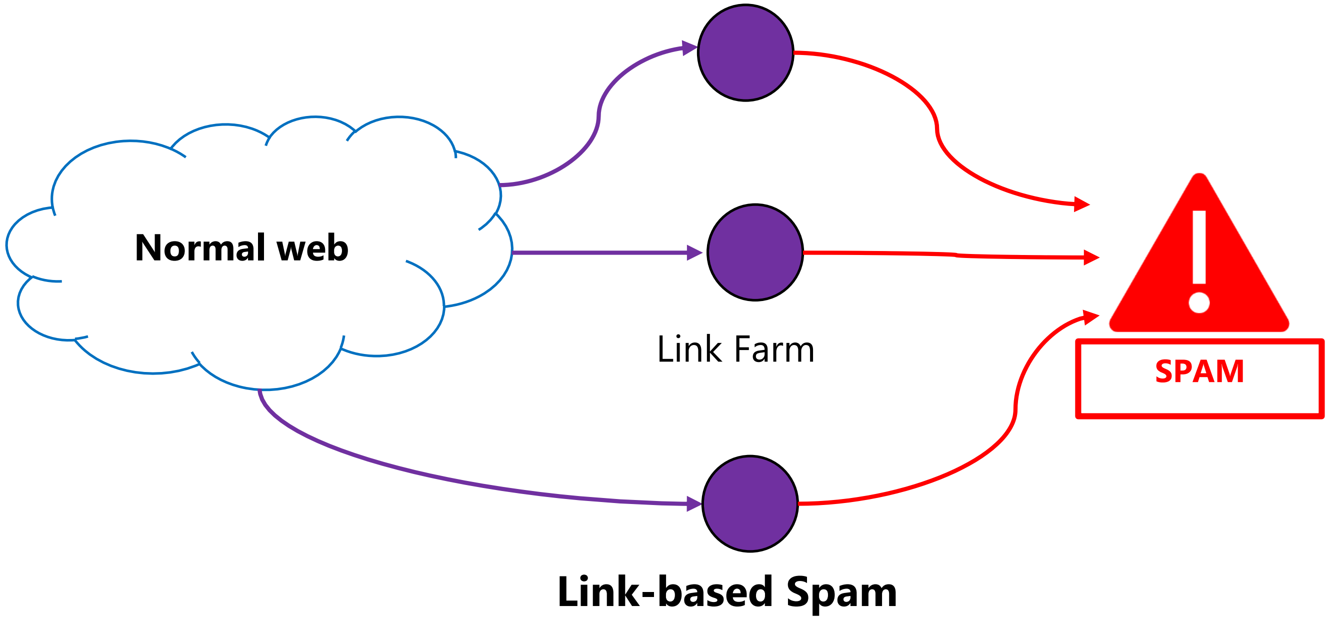
Scalable Anti-TrustRank with Qualified Site-level Seeds for Link-based Web Spam Detection

Joyce Jiyoung Whang
Sungkyunkwan University (SKKU)

Link-based Web Spam Detection

- **Web Spam**
  - spurious links to get higher-than-deserved rankings.
- Web spam detection algorithms exploit the **hyperlink structure**.



Contributions

- Collect and share **two real-world web graphs** with labels
- **Two-level analysis** of link spam
  - Page-level graph and site-level graph
  - ATR is useful to detect real-world link spam
- Effective and scalable **site-level seeding methodology** for ATR
- **Asynchronous ATR** significantly reduces the computational cost of ATR

Real-world Web Graphs

- Crawled by the NAVER search engine (https://www.naver.com/)

		page-level graph G	site-level graph H
W1	No. of normal nodes	797,718 (93.15%)	39,809 (68.63%)
	No. of spam nodes	47,301 (5.52%)	7,954 (13.71%)
	No. of undefined nodes	11,385 (1.33%)	10,239 (17.66%)
	No. of total nodes	856,404	58,002
	No. of labeled edges	3,929,401 (99.33%)	83,351 (85.67%)
	No. of edges	3,955,939	97,294
W2	No. of normal nodes	797,018 (91.20%)	39,984 (67.32%)
	No. of spam nodes	65,259 (7.47%)	8,846 (14.89%)
	No. of undefined nodes	11,684 (1.34%)	10,561 (17.78%)
	No. of total nodes	873,961	59,391
	No. of labeled edges	3,952,584 (99.33%)	84,373 (85.68%)
	No. of total edges	3,979,280	98,478

Site-level Examination

- A set of **human-labeled seeds**

  → An input of a web spam detection method

- Perform a **site-level examination** followed by refinement of page labels.

- Human experts examine web sites instead of pages.
  - All pages inside a spam site are spam.
  - A normal web site may contain some spam pages

    → Exploit the URL structure to label spam pages

Two-level Analysis of Link Spam

- Most existing methods focus on either **a page-level graph** or **a site-level graph**, and do not consider both of the graphs.
- We **generalize the structure of link spam** by analyzing the characteristics of link spam on the two different levels of graphs.
  - **Practical solutions for large-scale web spam detection problems**

Edge Classification

- **Page-level Graph**
  - Normal pages tend to point to other normal pages (**TrustRank**)
  - Spam pages tend be referred by other spam pages (**Anti-TrustRank**)

		$\lvert\mathcal{E}\rvert$	$E(\lvert\mathcal{E}\rvert)$	conclusion	p-value
G	normal $\longrightarrow$ normal	3,639,884	3,500,494	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	7.0×10^{-23}
	normal $\longrightarrow$ spam	2,157	208,725	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	7.9×10^{-28}
	spam $\longrightarrow$ normal	73,049	207,807	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	7.2×10^{-55}
	spam $\longrightarrow$ spam	214,311	12,375	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	9.2×10^{-63}
H	normal $\longrightarrow$ normal	56,647	57,840	$\lvert\mathcal{E}\rvert \neq E(\lvert\mathcal{E}\rvert)$	2.6×10^{-2}
	normal $\longrightarrow$ spam	17,551	11,771	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	5.6×10^{-13}
	spam $\longrightarrow$ normal	4,394	11,418	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	9.1×10^{-28}
	spam $\longrightarrow$ spam	4,759	2,321	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	9.2×10^{-21}

Edge Classification

- **Site-level Graph**
  - The number of edges **from normal nodes to spam nodes** is also significant as well as the edges **from spam nodes to spam nodes**.

		$\lvert\mathcal{E}\rvert$	$E(\lvert\mathcal{E}\rvert)$	conclusion	p-value
G	normal $\longrightarrow$ normal	3,639,884	3,500,494	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	7.0×10^{-23}
	normal $\longrightarrow$ spam	2,157	208,725	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	7.9×10^{-28}
	spam $\longrightarrow$ normal	73,049	207,807	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	7.2×10^{-55}
	spam $\longrightarrow$ spam	214,311	12,375	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	9.2×10^{-63}
H	normal $\longrightarrow$ normal	56,647	57,840	$\lvert\mathcal{E}\rvert \neq E(\lvert\mathcal{E}\rvert)$	2.6×10^{-2}
	normal $\longrightarrow$ spam	17,551	11,771	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	5.6×10^{-13}
	spam $\longrightarrow$ normal	4,394	11,418	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	9.1×10^{-28}
	spam $\longrightarrow$ spam	4,759	2,321	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	9.2×10^{-21}

