# Node2Vec

## Objective

- Prediction over nodes and edges
  - For node, predict the most probable labels of nodes in a network. (e.g., predict interest of users in social network, predict functional labels of proteins in a protein-protein interaction network)
  - For edge, predict if there is an edge between a pair of nodes. (e.g., identify real-world friends in social network, discover novel interactions between genes)

## Objective

- Feature learning over the network is important
- However, current technologies cannot make a good balance between efficiency and accuracy
  - PCA and Multi-Dimensional scaling approach are expensive for large real-world network, poor performance on latent relation prediction over network
  - Some recent work has applied single hidden layer feedforward neural network to do it, however, they focus on rigid notion of neighborhood, without flexibility. (e.g., triangle is rigid, rectangle is not, since it can be changed to parallelogram), insensitivity to connectivity pattern in network

# Objective

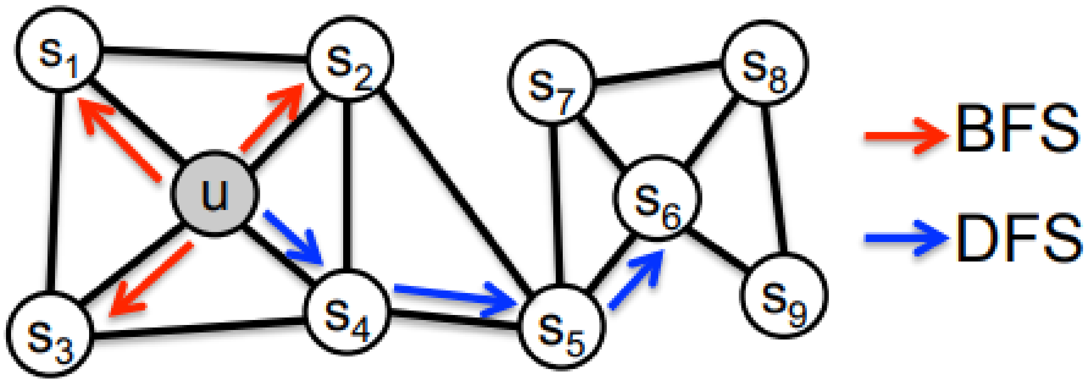


Figure 1: BFS and DFS search strategies from node $u$ ($k = 3$).

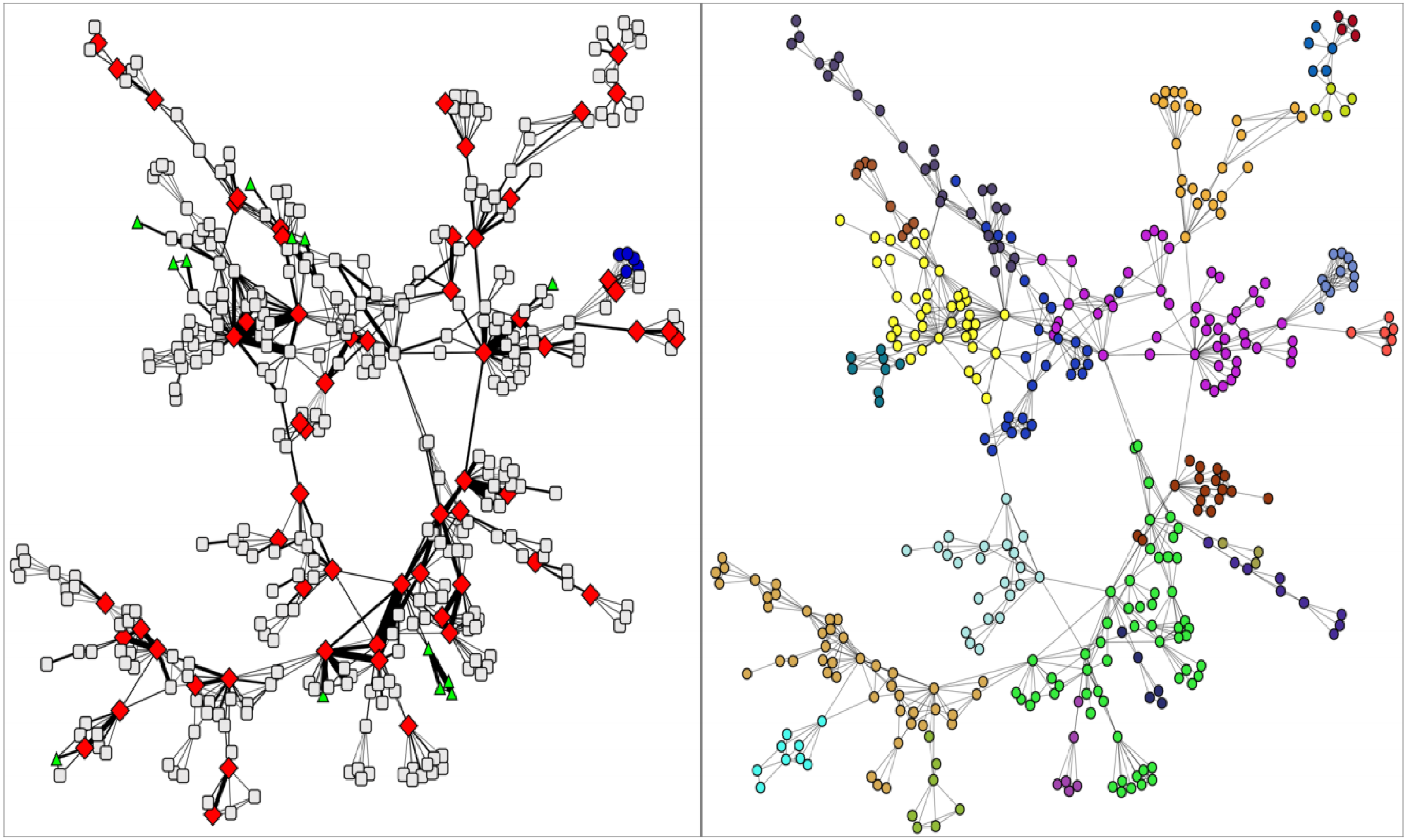
- Nodes can be organized based on community they belonged to (u and s1)
- Nodes can be organized based on the structural roles in the network (u and s6)
- Real-world network has a lot of such mixture of equivalence

