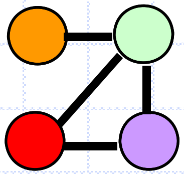
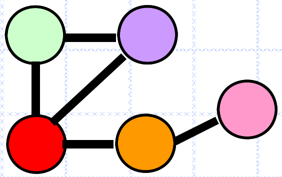
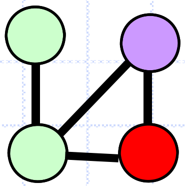
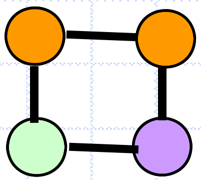
Entire Regularization Paths for Graph Data

Max Planck Institute for Biological Cybernetics

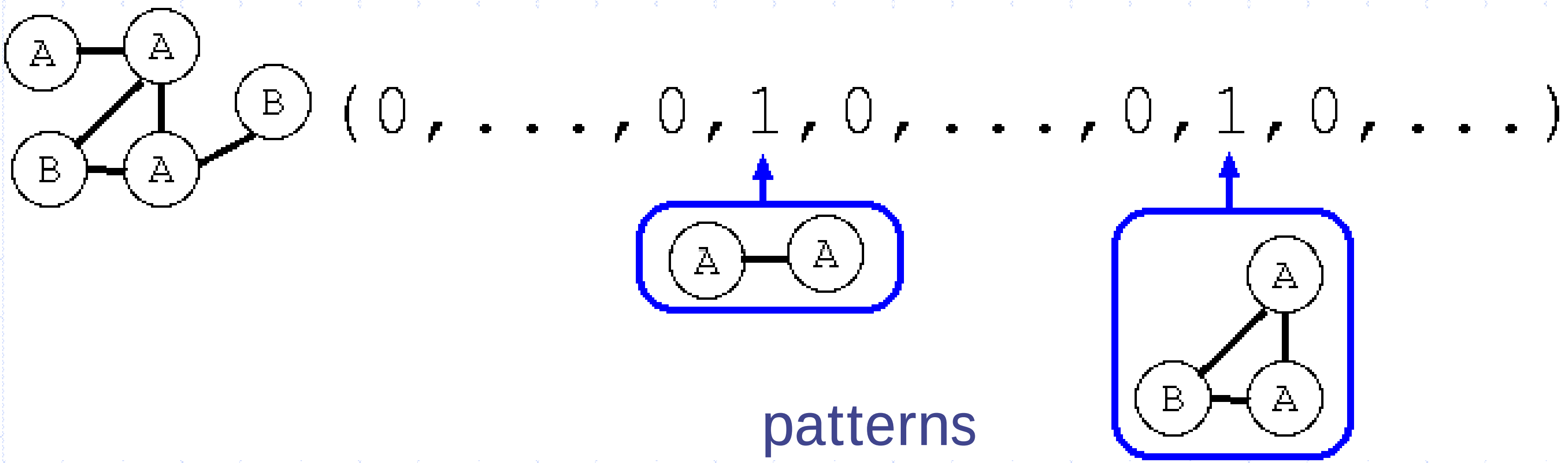
Koji Tsuda

Graph Regression

Training	Test
( ,-0.2)	( , ?)
( , 0.7)	
( ,-0.5)	

Substructure Representation

- 0/1 vector of *pattern* indicators
- Huge dimensionality!
- Need feature selection



Overview

- Entire regularization paths
  - LARS-LASSO (Efron et al., 2004), L1SVM
  - Forward selection of features
  - Trace the solution trajectory of L1-regularized learning
- Path following algorithm for graph data
  - Feature search -> pattern search
  - Branch-and-bound algorithm
  - DFS code tree, New Bound