Edge Classification

- Consider **an incident node of a between-site edge**

  (i) The site is **normal** and the page is **normal**

  (ii) The site is **normal** but the page is **spam**

  (iii) The site is **spam** and the page is **spam**

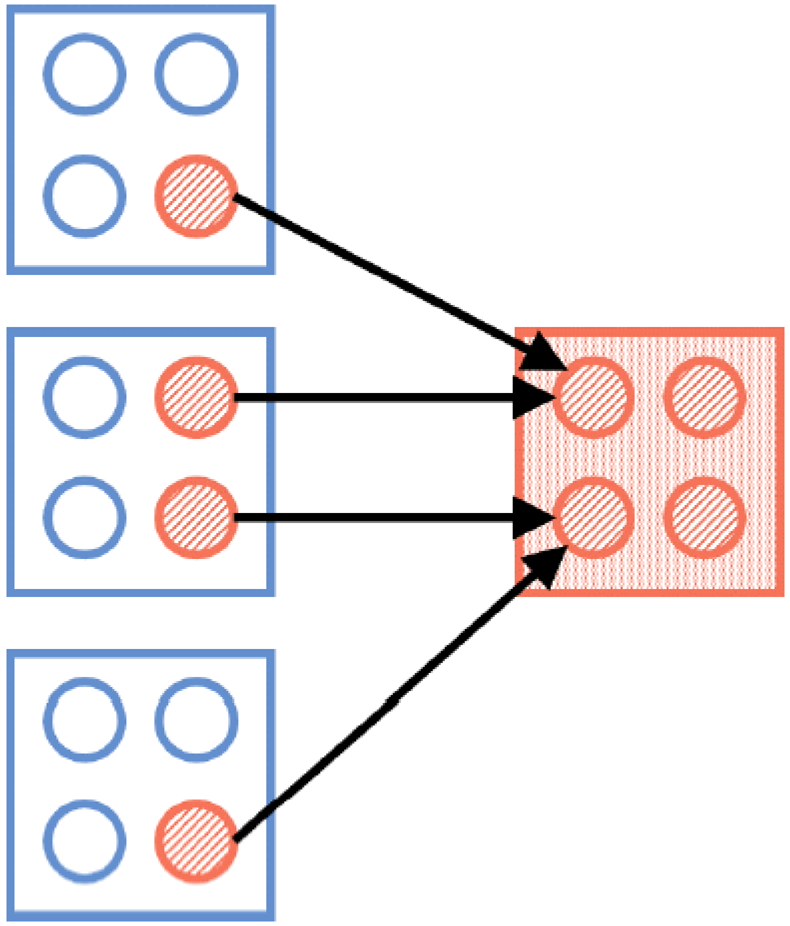
Edge Classification

- Three significant edge types: **NSNS, NSSS, SSSS**
  → spam to spam at the page-level graph
- NSSS: normal to spam at the site-level graph

Source		Destination		$\lvert\mathcal{E}\rvert$	$E(\lvert\mathcal{E}\rvert)$	conclusion	p-value
Site	Page	Site	Page				
Normal	Normal	Normal	Normal	857,565	666,284	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	2.0×10^{-20}
Normal	Normal	Normal	Spam	13	39,750	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	5.5×10^{-17}
Normal	Normal	Spam	Spam	1,205	5,611	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	5.1×10^{-10}
Normal	Spam	Normal	Normal	10,825	39,562	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	9.8×10^{-32}
Normal	**Spam**	**Normal**	**Spam**	52,392	2,357	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	4.9×10^{-55}
Normal	**Spam**	**Spam**	**Spam**	121,397	336	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	1.7×10^{-85}
Spam	Spam	Normal	Normal	5,953	7,361	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	1.3×10^{-5}
Spam	Spam	Normal	Spam	340	453	$\lvert\mathcal{E}\rvert < E(\lvert\mathcal{E}\rvert)$	2.6×10^{-3}
Spam	**Spam**	**Spam**	**Spam**	3,768	67	$\lvert\mathcal{E}\rvert > E(\lvert\mathcal{E}\rvert)$	2.0×10^{-52}

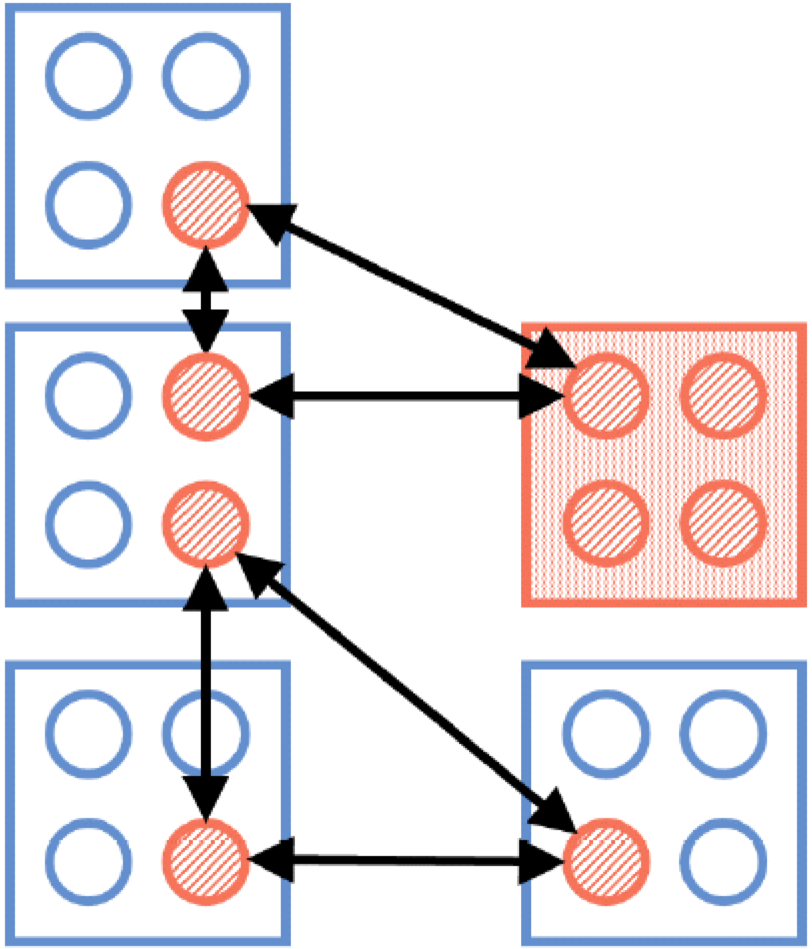
Web Spam via Two-level Edge Classification

- **Overpost**
  - A spammer makes a lot of postings in different normal sites to intrigue transactions into the targeting spam site.
  - The postings are spam pages which contain the links to the spam pages in the spam site.
  - This configuration makes the **NSSS** edge type.



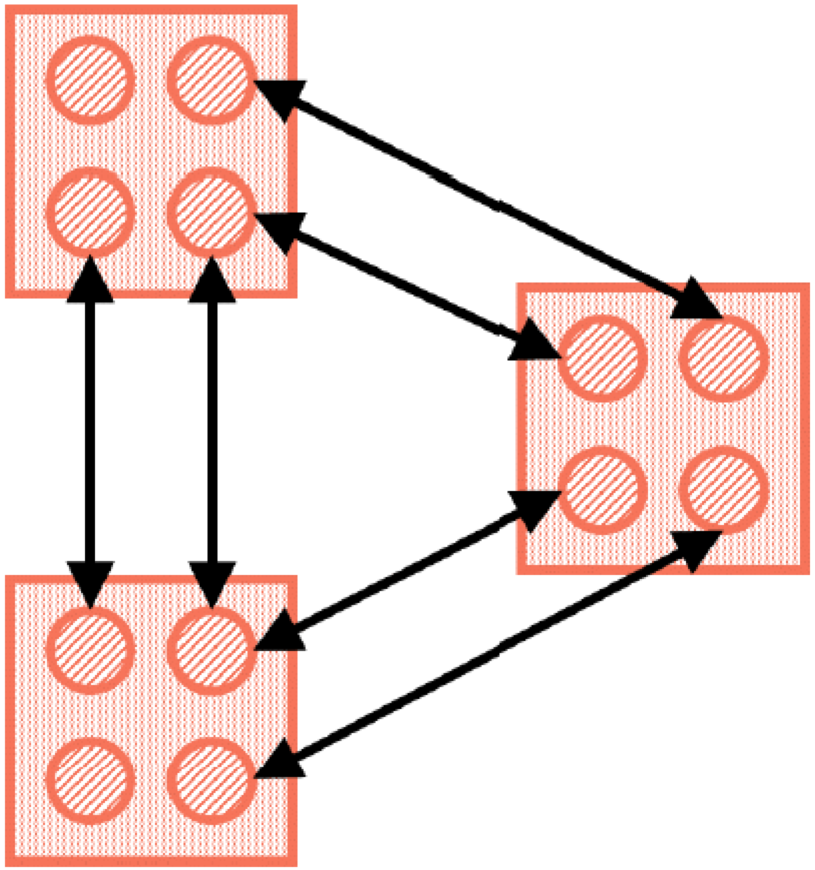
Web Spam via Two-level Edge Classification

- **Hacking**
  - A spammer hacks normal sites. The spammer makes spam pages in normal sites and the spam pages are linked to other spam pages.
  - We can observe the **NSSS** and **NSNS** edges.