## Motivation

- 1. **Homophily**: Ability to learn representations that embed nodes from the same network community closely together
- 2. **Structural equivalence**: Ability to learn representations where nodes share similar roles have similar embeddings (such as hub nodes, peripheral nodes)
- Refer to Figure 3.
- In my PhD work, I have the similar definition of role based on in-dgree/out-degree: provider, consumer, connector (page rank, probability similarity)
- Inspired by the skip-gram model, similar words tend to appear in similar word neighborhoods, they apply this idea in network

## Motivation

- node2vec
  - 2nd order random walk (sample network neighborhoods for nodes)
  - Word2vec (optimize a custom graph-based objective function using SGD)
- In addition, the authors also show how feature representation of individual nodes can be extend to pairs of nodes (i.e., edges), for link prediction



## Innovation

- They developed a flexible sampling strategy accommodate both homophily and structural equivalence
- Incorporate this sampling strategy into word2vec for feature representation learning

# Methods

## Overview of the framework

- Let G=(V,E) be a given network, map V to feature representation with d dimension. f is a matrix of size |V| by d parameters
- Ns(u) represents network neighborhood of node u with different sampling strategy S. S will be introduced in the following section

- Objective function:

$$\max_f \sum_{u \in V} \log Pr(N_S(u)|f(u)).$$

Simplify to

$$\max_f \sum_{u \in V} \Big[ -\log Z_u + \sum_{n_i \in N_S(u)} f(n_i) \cdot f(u) \Big]$$

Similar to Skim-gram loss function.

f(u) → input word

Ns(u) → context words

# Methods

## Overview of the framework

- The purpose is to maximize the log-probability of Ns(u) given the input embedding f(u)
- Softmax output is y_hat, normalized probability for each neighbor node
- The loss function is essentially a cross-entropy
- Optimize it with SGD

## Methods

### Overview of the framework

- However, skip-gram is not the innovative part of this paper
- Sliding window in word embedding cannot be applied here, a richer notion of a neighborhood is needed
- Ns(u) is not restricted to just immediate neighbors but can have vastly different structures depending on the sampling strategy S

# Methods

## Search strategy

- The problem of **sampling** neighborhoods of a source node can be viewed as a form of local **search**

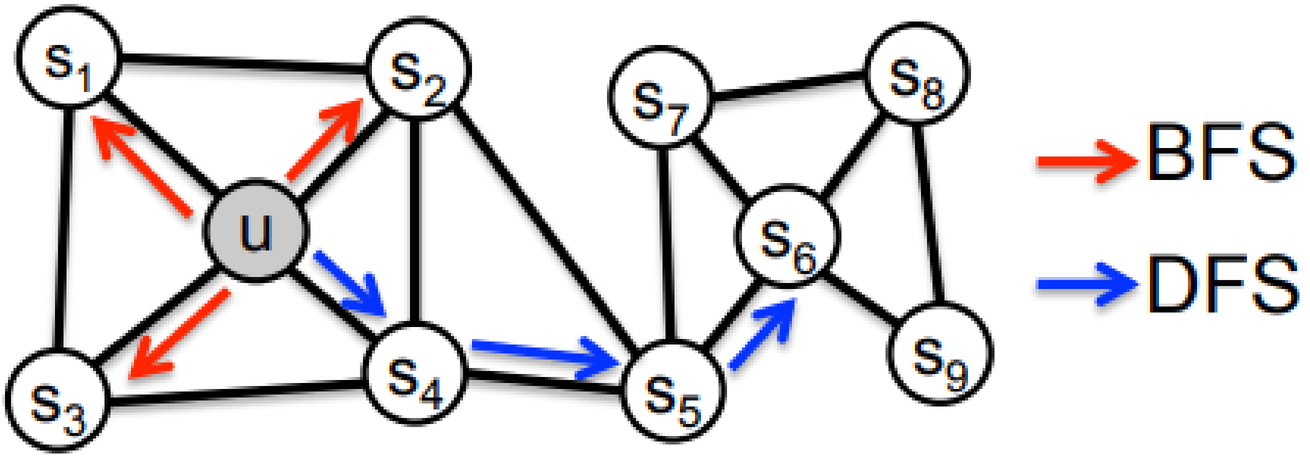


Figure 1: BFS and DFS search strategies from node $u$ ($k = 3$).

# Methods

## Search strategy

- Classic search strategies
  - Breadth-first sampling (BFS)
  - Depth-first sampling (DFS)
- BFS and DFS are two extreme scenarios in terms of the search space they explore
  - BFS, iterative, go as wide as possible, FIFO queue
  - DFS, recursive, go as deep as possible, LIFO stack
- **BFS**, suitable for **structural analysis**, microscopic view of neighborhood of every node
- **DFS**, suitable for **homophily**, macro-view of the neighborhood which is essential in inferring communities

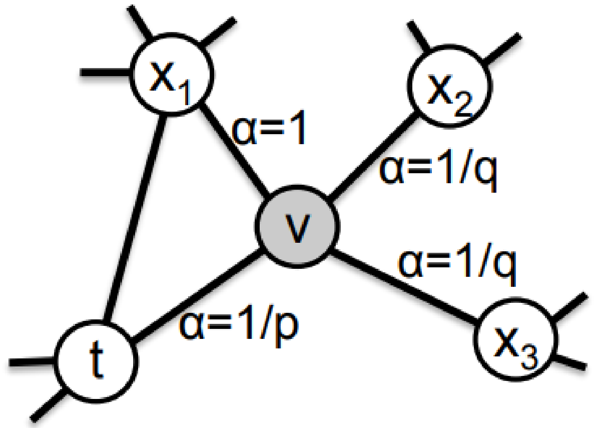
# Methods
## node2vec

- Flexible neighborhood sampling strategy to smoothly interpolate between BFS and DFS
- Biased random walks, static edge weight $w_{vx}$

$$P(c_i = x \mid c_{i-1} = v) = \begin{cases} \frac{\pi_{vx}}{Z} & \text{if } (v,x) \in E \\ 0 & \text{otherwise} \end{cases}$$

$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

$$\pi_{vx} = \alpha_{pq}(t,x) \cdot w_{vx}.$$



$$\alpha_{pq}(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$

Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from $t$ to $v$ and is now evaluating its next step out of node $v$. Edge labels indicate search biases $\alpha$.

- Because t already visited v, the algorithm will consider their shortest distance as 0, therefore, a 1/p bias will be set between t and v, the purpose is to control the probability of return
- Then the algorithm will evaluate its next step out of node v, which means, to go X1, X2 or X3?
- The algorithm will look back t's neighbor. Since t and x1 has a shortest distance as 1, so the bias between v and x1 is 1
- Similarly, since t and x2, x3 has shortest distance as 2, the bias between them is 1/q.

