Outline

1. Introduction
   - Graph clustering
2. Online Node Classification with Random Spanning Trees
   - Motivations
   - Problem
   - Compressing graph information with random spanning trees
3. Correlation Clustering with Adaptive Similarity Queries
   - Correlation clustering preliminaries
   - Active Correlation Clustering algorithms

Learning on graphs

I

Domains

Social networks (e.g., networks of customers and advertisers), hyperlinked Web pages, co-author networks, biological networks, communication networks, transportation networks, energy grids, neuroscience, physics, finance, and economics.

Main tasks

Graph clustering, node classification, link prediction, link classification, exploration/mining of graphs.

Main issue: Networks are now too large

- Many standard machine learning techniques are unfeasible.
- Algorithms working in time sublinear in the graph size may be needed.

Graph Clustering – Main research results

Graph clustering publications

- **Random Spanning Trees and the Prediction of Weighted Graphs**
  [*JMLR 2013 – joint work with N. Cesa-Bianchi, C. Gentile, G. Zappella*].
  Sequentially predict the binary labels on the nodes of an arbitrary weighted graph.

- **Correlation Clustering with Adaptive Similarity Queries**
  [*NeurIPS 2019 – joint work with M. Bressan, N. Cesa-Bianchi, A. Paudice*].
  Use as few pairwise similarity queries as possible to partition an input set of nodes into clusters.

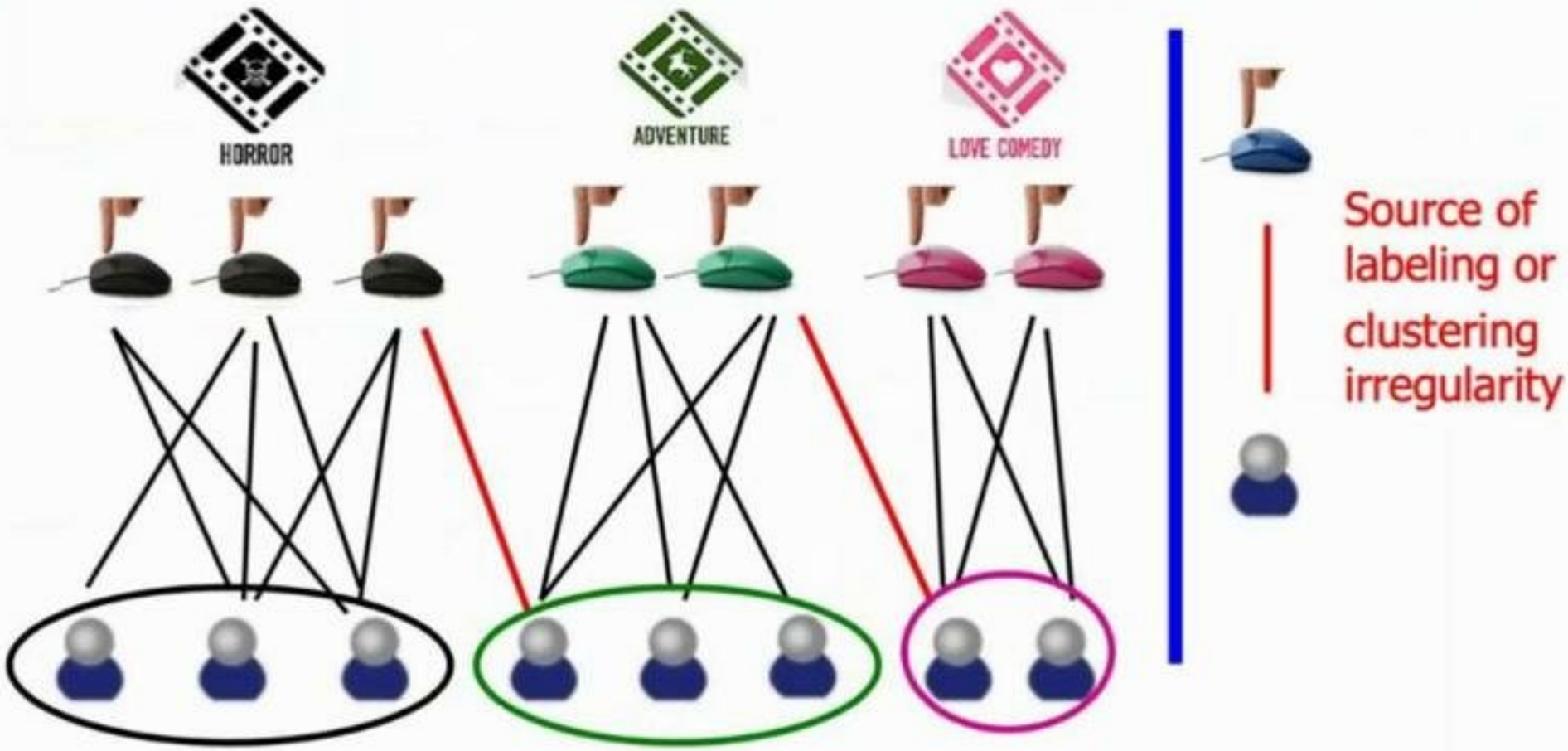
Outline

1. Introduction
   - Graph clustering
2. Online Node Classification with Random Spanning Trees
   - Motivations
   - Problem
   - Compressing graph information with random spanning trees
3. Correlation Clustering with Adaptive Similarity Queries
   - Correlation clustering preliminaries
   - Active Correlation Clustering algorithms

Online node classification – Motivation examples

Targeted advertising problem

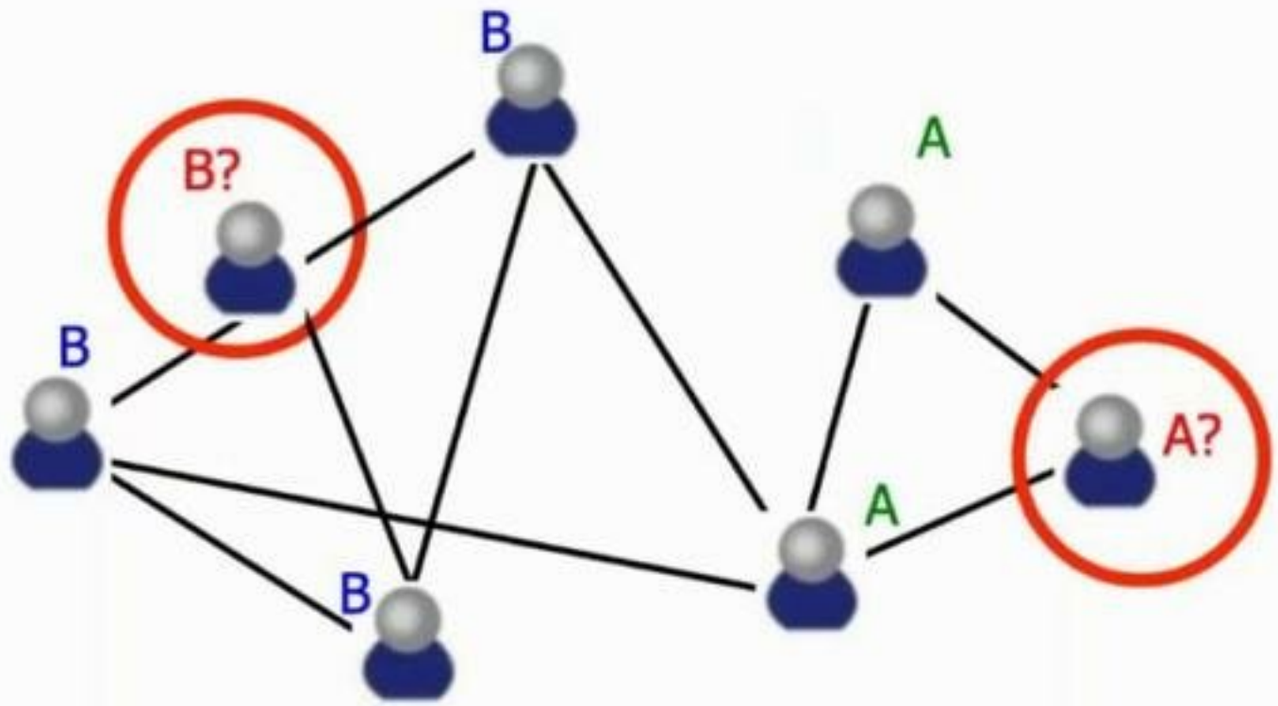
Input: **Arbitrary** (weighted) graph $G(V, E)$.
Targeting each user with the product she is most likely to buy.