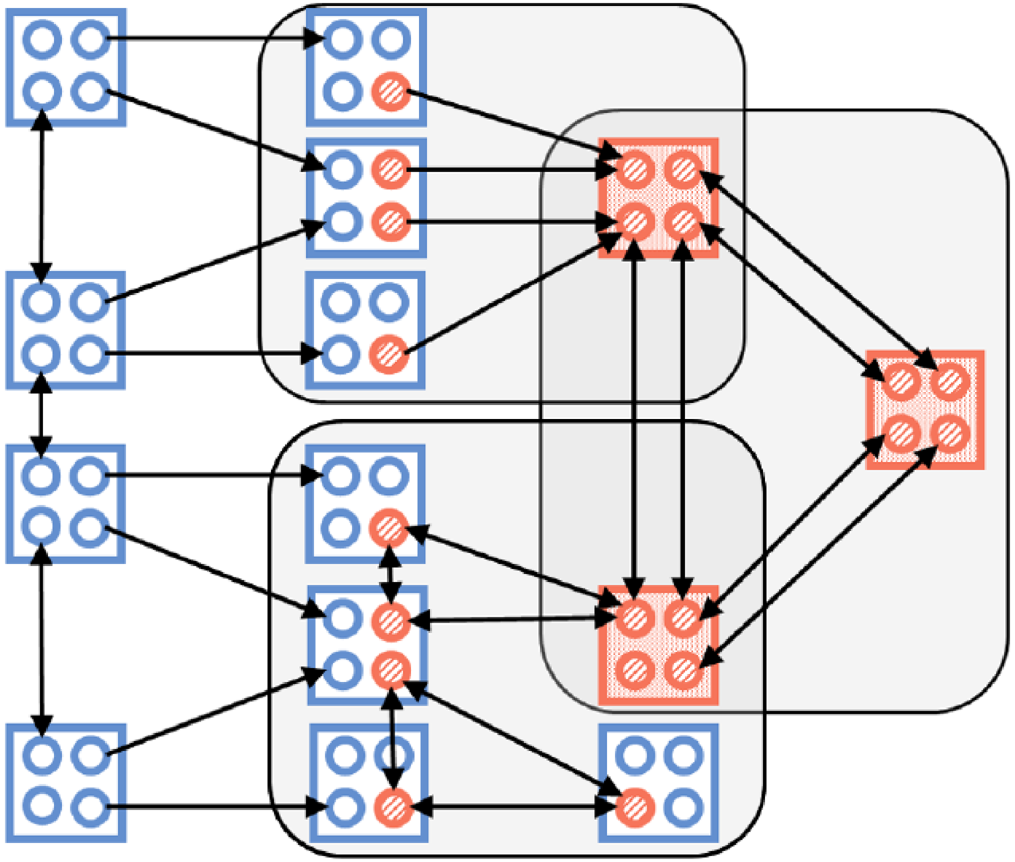
Web Spam via Two-level Edge Classification

- **Link Farm**
  - Some spam sites and spam pages are designed to be densely connected with each other to raise PageRank scores so that they can be indexed by a search engine.
  - We observe **SSSS** edge types.



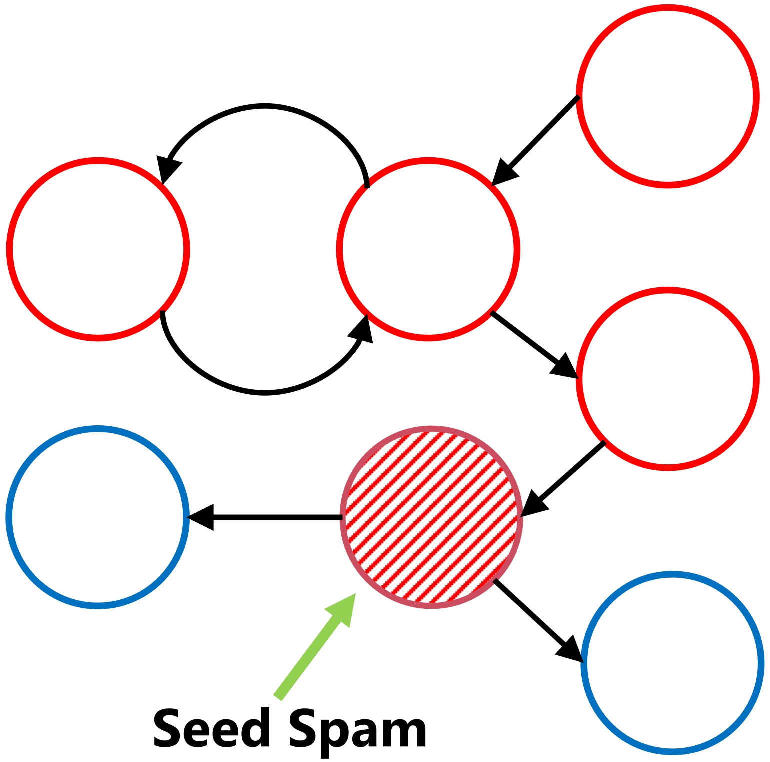
Web Spam via Two-level Edge Classification

- Real-world link spam can be explained by a combination of the aforementioned **building blocks**.



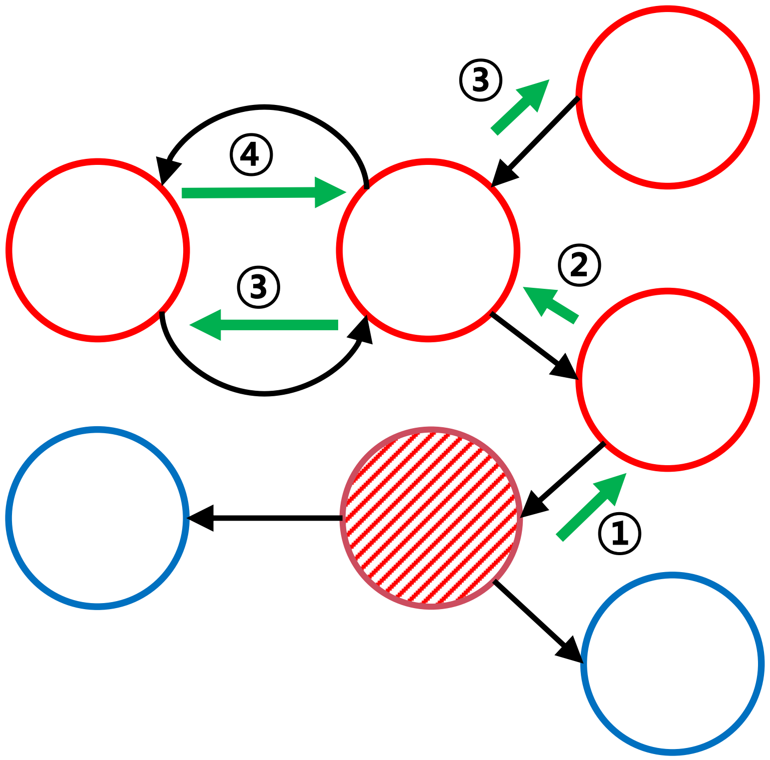
Anti-TrustRank with Qualified Site-level Seeds

- Anti-TrustRank (ATR)
  - Spam pages are likely to be referenced by other spam pages.
  - **Carefully select seed spam pages.**
  - Assign ATR scores to the seed spam pages.



Anti-TrustRank with Qualified Site-level Seeds

- Anti-TrustRank (ATR)
  - From the seeds, the ATR scores are propagated to incoming neighbors of the nodes so that **the pages having links to the spam pages end up with having high ATR scores**.
  - Pages with high ATR scores are considered as spam pages.



Anti-TrustRank with Qualified Site-level Seeds

- The spam seeds should be examined by human experts to get labels.
- Human experts conduct a site-level examination.
- **Represent each site as a feature vector** and build a classifier that predicts the probability of being spam.
- We **prioritize the websites** according to the probability for the **site-level examination**.