Path Following Algorithms

- LASSO regression

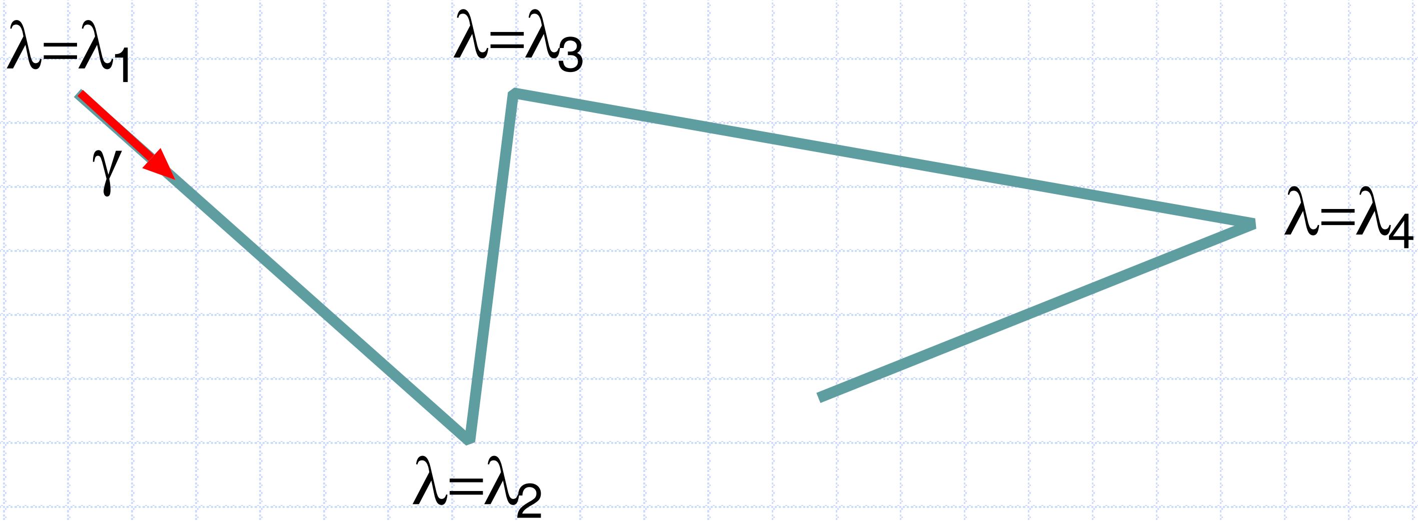
$$\boldsymbol{\beta}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, L(\boldsymbol{y}, X\boldsymbol{\beta}) + \lambda\|\boldsymbol{\beta}\|_1.$$

- Follow the complete trajectory of $\boldsymbol{\beta}(\lambda)$
  - λ : Infinity to Zero

- Active feature set $\mathcal{A}$
  - Features corresponding to nonzero weights

Piecewise Linear Path

- At a turning point,
  - A new feature included into the active set, or
  - An existing feature excluded from the active set



Practical Merit of Path Following

- Cross validation by grid search
  - Has to solve QP many times
  - Especially time-consuming for graph data

- Path following does not include QP
- Determine the CV-optimal regularization parameter in the finest precision