Online node classification – Motivation examples

Targeted advertising problem

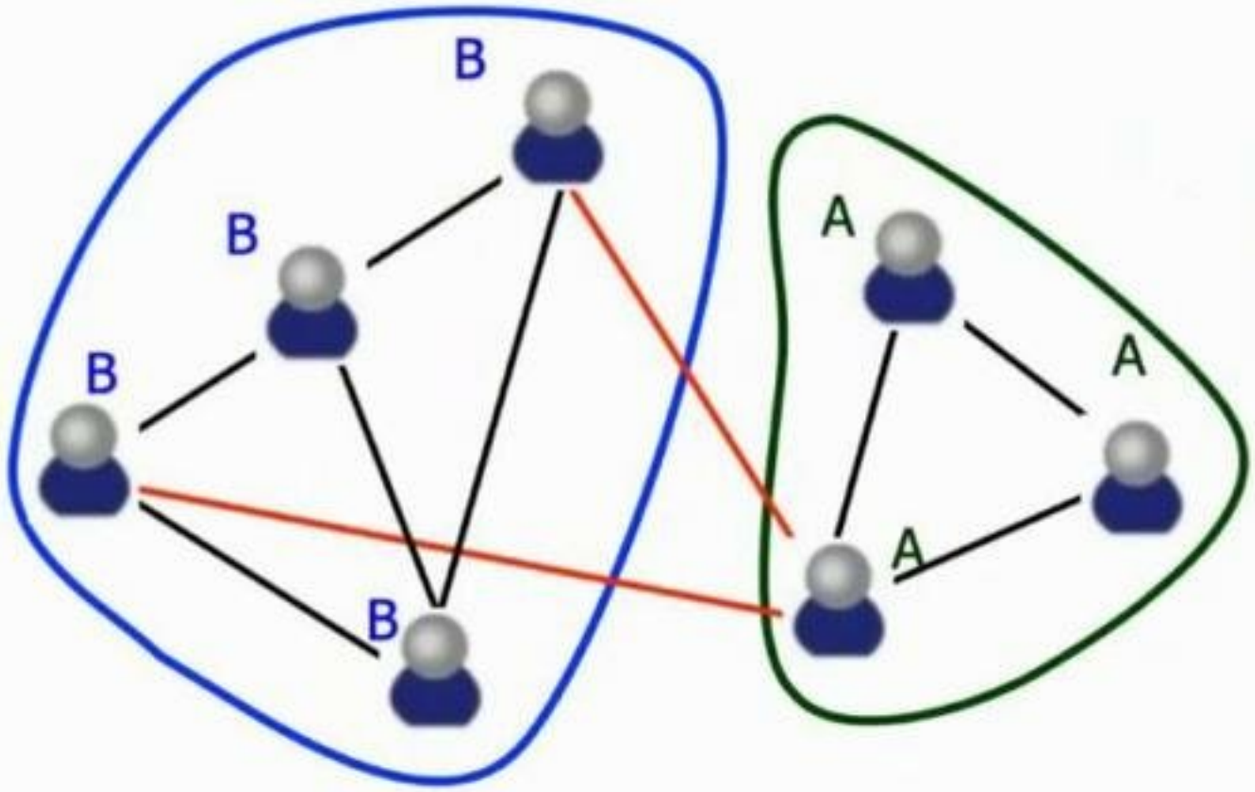
Input: **arbitrary** (weighted) graph $G(V, E)$.
Targeting each member of a social network with the product she is most likely to buy (product A vs. product B).



Online node classification – Motivation examples

Targeted advertising problem

Input: **arbitrary** (weighted) graph $G(V, E)$.
Targeting each member of a social network with the product she is most likely to buy (product A vs. product B).



Source of labeling or clustering irregularity

Online node classification

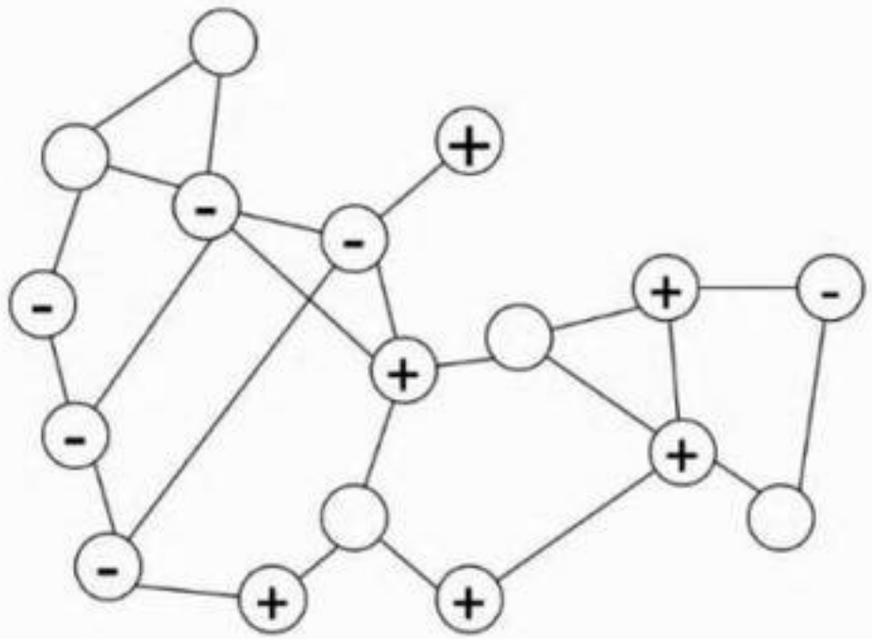
I

Online node classification problem

Input: **arbitrary** (weighted) graph $G(V, E)$.