Anti-TrustRank with Qualified Site-level Seeds

- **Our features to model a site**
  - entro-in-p: the entropy of the indegrees of pages within a site

in-p: indegree of each page in the site h
out-p: outdegree of each page in the site h
dist: the distances from the site h to all other reachable sites on $\bar{H}$

entro-in-p: entropy of ***in-p***	**reachability**: no. of reachable sites on $\bar{H}$
entro-out-p: entropy of ***out-p***	**cluster**: whether h belongs to a spam cluster
mean-dist: mean of ***dist***	**dmnt-ratio**: max. weight/degree of h on $\bar{H}_w$
std-dist: standard deviation of ***dist***	**no-page**: no. of pages in the site h
max-dist: maximum of ***dist***	**in-page**: no. of pages having an edge to h
within-site: no. of within-site edges	**out-page**: no. of pages having an edge from h
in-h: indegree of the site h on H	**one-hop**: no. of one-hop distant sites on $\bar{H}$
out-h: outdegree of the site h on H	**two-hop**: no. of two-hop distant sites on $\bar{H}$

Anti-TrustRank with Qualified Site-level Seeds

- **Classification performance of the features**
  - Our features show better performance than node2vec features.

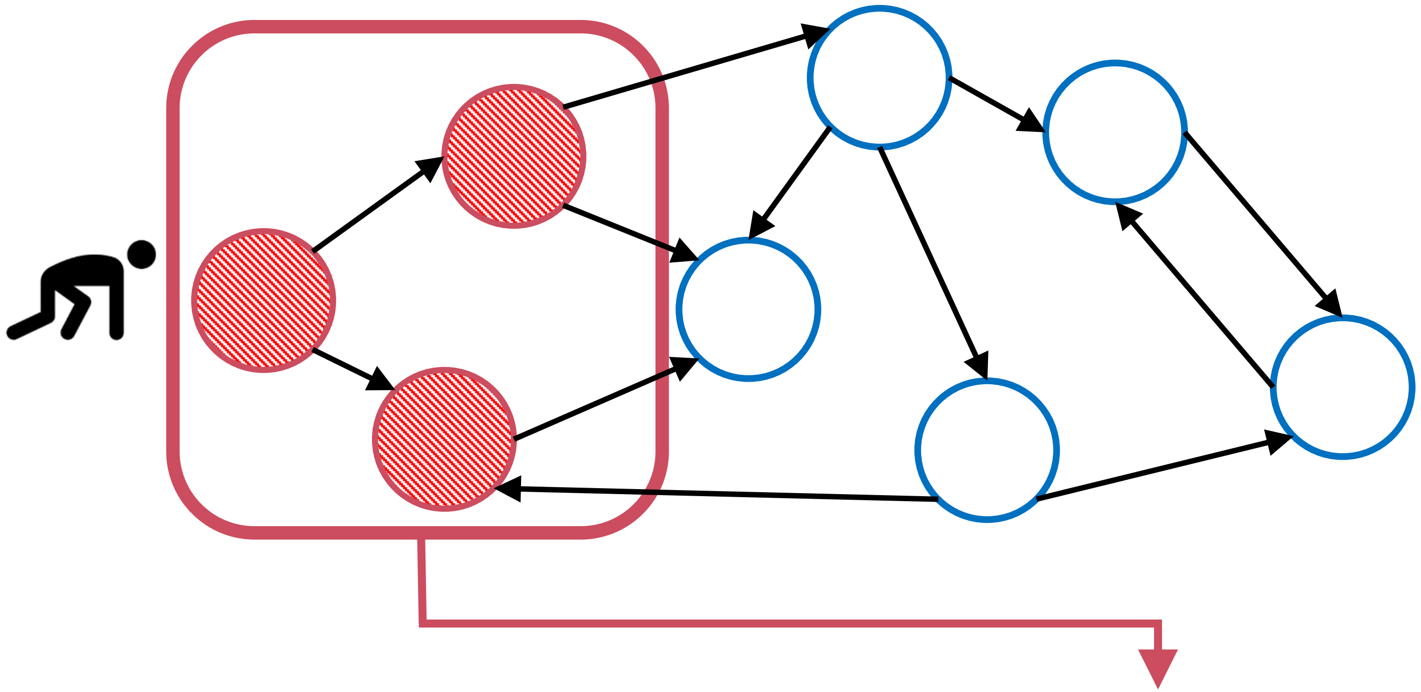
	W1		W2	
	node2vec	Our Features	node2vec	Our Features
Accuracy	83.9%	**88.0%**	82.7%	**88.1%**
Normal F1	90.6%	**92.1%**	89.7%	**92.2%**
Spam F1	46.1%	**86.1%**	45.1%	**86.1%**
Avg. Precision	70.5%	**88.8%**	70.2%	**89.0%**
Avg. Recall	66.8%	**89.4%**	65.7%	**89.3%**
Avg. F1	68.3%	**89.1%**	67.4%	**89.1%**

Work-Efficient Anti-TrustRank

- Computing **Anti-TrustRank (ATR)** scores is identical to computing the **personalized PageRank (PPR)** scores on the reverse graph.
  - Spam seeds in ATR → personalization set (predefined nodes) in PPR
- We propose **asynchronous Anti-TrustRank algorithms**
  - Reduce the computational cost of the traditional ATR algorithm
  - Without degrading performance in spam detection
  - Convergence analysis

Personalized PageRank

- **Randomly jump to one of the predefined nodes.**

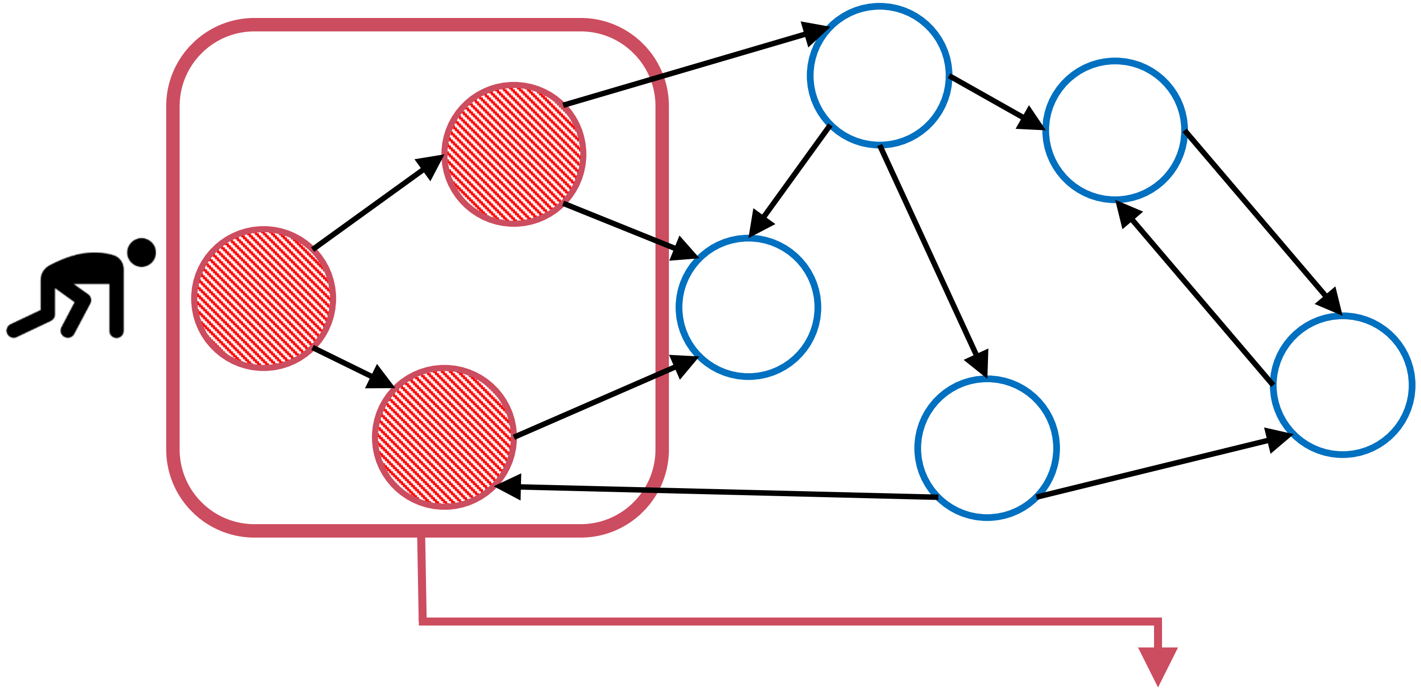


$$\boldsymbol{x} = \alpha \boldsymbol{P}^{T}\boldsymbol{x} + (1-\alpha)\boldsymbol{e_s}$$

($\boldsymbol{e_s}$: Personalized vector)

Anti-TrustRank

- Randomly jump to one of spam seeds on the reverse graph.



