Machine Learning Summer School Kyoto 2012

Graph Mining

Chapter 1: Data Mining

National Institute of Advanced Industrial Science and Technology

Koji Tsuda

National Institute of Advanced Industrial Science and Technology (AIST)

- Research institute under METI for 6 fields: Life Science, Informatics, Environment and Energy, Nanotechnology and Materials, Geological Survey, Standard
- 2500 scientists, 700 administrative staff, 5200 scientists from outside
- Since 2001

Computational Biology Research Center

- Odaiba, Tokyo
- Developing Novel Methods and Tools for
  - Genome Informatics
  - Molecular Informatics
  - Cellular Informatics
- Diverse Collaboration with Companies and Universities

Koji Tsuda: Short Bio

- 1998 PhD in Kyoto Univ, Join ETL
- 2000 GMD FIRST (Berlin, Germany)
- 2001 Join CBRC/AIST
- 2003-2004, 2006-2008 Max Planck Institute for Biological Cybernetics, Tuebingen, Germany
- 2009 Back to CBRC, Machine Learning Group Leader

About this lecture

- How to extract knowledge from structured data
- Itemset mining, tree mining, graph mining
  - "Reverse Search Principle"
- Learning from structured data
- Kernels for structured data

Chapter 1 (Data Mining)

- 1. Structured Data in Biology
- 2. Itemset Mining
- 3. Closed Itemset Mining
- 4. Ordered Tree Mining
- 5. Unordered Tree Mining
- 5. Graph Mining
- 6. Dense Module Enumeration

Agenda 2 (Learning from Structured Data)

- 1. Preliminaries
- 2. Graph Clustering by EM
- 3. Graph Boosting
- 4. Regularization Paths in Graph Classification
- 5. Itemset Boosting for predicting HIV drug resistance

Agenda 3 (Kernel)

- 1. Kernel Method Revisited
- 2. Marginalized Kernels (Fisher Kernels)
- 3. Marginalized Graph Kernels
- 4. Weisfeiler-Lehman kernels
- 5. Reaction Graph kernels
- 6. Concluding Remark