Nodes are issued one by one in an **arbitrary** order $i_1, i_2, \ldots, i_n$.

At each time step t, learner predicts the label of i_t, and then the true label $y(i_t)$ is revealed.



Complexity measure: *Effective resistance weighted* cutsize $\Phi_R(G)$ of $G(V, E)$.

Performance measure: Number of mistakes.

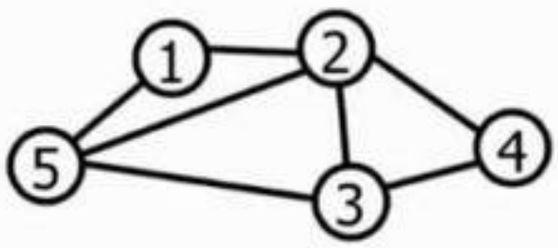
Weighted Tree Algorithm – Graph linearization overview

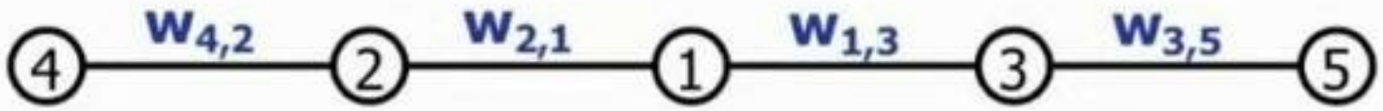
Sub-linear time compression for sub-linear time nearly optimal prediction

Given **any** (unlabeled) weighted graph $G(V, E)$, find

1. a vertex permutation, and
2. a set of $n - 1$ weights

to predict with a **NN method** on the generated **line-graph** with **strong** theoretical performance guarantees. *Random graph "fingerprint".*





- **Scalability**: Very fast (**sub-linear** time for most inputs).
- **Accuracy analysis**: **Nearly optimal** mistake bound.
- **Very good practical performance**.
- **Parallelizable**.

