Being Prepared In A Sparse World: The Case of KNN Graph Construction

François Taïani – ASAP – Inria Rennes

UMR IRISA

UNIVERSITÉ DE RENNES 1

Inria INVENTEURS DU MONDE NUMÉRIQUE

esir ECOLE SUPERIEURE D'INGENIEURS DE RENNES

Ideal computer system

- structured, predictable, open, evolvable

A Distributed System Today …

Standards

JSON
JavaScript Object Notation

External developers

External services

facebook

twitter

foursquare™

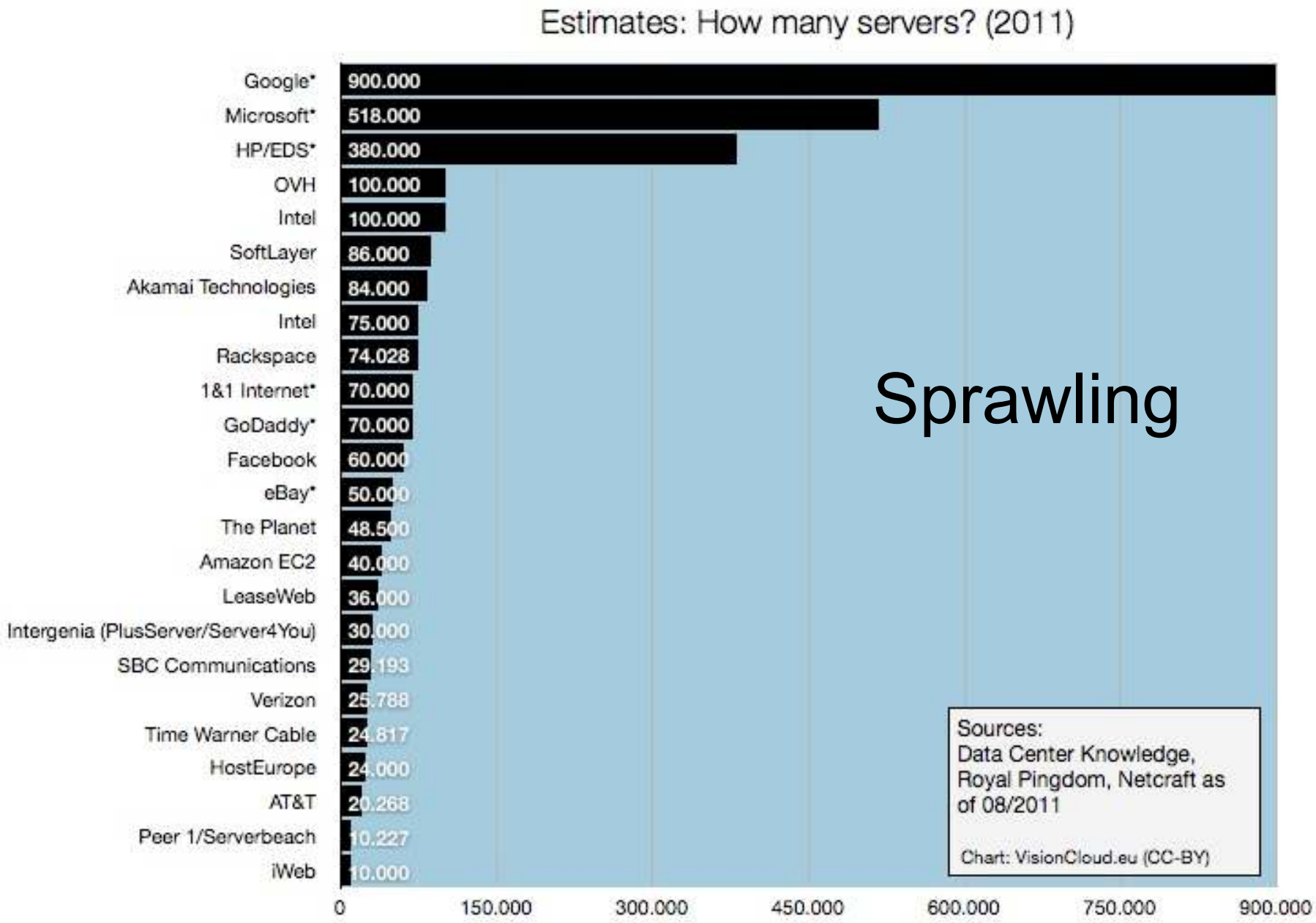
Geosocial app, est. 2009

45M Users

Middleware

Today's computer systems

- ➔ sprawling, chaotic, complex, unmanageable?



Source: "How Many Servers Worldwide?", Mirko Lorenz, http://www.visioncloud.eu/content.php?s=191,324


Chaotic
Google

Complex

Portability

Interoperability

Transparency

...

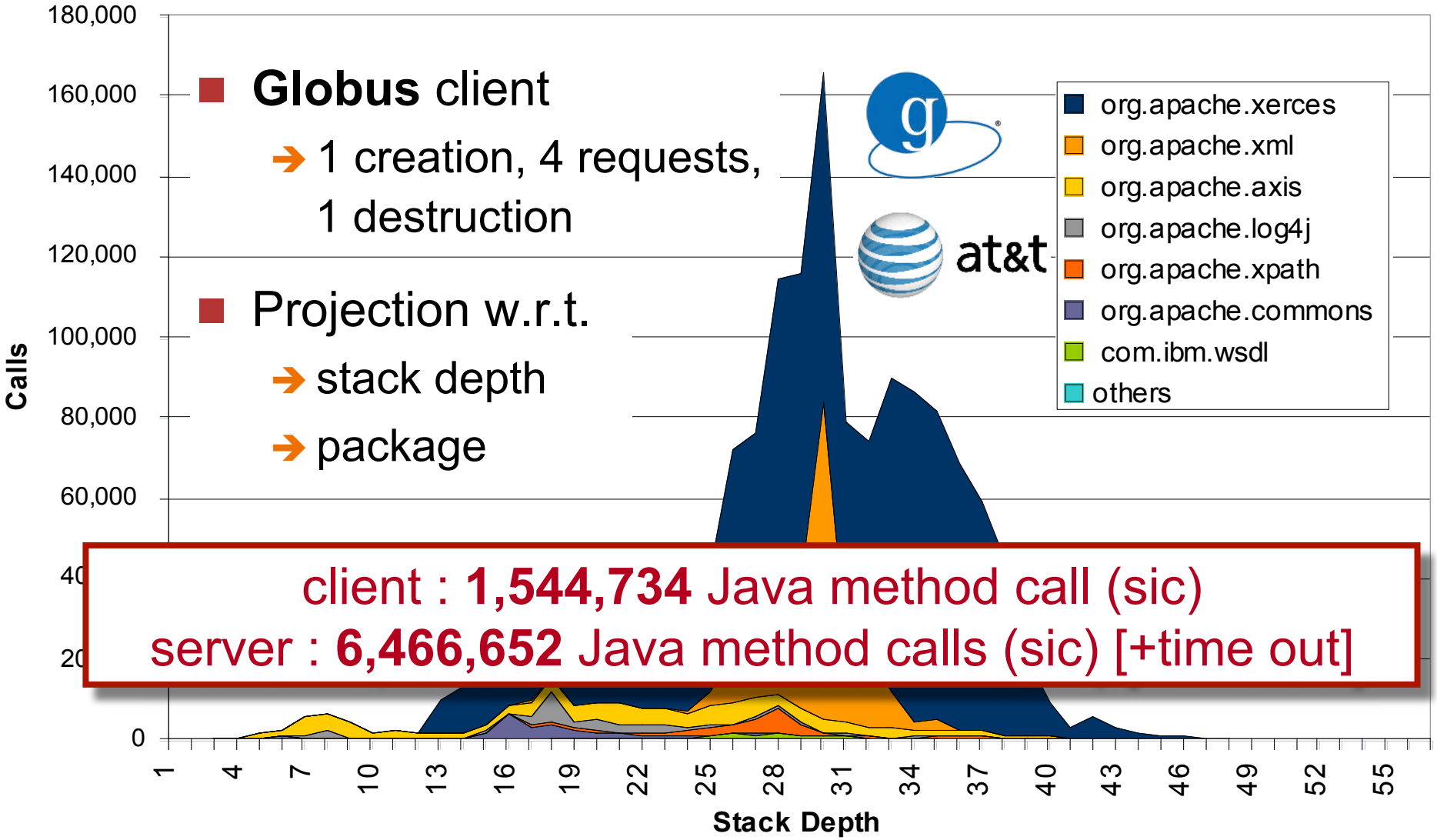
one RPC request,

- **2065** individual invocations
- > **50** C-functions
- > **140** C++ classes

Source: [TKF2009]

Unmanageable?