## Methods
### node2vec

- From the above example, we can figure out p is actually control the return step, and q actually control the step of walk to outside world
- p: return parameter. Control the likelihood of revisit. Large p ensure less revisit. Small p lead a back step
- q: in-out parameter. BFS while q>1 and DFS while q<1
- P and q have to be optimized during cross-validation

## Methods
### node2vec

- Three phases of node2vec
  - Preprocessing to compute transition probabilities
  - Random walk simulations (get the sample, similar as window size for text)
  - Skip-gram, Optimization using SGD

# Methods

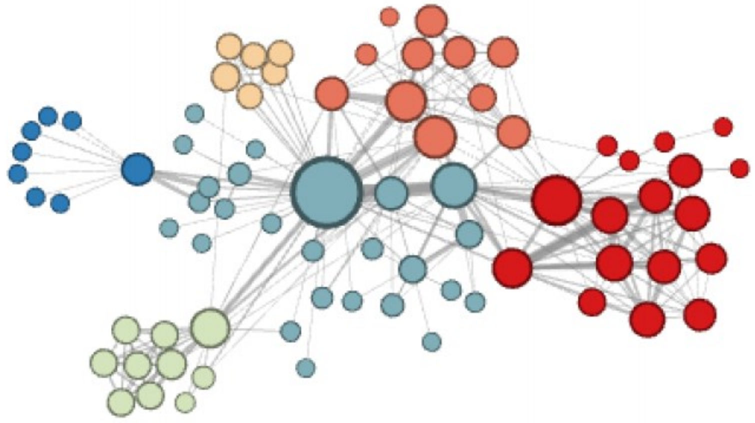
## Learning edge features

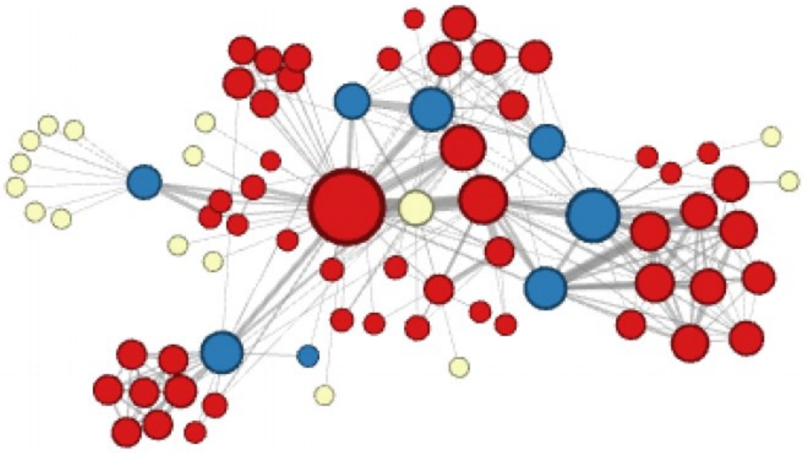
- Extend random walk to pairs of nodes using a bootstrapping approach over feature representations of the individual nodes
- For operators to generate a paired representation over two feature vector f(v) and f(u)
  - Average
  - Hadamard
  - Weighted-L1
  - Weighted-L2

# Experiments
## Case Study

- Les Miserables Network
  - 77 nodes
  - 254 edges
  - d=16

P=1, q=0.5, DFS community detection (homophily)





P=1, q=2, BFS structural detection

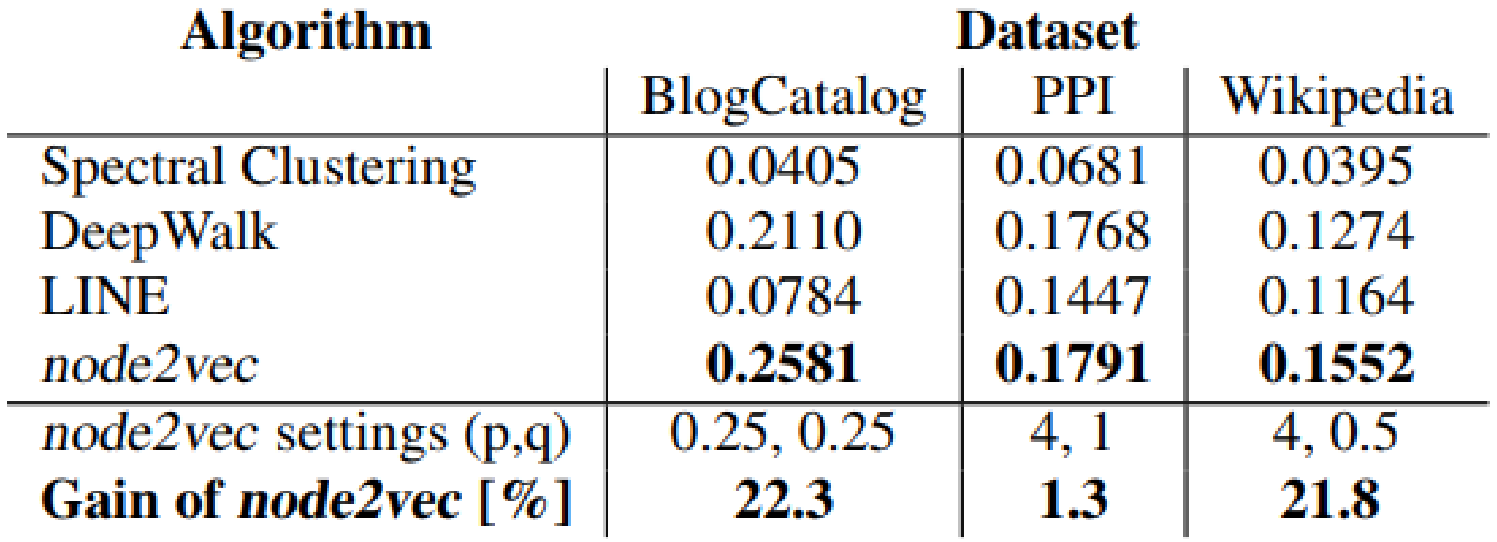
# Experiments

## Setup

- d=128
- r=10 (# of random walks)
- l=80 (fixed length)
- k=10 (# of neighborhood)
- 10-fold cross-validation on 10% labeled data
- Grid search over p and q, {0.25, 0.5, 1, 2, 4}

