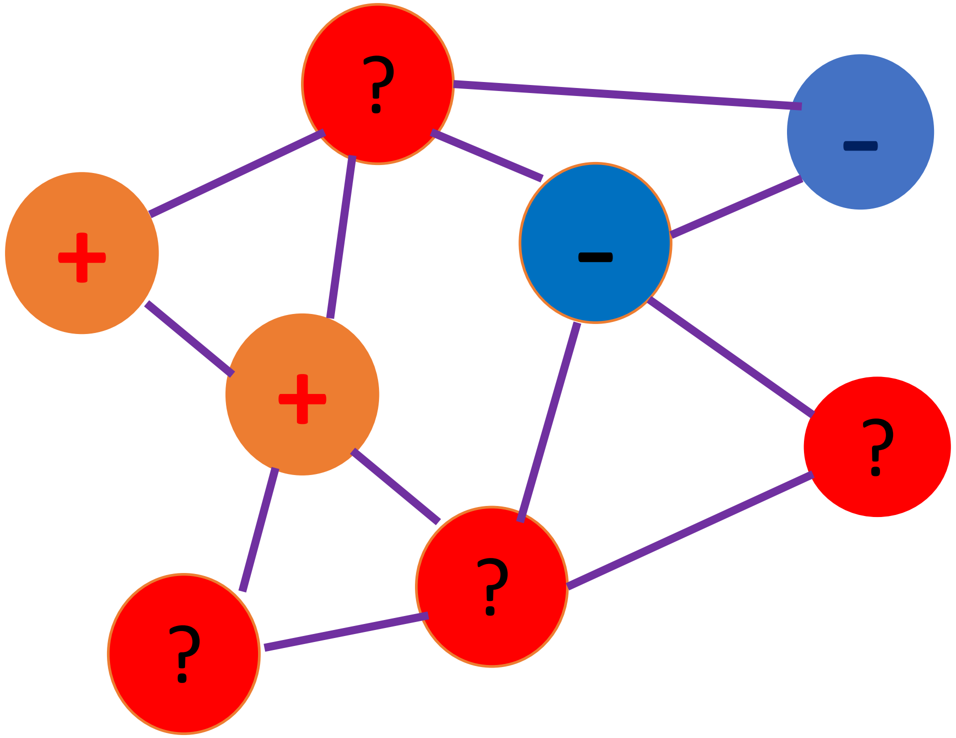
Graph-based Security and Privacy Analytics via Collective Classification with Joint Weight Learning and Propagation

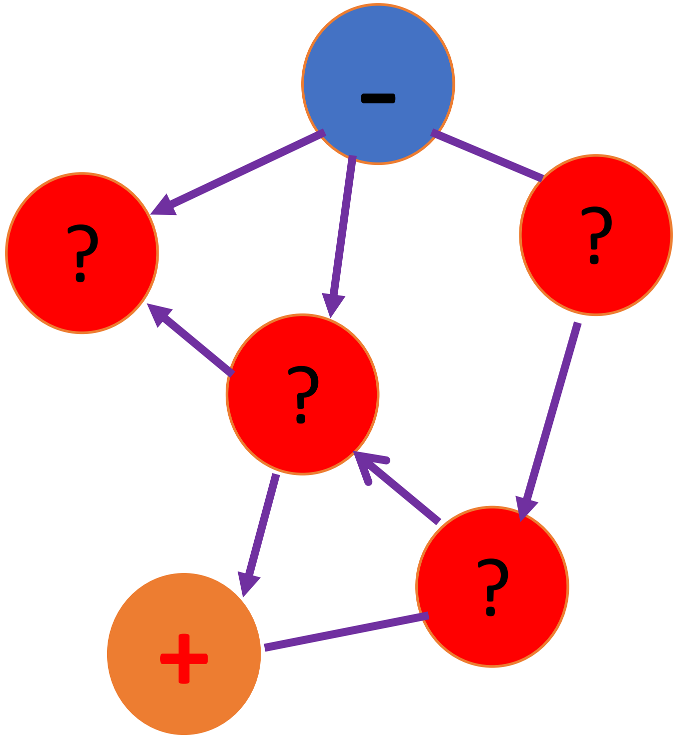
Binghui Wang, Jinyuan Jia, and Neil Zhenqiang Gong