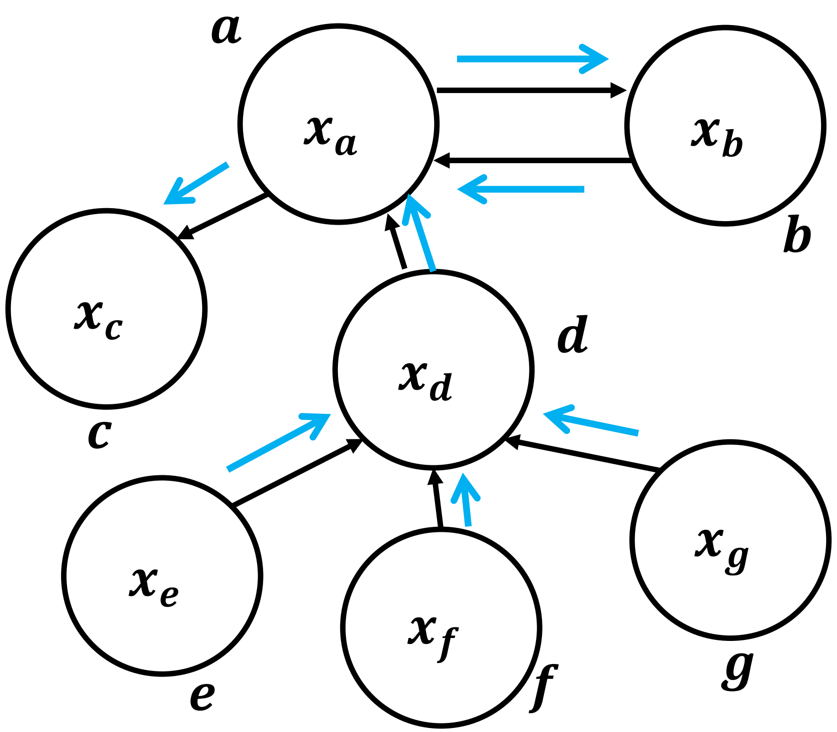
$$x = \alpha P^T x + (1 - \alpha) e_s$$

(e_s : Personalized vector)

Synchronous Anti-TrustRank

SYNC ATR

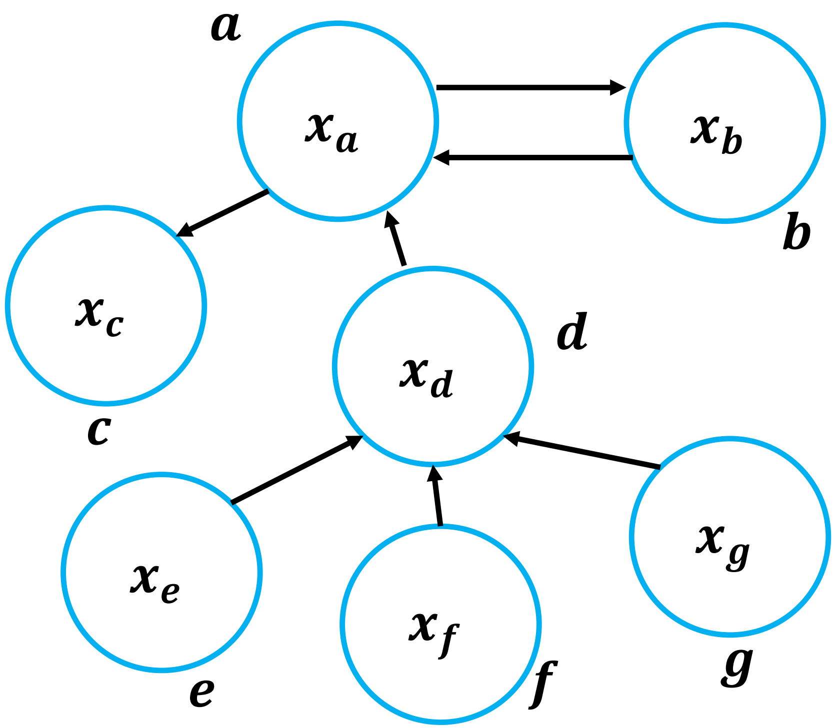
The ATR scores are updated after all the nodes re-compute the ATR scores.



Synchronous Anti-TrustRank

SYNC ATR

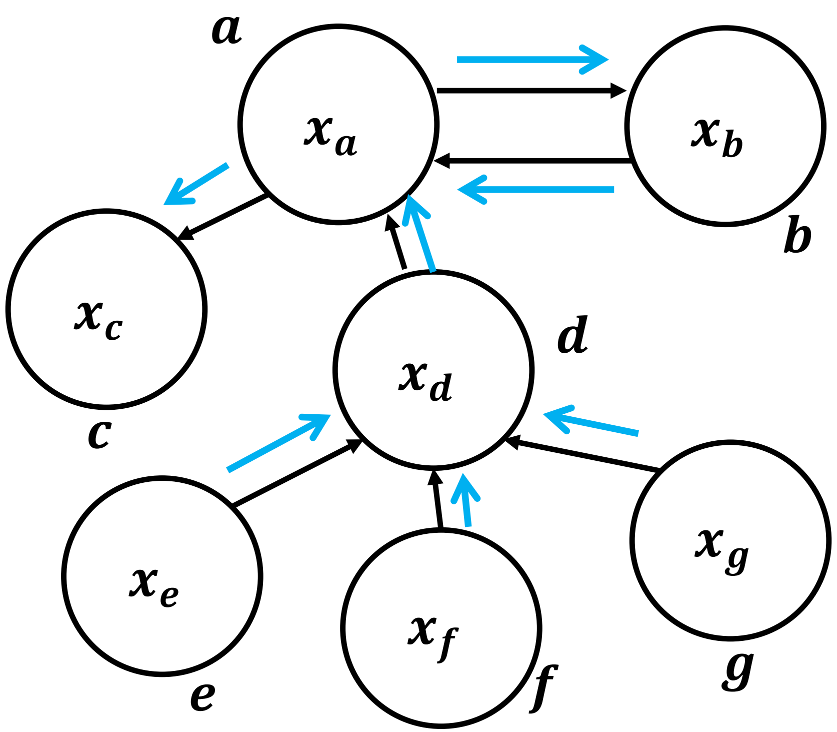
The ATR scores are updated after all the nodes re-compute the ATR scores.