- **Globus** client
  - 1 creation, 4 requests, 1 destruction
- Projection w.r.t.
  - stack depth
  - package



client : **1,544,734** Java method call (sic)
server : **6,466,652** Java method calls (sic) [+time out]

The Impact of Web Service Integration on Grid Performance. Taïani, Hiltunen, Schlichting, HPDC-14, 2005

Unmanageable?

ars technica

ALL APPLE ASK ARS BUSINESS GADGETS GAMING MICROSOFT OPEN SOURCE SCIE

TECH POLICY

NEWS GUIDES REVIEWS FUTURE OF CARS

› Gear & Gadgets Essential toys, tools, and hardware

Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awarded a $1 million prize to a developer team in 2009 for an algorithm that increased the accuracy of the company's recommendation engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: a combination of too much engineering effort for the results, and a shift from movie recommendations to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.

LATEST TO

Unmanageable?

ars technica

ALL APPLE ASK ARS BUSINESS GADGETS GAMING MICROSOFT OPEN SOURCE SCIE

TECH POLICY

NEWS GUIDES REVIEWS FUTURE OF CARS

› Gear & Gadgets Essential toys, tools, and hardware

Netflix never used its $1 million algorithm due to engineering costs

By Casey Johnston | Published April 13, 2012 4:25 PM

Netflix awa **$1 million prize** veloper team in 2009 for an algorithm that increased the accuracy of the compar **recommendation** engine by 10 percent. But today it doesn't use the million-dollar code, and has no plans to implement it in the future, Netflix announced on its blog Friday. The post goes on to explain why: a combination of too much **too much engineering effort** n movie recommendations to the "next level" of personalization caused by the transition of the business from mailed DVDs to video streaming.

LATEST TO

NETFLIX

amazon webservices™

Apache

Large **Dynamic** **Complex**

Which **practical approaches** for **scale** and **performance**?

$1 million prize

recommendation

too much engineering effort

Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neIghbor Fast and eFficient
- Evaluation

Co-authors

- Joint work with

Antoine Boutet

Nupur Mittal

Anne-Marie Kermarrec

- Published at ICDE 2016

Outline

- **The problem: KNN graph construction**
- The intuition: Is greed all there is?
- KIFF: K-nearest neIghbor Fast and eFficient
- Evaluation

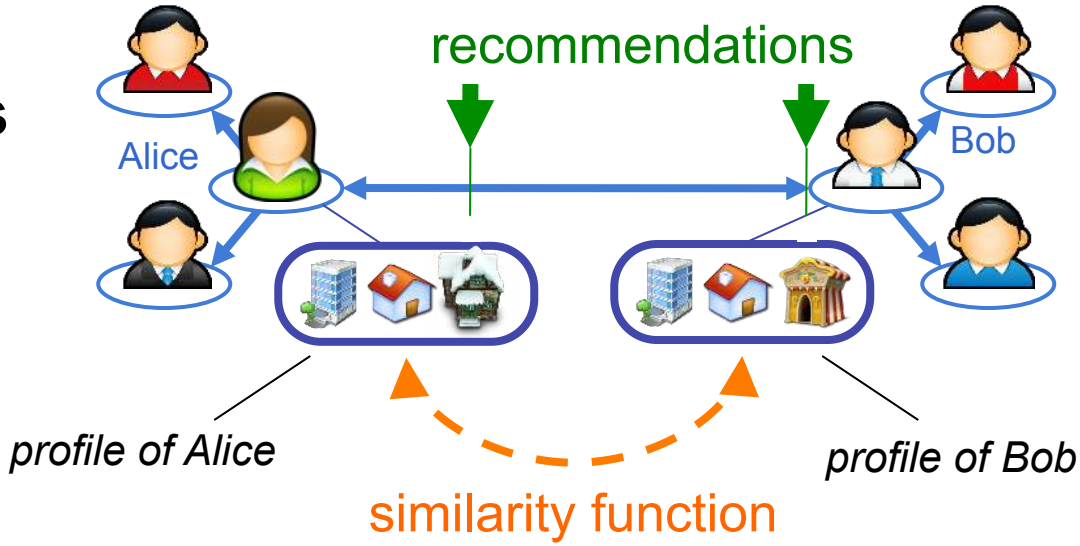
KNN Graph Construction

- **Entities** (e.g. users)

- **Similarity** function

- Goal: for each entity find ***k* closest entities**

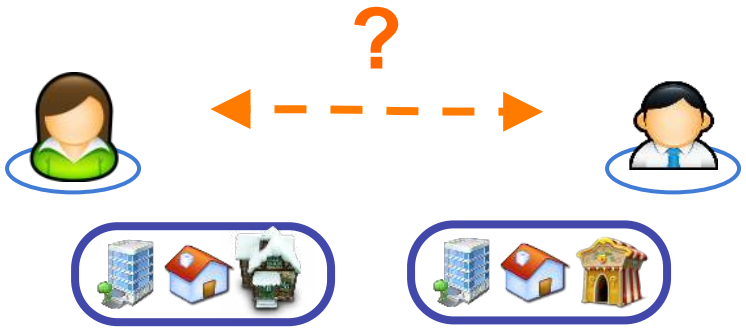
- Many **applications**
  - ➔ search
  - ➔ recommendation,
  - ➔ learning, ...



Challenges

- Brute force not scalable: O($n^2 \times \log(k)$)

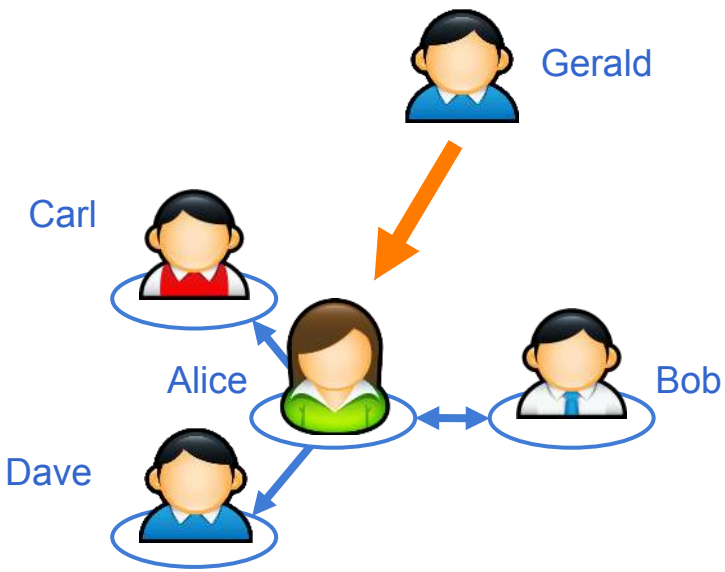
- Alternatives: Approximate KNN Graph
  - ➔ Using Locality Sensitive Hashing (LSH)
  - ➔ Using Greedy Construction: best at the moment
    Vicinity [VS05], T-Man [JMB09], NNDescent [DML11], Hyrec [BFGKP14]



Greedy KNN Construction

Parallel-iterative algorithm,

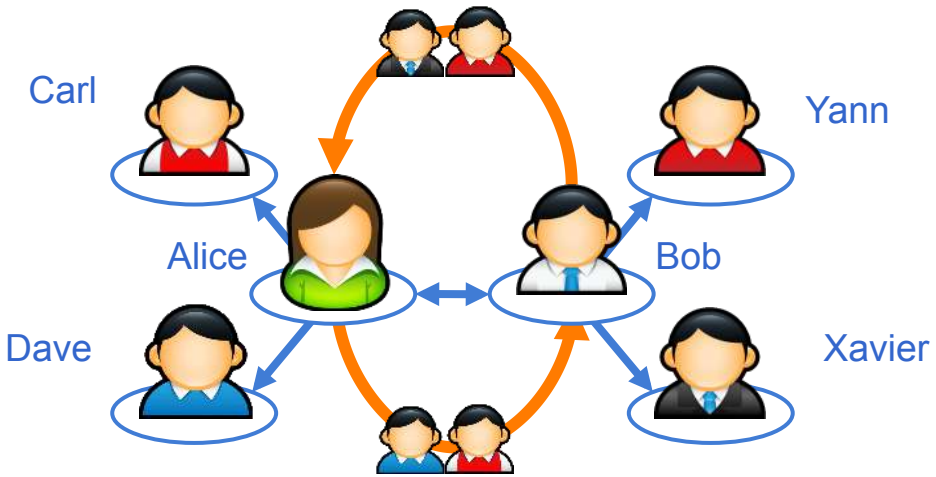
- From a **random** graph
- Each node looks for **potential new neighbours:**
  - ➔ **(1) among random nodes**