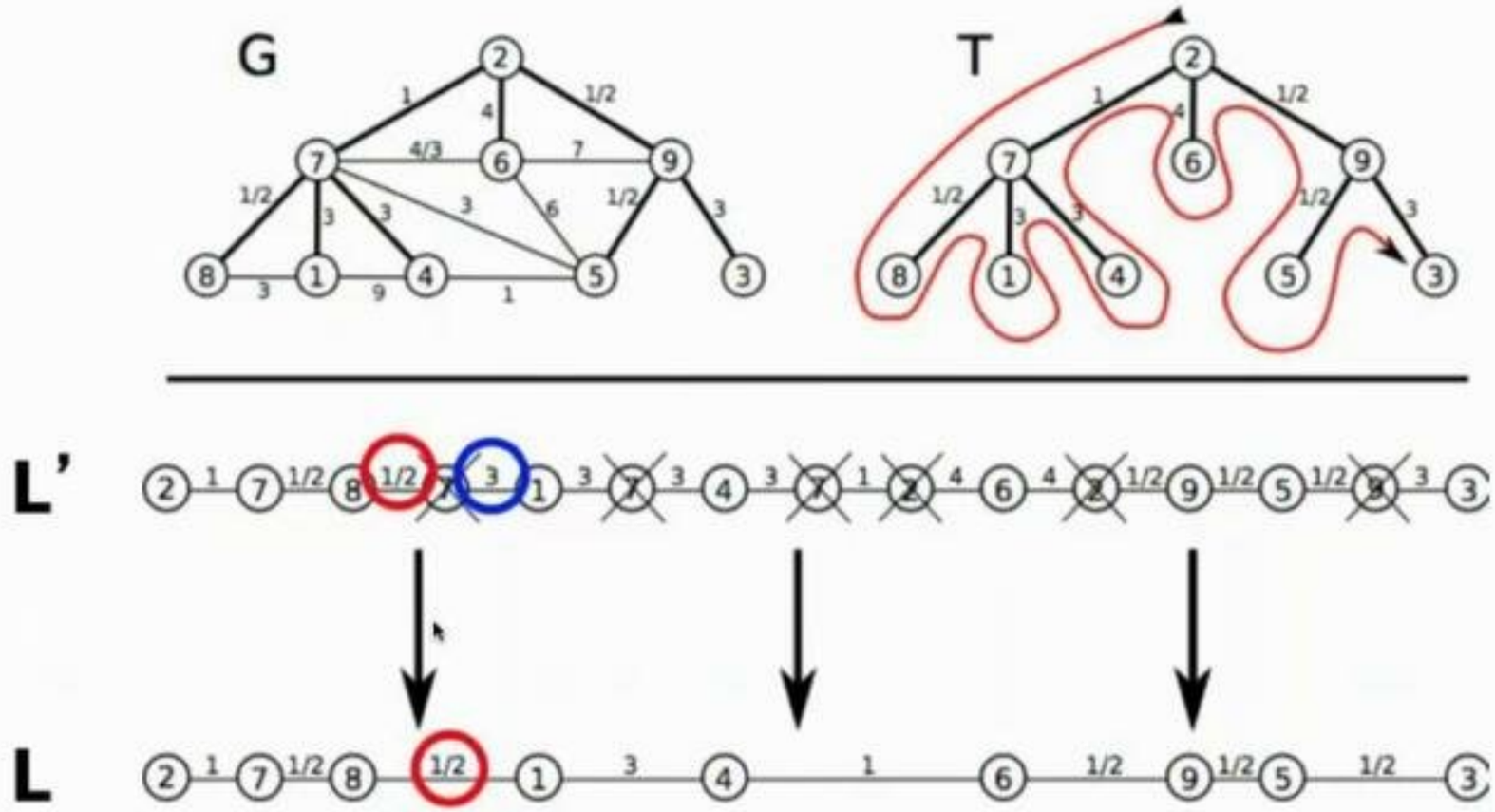
Weighted Tree Algorithm – WTA

I

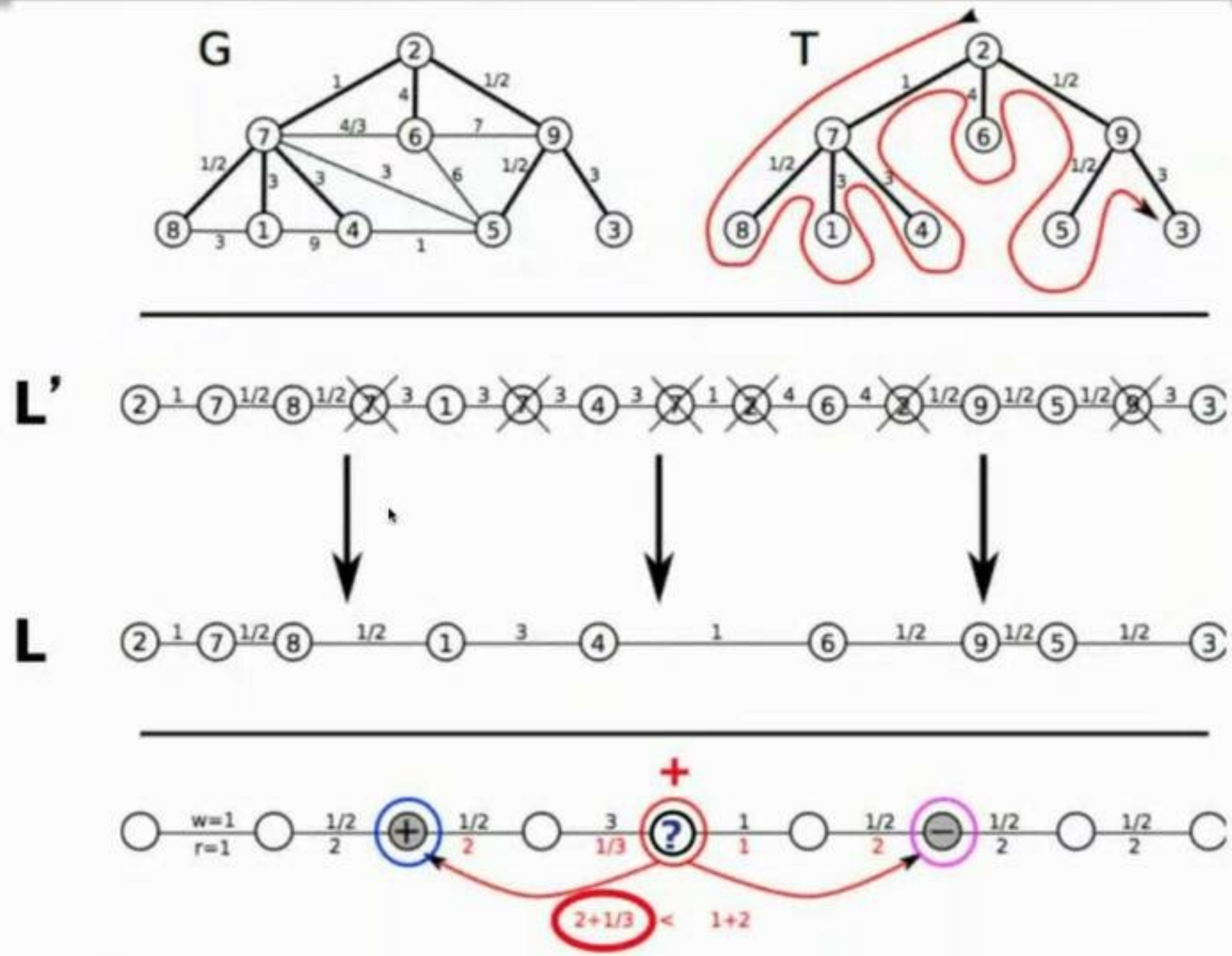
The three steps of WTA – [JMLR 2013; ICML 2010; NIPS/NeurIPS 2009 (workshop)]

1. Draw a **Random Spanning Tree** T.
2. Turn T into **line-graph** L.
3. Run a fast **Nearest Neighbor** method operating on L with **resistance distance metric**.

Weighted Tree Algorithm – WTA



Weighted Tree Algorithm – WTA



In a line graph $r_{p,q}$ is equal to the sum of $1/w_{i,j}$ for all edges (i,j) on the path connecting p and q.

WTA Analysis

Lower bound: Theorem

Any algorithm can be forced to make $\Omega(\Phi_R(G))$ mistakes in expectation on any input graph $G(V, E)$.

WTA Analysis

Lower bound: Theorem

Any algorithm can be forced to make $\Omega(\Phi_R(G))$ mistakes in expectation on any input graph $G(V, E)$.

WTA Upper bound: Theorem

On any input graph where the ratio of any pair of edge weights is polynomially bounded in n, the expected number of mistakes m_G of WTA satisfies

$$\mathbb{E}[m_G] = \Phi_R(G)\log(n) \ .$$

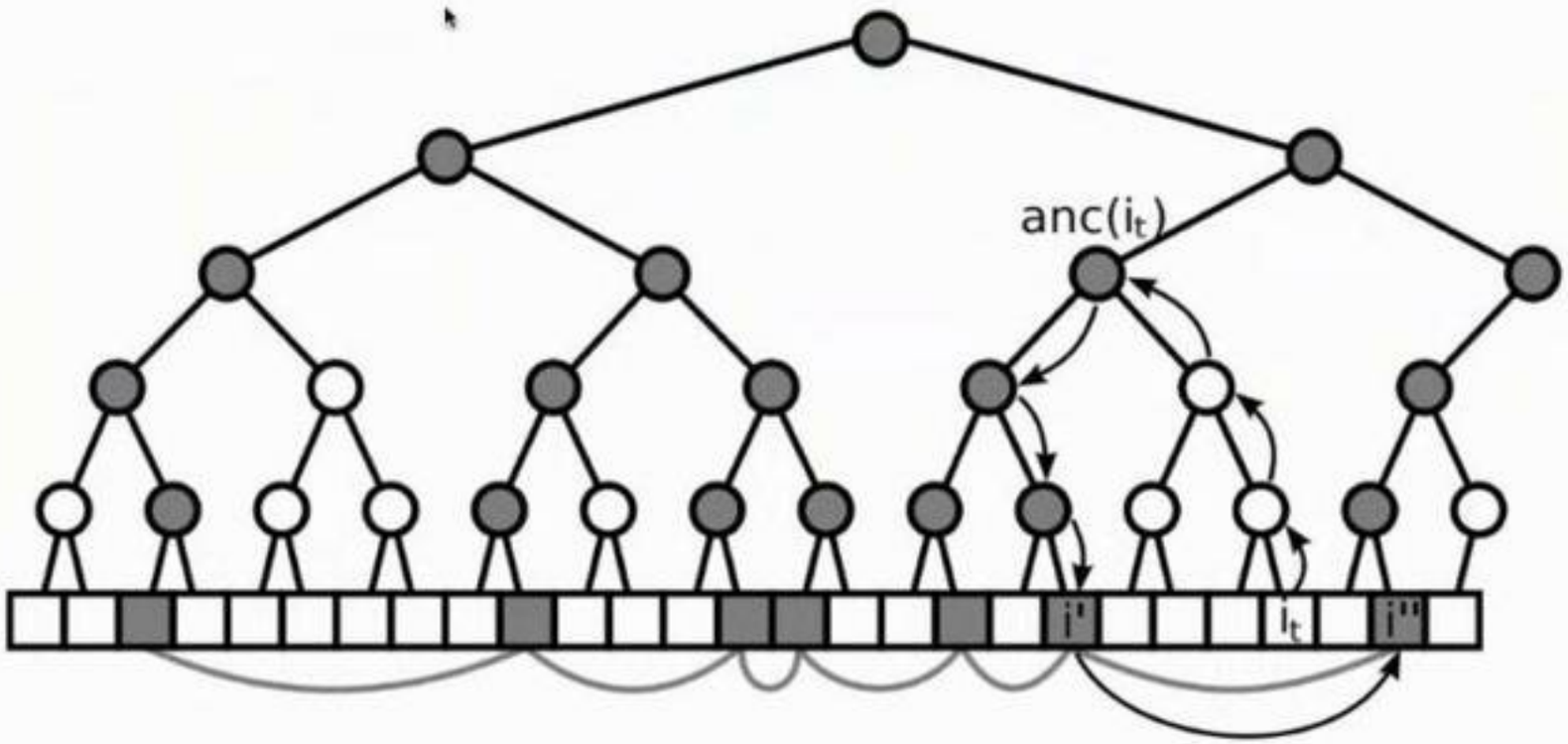
This result holds for much weaker constraints on the edge weights.
Robustness can be increased by generating multiple line-graphs. **Parallelization**.