Department of Electrical and Computer Engineering

IOWA STATE UNIVERSITY

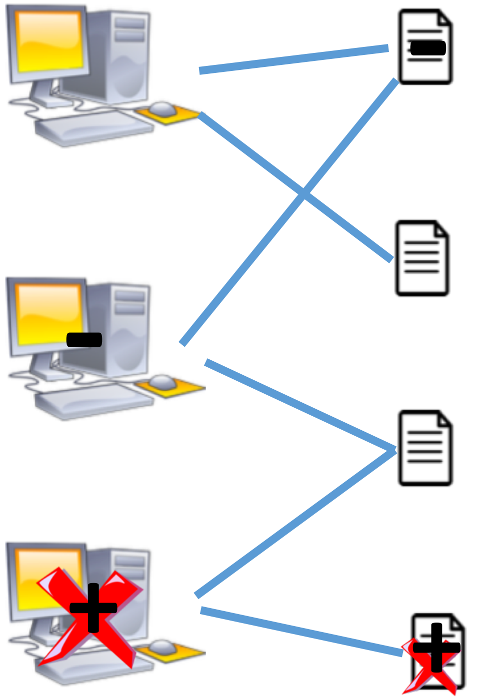
What is Collective Classification?





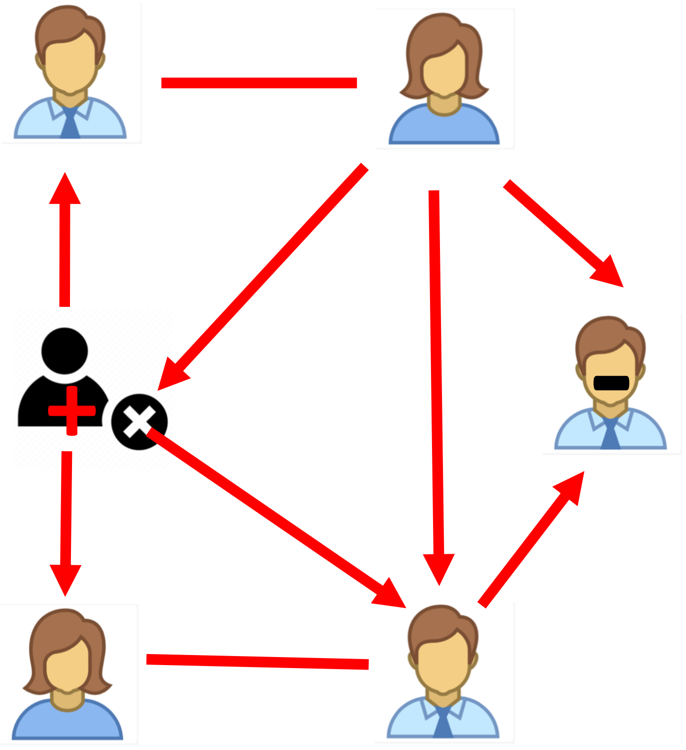
Modeling Security & Privacy Problems as Collective Classification