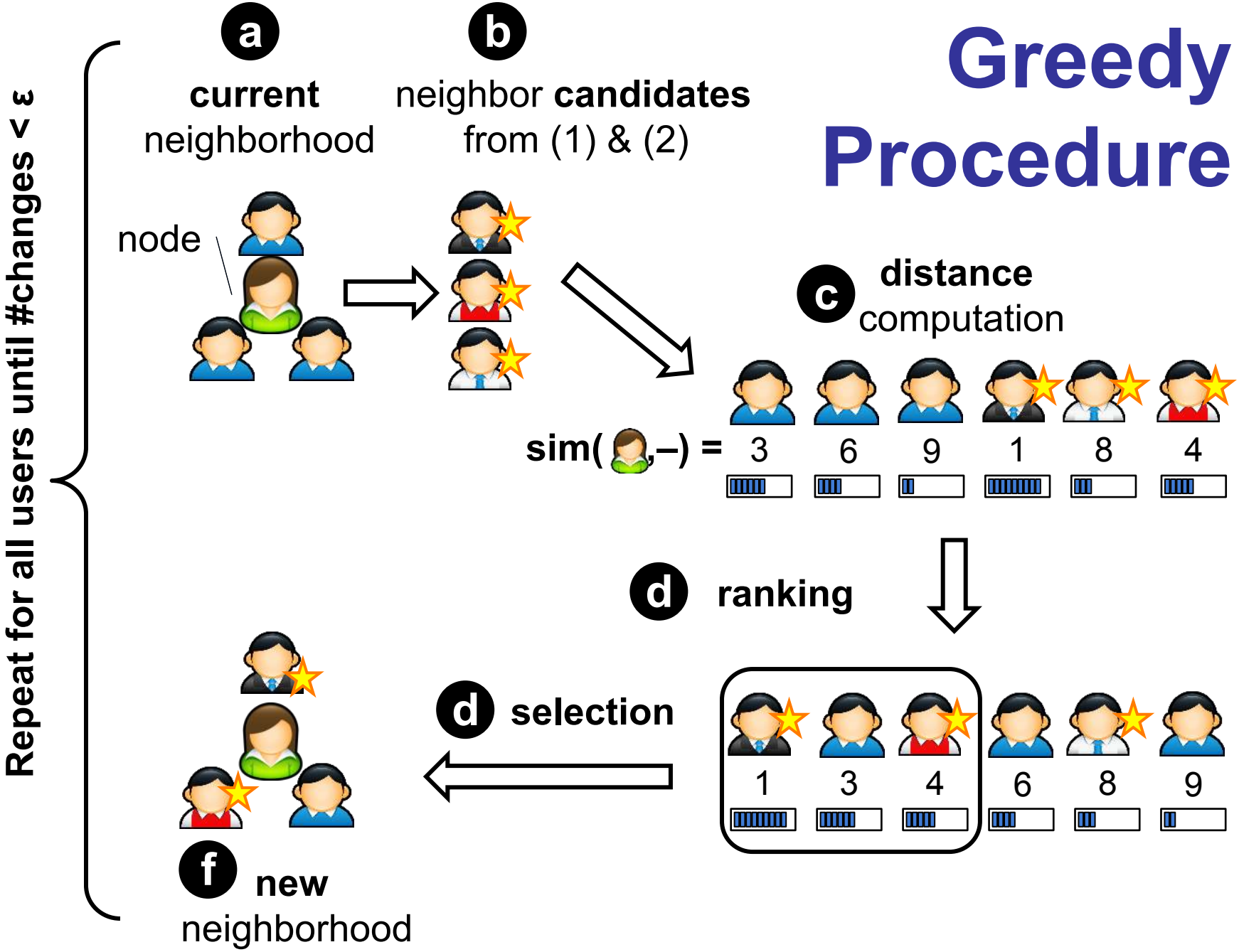
Greedy KNN Construction

Parallel-iterative algorithm,

- From a **random** graph
- Each node looks for **potential new neighbours:**
  - ➔ **(1) among random nodes (*optional*)**
  - ➔ **(2) among "friends of friends"**



Greedy Procedure

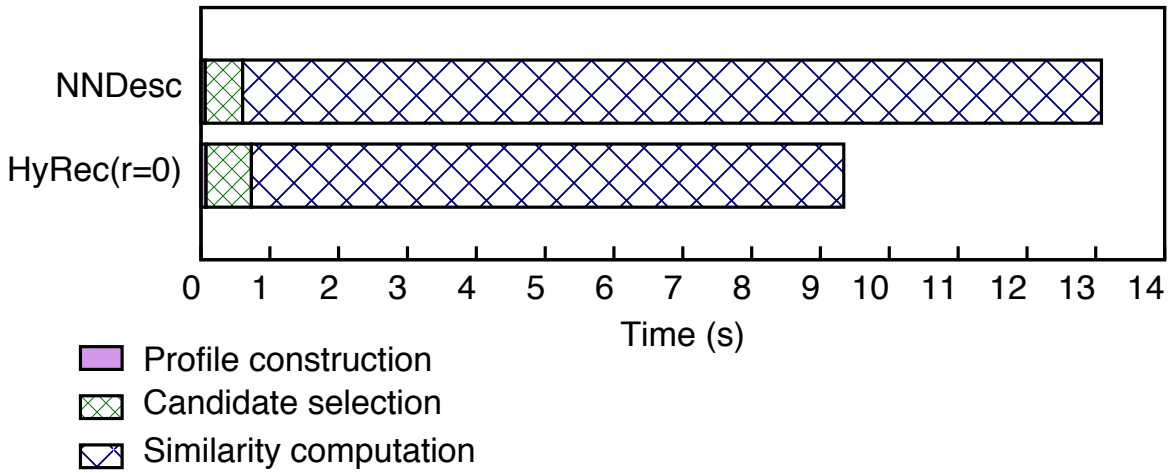


Outline

- The problem: KNN graph construction
- **The intuition: Is greed all there is?**
- KIFF: K-nearest neIghbor Fast and eFficient
- Evaluation

Is Greed all there is?

- Observation 1: **Similarity** remains the **bottleneck**



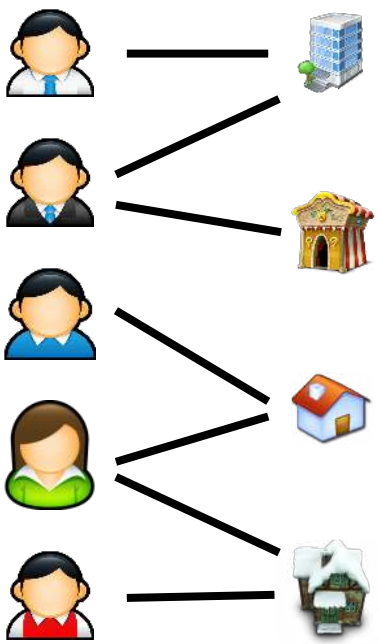
  - ➔ 90% of execution time spent on similarity (Wikipedia dataset)

- Observation 2: **Datasets** are (often) **sparse**
  - ➔ Many datasets use item-based profiles
  - ➔ Most items little shared : **sparse**

The Problem with Sparsity

- Density: $|E| \div (|U| \times |I|)$
  - ➔ $|E|$ = # ratings, $|U|$ = #users, $|I|$ = #items

Dataset	Density
Wikipedia	0.7127%
Arxiv	0.1124%
Gowalla	0.0029%
DBLP	0.0011%



density = 35%

- Only a few rugs ("ratings") on the ladder
  - ➔ 2 random nodes unlikely to be close
  - ➔ Hence greedy processes slow to start