# Experiments
## Multi-label classification

One-vs-rest logistic regression classifier with L2 regularization

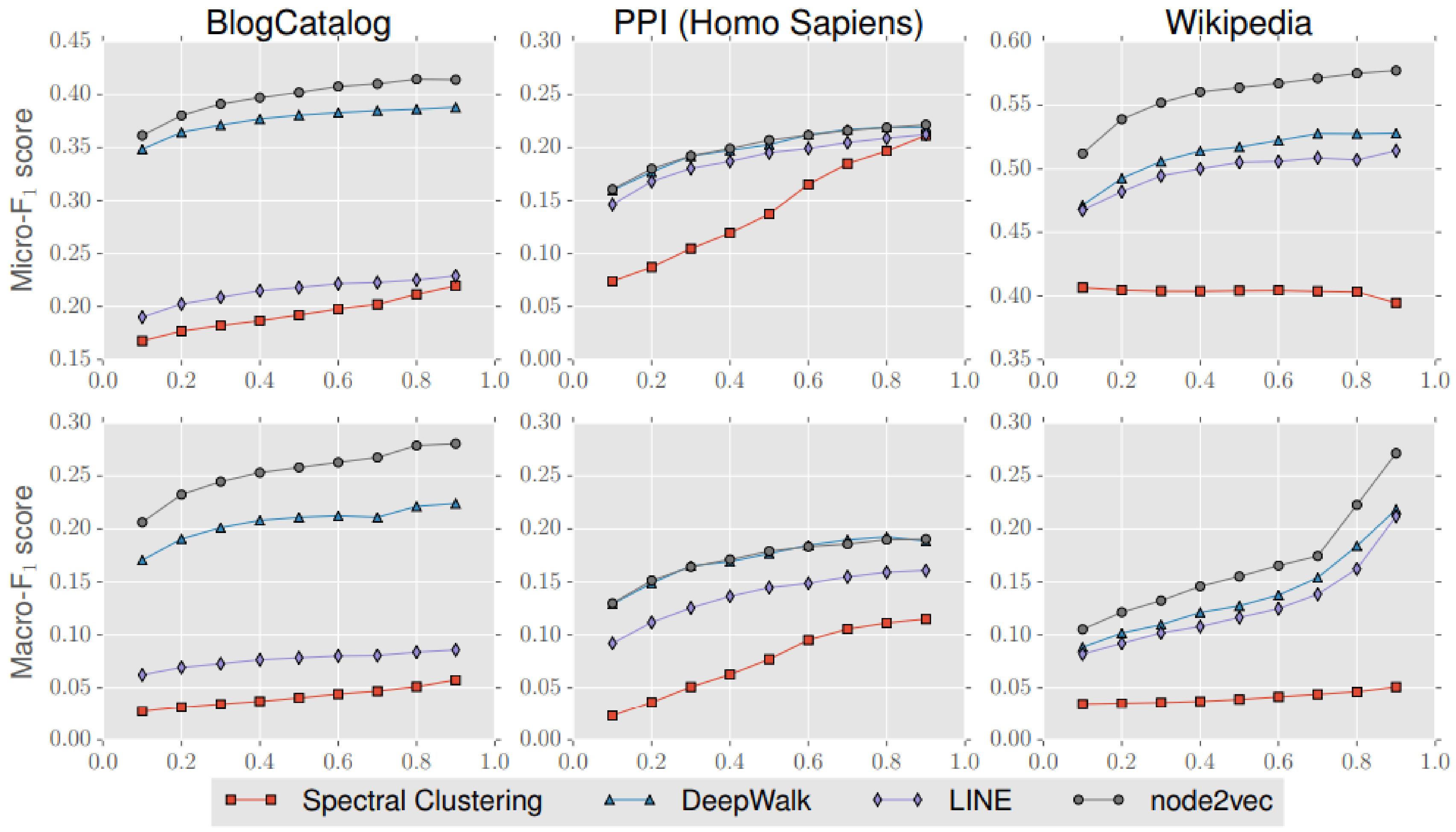


| **Algorithm** | **Dataset** | | |
|---|---|---|---|
| | BlogCatalog | PPI | Wikipedia |
| Spectral Clustering | 0.0405 | 0.0681 | 0.0395 |
| DeepWalk | 0.2110 | 0.1768 | 0.1274 |
| LINE | 0.0784 | 0.1447 | 0.1164 |
| *node2vec* | **0.2581** | **0.1791** | **0.1552** |
| *node2vec* settings (p,q) | 0.25, 0.25 | 4, 1 | 4, 0.5 |
| **Gain of *node2vec* [%]** | **22.3** | **1.3** | **21.8** |

Table 2: Macro-$F_1$ scores for multilabel classification on BlogCatalog, PPI (Homo sapiens) and Wikipedia word cooccurrence networks with 50% of the nodes labeled for training.