Malware detection



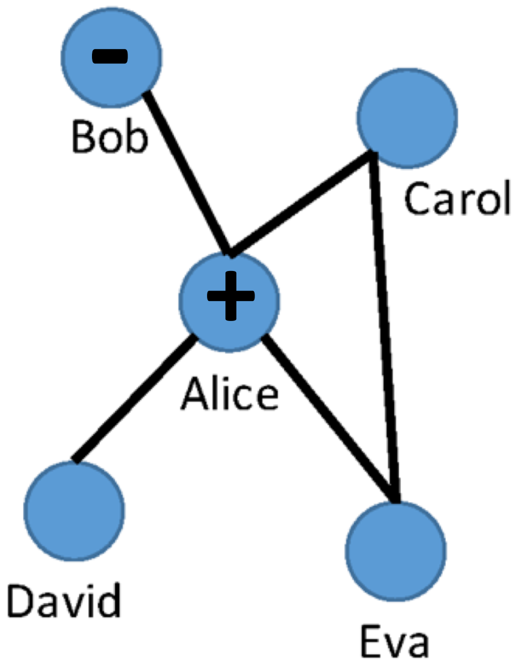
Host-file graph

Sybil detection



Social graph

Attribute inference



Social graph

Existing Collective Classification Methods

- Studied by multiple research communities
  - Networking, security, machine learning, data mining, etc.

- Classified as Random walk (RW) and Loopy belief propagation (LBP)

- Three key steps:
  - Step I: assign nodes' prior scores based on a training dataset
  - Step II: assign (fixed/equal) weight to every edge in the graph
  - Step III: obtain nodes' posterior scores by propagating nodes' prior scores among the weighted graph; larger posterior score indicates a higher likelihood to be positive

Fundamental Limitation of Existing Methods

- Assign small weights to a large number of homogeneous edges
  - *homogeneous* edge (u,v) => u and v *have the same label => large weight*
- Assign large weights to a large number of heterogeneous edges
  - *heterogeneous* edge (u,v) => u and v *have different labels => small weight*
- Limited success in security and privacy problems having a large amount of heterogeneous edges
  - e.g., Sybil detection in weak-trust social networks (like Twitter)

Our Work: Joint Weight Learning and Propagation

- Jointly learning edge weights and propagating posterior scores
- Applicable to both RW-based and LBP-based methods
- Applicable to both undirected and directed graphs
- Applicable to various graph-based security and privacy problems
    - Sybil detection in social networks
    - Fake review detection
    - Attribute inference in social networks
    - Malware detection
    - Malicious website detection
    - …

Outline

- Background
- Methodology
- Evaluation
- Conclusion

Outline

- Background
- Methodology
- Evaluation
- Conclusion

Collective Classification

- Nodes' posterior scores are solutions to a system of equations:

$$\mathbf{p} = f(\mathbf{q}, \mathbf{W}, \mathbf{p})$$

  - **q, p**: nodes' prior and posterior scores
  - **W**: edge weight matrix
  - ***f***: different methods use different function *f*

- Iteratively updating the posterior scores:

$$\mathbf{p}^{(t+1)} = f(\mathbf{q}, \mathbf{W}, \mathbf{p}^{(t)}),\ \mathbf{p}^{(0)} = \mathbf{q}.$$

LBP on Undirected Graphs

- Function f

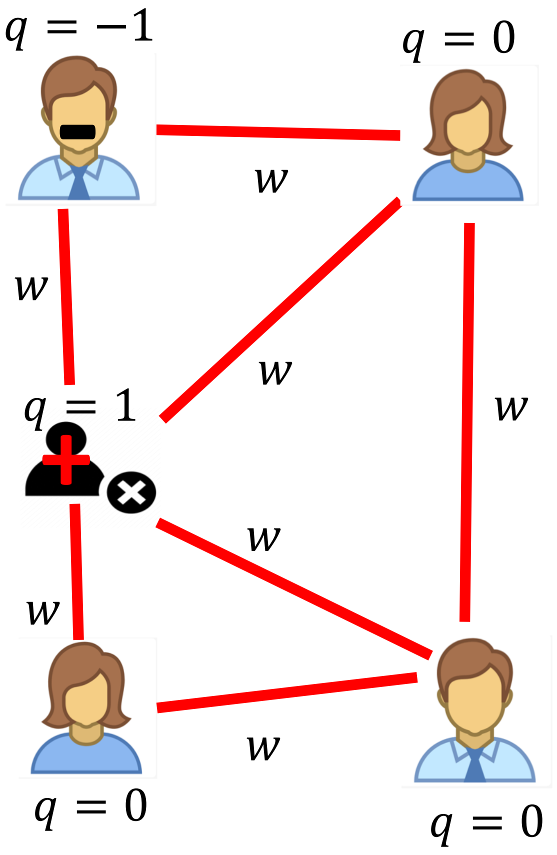
$$f(\mathbf{q}, \mathbf{W}, \mathbf{p}) = \mathbf{q} + \mathbf{W}\mathbf{p},$$