Pseudo code of path following

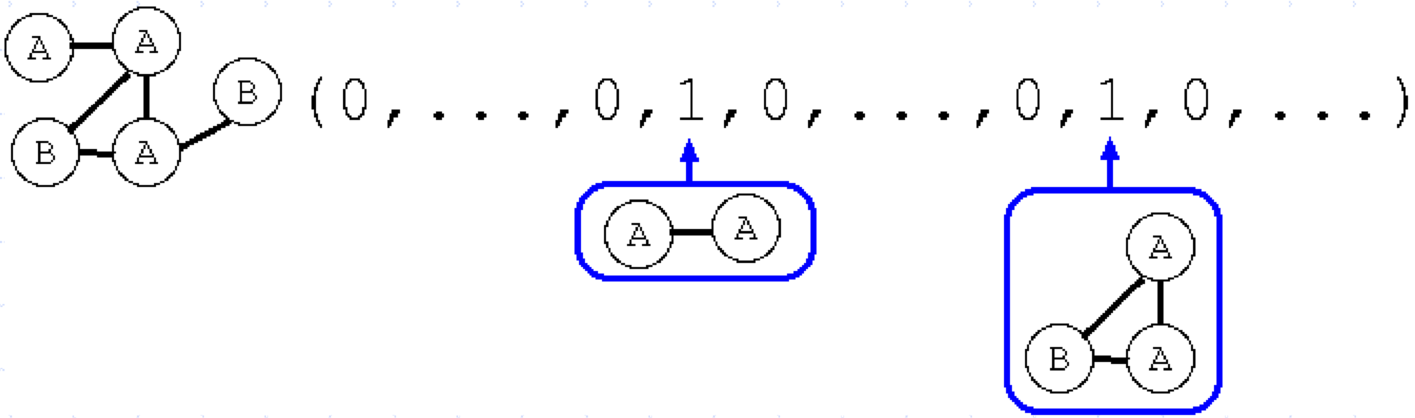
- Set initial point $\boldsymbol{\beta}$ and direction $\boldsymbol{\gamma}$
- Do
  - d1 = Step size if next event is inclusion
  - d2 = Step size if next event is exclusion
  - d = min(d1,d2)
  - $\boldsymbol{\beta} = \boldsymbol{\beta} + d\boldsymbol{\gamma}$
  - Update the active feature set
  - Set the next direction $\boldsymbol{\gamma}$
- Until all features are included

Main Search Problem

Feature space of patterns

- Graph training data $\mathcal{G} = \{G_i\}_{i=1}^{n}$
- Set of all subgraphs (patterns) $\mathcal{T}$
- Each graph is represented as

$$\boldsymbol{x}_i = (x_{it})_{t \in T}\ , x_{it} = I(t \subseteq G_i)$$



Main Search problem

- Step size if pattern t is included next

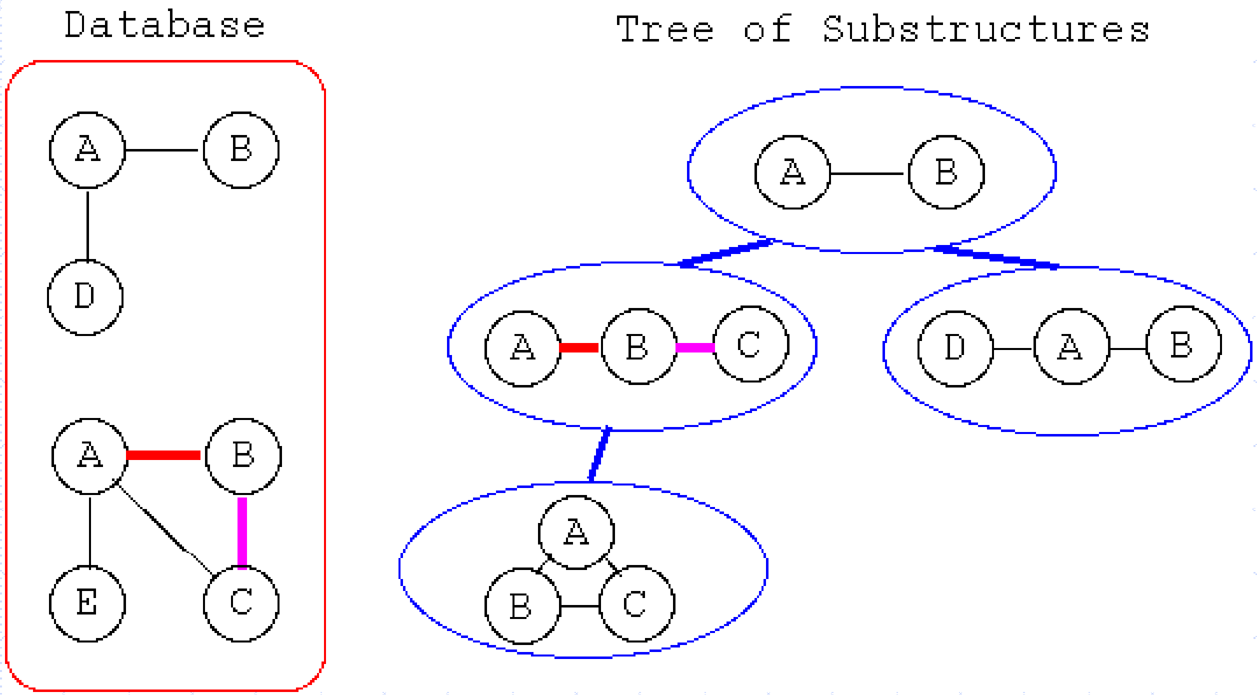
$$d_t = \min_{+} \left\{ \frac{\rho_0 - \sum_i w_i x_{it}}{\eta_0 - \sum_i v_i x_{it}}, \quad \frac{\rho_0 + \sum_i w_i x_{it}}{\eta_0 + \sum_i v_i x_{it}} \right\}.$$