KIFF's Intuition

Greedy KNN approaches

- Assume no initial structure
- Start from a random graph

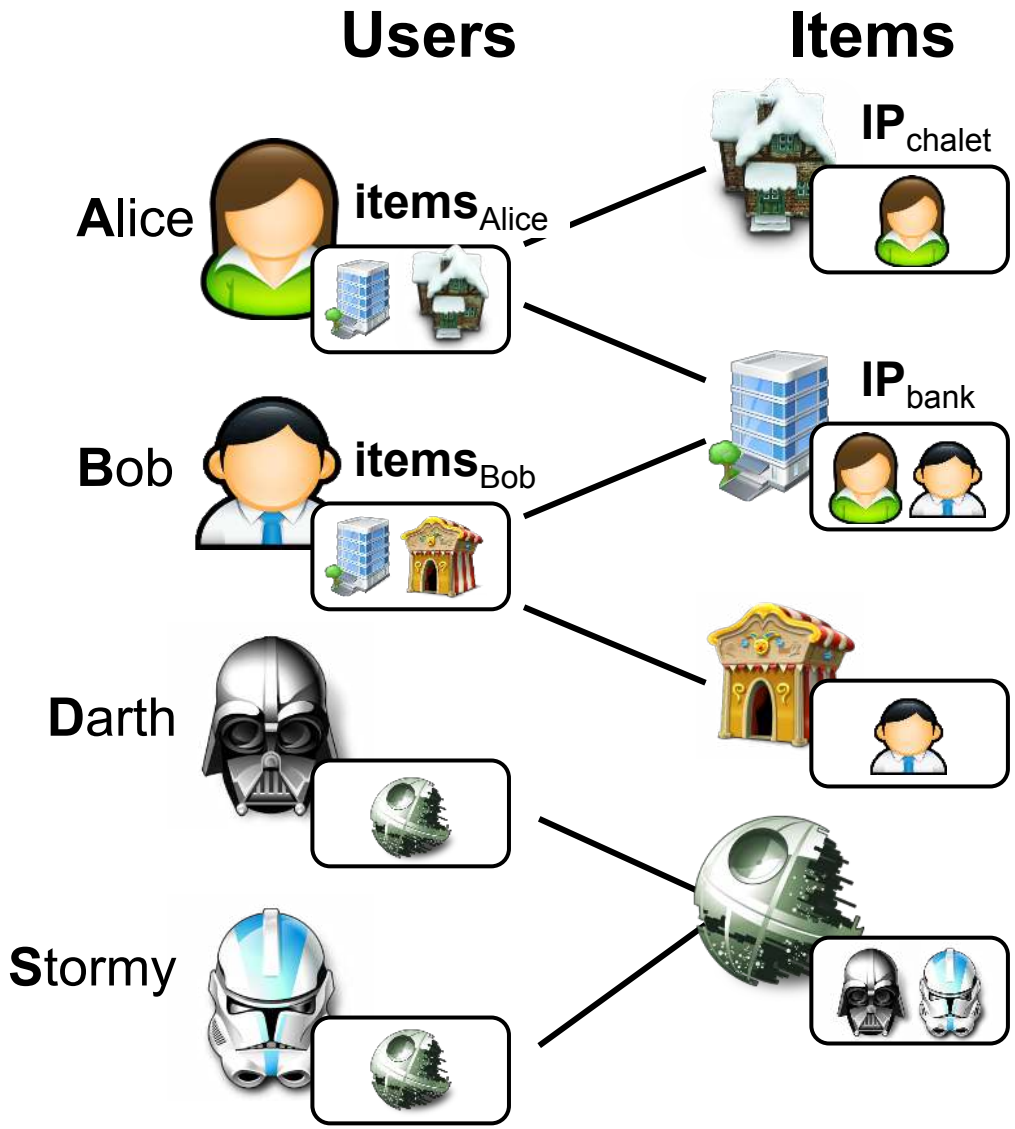
In practice

- Underlying bipartite user / item graph
- Can be used to **bootstrap** the greedy process
- Use items to create **Ranked Candidate Sets**

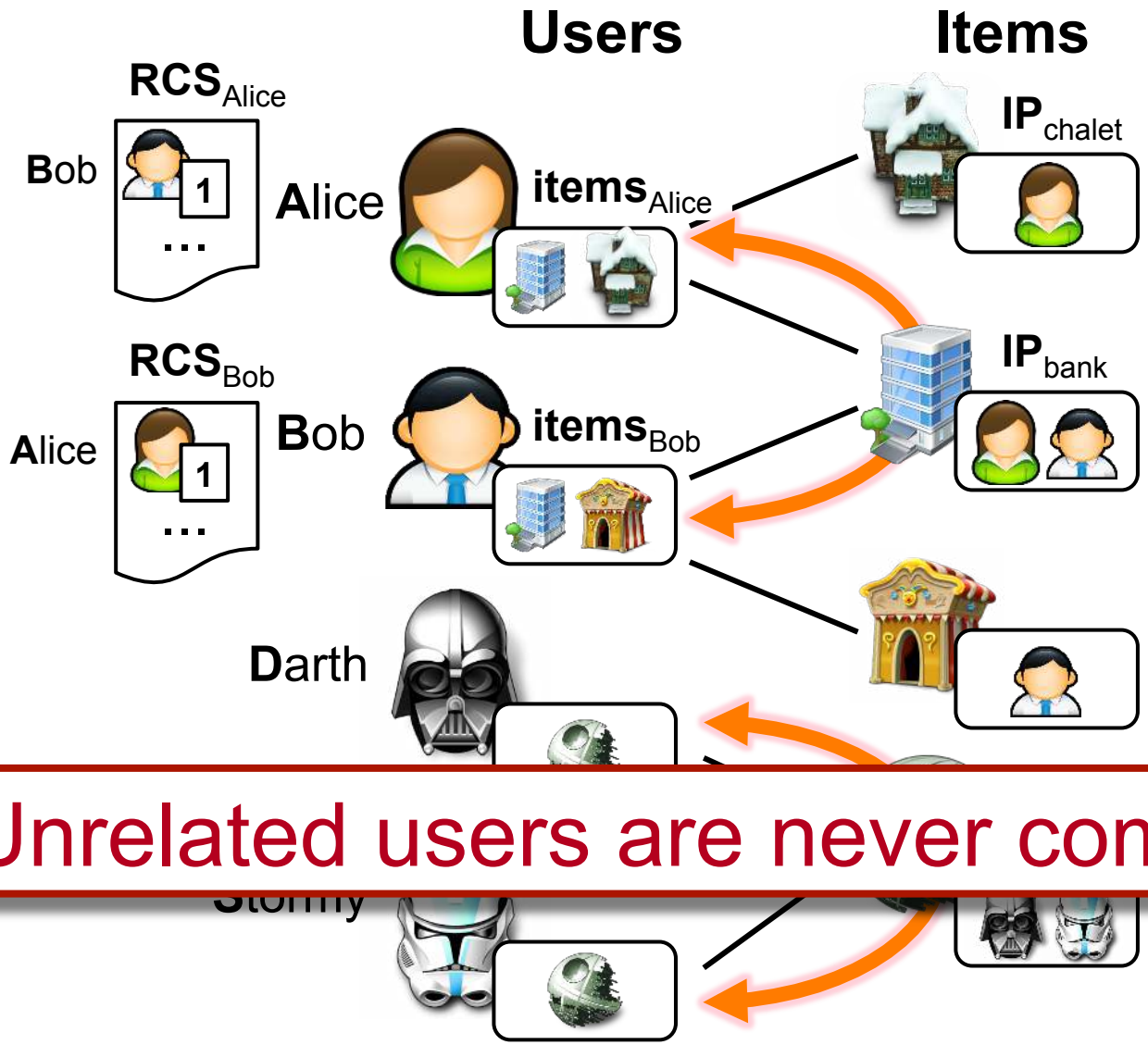
Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- **KIFF: K-nearest nelghbor Fast and eFficient**
- Evaluation