- Nodes' prior scores

$$q_u = \begin{cases} \theta & \text{if } u \in L_P \\ -\theta & \text{if } u \in L_N \\ 0 & \text{otherwise,} \end{cases}$$

  - L_p, L_N: labeled positive and labeled negative nodes
  - $\theta > 0$: strength of the prior

- Edge weight
  - $w_{uv}>0$: u, v likely to have the same label
  - $w_{uv}<0$: u, v likely to have different labels
  - $\mathrm{w}_{\mathrm{uv}}=\mathrm{w} > 0$, i.e., assume ***all edges*** homogeneous!



Outline

- Background
- Methodology
- Evaluation
- Conclusion

Motivation

- Existing methods assign large weights to a large number of heterogeneous edges
- Existing methods assign small weights to a large number of homogeneous edges
- Can we adaptively learn edge weights such that heterogeneous (homogeneous) edges have small (large) weights?

Goals

- Goal 1: *final* posterior scores of labeled nodes should be close to nodes' labels

- Quantifying Goal 1:

$$L(\mathbf{W}) = \frac{1}{2} \sum_{l \in L} (p_l - y_l)^2,$$

  - $y_l = 1$, if l is labeled positive
  - $y_l = -1$, if l is labeled negative
  - L(W): loss function over the training dataset

Goals

- Goal 2: edge weights and *final* posterior scores be *consistent*
  - u and v predicted the *same label* => edge (u,v) *homogeneous*
  - u and v predicted *different labels* => edge (u,v) *heterogeneous*

- Quantifying Goal 2:
  - $p_u p_v > 0$ => $w_{uv} > 0$
  - $p_u p_v < 0$ => $w_{uv} < 0$
  - C(W): regularization term

$$\mathrm{C}(\mathbf{W}) = \sum_{(u,v)\in E} p_u p_v w_{uv},$$

Learning Edge Weights via Gradient Descent

- Optimization problem:

$$\min_{\mathbf{W}} \mathcal{L}(\mathbf{W}) = \mathrm{L}(\mathbf{W}) - \lambda \mathrm{C}(\mathbf{W}), \qquad f(\mathbf{q}, \mathbf{W}, \mathbf{p}) = \mathbf{q} + \mathbf{W}\mathbf{p},$$

- Gradient descent: $w_{uv} \leftarrow w_{uv} - \gamma \frac{\partial \mathcal{L}(\mathbf{W})}{\partial w_{uv}},$

- Solving a linear system for each edge:

$$\frac{\partial \mathbf{p}}{\partial w_{uv}} = \mathbf{1}_v + \mathbf{W} \frac{\partial \mathbf{p}}{\partial w_{uv}}.$$

Computationally infeasible for large graphs!

Alternative Goals

- Computational challenge due to two goals using final posterior scores
- Instead, quantify the two goals using the current posterior scores
- Given posterior scores $p^{(t)}$ and edge weights $W^{(t-1)}$, we learn $W^{(t)}$
  - Goal 1’: posterior scores $p^{(t+1)}$of labeled nodes should be close to their labels
  - Goal 2’: edge weights $W^{(t)}$ and posterior scores $p^{(t)}$ should be *consistent*

Joint Weight Learning and Propagation

- Propagating posterior reputation scores $p^{(t)}$:

$$\mathbf{p}^{(t+1)} = f(\mathbf{q}, \mathbf{W}^{(t)}, \mathbf{p}^{(t)}).$$

- Learning weight matrix $W^{(t)}$:

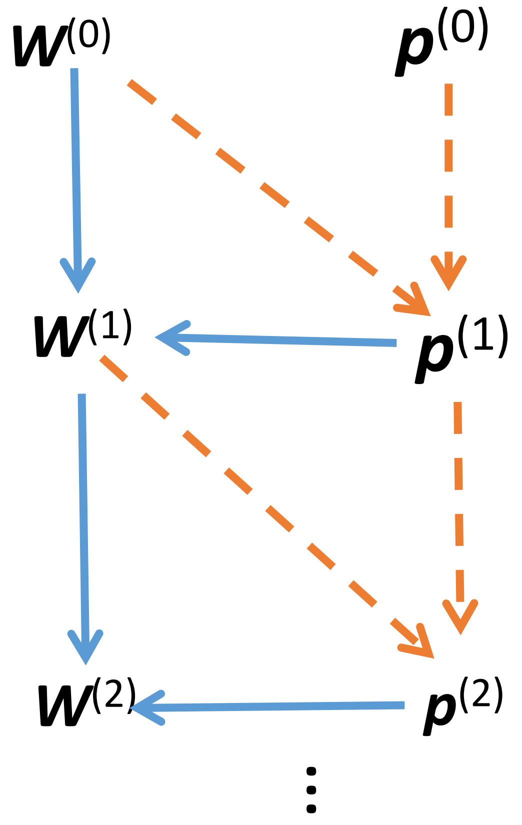
$$\min_{\mathbf{W}^{(t)}} \mathcal{L}(\mathbf{W}^{(t)}) = \frac{1}{2}\sum_{l \in L}(p_l^{(t+1)} - y_l)^2 - \lambda \sum_{(u,v)\in E} p_u^{(t)} p_v^{(t)} w_{uv}^{(t)},$$

- Gradient descent ($p^{(t)}$ is known):

$$\frac{\partial \mathcal{L}(\mathbf{W}^{(t)})}{\partial w_{uv}^{(t)}} = \sum_{l \in L}(p_l^{(t+1)} - y_l)\frac{\partial p_l^{(t+1)}}{\partial w_{uv}^{(t)}} - \lambda p_u^{(t)} p_v^{(t)}.$$

LBP for undirected graphs:

$$\frac{\partial p_l^{(t+1)}}{\partial w_{uv}^{(t)}} = \begin{cases} p_v^{(t)} & \text{if } u = l \\ p_u^{(t)} & \text{if } v = l \\ 0 & \text{otherwise} \end{cases}$$



Computationally efficient!

Outline

- Background
- Methodology
- Evaluation
- Conclusion

Experimental Setup

- Application scenarios
  - Security problem: Sybil detection & fake review detection
  - Privacy problem: Attribute inference attack

- Datasets

Dataset	#Nodes	#Edges	Ave. degree
Twitter	41,652,230	1,468,364,884	71
Sina Weibo	3,538,487	652,889,971	369
Yelp	520,230	718,144	3
Google+	5,735,175	30,644,909	11

Experimental Setup

- Training datasets
  - Twitter: 3000 Sybils and 3000 benign users
  - Sina Weibo: 980 labeled users
  - Yelp: 1000 fake reviews and 1000 genuine reviews
  - Google+: 75% users who have at least one city

- Evaluation metrics
  - AUC
  - Learnt edge weights
  - Scalability

Compared Methods

- RW-based methods
  - For undirected graphs: RW-N, RW-P, RW-B, RW-FLW
  - For directed graphs: RW-N-D, RW-P-D
- LBP-based methods
  - For undirected graphs: LBP-U, LBP-FLW-U
  - For directed graphs: LBP-D
- Our proposed methods
  - LBP-JWP-w/o, LBP-JWP-L1, LBP-JWP-L2, LBP-JWP