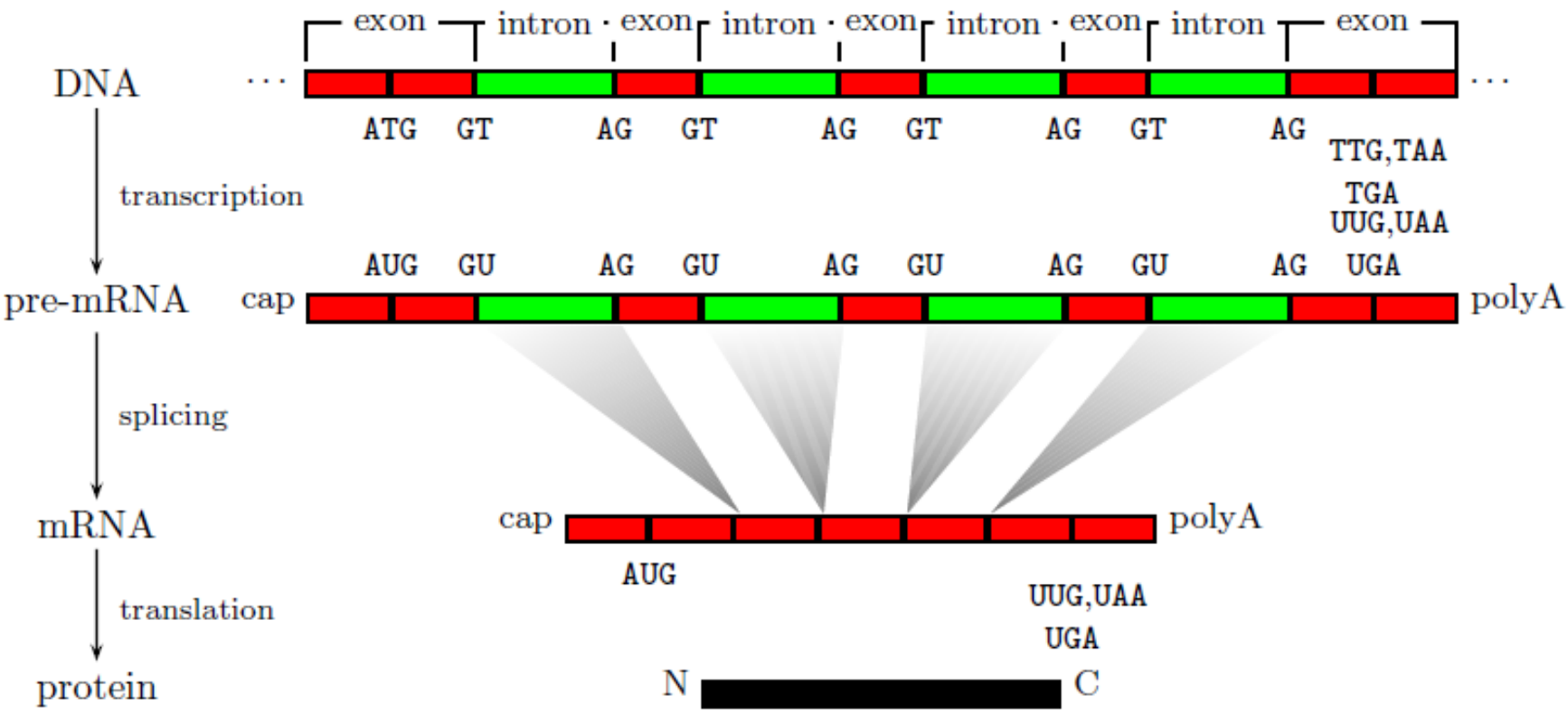
Part 1: Structural Data in Biology

Biological Sequences

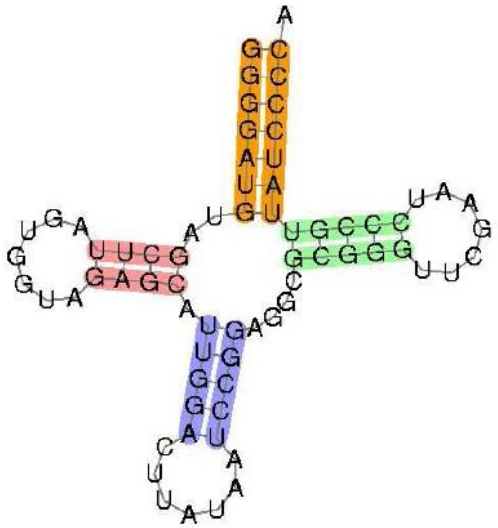
- DNA sequences (A,C,G,T)
  - Gene Finding, Splice Sites
- RNA sequences (A,C,G,U)
  - MicroRNA discovery, etc.
- Amino acid sequences (20 symbols)
  - Remote homolog detection, Fold recognition etc.

Structures hidden in sequences (I)

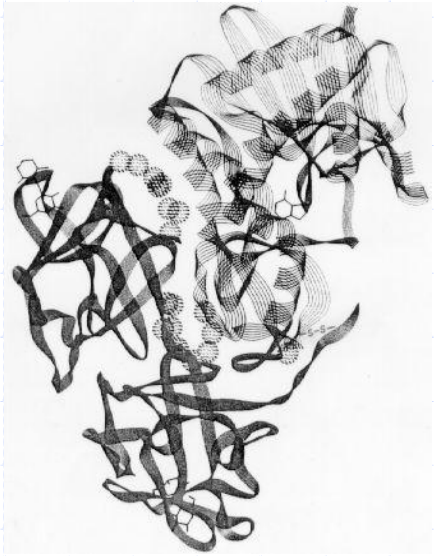
- Exon/intron of DNA (Gene)



Structures hidden in sequences (II)



RNA
Secondary
Structure



Protein
3D Structures

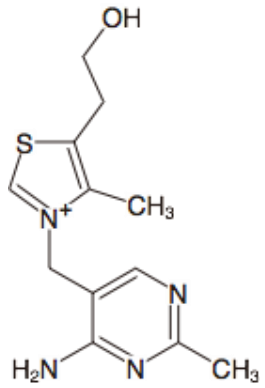
- It is crucial to infer hidden structures and exploit them for classification

→ Biological Graphs

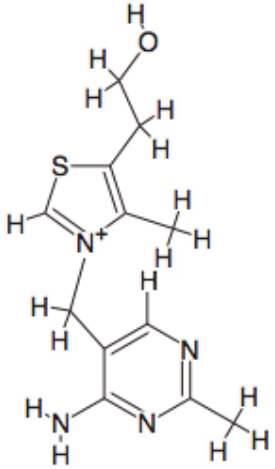
Molecular graphs

graph =
a set of **dots**
& **lines**
(or nodes &
edges)