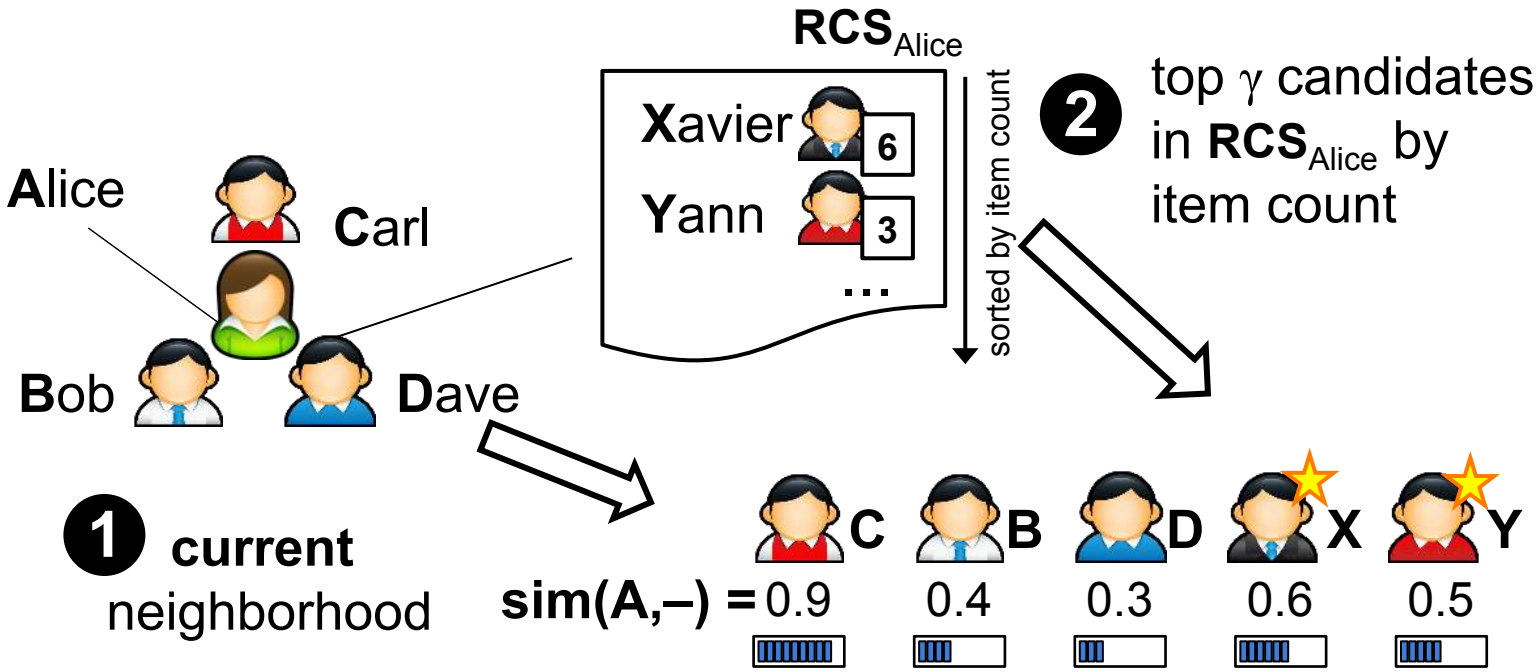
RCS Construction

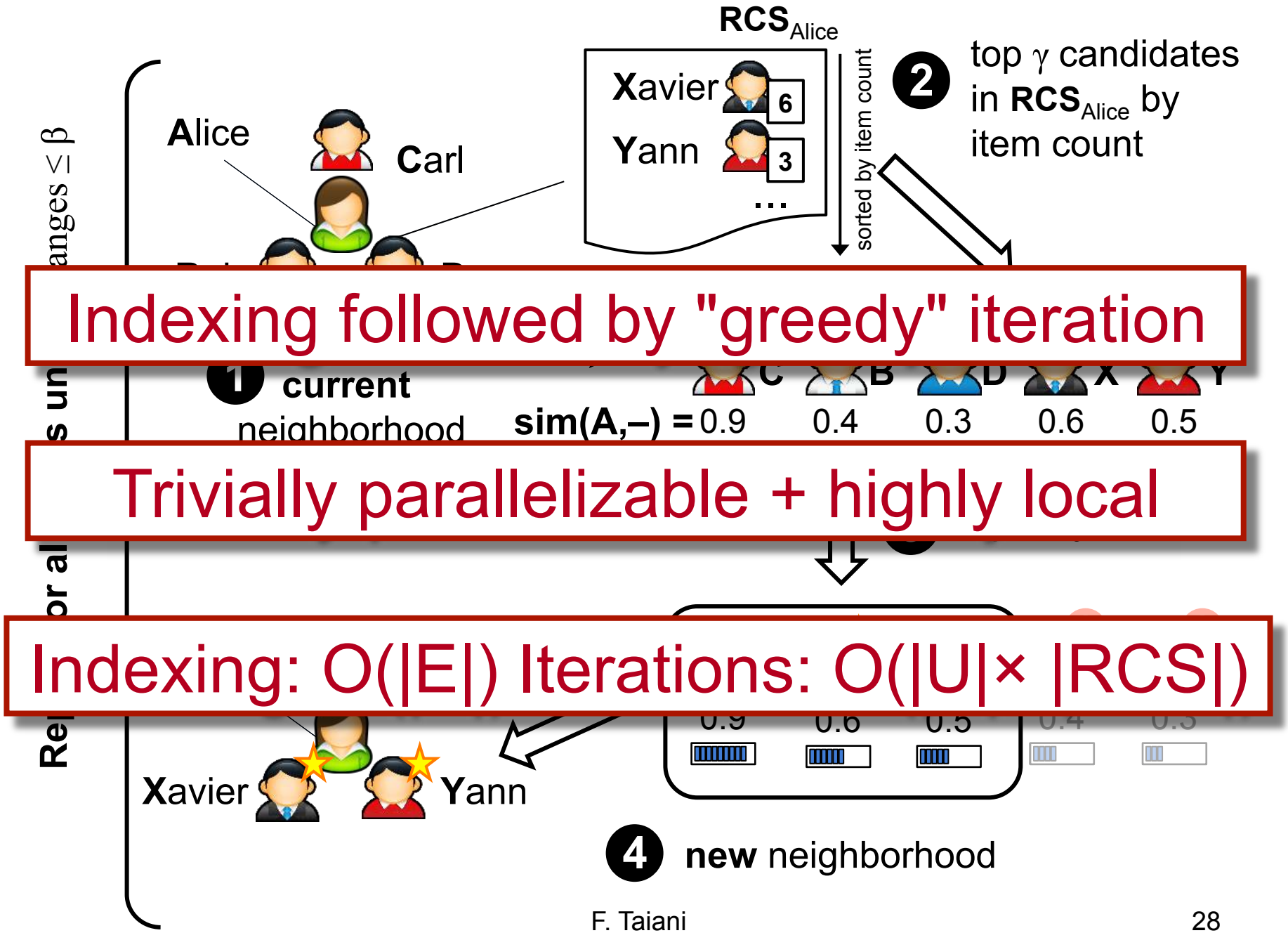


RCS Construction



Unrelated users are never compared


RCS$_{Alice}$
Xavier
6
Yann
3
...
sorted by item count
2
top γ candidates in RCS$_{Alice}$ by item count
Alice
Carl
Bob
Dave
1
current neighborhood
sim(A,–) =
C
B
D
X
Y
0.9
0.4
0.3
0.6
0.5



Indexing followed by "greedy" iteration

Trivially parallelizable + highly local

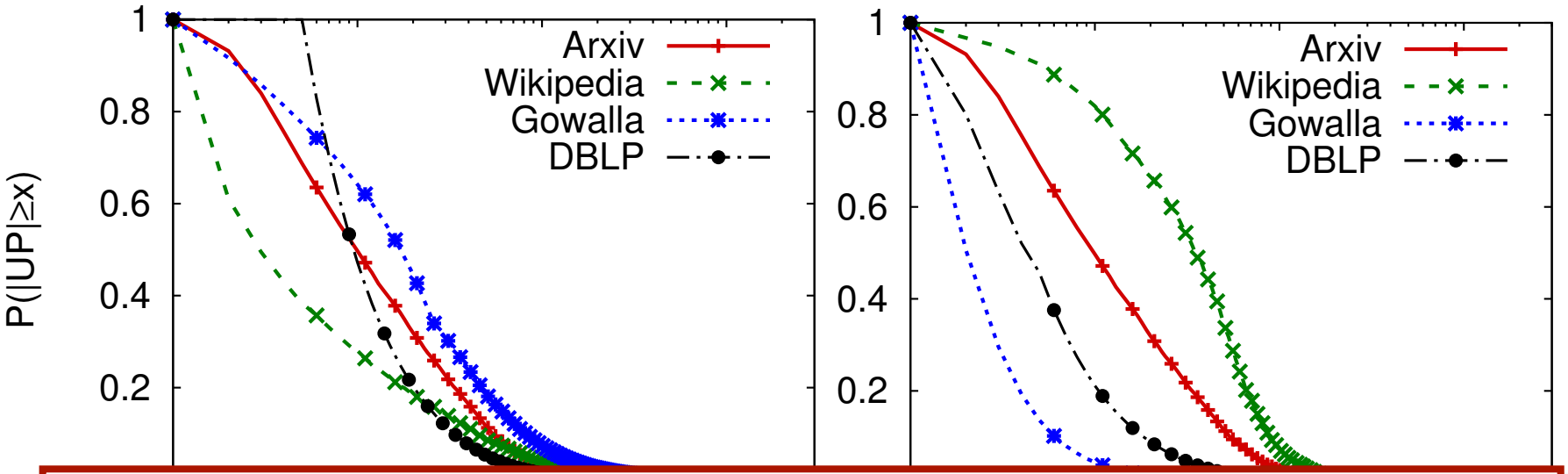
Indexing: O(|E|) Iterations: O(|U|× |RCS|)