# Experiments
## Multi-label classification

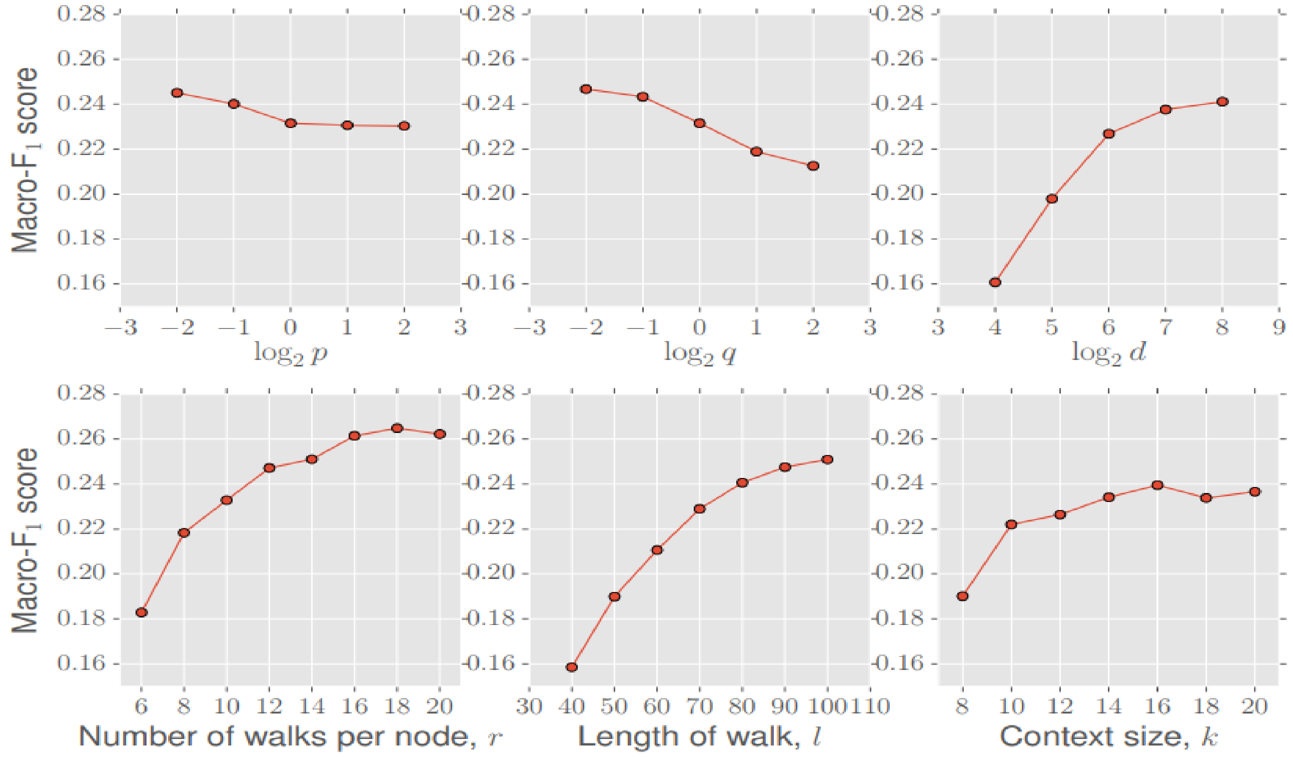
One-vs-rest logistic regression classifier with L2 regularization



# Experiments

## Parameter Sensitivity

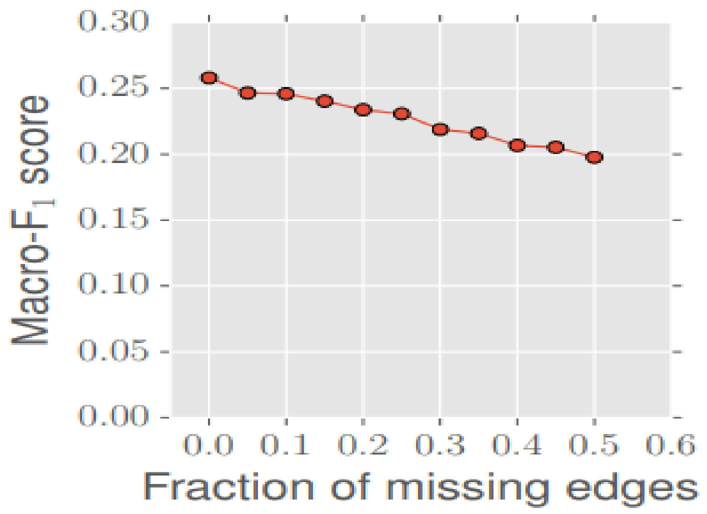
- BlogCatalog dataset
- 50-50 split between labeled and unlabeled data