Synchronous Anti-TrustRank

SYNC ATR

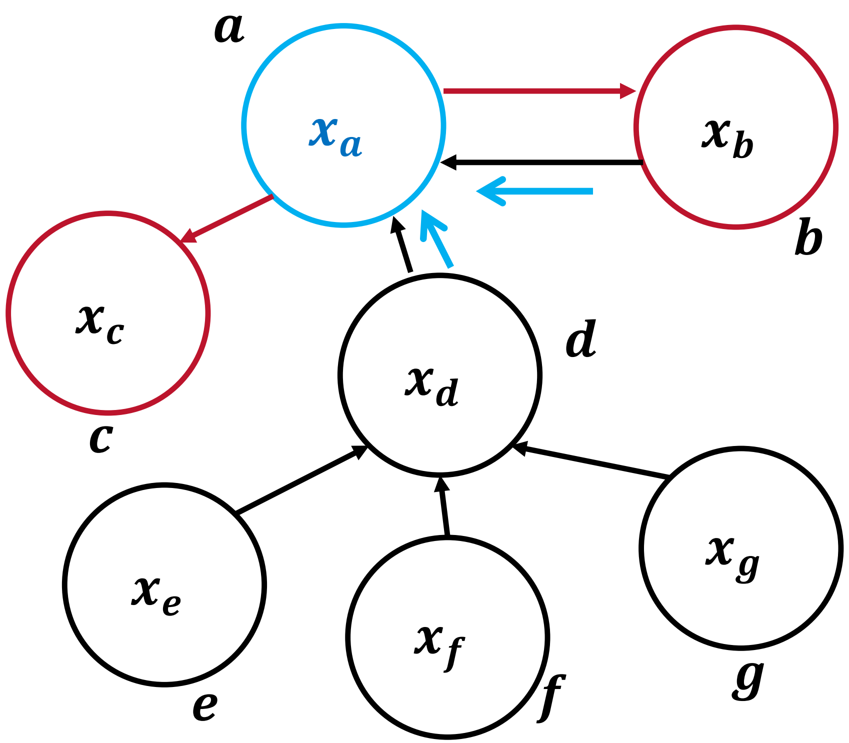
The ATR scores are updated after all the nodes re-compute the ATR scores.



Asynchronous Anti-TrustRank

ASYNC ATR

Worklist A set of nodes whose ATR scores need to be updated



Worklist

[b, c, d, e, f, g, **b**, **c**]

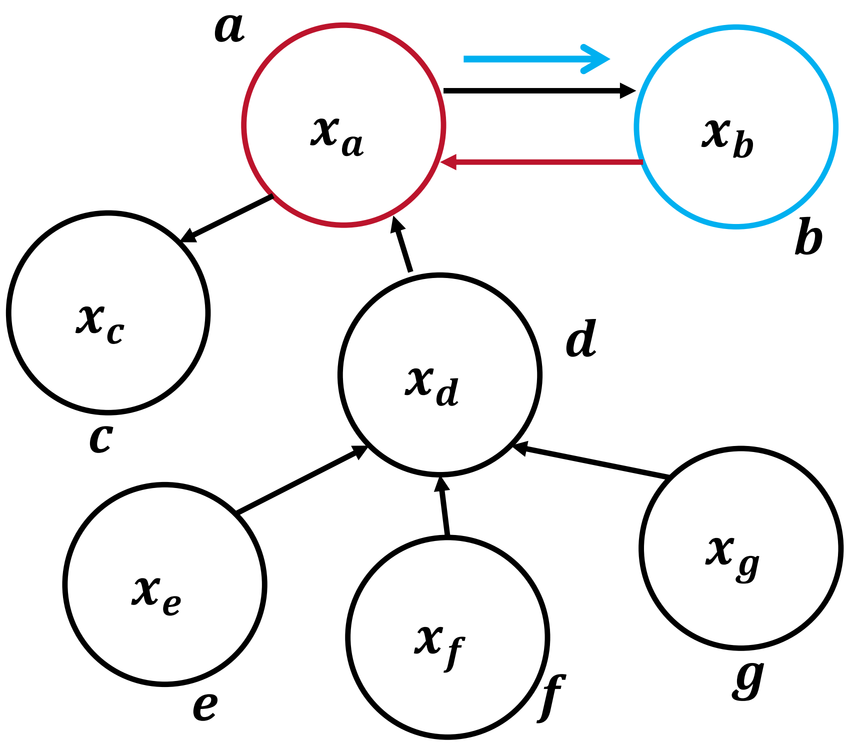
Pop a

Push b, c

Asynchronous Anti-TrustRank

ASYNC ATR

Worklist A set of nodes whose ATR scores need to be updated



Worklist

[c, d, e, f, g, **b**, **c**, **a**]

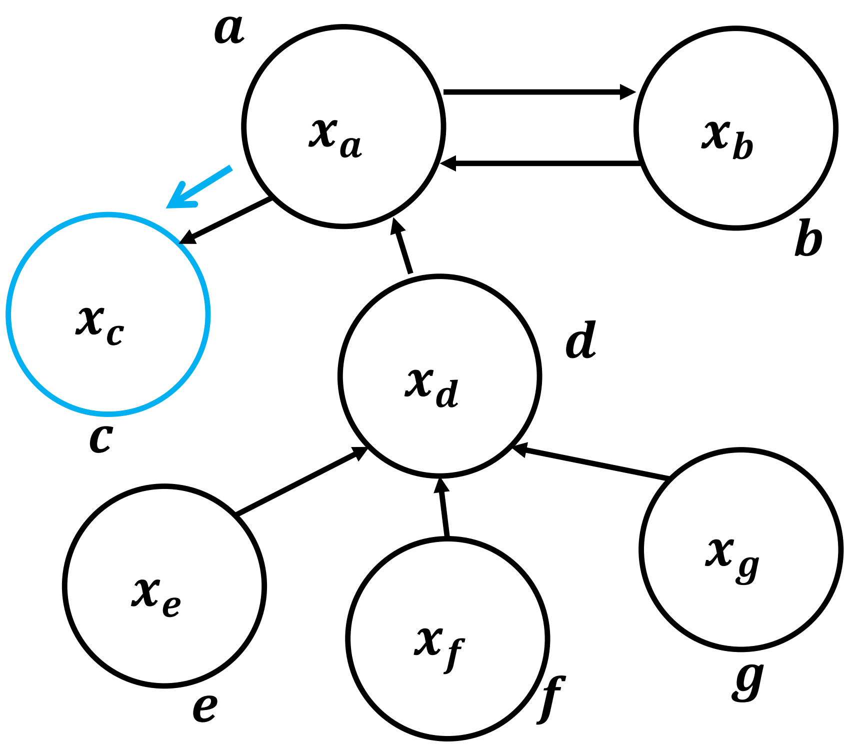
Pop b

Push a

Asynchronous Anti-TrustRank

ASYNC ATR

Worklist A set of nodes whose ATR scores need to be updated



Worklist

[d, e, f, g, **b**, **c**, **a**]

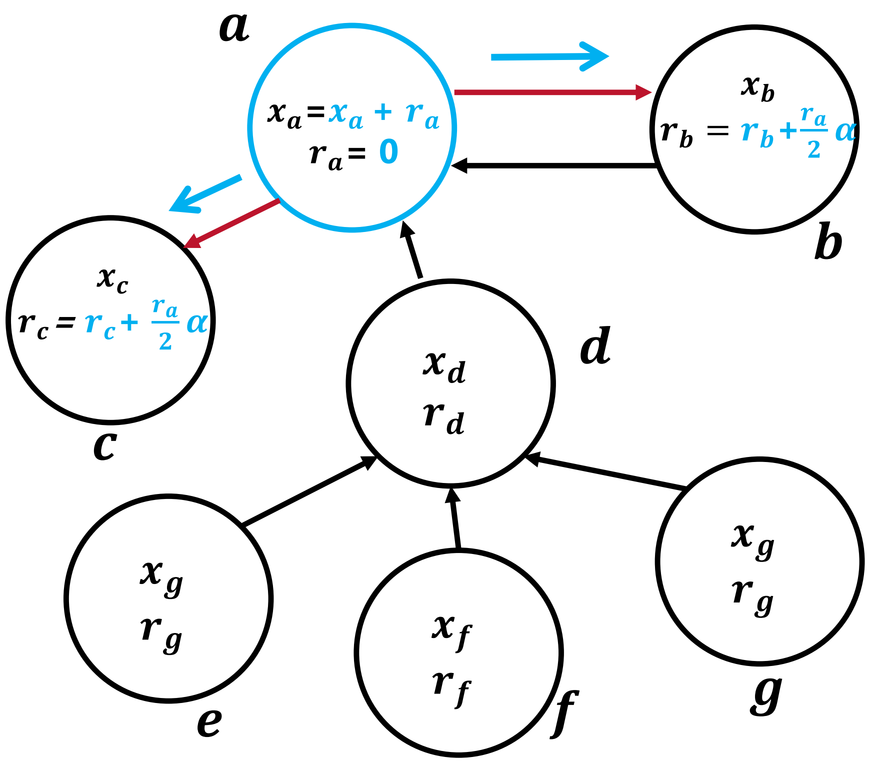
Pop c

Residual-based Asynchronous Anti-TrustRank

RASYNC ATR

new ATR = current ATR + current residual (explicitly maintain the residual of each node)