Outline

- The problem: KNN graph construction
- The intuition: Is greed all there is?
- KIFF: K-nearest neIghbor Fast and eFficient
- **Evaluation**

Evaluation: Datasets

Dataset	#Users $\|U\|$	#Items $\|I\|$	#Ratings $\|E\|$	Density	Avg. $\|UP_u\|$	Avg. $\|IP_i\|$
Wikipedia	6,110	2,381	103,689	0.7127%	16.9	43.5
Arxiv	18,772	18,772	396,160	0.1124%	21.1	21.1
Gowalla	107,092	1,280,969	3,981,334	0.0029%	37.1	3.1
DBLP	715,610	1,401,494	11,755,605	0.0011%	16.4	8.3



Long tail profile size distribution

Evaluation: Metrics

- Wall-clock **computation time**
- **Recall**
  - knn_u : **ideal** KNN neighborhood for user *u*
  - $\widehat{knn}_u$: **approximated** KNN neighborhood for user *u*

$$recall_{\widehat{G}_{\mathrm{KNN}}}(u) = \frac{|\widehat{knn}_u \cap knn_u|}{k}$$

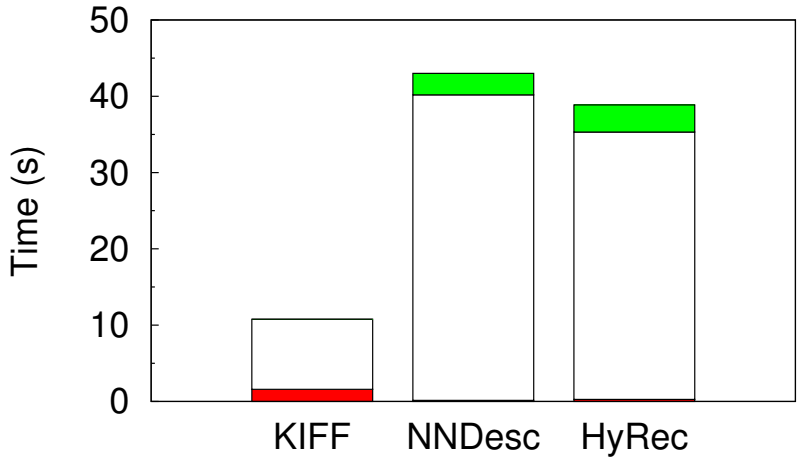
$$recall(\widehat{G}_{\mathrm{KNN}}) = \mathop{\mathbb{E}}_{u \in U} recall_{\widehat{G}_{\mathrm{KNN}}}(u)$$

- **Scan rate**

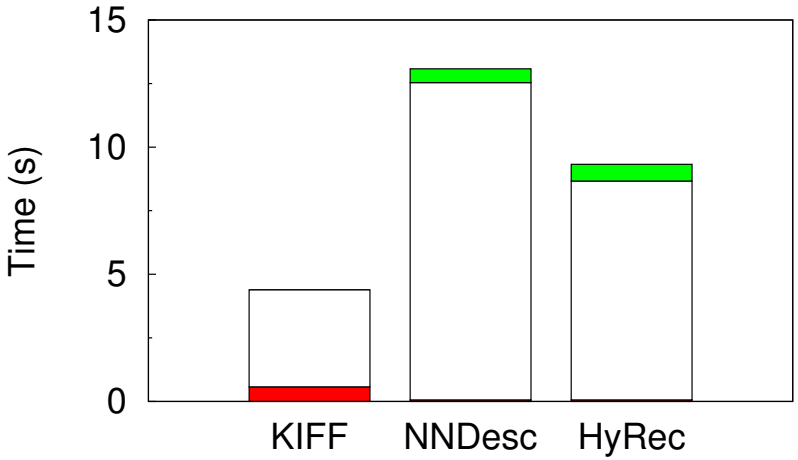
$$scanrate = \frac{\#(similarity\ \ evaluations)}{|U| \times (|U| - 1)/2}$$

Overall Performance

Competitor	*KIFF*'s gain speed-up	*KIFF*'s gain Δrecall	
NN-Descent	×15.42	+0.14	[DML11]
HyRec	×12.51	+0.23	[BFGKP14]
Average	*×13.97*	*+0.19*	



Arxiv

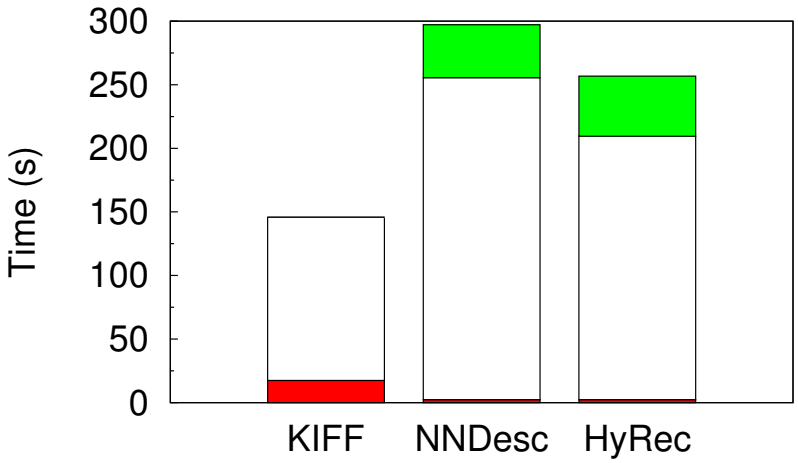


Wikipedia

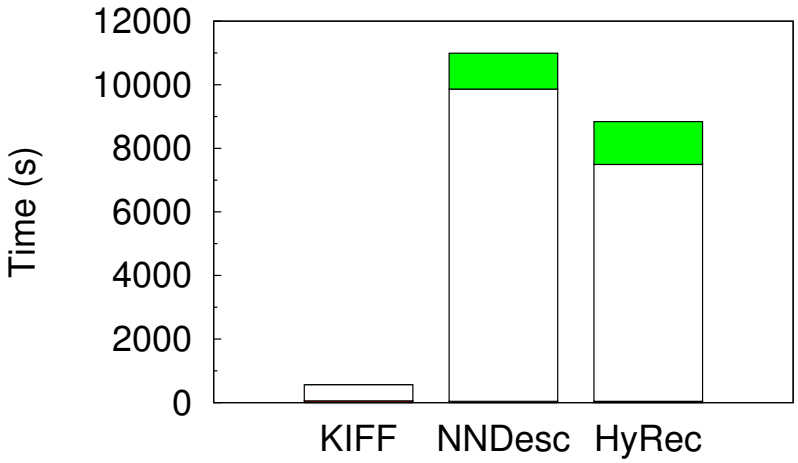
Preprocessing
Similarity computation
Candidate selection

Overall Performance

Competitor	*KIFF*'s gain speed-up	*KIFF*'s gain Δrecall	
NN-Descent	×15.42	+0.14	[DML11]
HyRec	×12.51	+0.23	[BFGKP14]
Average	*×13.97*	*+0.19*	



Gowalla



DBLP

Faster (x14), Better (+20%)