Structure: Thiamine (Vitamin B_1)

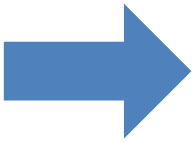


implicit
hydrogens

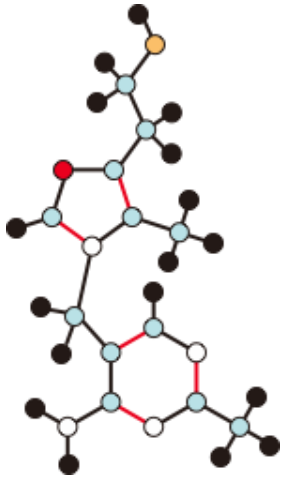


explicit
hydrogens

abstraction



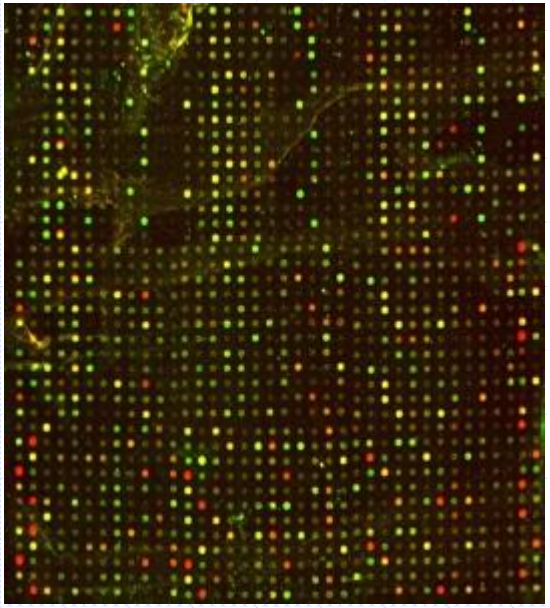
molecular graph



Hydrogen
Carbon
Oxygen
Nitrogen
Sulfur
single bond
double bond

Gene Expression Data

- Measurement of many mRNAs in the cell
- Rough estimate of amount of proteins
- Time-series or not
- Snapshot of the underlying dynamic system