w_i, v_i, ρ_0, η_0 : constants computed from active set

- Find pattern $t \in \mathcal{T}$ that minimizes d_t

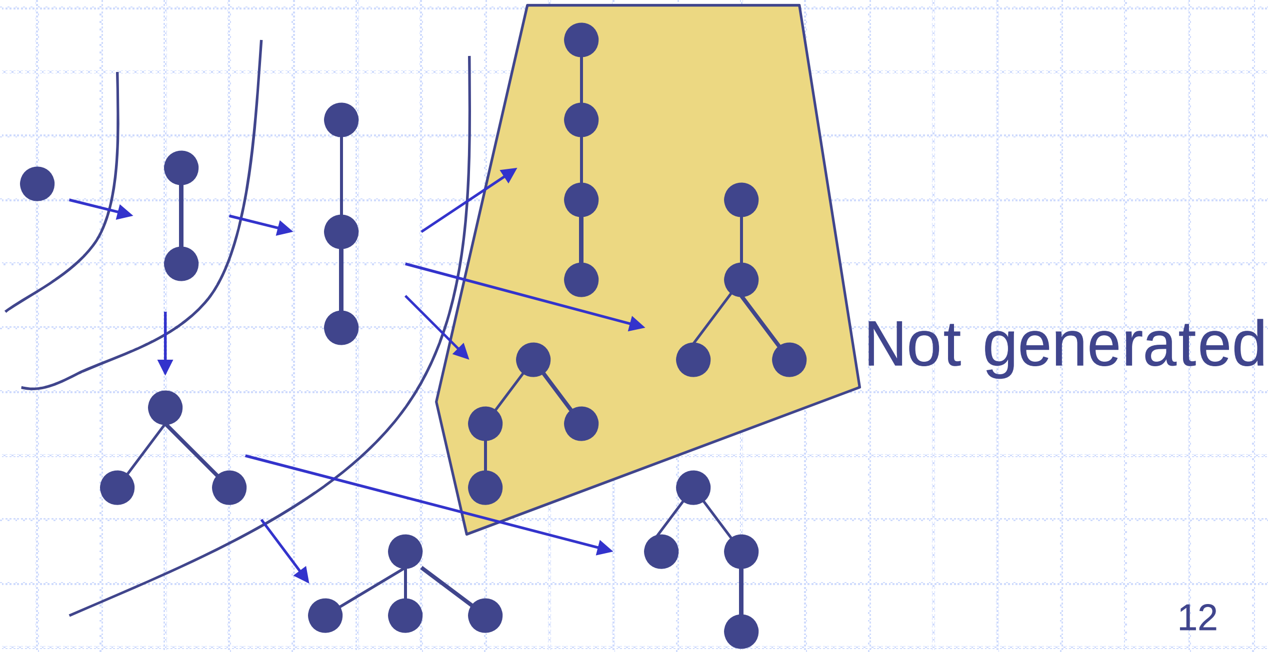
Tree-shaped Search Space

- Each node has a pattern
- Generate nodes from the root:
  - Add an edge at each step



Tree Pruning

- If it is guaranteed that the optimal pattern is not in the downstream, the search tree can be pruned



Theorem (Pruning condition)