AUC Performance

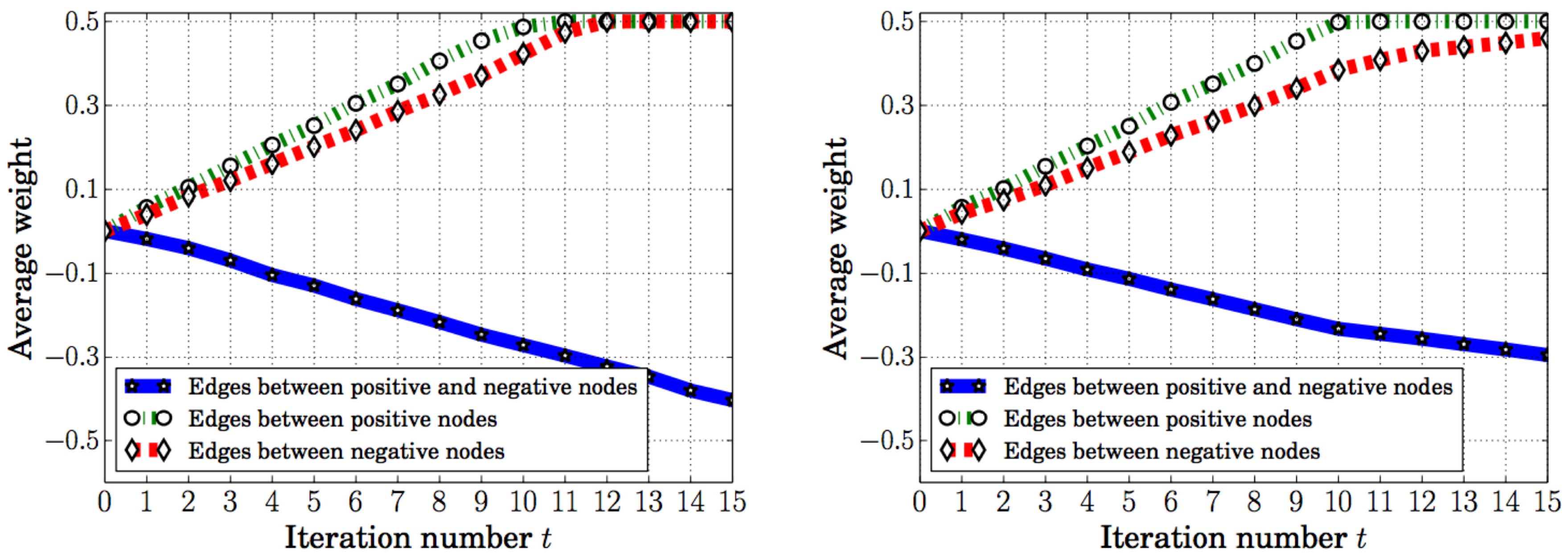
Methods		Twitter	Sina Weibo	Yelp	Google+
RW	RW-N-U	0.57	0.61	0.55	0.59
	RW-P-U	0.58	0.61	0.57	0.58
	RW-LFW-U	0.53	0.54	0.48	0.57
	RW-B-U	0.63	0.68	0.58	0.63
LBP	LBP-U	0.64	0.68	0.58	0.66
	LBP-FLW-U	0.62	0.66	0.58	0.66
Ours	LBP-JWP-w/o-U	0.69	0.74	0.60	0.69
	LBP-JWP-L1-U	0.65	0.70	0.59	0.66
	LBP-JWP-L2-U	0.68	0.72	0.60	0.68
	LBP-JWP-U	**0.73**	**0.77**	**0.62**	**0.72**

Methods		Twitter	Sina Weibo
RW	RW-N-D	0.60	0.66
	RW-P-D	0.63	0.64
LBP	LBP-D	0.72	0.80
Ours	LBP-JWP-w/o-D	0.75	0.82
	LBP-JWP-L1-D	0.72	0.79
	LBP-JWP-L2-D	0.73	0.80
	LBP-JWP-D	**0.78**	**0.85**