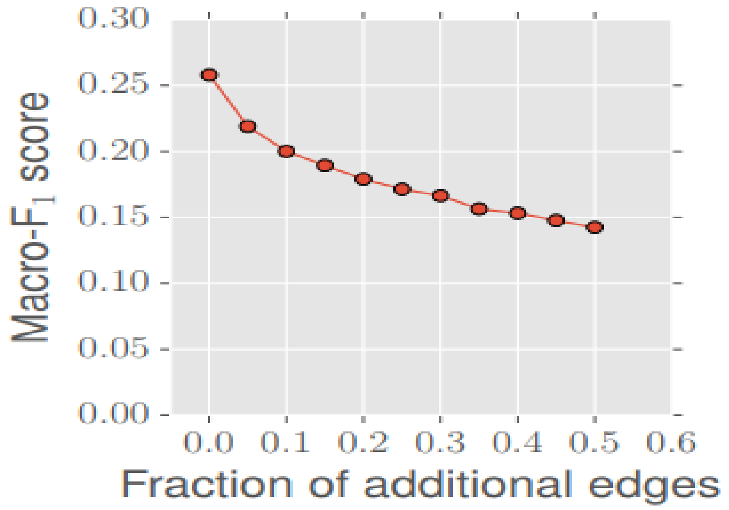


# Experiments

## Perturbation Analysis

Gradually slow down for the increment of missing edges and additional edges

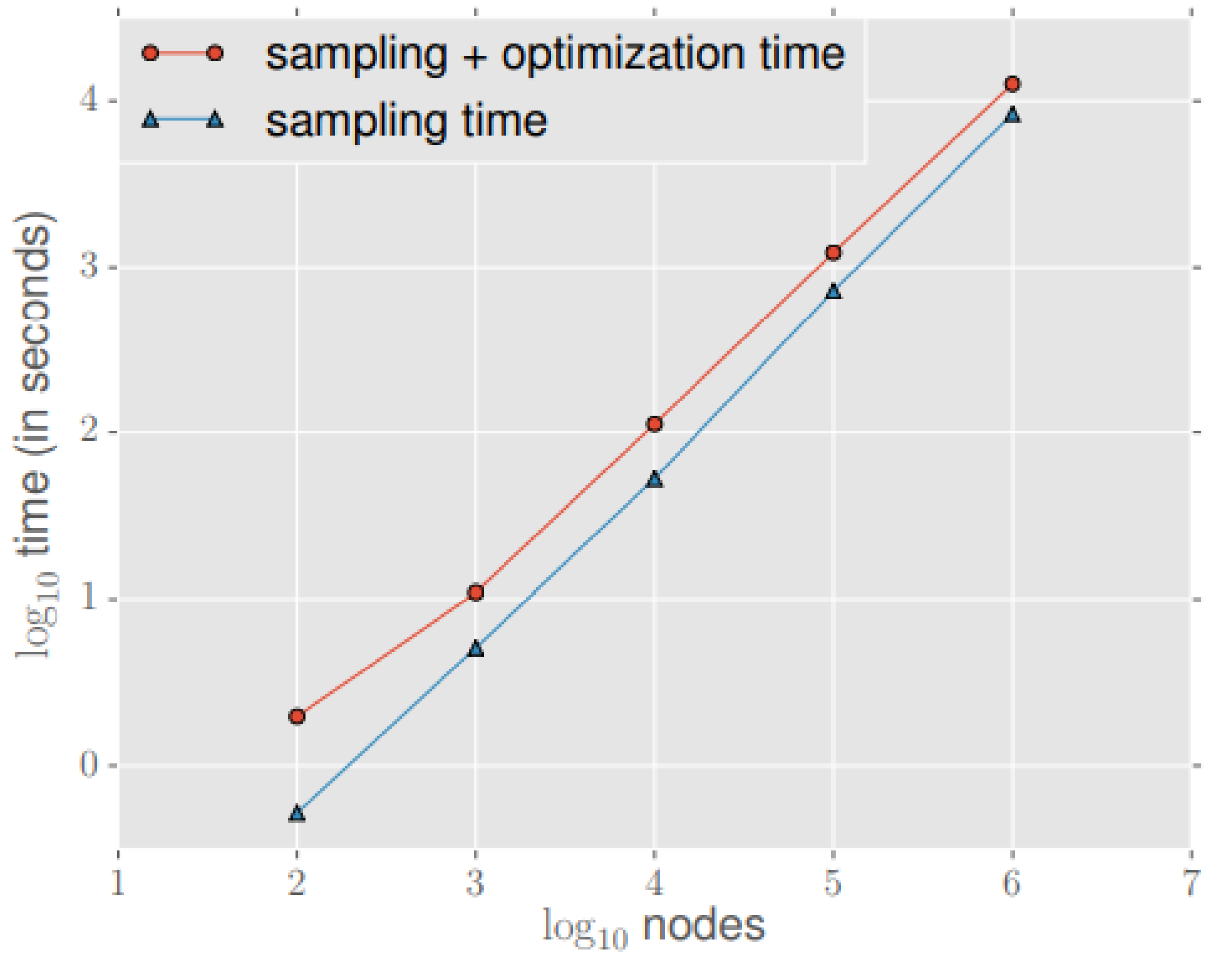




# Experiments
## Scalability

Negative sampling

Asynchronous SGD with spark



## Experiments
### Link prediction

- Network with a certain fraction of edges removed, try to see if the node2vec can predict these missing edges
- To obtain positive examples: remove 50% edges chosen randomly, and make sure the remaining nodes are still connected as a graph
- To get negative examples: randomly sample an equal number of node pairs from the network which have no edge connecting them

| **Op** | **Algorithm** | **Dataset** | | |
|---|---|---|---|---|
| | | Facebook | PPI | arXiv |
| | Common Neighbors | 0.8100 | 0.7142 | 0.8153 |
| | Jaccard's Coefficient | 0.8880 | 0.7018 | 0.8067 |
| | Adamic-Adar | 0.8289 | 0.7126 | 0.8315 |
| | Pref. Attachment | 0.7137 | 0.6670 | 0.6996 |
| (a) | Spectral Clustering | 0.5960 | 0.6588 | 0.5812 |
| | DeepWalk | 0.7238 | 0.6923 | 0.7066 |
| | LINE | 0.7029 | 0.6330 | 0.6516 |
| | *node2vec* | 0.7266 | 0.7543 | 0.7221 |
| (b) | Spectral Clustering | 0.6192 | 0.4920 | 0.5740 |
| | DeepWalk | **0.9680** | 0.7441 | 0.9340 |
| | LINE | 0.9490 | 0.7249 | 0.8902 |
| | *node2vec* | **0.9680** | **0.7719** | **0.9366** |
| (c) | Spectral Clustering | 0.7200 | 0.6356 | 0.7099 |
| | DeepWalk | 0.9574 | 0.6026 | 0.8282 |
| | LINE | 0.9483 | 0.7024 | 0.8809 |
| | *node2vec* | 0.9602 | 0.6292 | 0.8468 |
| (d) | Spectral Clustering | 0.7107 | 0.6026 | 0.6765 |
| | DeepWalk | 0.9584 | 0.6118 | 0.8305 |
| | LINE | 0.9460 | 0.7106 | 0.8862 |
| | *node2vec* | 0.9606 | 0.6236 | 0.8477 |

Table 4: Area Under Curve (AUC) scores for link prediction. Comparison with popular baselines and embedding based methods bootstapped using binary operators: (a) Average, (b) Hadamard, (c) Weighted-L1, and (d) Weighted-L2 (See Table 1 for definitions).

# Discussion

- Semantic relations between ontology entities are not incorporated into consideration, e.g., SameAs, seeAlso, hasSynonym.
- For ontology, an interesting evaluation can be made. If there exists a triple <A SameAs B>, let’s see if the node2vec can represent the similar embeddings for node A and B. From this way, we can view the gap between structural similarity versus semantic similarity (**If semantic relation can be represented by node embedding**)
- Random work only choose {0, 1, 2}. Mine is three level

## Discussion

- If more than two step, and the graph has circle, is it possible to trap in a infinite loop? For example, if t->v->x1->t, if the bias not set properly, it can never walk out of this small loop

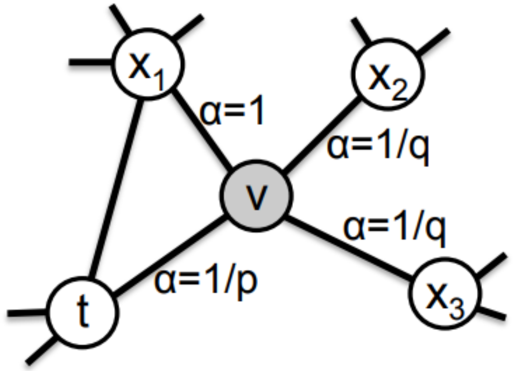
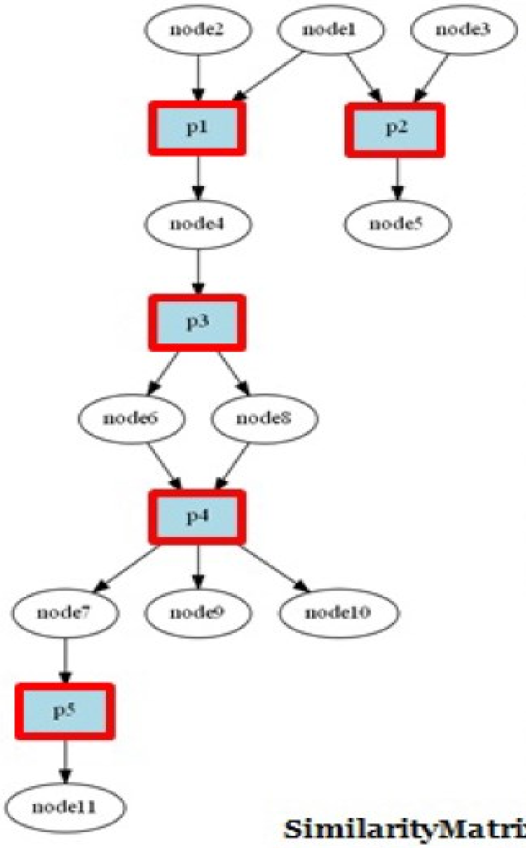


Figure 2: Illustration of the random walk procedure in *node2vec*. The walk just transitioned from $t$ to $v$ and is now evaluating its next step out of node $v$. Edge labels indicate search biases $\alpha$.