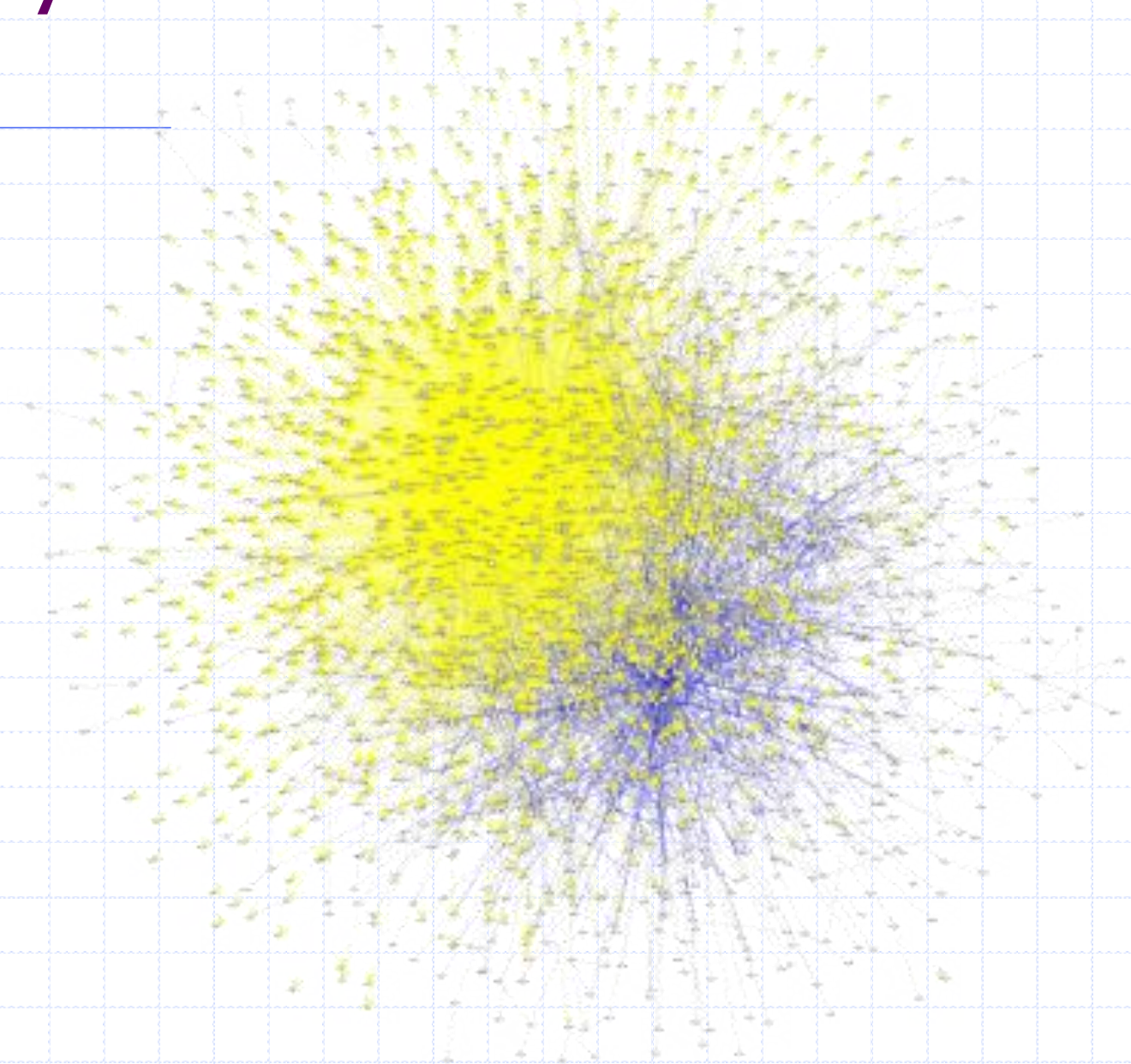


Biological Networks

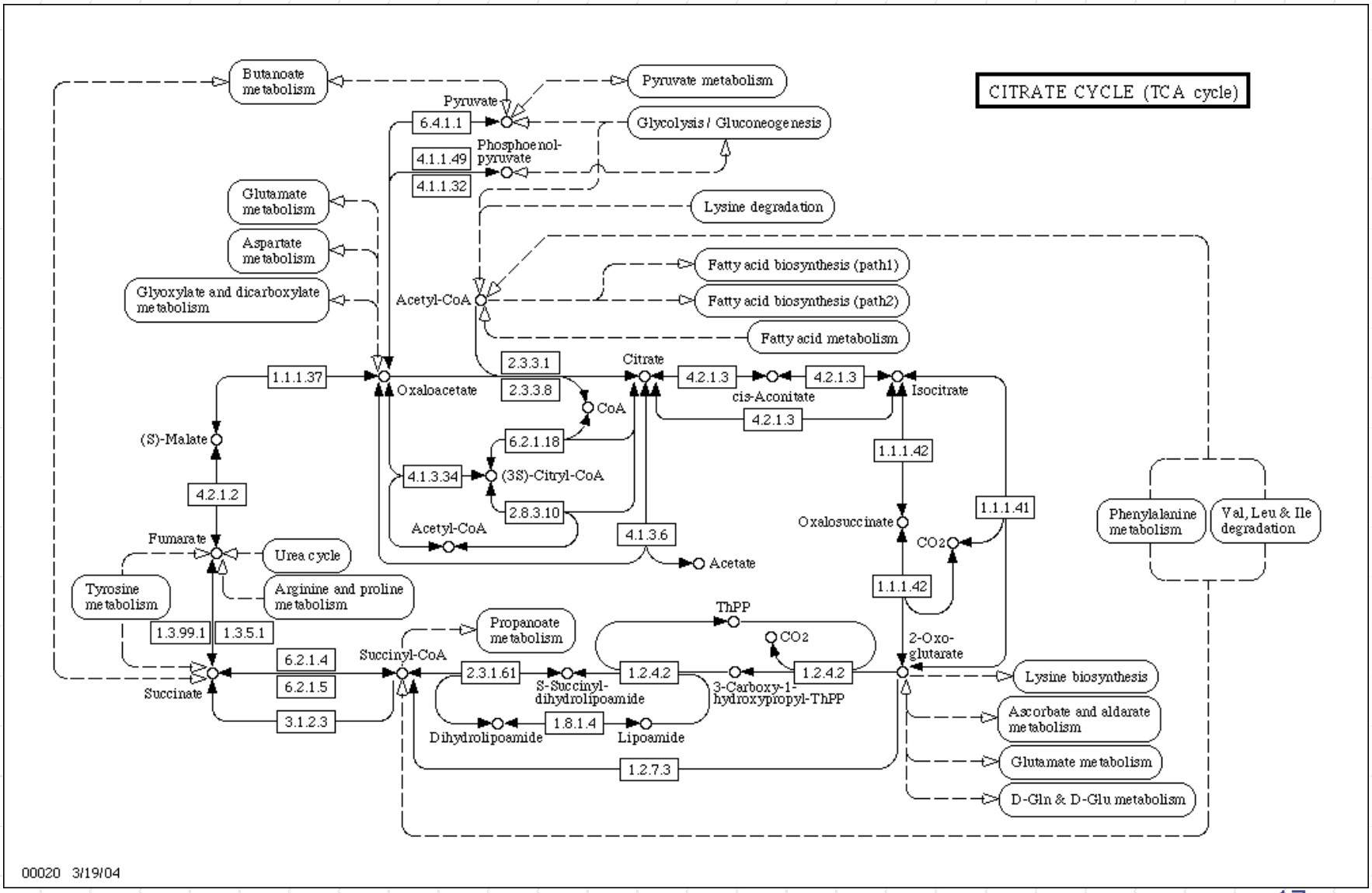
- Protein-protein physical interaction
- Metabolic networks
- Gene regulatory networks
- Network induced from sequence similarity