- Traversed up to pattern t
- d_t^* :Minimum value so far
- No better pattern in the downstream, if

$$b_{\boldsymbol{w}} + d_t^* b_{\boldsymbol{v}} < |\rho_0| - d_t^* |\eta_0|.$$

where

$$b_{\boldsymbol{w}} = \max\left\{ \sum_{w_i < 0} |w_i| x_{it}, \sum_{w_i > 0} |w_i| x_{it} \right\}.$$

Reusing the search space

- Main search is solved repeatedly with different parameters
- More efficient to reuse the search space in next iterations
  - Node generation is expensive due to the minimum DFS code check
- Whole tree of patterns is kept in memory and progressively extended

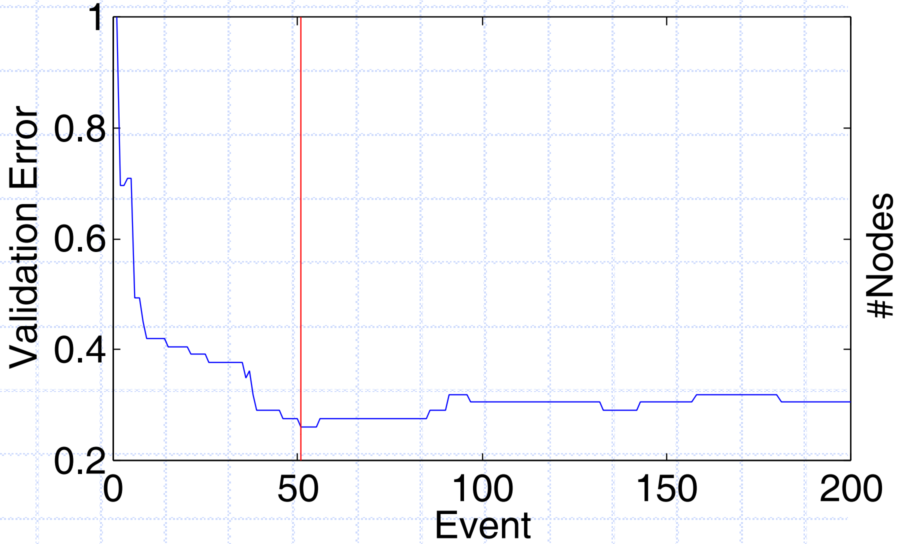
Experiments

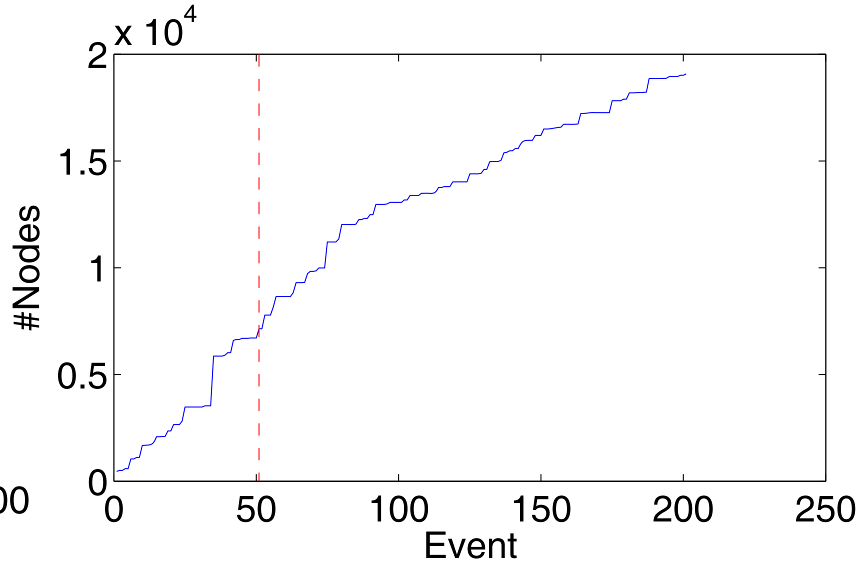
- Naïve Method
  - Enumerate all patterns whose edge size is smaller than **maxpat**
  - Then, LAR-LASSO is applied
- CPDB dataset
  - 683 training graphs (chemical compounds)
  - Classification dataset (mutagenetic or not)
  - Converted to regression problem (y=1,-1)

