.073	0.066	0.068	**0.062**
15	0.071	0.065	0.068	**0.061**
20	0.068	0.063	0.068	**0.061**
25	0.068	0.066	0.069	**0.064**
30	0.068	0.067	0.071	**0.066**
35	0.071	**0.069**	0.072	**0.069**
40	0.074	0.073	0.075	**0.072**
45	0.079	0.076	0.077	**0.075**
50	0.081	0.080	0.079	**0.077**

EXPERIMENTAL SETTINGS

- **Flixster (映画レビューSNS) dataset**
  - 30k users, 425k social links, 12k movies
  - “*User* u rated *Item* m on *Time* t” 形式のログ
    - 著者らが ICDM’12 で提案した手法で、Topic数 Z (= 10) を決め打って 各link の p と各movieの γ を推測したデータを用いる。

FINAL RESULT

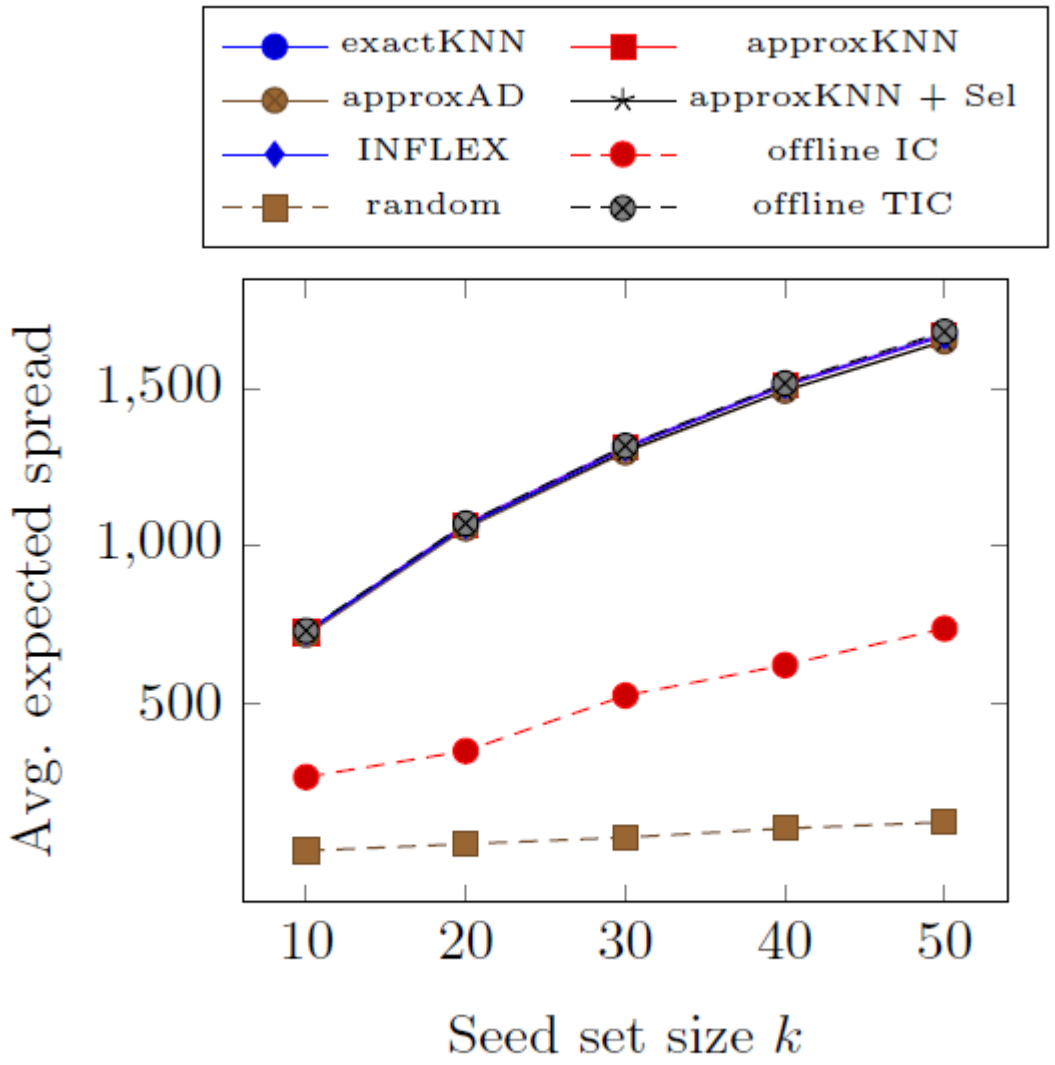


Figure 8: Expected spread comparison.