- Thousands of nodes (genes/proteins)
- 100000s of edges (interactions)

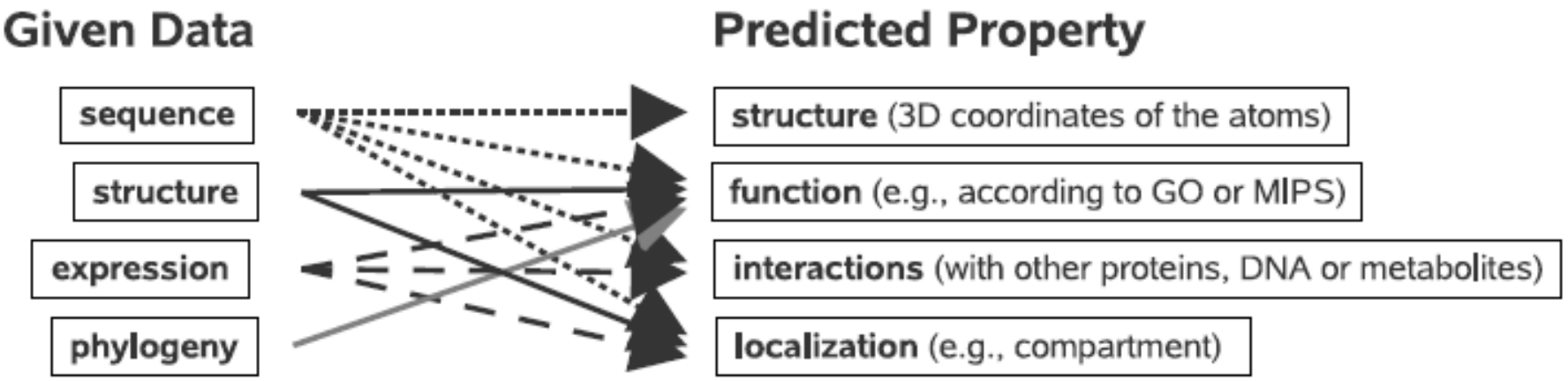
Physical Interaction Network



Metabolic Network



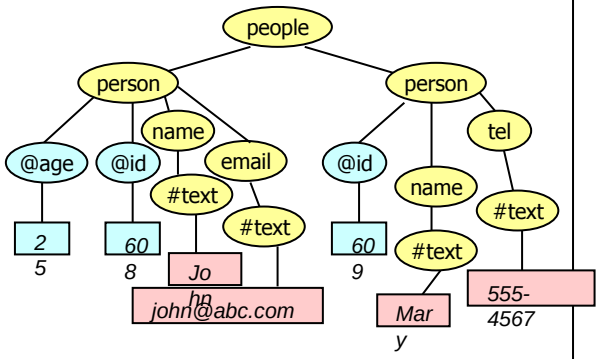
Many possible prediction problems..