How to measure the computational cost of our method

- Data divided into 90% train and 10% validation
- Record
  - Number of nodes in tree
  - Computation time

at the point of minimum validation error

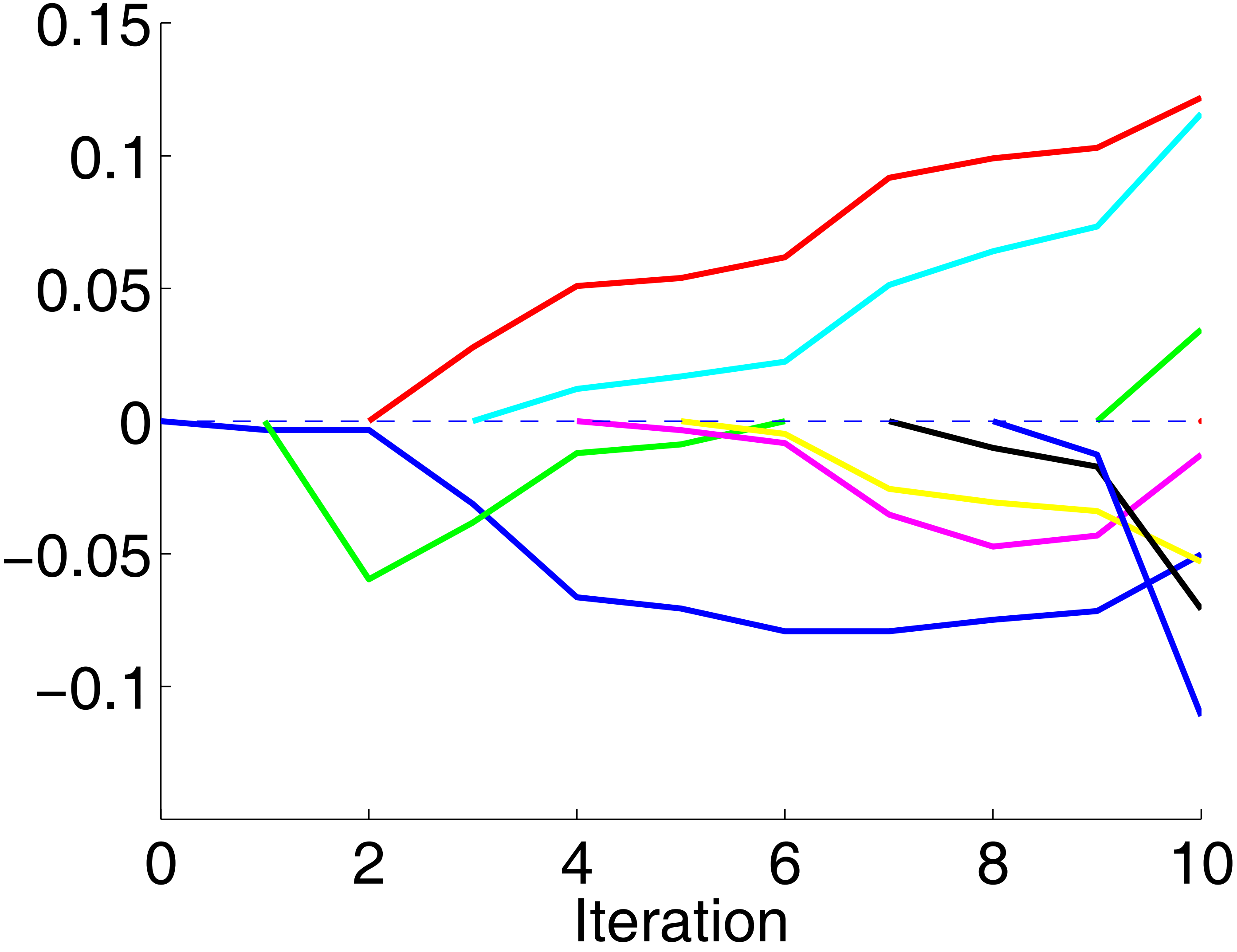




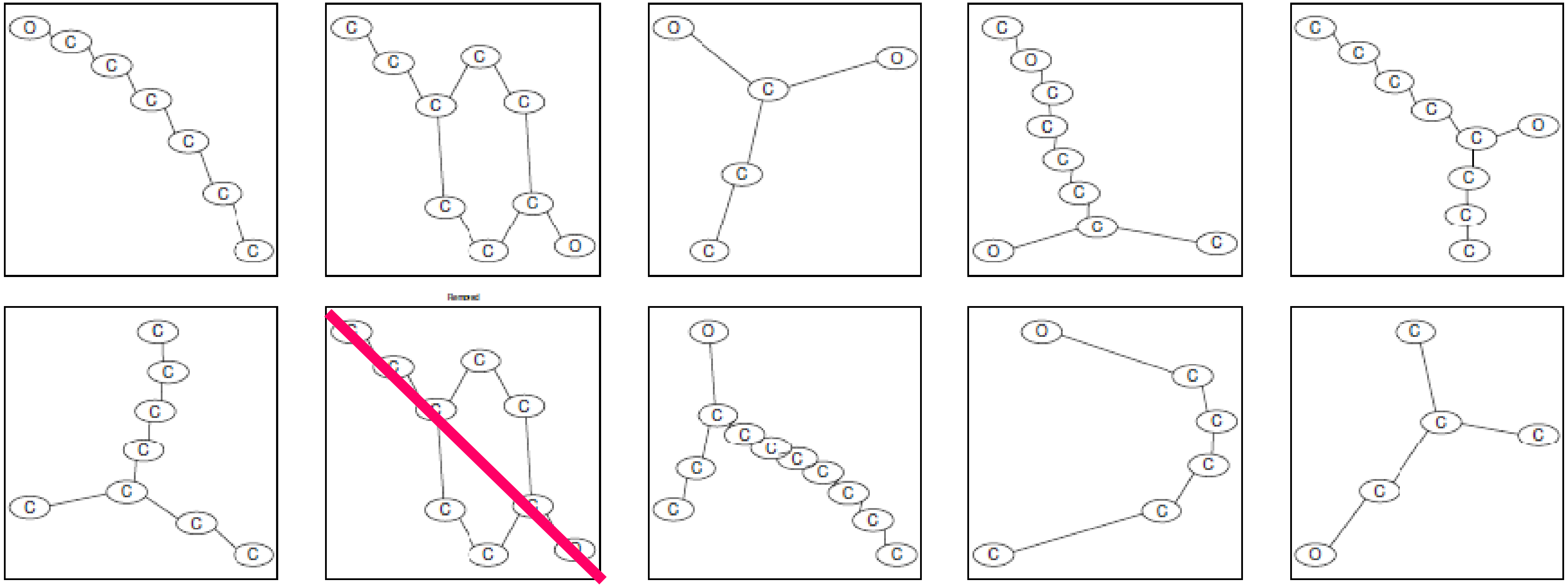
Computational Cost

	Naive		Progressive	
maxpat	#nodes	time	#nodes	time
5	2142	0.51	1171	0.38
6	5717	1.39	2111	0.67
7	14309	3.79	3614	1.36
8	33862	9.93	4936	1.90
9	75814	25.64	6605	2.42
10	161858	65.60	7961	2.80
11	332553	164.20	8613	3.00
12	665202	397.95	8857	3.12
13	1302273	931.61	8964	3.11
∞	N/A	N/A	9088	3.15

Solution Path



Events



Conclusion

- Path following implemented for graph data
- Pattern search by the DFS code tree
- Hinge loss: To do
  - Search criterion more complicated
- Easily combined with itemset mining, tree mining, sequence mining

gboost MATLAB toolbox

- Graph classification by LPBoost + DFS Code Tree
- Includes an implementation of gspan
- www.kyb.mpg.de/people/nowozin/gboost
- Path following code will be available soon