Online WTA computational complexity

Naive implementation: Total time $n \log(n)$.

Implementation for speeding up the computation: Build a binary tree T' on the line graph L. The leaves of T' (nodes in L) whose labels have been required in the past are connected via a doubly linked list.

Amortized time per prediction: $\mathcal{O}(1)$
Worst case time per prediction: $\mathcal{O}(\log(n))$.
Space: At most linear in the size of $G(V, E)$.



WTA – Experiments

We compare WTA with fast algorithms for weighted graphs

- Perceptron algorithm with graph laplacian kernel (GPA).
- Label propagation (LABPROP) – batch transductive learning method.
- Online/Weighted majority vote (OMV / WMV).

WTA – Experiments

We compare WTA with fast algorithms for weighted graphs

- Perceptron algorithm with graph laplacian kernel (GPA).
- Label propagation (LABPROP) – batch transductive learning method.
- Online/Weighted majority vote (OMV / WMV).

We combined WTA and GPA with spanning trees drawn in different ways:

- Uniformly generated random spanning tree (RST).
- Depth first spanning trees (DFST).
- Minimum spanning tree – minimizing the sum of the resistors (MST).
- Shortest path spanning tree (SPST).

WTA – Experiments

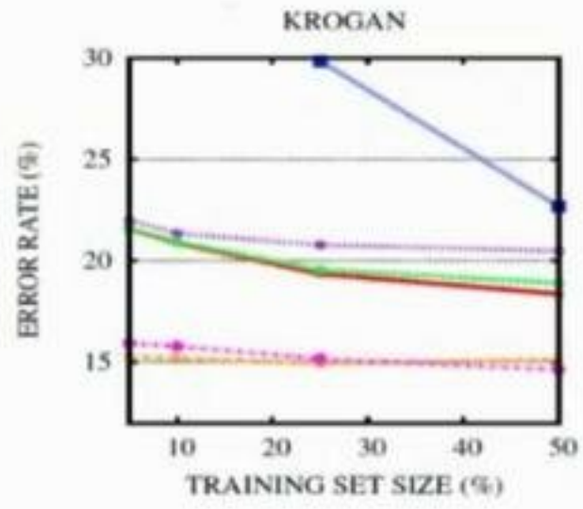
Datasets

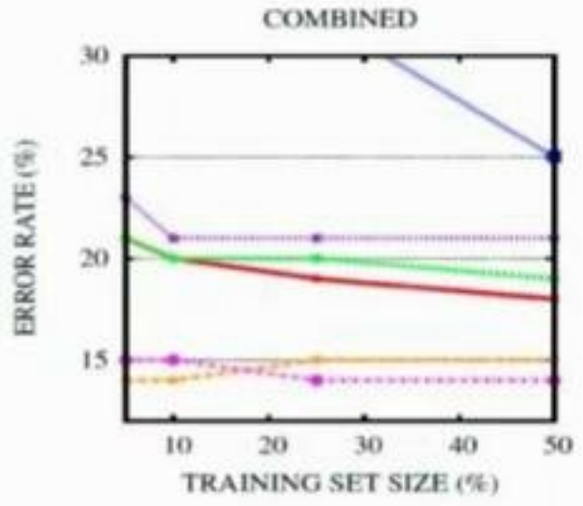
- **Text categorization**: RCV1-100 (100-NN – first 10,000 documents).
- **OCR**: USPS-10 and USPS-100 (9298 nodes; 10-NN and 100-NN).
- **Spam detection**: Webspam (110,900 nodes and 1,836,136 edges).
- **Bioinformatics**: Two protein-protein interaction datasets, KROGAN and COMBINED (each graph has about 2000-3000 nodes and 6000 edges).

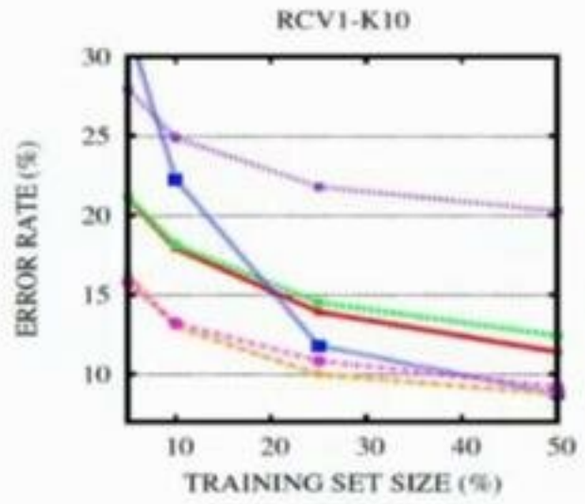
Experiment protocol (batch)

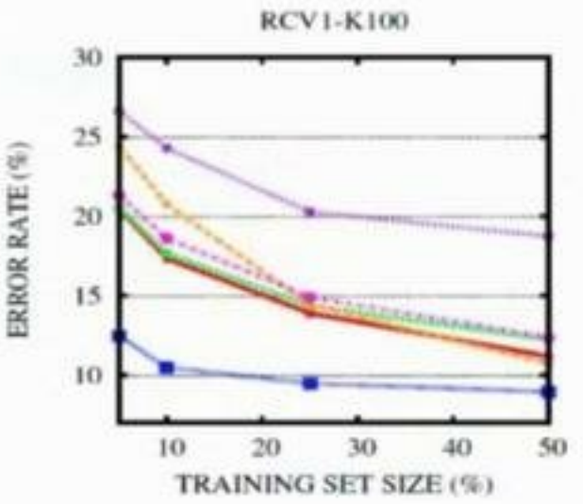
- One VS Rest scheme – binarization of the multiclass problems.
- Training set size: 5%, 10%, 25% and 50%.