# Feichen’s work

## predicate-oriented pattern analysis



**Level1: ($P_1$, $P_2$), ($P_1$, $P_3$)**

$PS_s(P_1, P_2) = \frac{1}{3 \cdot 3} = \frac{1}{9} = 0.11$

$PS_s(P_1, P_3) = \frac{1}{3 \cdot 3} = \frac{1}{9} = 0.11$

$PS_s(P_3, P_4) = \frac{2^2}{3 \cdot 5} = \frac{4}{15} = 0.27$

$PS_s(P_4, P_5) = \frac{1}{5 \cdot 2} = \frac{1}{10} = 0.1$

**Level2: ($P_1$, $P_4$)**

$PS_c(P_1, P_4)$
$= PSs(P_1, P_3) * PSs(P_3, P_4)$
$= \frac{1}{9} * \frac{4}{15} = \frac{4}{135} = 0.03$

$PS_c(P_3, P_5)$
$= PSs(P_3, P_4) * PSs(P_4, P_5)$
$= \frac{4}{15} * \frac{1}{10} = \frac{4}{150} = 0.026$

**Level3: ($P_1$, $P_5$)**

$PS_c(P_1, P_5) =$
$Max(PSc(P_1, P_4) * PSs(P_4, P_5),$
$PS_s(P_1, P_3) * PSc(P_3, P_5)) =$
$Max(\frac{4}{135} * \frac{1}{10}, \frac{1}{9} * \frac{4}{150}) = 0.0029$

$PS_c(P_2, P_3) = 0$
$PS_c(P_2, P_4) = 0$
$PS_c(P_2, P_5)) = 0$

**SimilarityMatrix(SM)**

| | $p_1$ | $p_2$ | $p_3$ | $p_4$ | $p_5$ |
|---|---|---|---|---|---|
| $p_1$ | 1 | 0.11 | 0.11 | 0.03 | 0.0029 |
| $p_2$ | 0.11 | 1 | 0 | 0 | 0 |
| $p_3$ | 0.11 | 0 | 1 | 0.27 | 0.026 |
| $p_4$ | 0.03 | 0 | 0.27 | 1 | 0.2 |
| $p_5$ | 0.0029 | 0 | 0.026 | 0.2 | 1 |