Filtering out unnecessary work in the worklist



Worklist

[b, c, d, e, f, g]

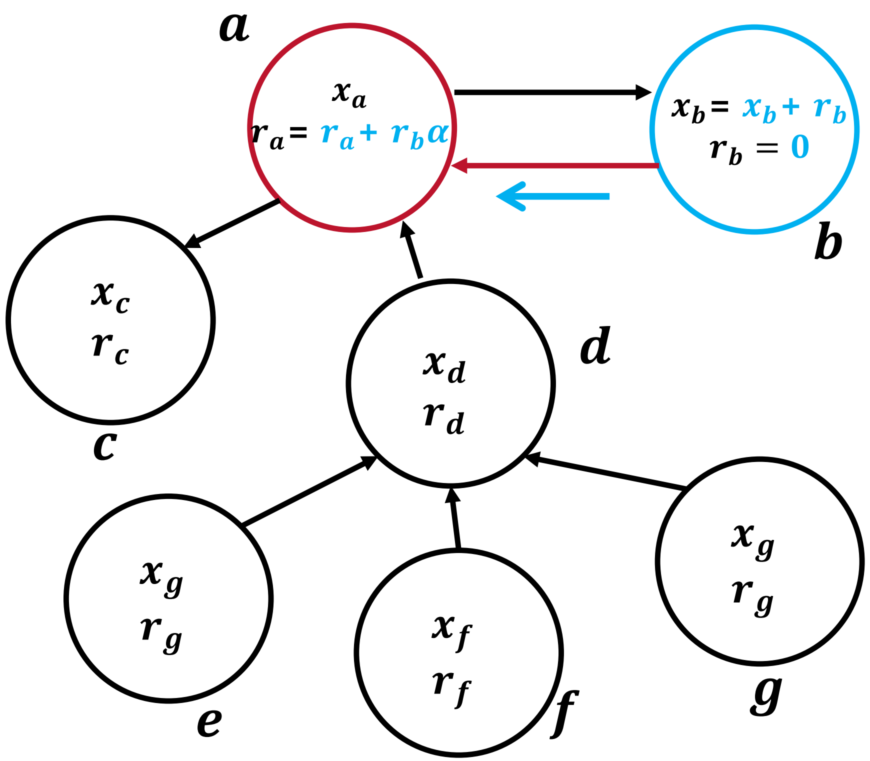
Pop a

Residual-based Asynchronous Anti-TrustRank

RASYNC ATR

new ATR = current ATR + current residual (explicitly maintain the residual of each node)

Filtering out unnecessary work in the worklist



Worklist

[c, d, e, f, g, **a**]

Pop b

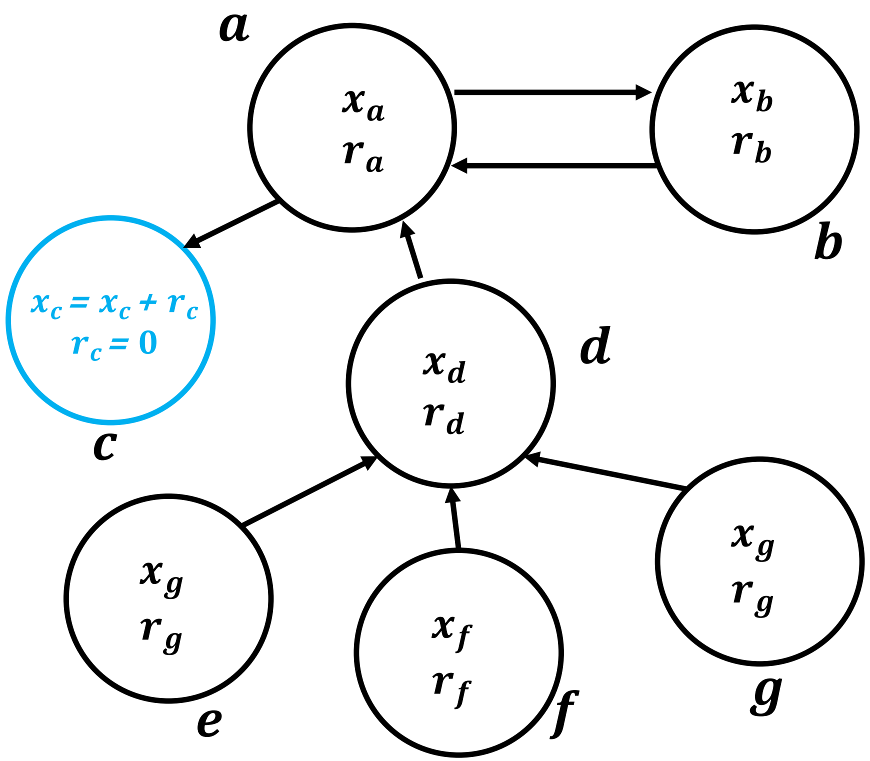
Push a

Residual-based Asynchronous Anti-TrustRank

RASYNC ATR

new ATR = current ATR + current residual (explicitly maintain the residual of each node)

Filtering out unnecessary work in the worklist



Worklist

[d, e, f, g, **a**]

Pop c

Convergence of Asynchronous Anti-TrustRank

The Anti-TrustRank $\boldsymbol{x}$ is computes as follow:

$$\boldsymbol{x} = \boldsymbol{\alpha P^T x} + (\mathbf{1} - \boldsymbol{\alpha})\boldsymbol{e_s}.$$

Where $\boldsymbol{P}$ is a row-stochastic matrix ($\boldsymbol{P} = \boldsymbol{D^{-1}A}$) and $\boldsymbol{e_s}$ is the personalized vector.
This is the linear system :

$$\left(\mathbf{1} - \boldsymbol{\alpha P^T}\right)\boldsymbol{x} = (\mathbf{1} - \boldsymbol{\alpha})\boldsymbol{e_s}.$$

and the residual :

$$\boldsymbol{r} = (\mathbf{1} - \boldsymbol{\alpha})\boldsymbol{e_s} - \left(\mathbf{1} - \boldsymbol{\alpha P^T}\right)\boldsymbol{x} = \boldsymbol{\alpha P^T x} + (\mathbf{1} - \boldsymbol{\alpha})\boldsymbol{e_s} - \boldsymbol{x}.$$

When the *j*-th node is processed, the residual is decreased by $\boldsymbol{r_j^k}(\mathbf{1} - \boldsymbol{\alpha})$.

$$\boldsymbol{e^T r^{(k+1)}} = \boldsymbol{e^T r^{(k)}} - \boldsymbol{r_j^k}(\mathbf{1} - \boldsymbol{\alpha}).$$

Asynchronous Anti-TrustRank Algorithms

Asynchronous Anti-TrustRank

- Require much fewer Anti-TrustRank updates as well as arithmetic operations with the same precision by maintaining a working set.

Residual-based Asynchronous Anti-TrustRank

- Significantly reduces the number of arithmetic computations.
- Able to effectively reduce the size of the working set by filtering out unnecessary computations.