Our methods consistently outperform existing ones

Jointly edge weight learning and propagation indeed enhances performance

Learnt Edge Weights



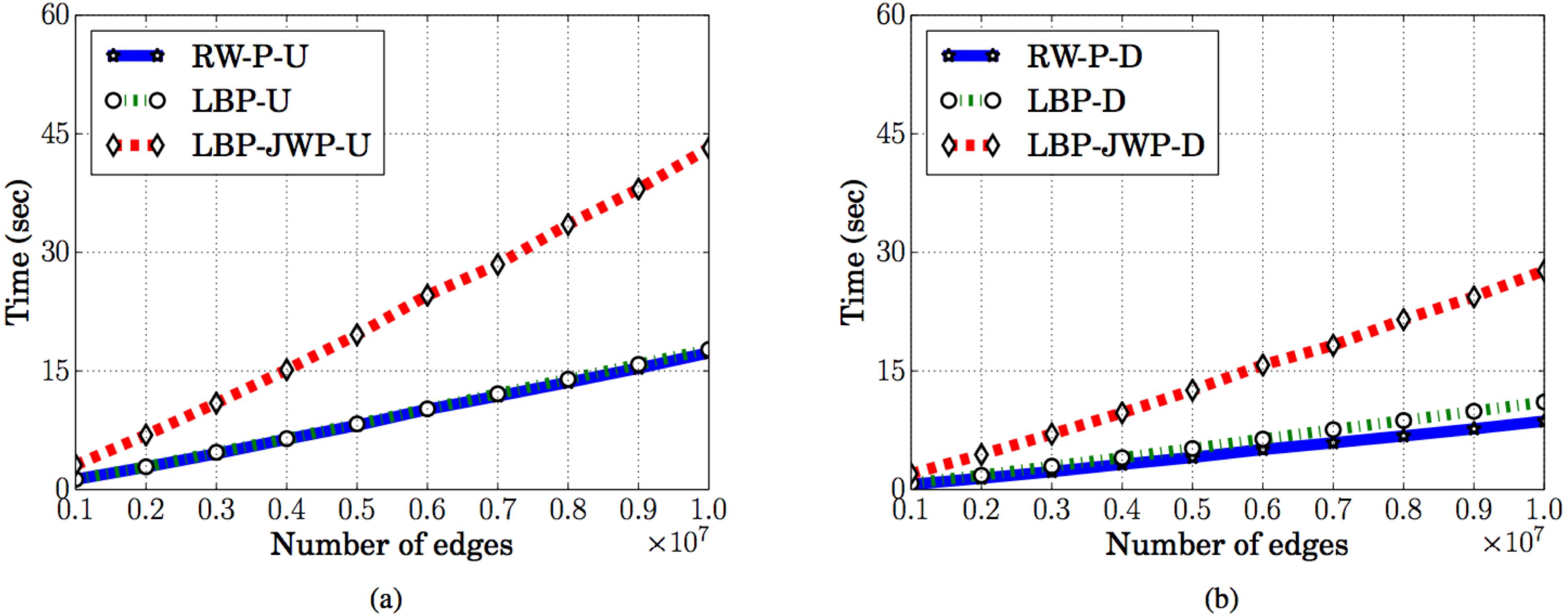
(a) LBP-JWP-U

(b) LBP-JWP-D

The average edge weights between positive nodes and negative nodes decrease

The average edge weights between negative (or positive) nodes increase

Scalability



Our methods are only 2-3 times slower than state-of-the-art methods

Outline

- Background
- Methodology
- Evaluation
- Conclusion

Conclusion

- We propose a general framework to learn edge weights for graph-based security and privacy analytics
- Our framework is applicable to both RW-based and LBP-based methods, and both undirected and directed graphs
- Iteratively learning edge weights can enhance performance for various graph-based security and privacy applications, with an acceptable computational overhead