Si

Candidate selection

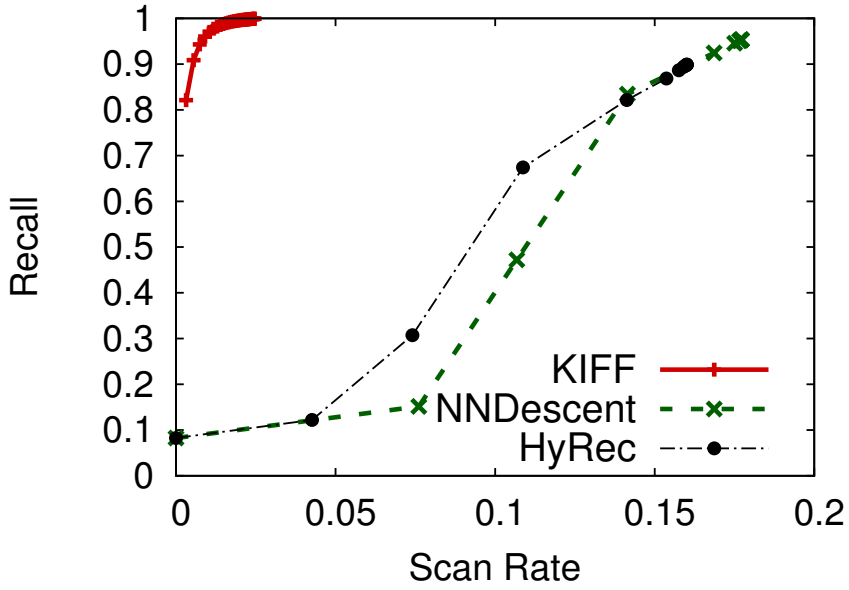
Performance Details

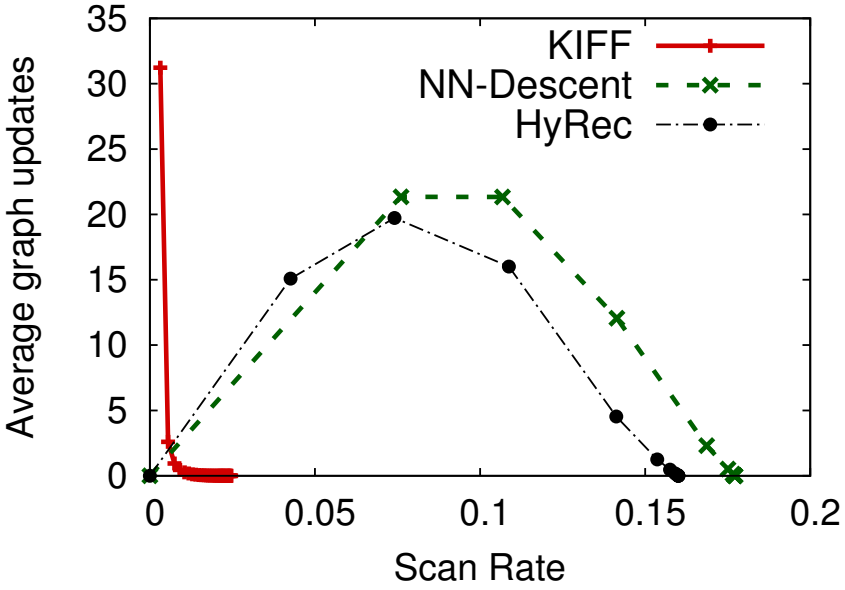
datasets	Approach	recall	wall-time (s)	scan rate	#iter.
Arxiv	**NN-Descent**	0.95	41.8	17.6%	9
	HyRec	0.90	38.6	16.0%	12
	KIFF	**0.99**	**10.7**	2.5%	36
	KIFF's Gain	*+0.06*	*×3.7*		
Wikipedia	**NN-Descent**	0.97	13.1	51.69%	7
	HyRec	0.95	9.4	44.64%	8
	KIFF	**0.99**	**4.4**	7.37%	22
	KIFF's Gain	*+0.03*	*×2.5*		
Gowalla	**NN-Descent**	0.69	307.9	3.67%	16
	HyRec	0.56	253.2	2.69%	22
	KIFF	**0.99**	**146.6**	0.84%	115
	KIFF's Gain	*+0.36*	*×1.9*		
BLP	**NN-Descent**	0.78	10,890.2	3.08%	19
	HyRec	0.63	8,829.9	2.37%	26

Much reduced scan rate

KIFF's Scan Rate

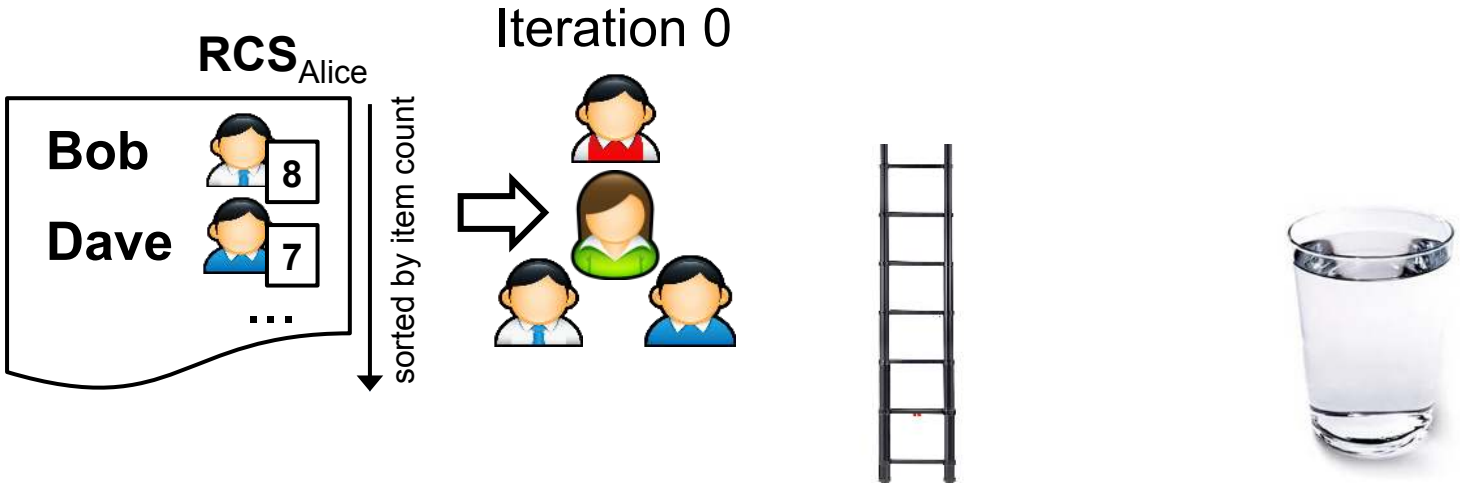
- Arxiv Dataset





KIFF: First iterations yield highest gains

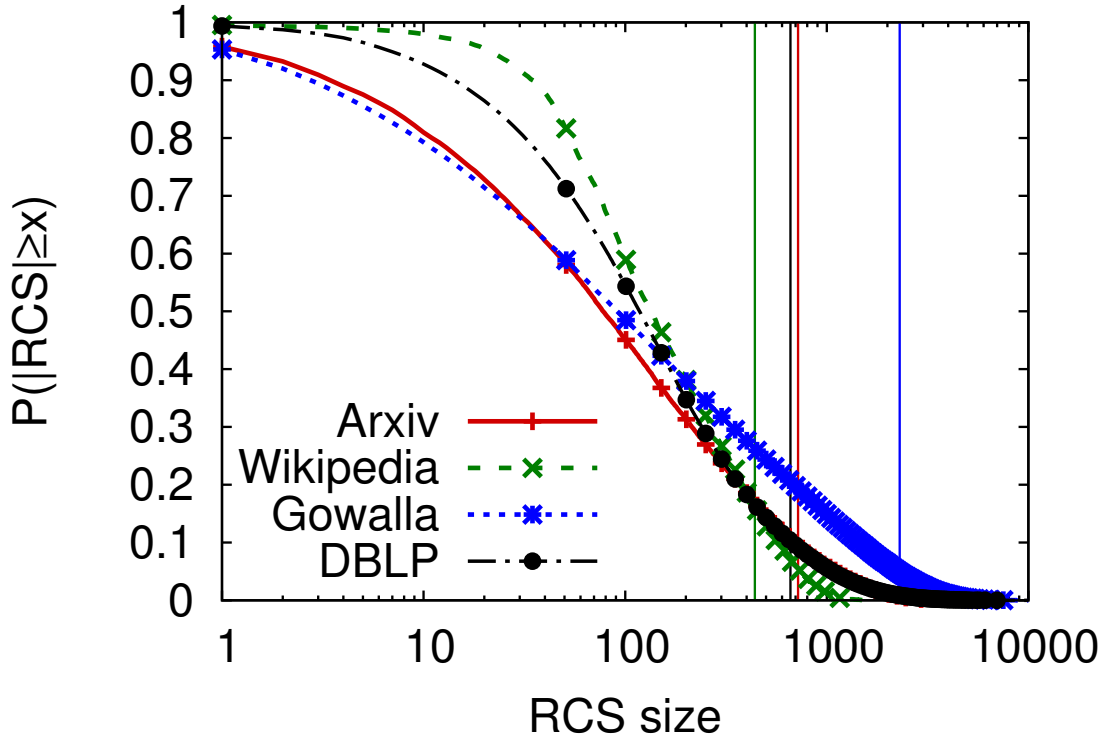
Impact of RCS on Bootstrap



	Recall	
Dataset	Top k from RCS	Random
Arxiv	0.82	0.08
Wikipedia	0.54	0.01
Gowalla	0.55	0.15
DBLP	0.79	0.09