# Feichen's work

## MedKDD

Leverage predicate pattern to detect link between nodes in heterogeneous domains

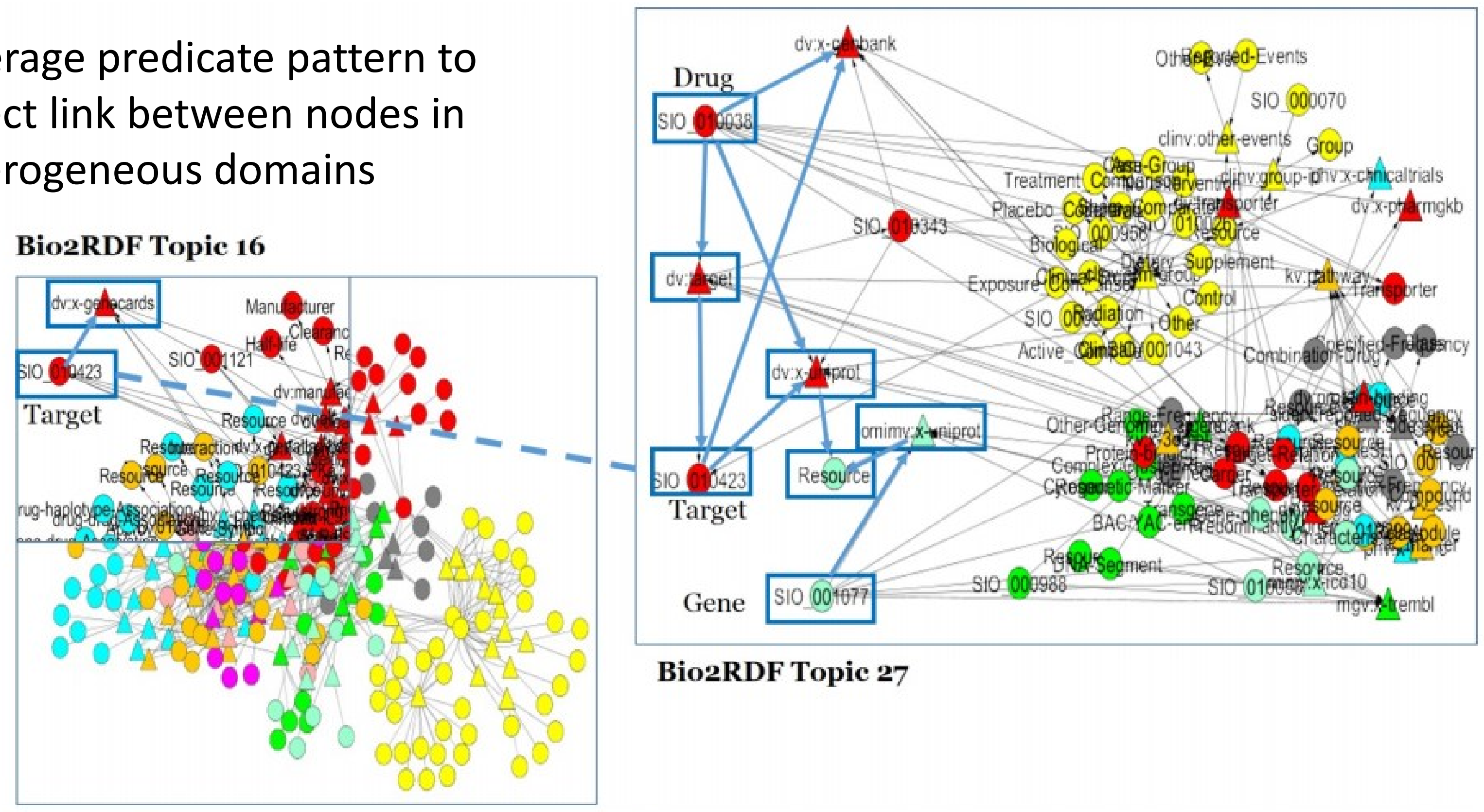
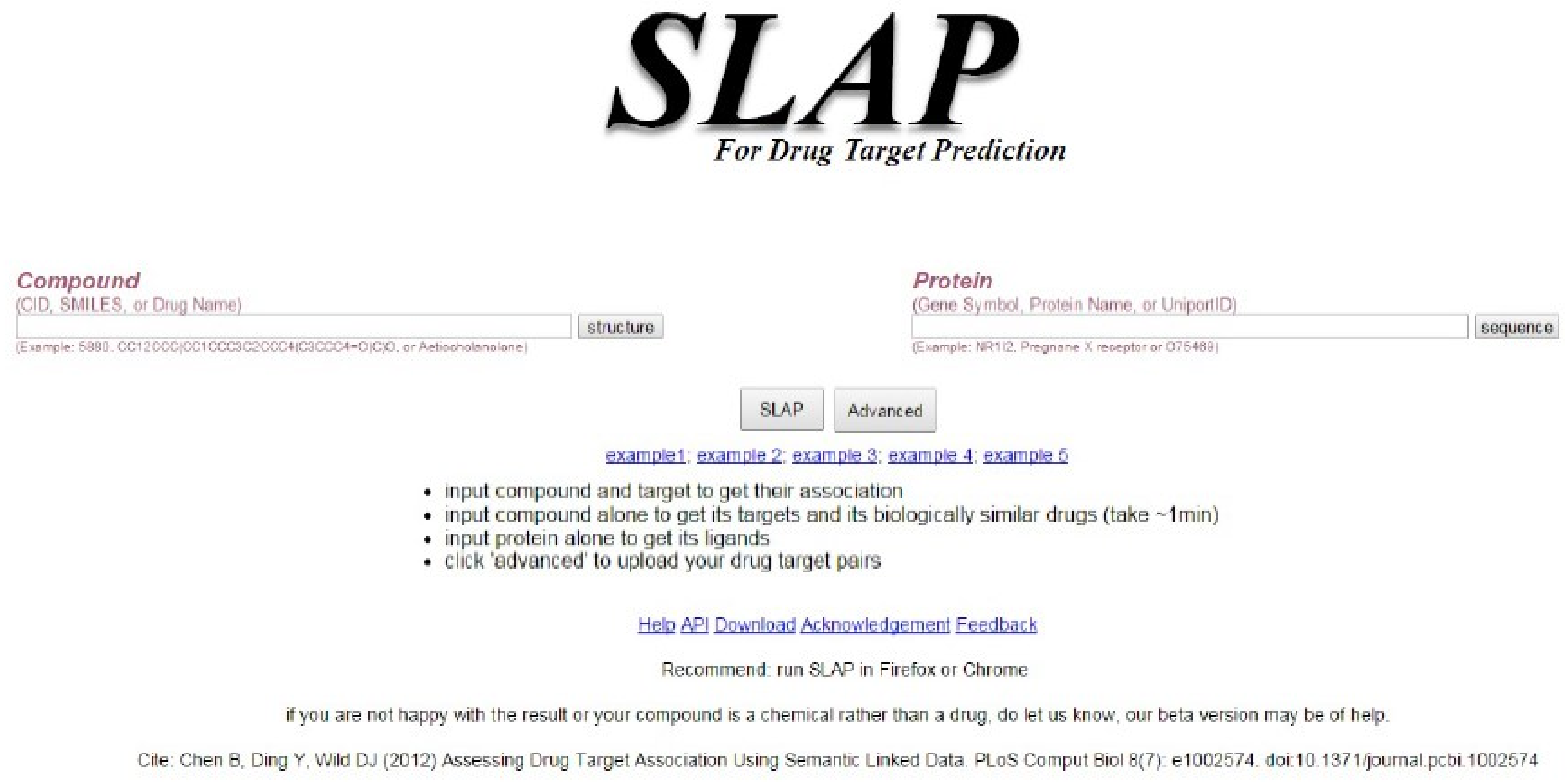


**Fig 10. The Cross Domain Query Graphs for Topic 16 and Topic 27.** In the cross domain query graph, the circle represents a concept and the triangle represents a predicate. A color is assigned to each domain as follows: DrugBank: Red; HGNC: Pink; MGI: Green; PharmGKB: Cyan; ClinicalTrials: Yellow; OMIM: Sky Blue; SIDER: Gray; KEGG: Orange; CTD: Magenta. The cross domain query graph is composed with the paths between Topic 27 and Topic 16 such as i) {*dv:Drug (SIO_010038)* → *dv:target* → *dv:Target(SIO_010423)* → *dv:x-genecards* → *dv:Gene(SIO_001121)*}; ii){*DrugBank*:⟨*dv:Drug (SIO_010038)* → *dv:target* → *dv:Target(SIO_010423)* → *dv:x-uniprot*⟩ ⇒ *OMIM:omimv:Uniprot*}; iii) {*omimv:Resource* → *omimv:x-uniprot* → *omimv:Uniprot*}.

# Bin’s work
## Semantic Link Association Prediction (SLAP)



***Compound***
(CID, SMILES, or Drug Name)
(Example: 5880, CC12CCC(CC1CCC3C2CCC4(C3CCC4=O)C)O, or Aetiocholanolone)

***Protein***
(Gene Symbol, Protein Name, or UniportID)
(Example: NR1I2, Pregnane X receptor or O75469)

example1; example 2; example 3; example 4; example 5

- input compound and target to get their association
- input compound alone to get its targets and its biologically similar drugs (take ~1min)
- input protein alone to get its ligands
- click 'advanced' to upload your drug target pairs

Help API Download Acknowledgement Feedback

Recommend: run SLAP in Firefox or Chrome

if you are not happy with the result or your compound is a chemical rather than a drug, do let us know, our beta version may be of help.

Cite: Chen B, Ding Y, Wild DJ (2012) Assessing Drug Target Association Using Semantic Linked Data. PLoS Comput Biol 8(7): e1002574. doi:10.1371/journal.pcbi.1002574