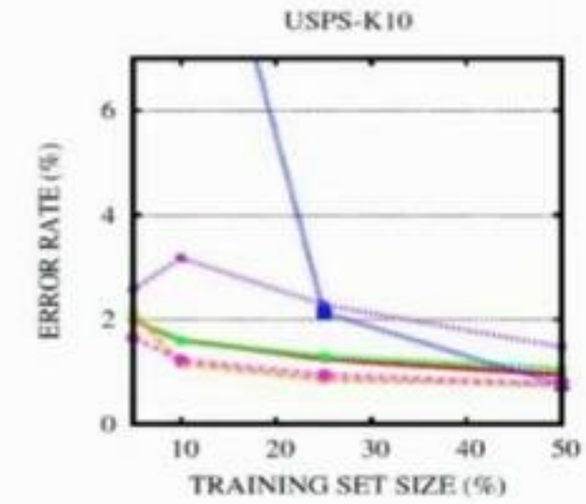
WTA – Experiments

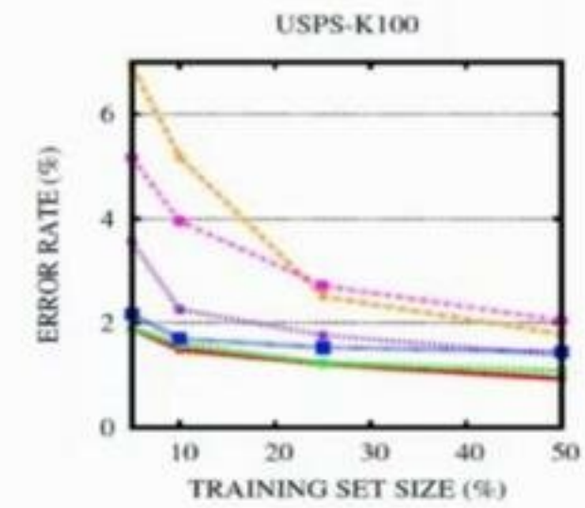






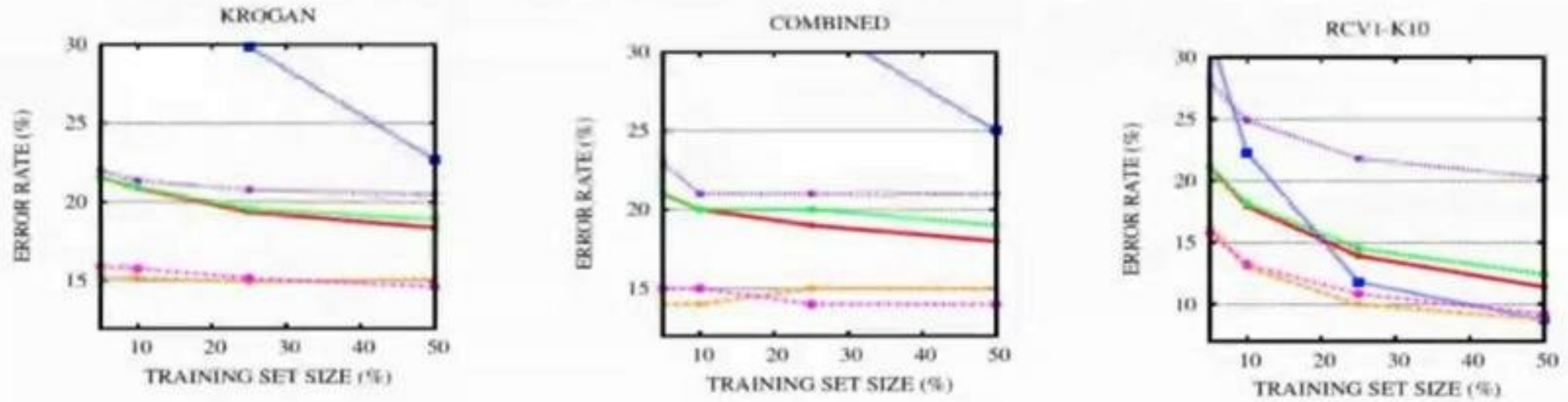






WTA+MST
NWWTA+MST
GPA+MST
OMV
LABPROP
17WTA+RST = **Committee of 17 RST**

WTA – Experiments



Outline

1. Introduction
   - Graph clustering
2. Online Node Classification with Random Spanning Trees
   - Motivations
   - Problem
   - Compressing graph information with random spanning trees
3. Correlation Clustering with Adaptive Similarity Queries
   - Correlation clustering preliminaries
   - Active Correlation Clustering algorithms

Correlation Clustering – Problem and Motivations

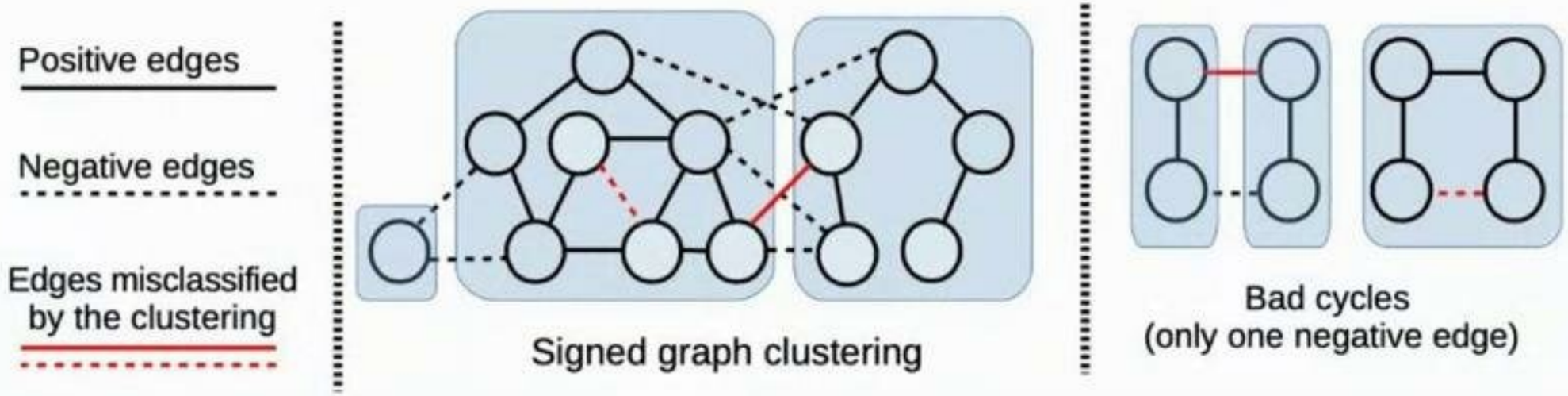
Correlation Clustering problem

- Given a set $V = [n]$ of elements, a function $\sigma : V^2 \to \{-1, +1\}$ establishes whether any two distinct elements of V are similar or not.
- **Clustering error**: a pair of elements having similarity -1 and belonging to the same cluster, or having similarity $+1$ and belonging to different clusters.
- **Objective**: Cluster the points in V **minimizing the number of errors**.

$$\Delta_C = \sum_{C(u)=C(v)} \mathbb{I}\big(\sigma(u,v) = -1\big) + \sum_{C(u)\neq C(v)} \mathbb{I}\big(\sigma(u,v) = +1\big) \ .$$

- **NP-hard** even when σ is fully known beforehand, and even APX-hard.
- There are no *a priori* restrictions on the number of clusters to be used.
- **Applications**: Entity resolution, image analysis, social media analysis, data integration and biology.