Termination Criteria

Repeat for all users until #changes ≤ β

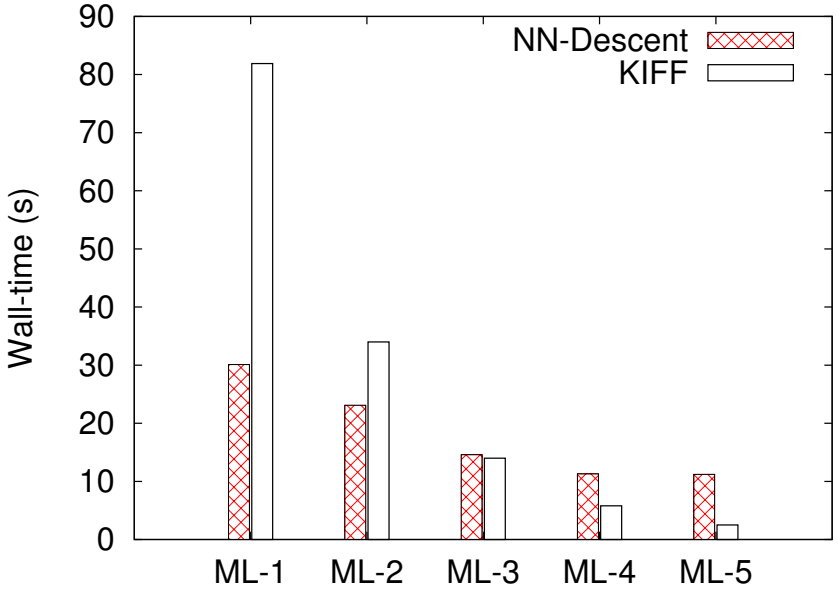


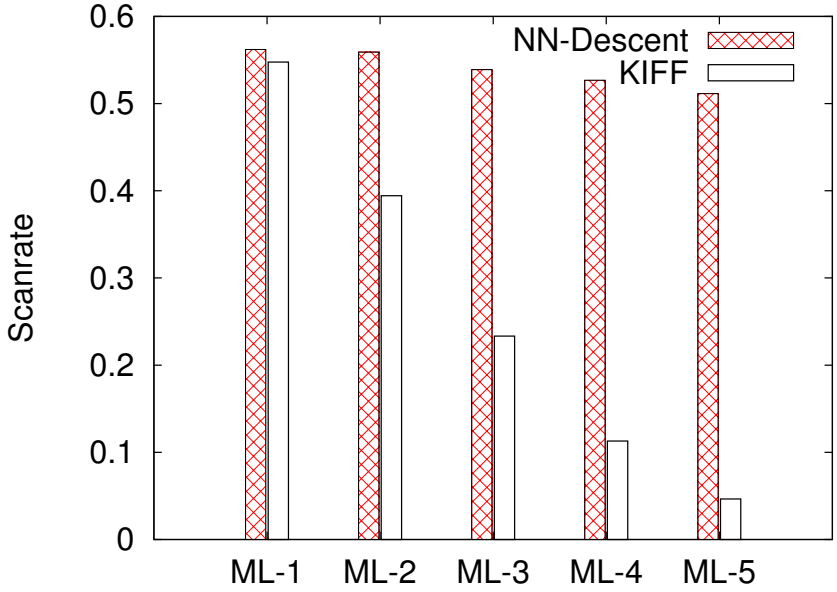
- Vertical bars: RCS truncation imposed by KIFF

Termination only impacts minority of users

Effect of Density

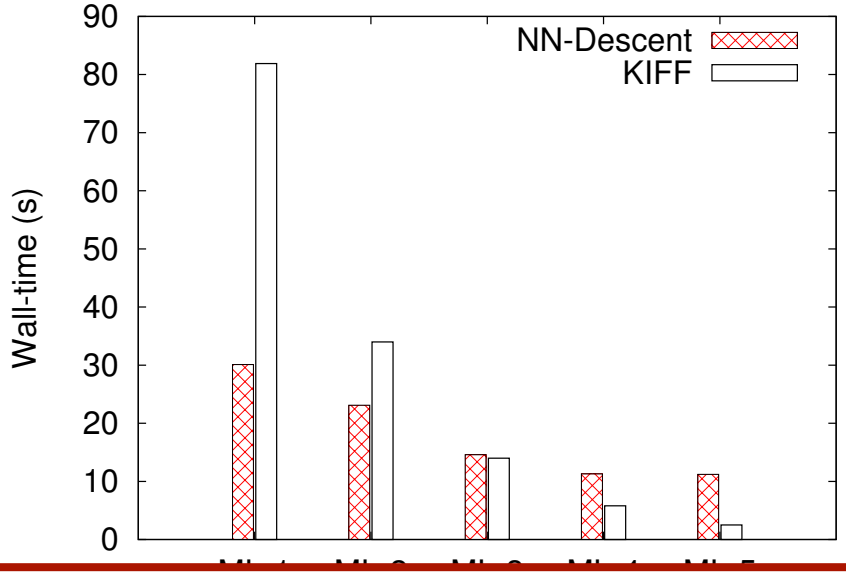
Dataset	Ratings	Density	average $\lvert RCS \rvert$
ML-1	1,000,209	4.47%	2,892.7
ML-2	500,009	2.23%	2,060.6
ML-3	255,188	1.14%	1,125.4
ML-4	131,668	0.59%	510.8
ML-5	68,415	0.30%	202.5

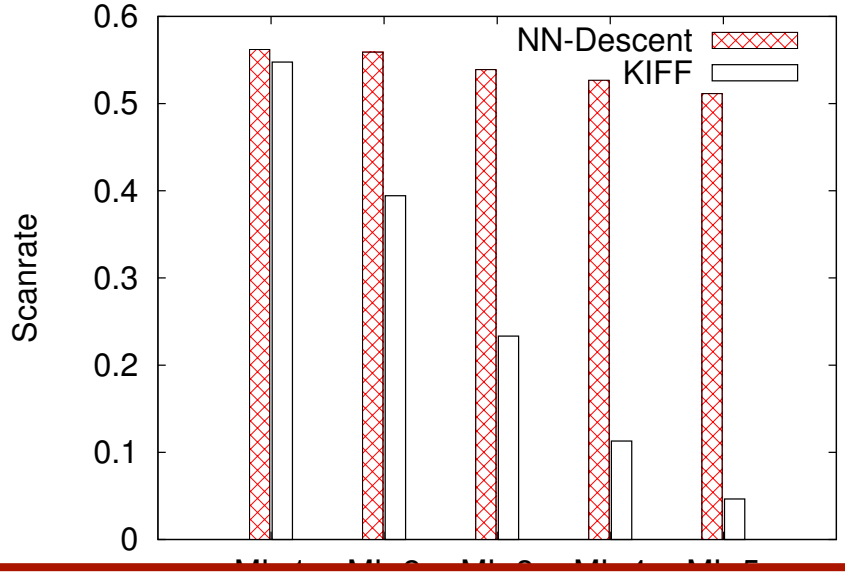




Effect of Density

Dataset	Ratings	Density	average $\lvert RCS \rvert$
ML-1	1,000,209	4.47%	2,892.7
ML-2	500,009	2.23%	2,060.6
ML-3	255,188	1.14%	1,125.4
ML-4	131,668	0.59%	510.8
ML-5	68,415	0.30%	202.5





Scan rate grows with density, hurting perf

Conclusion

- Novel KNN construction algorithm
- Intuition : reduce similarity computations
- Faster (x14) and more accurate (+20%) than SotA
- Performs best on sparse datasets
- Future: finer complexity analysis / distributed version

Thank you

Some References

- [JMB09] M. Jelasity, A. Montresor, and O. Babaoglu. 2009. **T-Man: Gossip-based fast overlay topology construction**. *Comput. Netw.* 53, 13 (August 2009), 2321-2339.
- [VS05] S. Voulgaris and M. v. Steen, **Epidemic-style management of semantic overlays for content-based searching**, in Euro-Par, 2005, pp. 1143– 1152.
- [DML11] W. Dong, C. Moses, and K. Li, **Efficient k-nearest neighbor graph construction for generic similarity measures**, in WWW, 2011, pp. 577– 586.
- [BFGKP14] A. Boutet, D. Frey, R. Guerraoui, A.-M. Kermarrec, and R. Patra, **HyRec: Leveraging Browsers for Scalable Recommenders**, in Middleware, 2014, pp. 85–96.