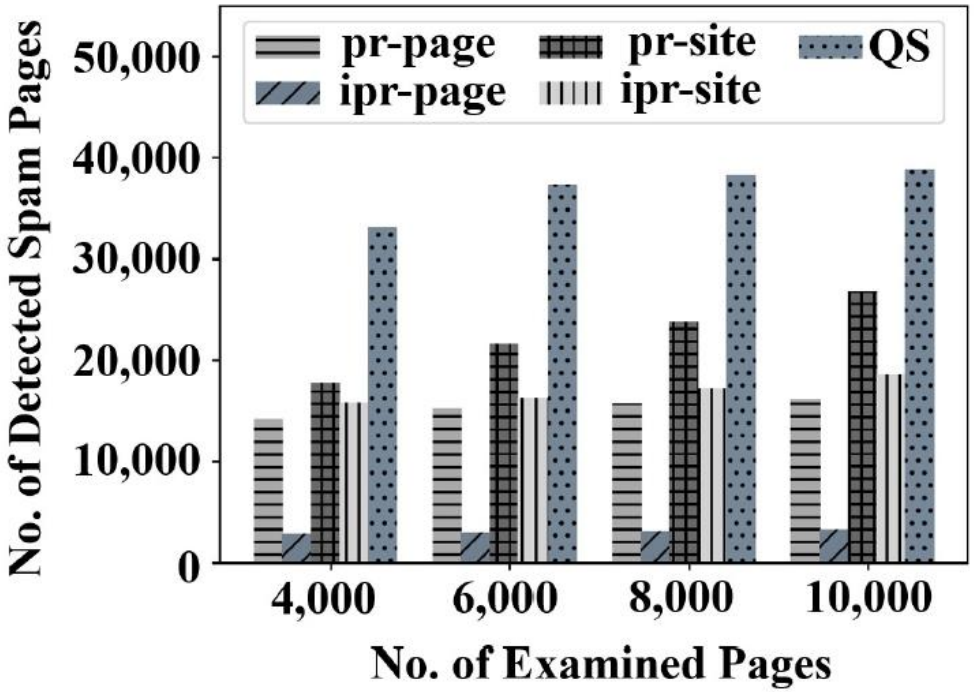
Experimental Results

- **Performance in web spam detection**
  - Our method (**QS**) significantly outperforms other methods.

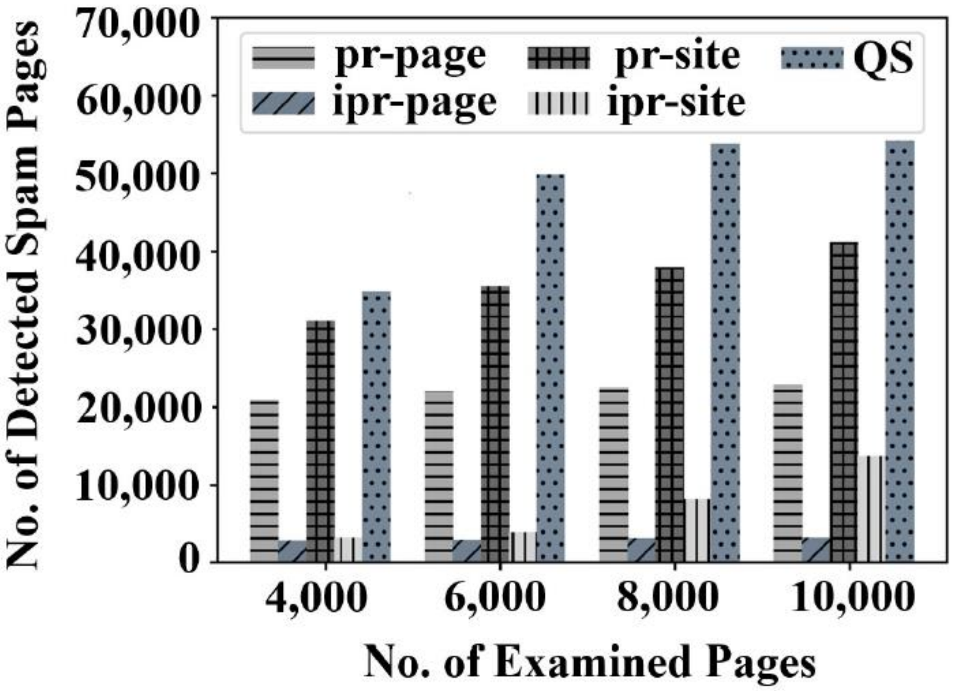
No. of Examined Pages		*lfeat*	*nvec*	*trust*	*pr-page*	*ipr-page*	*pr-site*	*ipr-site*	*QS*
4,000 (0.47% examined)	Accuracy	60.80%	94.50%	26.33%	96.00%	94.73%	96.41%	96.25%	**98.22%**
	F1 score	15.90%	5.80%	13.04%	45.67%	11.22%	53.90%	49.95%	**81.52%**
	Precision	9.00%	68.30%	6.98%	95.56%	98.43%	96.14%	99.05%	97.67%
	Recall	66.20%	3.00%	99.83%	30.01%	5.95%	37.45%	33.39%	69.95%
6,000 (0.70% examined)	Accuracy	89.20%	94.60%	27.39%	96.12%	94.75%	96.86%	96.31%	**98.71%**
	F1 score	21.40%	22.10%	13.21%	48.20%	11.75%	61.98%	51.03%	**87.22%**
	Precision	18.00%	57.00%	7.07%	95.51%	98.27%	96.34%	99.01%	97.60%
	Recall	26.40%	13.70%	99.87%	32.23%	6.25%	45.69%	34.37%	78.83%
10,000 (1.17% examined)	Accuracy	84.30%	94.40%	35.02%	96.21%	94.78%	97.47%	96.58%	**98.88%**
	F1 score	21.70%	30.90%	14.53%	50.16%	12.77%	71.46%	56.28%	**89.12%**
	Precision	15.10%	49.40%	7.84%	95.14%	98.09%	96.75%	98.89%	97.42%
	Recall	38.80%	22.50%	99.83%	34.06%	6.83%	56.65%	39.33%	82.13%

Experimental Results

- **No. of detected spam pages of the ATR algorithm with different seeding methods**
  - Our seeding method (**QS**) detects the largest number of spam pages.



(a) W1 dataset



(b) W2 dataset

Experimental Results

- **async** and **rasync** save much computation compared to **sync**.
- **rasync** reduces the number of arithmetic operations compared to **async**.

		sync	*async*	*rasync*
$\epsilon = 10^{-4}$	No. of Detected Spam Pages	33,088	33,029	33,029
$e = 4000$	F1 Score	81.52 %	81.67 %	81.67 %
	No. of ATR updates	51,384,240	46,680	46,454
	No. of Arithmetics	578,549,460	11,170,087	1,765,129
	Run Time (milliseconds)	7,596	339	87
$\epsilon = 10^{-8}$	No. of Detected Spam Pages	33,088	33,088	33,088
$e = 4000$	F1 Score	81.52 %	81.52 %	81.52 %
	No. of ATR updates	100,199,268	83,961	83,972
	No. of Arithmetics	1,128,171,447	13,009,448	2,673,169
	Run Time (milliseconds)	14,952	358	99

Experimental Results

- Run Time (milliseconds) of the algorithms
  - **rasync** is the fastest method.

		sync	*async*	*rasync*	*bstab*	*brppr*
W1	e=4,000, ϵ=10^{-4}	7,596	339	87	566	678
	e=4,000, ϵ=10^{-8}	14,952	358	99	1,217	680
	e=10,000, ϵ=10^{-4}	7,678	350	98	678	822
	e=10,000, ϵ=10^{-8}	14,628	374	111	1,775	829
W2	e=4000, ϵ=10^{-4}	6,526	556	148	821	726
	e=4,000, ϵ=10^{-8}	13,841	1,205	374	1,926	742
	e=10,000, ϵ=10^{-4}	6,212	607	169	707	968
	e=10,000, ϵ=10^{-8}	13,174	1,406	453	1,546	948

Experimental Results

- Parallel sync, async, and rasync on distributed machines
  - **rasync** is the fastest method.

Data Information			Run Time (minutes)		
No. of nodes	No. of edges	Size of $\mathcal{S}$	*sync*	*async*	*rasync*
59,180,800	82,824,237	2,340,940	86	94	37
152,595,632	274,392,463	3,329,026	191	162	69
57,135,532	732,008,321	4,381,555	516	351	121
556,047,762	1,207,335,482	5,016,499	>2,116	>1,413	163

Conclusion

- We develop a **site-level seeding methodology** for the ATR algorithm, which leads to remarkably boosting up the performance of the ATR algorithm.
- We design a work-efficient **asynchronous ATR algorithm** which significantly reduces the computational cost of the traditional ATR method while guaranteeing convergence.
- Our methodologies can be **integrated into other spam detection models** in practice, e.g., considering both TrustRank and Anti-TrustRank.

Big Data Lab

- Email : jjwhang@skku.edu
- Homepage : http://bigdata.cs.skku.edu
- Office : Engineering Building 2, #27326
- Lab : Engineering Building 2, #26315B