Correlation Clustering example for a *non-complete* input signed graph



CC – Minimize the number of disagreements

- Bansal, Blum, Chawla (2004): $\mathcal{O}(1)$ approximation.
- Charikar, Guruswami, Wirth (2005): 4 approximation.
- Ailon, Charikar, Newman (2005): 3 approximation.
- Chawla, Makarychev, Schramm, Yaroslavtsev (2015): 2.06 approximation.

Correlation Clustering – Related Problems

Aggregating Inconsistent Information

- **Rank Aggregation:** Find a permutation π minimizing the sum of distances $\sum_{i=1}^{k} d(\pi, \pi_i)$ where $d(\pi, \rho)$ is the number of ordered pairs (i, j) such that $i <_\pi j$ but $j <_\rho i$ (the Kemeny distance).
- **Consensus Clustering:** Find one clustering C that minimizes $\sum_{i=1}^{k} d(\mathcal{C}, \mathcal{C}_i)$, where the distance $d(\mathcal{C}, \mathcal{D})$ between two clusterings is the number of unordered pairs $i, j \in V$ that are clustered together by one and separated by the other.
- **Feedback Arc Set Problem**: Find a linear order of a given tournament minimizing the number of backward edges.

One Solution

These three problems and CC can be solved **all** by applying **the same** strategy obtaining **constant factor approximations** (*Ailon et al. J. ACM, 2008*).

Active Correlation Clustering – Problems and Motivations

Active (Query-based) Correlation Clustering for complete graphs

Adaptively select and **query** the signs of a sequence of pairs of nodes to solve CC given a query budget Q.

Motivations: Similarity information is costly to obtain (require a complex computation, and possibly interaction with human experts).

Active Correlation Clustering – Lower bound

General Lower Bound

Any (possibly randomized and adaptive) algorithm asking Q queries is forced to output a clustering with (expected) cost of

$$OPT + \Omega\left(\frac{n^3}{Q}\right).$$

OPT denotes the cost of an optimal solution to CC.

Correlation Clustering – KwikCluster

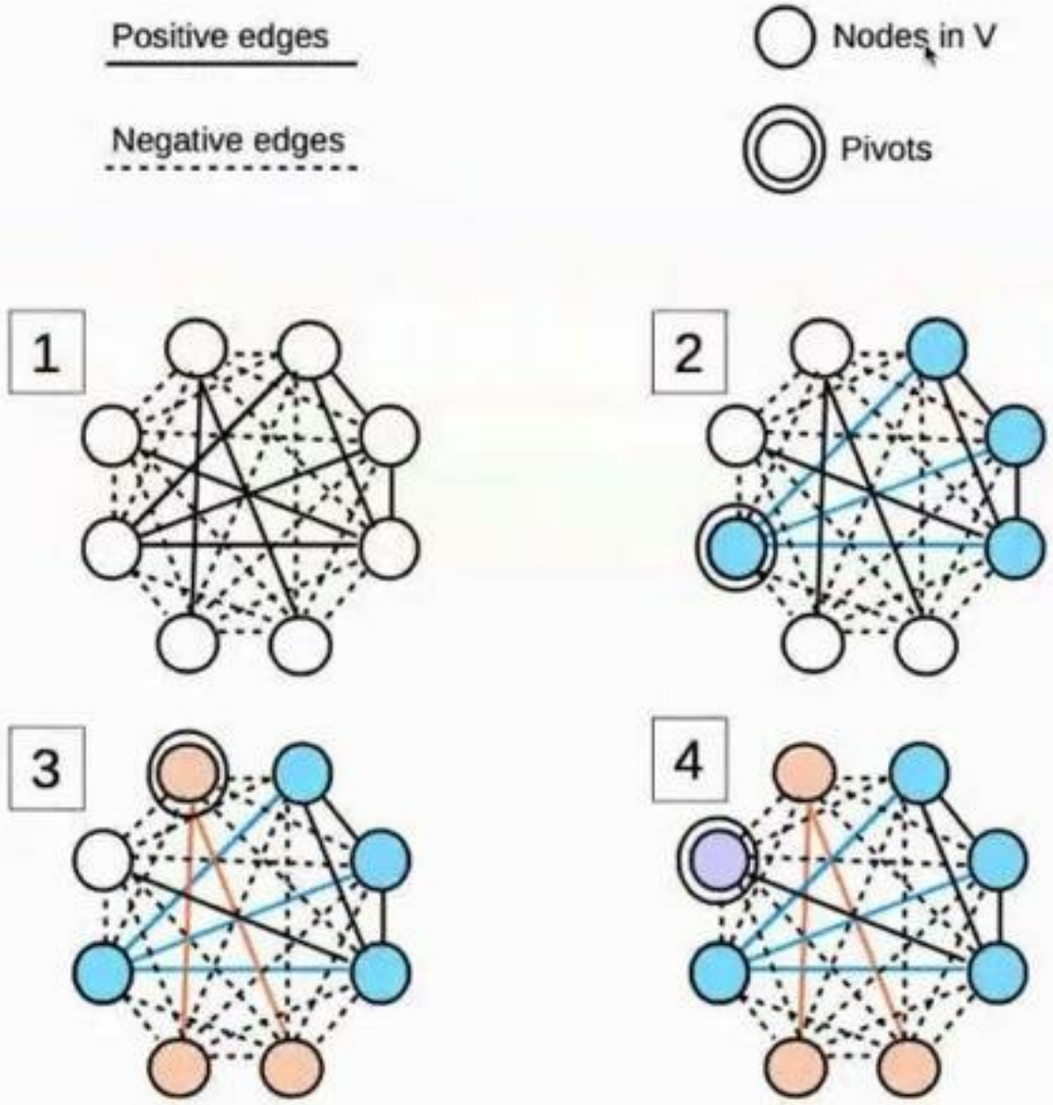
KwikCluster description [Ailon et al., J. ACM 2008]

- Select u.a.r. (uniformly at random) a node $\pi \in V$.
- Create a new cluster $\mathcal{C}_\pi$ containing π and all nodes $u \in V$ such that $\sigma(\pi, u) = 1$.
- Remove the nodes of $\mathcal{C}_\pi$ from V.

  If V becomes empty output the clustering.

Performance:

For *any* input sign function σ, we have $\mathbb{E}\Delta_{C_{KC}} \leq 3OPT$.



Active Correlation Clustering – ACC algorithm

ACC (Active Correlation Clustering)

- Set $f(n) \leq n$ such that $nf(n) \leq Q$.
- Run for (at most) $f(n)$ rounds:
  - At each round r select u.a.r. a pivot π_r from the residual vertex set V_r, **choose u.a.r.** $f(n_r)$ **elements in** $V_r \setminus \{\pi_r\}$, and ask σ for their similarity with the pivot.
  - **If no positive** similarities are observed, **declare** π_r **singleton**, otherwise ask for the remaining similarities and form the cluster with π_r and its neighbors.
- **After** $f(n)$ **rounds**, declare any remaining node in $V_{f(n)}$ singleton.

Active Correlation Clustering – ACC algorithm

ACC (Active Correlation Clustering)

- Set $f(n) \leq n$ such that $nf(n) \leq Q$.
- Run for (at most) $f(n)$ rounds:
  - At each round r select u.a.r. a pivot π_r from the residual vertex set V_r, **choose u.a.r.** $f(n_r)$ **elements in** $V_r \setminus \{\pi_r\}$, and ask σ for their similarity with the pivot.
  - **If no positive** similarities are observed, **declare** π_r **singleton**, otherwise ask for the remaining similarities and form the cluster with π_r and its neighbors.
- **After** $f(n)$ **rounds**, declare any remaining node in $V_{f(n)}$ singleton.

ACC Performance

For **any** input sign function σ and query budget Q, we have

$$\mathrm{E}\Delta_{C_{ACC}} \leq 3OPT + \mathcal{O}(n^3/Q)$$

Instance-based Active Correlation Clustering

The number of queries can be further reduced by testing the size of the residual graph in each round.

Active Correlation Clustering with Early Stop Strategy (ACCESS)

At each round, test the number of edges of the residual graph. If a small number of edges is left, stop and declare the remaining nodes singleton, otherwise perform a round of ACC.

Instance-based Active Correlation Clustering

The number of queries can be further reduced by testing the size of the residual graph in each round.

Active Correlation Clustering with Early Stop Strategy (ACCESS)

At each round, test the number of edges of the residual graph. If a small number of edges is left, stop and declare the remaining nodes singleton, otherwise perform a round of ACC.

ACCESS Performance

ACCESS has the same guarantees of ACC on the expected cost and the expected of queries. However, on some graph ACCESS performs **much less queries**.

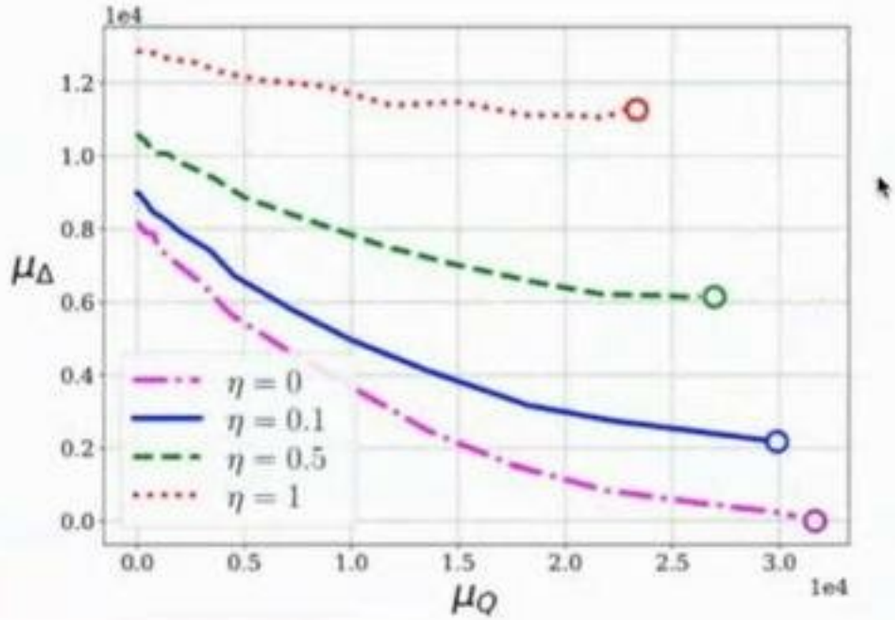
ACCESS vs. ACC

Let $f(n) = \sqrt{n}$. If G contains $n^{1/3}$ cliques of $n^{2/3}$ nodes each, then ACC performs $O\left(n^{3/2}\right)$ queries, while ACCESS performs only $O\left(n^{4/3}\ln(n)\right)$ queries.

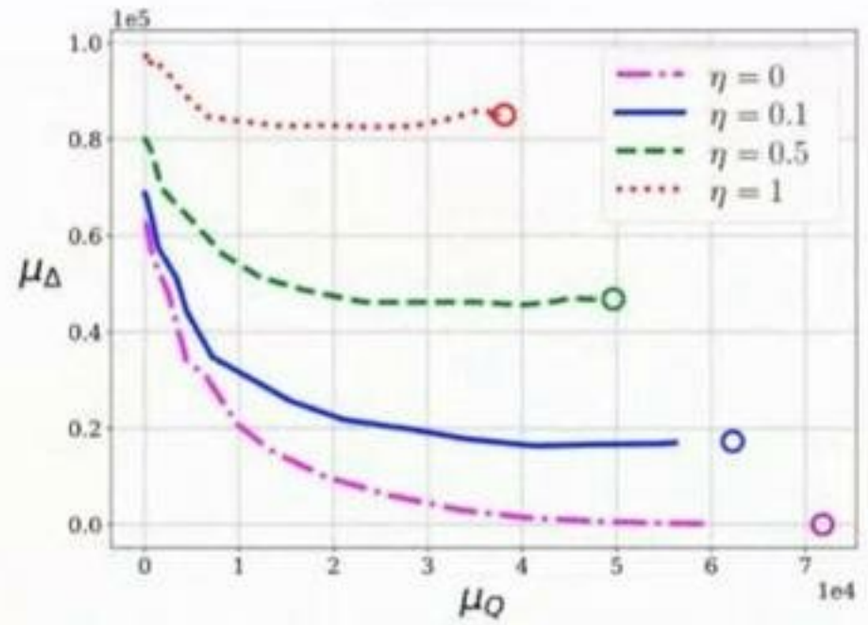
ACC algorithm – Preliminary experiments

Experimental setting

- Every dataset provides a ground-truth partitioning of nodes with $OPT = 0$ (real-world dataset "Cora" with $\approx 2K$ nodes and synthetic dataset "Skew" with $\approx 1K$ nodes).
- We perturbed the dataset by flipping the label of each edge independently with probability $p = \eta \left|E^+\right| / \binom{n}{2}$, where $\left|E^+\right| = \#\text{positive edges}$.
- μ_Q is the *average* number of queries, and μ_Δ is the *average* clustering cost .



(a) skew.



(b) cora.

Conclusions and Ongoing research

- Motivated by real-world applications, we have studied the problem of **clustering within an online/active learning setting**.
- We described **fast and efficient** online and active learning algorithms, which provably achieve a nearly **optimal trade-off** number of mistakes and an **optimal trade-off** between the number of errors and queries.
- These methods lend themselves to **time and space efficient implementations.**

Conclusions and Ongoing research

- Motivated by real-world applications, we have studied the problem of **clustering within an online/active learning setting**.
- We described **fast and efficient** online and active learning algorithms, which provably achieve a nearly **optimal trade-off** number of mistakes and an **optimal trade-off** between the number of errors and queries.
- These methods lend themselves to **time and space efficient** implementations.

Future research directions

- Exploit the graph compression using Random Spanning Trees for Graph Neural Networks.
- Study the Active Correlation Clustering problem when the similarity function is not binary (similarity real weights).
- Extend the active learning results to the related problems (Rank Aggregation, Consensus Clustering, Minimum Feedback Arc Set Problem).