Part 2: Itemset mining

Data Mining

- A formal study of efficient methods for extracting interesting rules and patterns from massive data



- Frequent itemset mining (Agrawal and Srikant 1994)
- Closed pattern mining
- Structured data mining (Sequence, Trees, and Graphs)

Frequent Itemset Mining

[Agrawal, Srikant, VLDB'94]

- Finding all "frequent" sets of elements (items) appearing σ times or more in a database

Minsup σ= **2**

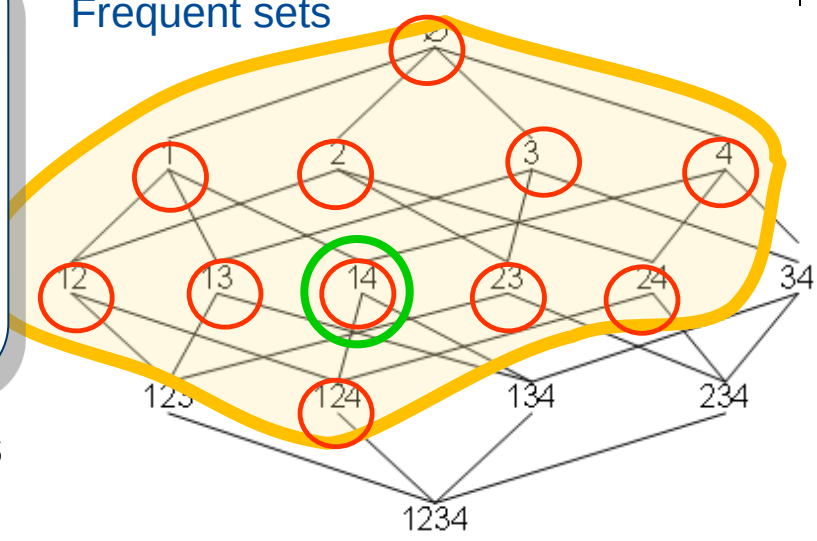
	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

Frequent sets
∅,
1, 2, 3, 4,
12, 13, 14,
23, 24, 124

X = {2, 4} appears **three times**, thus **frequent**

Frequent sets



The itemset lattice $(2^{\Sigma}, \subseteq)$

Definitions: Database

- A set $\mathbf{\Sigma} = \{ 1, ..., n \}$ of items (elements)
- Transaction database
  - A set $\mathbf{T} = \{ t_1, ..., t_m \}$ of subsets of $\mathbf{\Sigma}$
  - Each subset $t \subseteq \mathbf{\Sigma}$ is called a transaction

L = {1, 2, 3, 4}

- **Alphabet of items**

id	transaction
t1	1, 3
t2	2, 4
t3	1, 2, 3, 4
t4	1, 2, 4

•22

Definitions: Frequent sets

- Itemset X appears in transaction t: $X \subseteq t$
- Occurrence of X in database ***T***:
  $Occ(X, \boldsymbol{T}) = \{ t \in \boldsymbol{T} : X \subseteq t \}$
- Frequency of X: $Fr(X, T) = |\, Occ(X, T) \,|$
- Minimum support (minsup): $0 \leqq \sigma \leqq |\boldsymbol{T}|$
- X is frequent in T if $Fr(X, T) \geqq \sigma$.

$I = \{1, 2, 3, 4\}$

Alphabet of items

Transaction database

id	transaction
t1	1, 3
t2	2, 4
t3	1, 2, 3, 4
t4	1, 2, 4

Occurrences and frequencies of itemsets

Occ(**3**, T) = {t1, t3}
Fr(**3**, T) = 2

Occ(**24**, T) = {**t**2, **t**3, **t**4},
Fr(**24**, T) = 3

•23

Market Basket Data

- Popular application of itemset mining
- Business and Market data analysis

- Transaction Data of purchase

ID	Chips	Mustard	Sausage	Softdrink	Beer
001	1	0	0	0	1
002	1	1	1	1	1
003	1	0	1	0	0
004	0	0	1	0	1
005	0	1	1	1	1
006	1	1	1	0	1
007	1	0	1	1	1
008	1	1	1	0	0
009	1	0	0	1	0

- a transaction or a "basket"

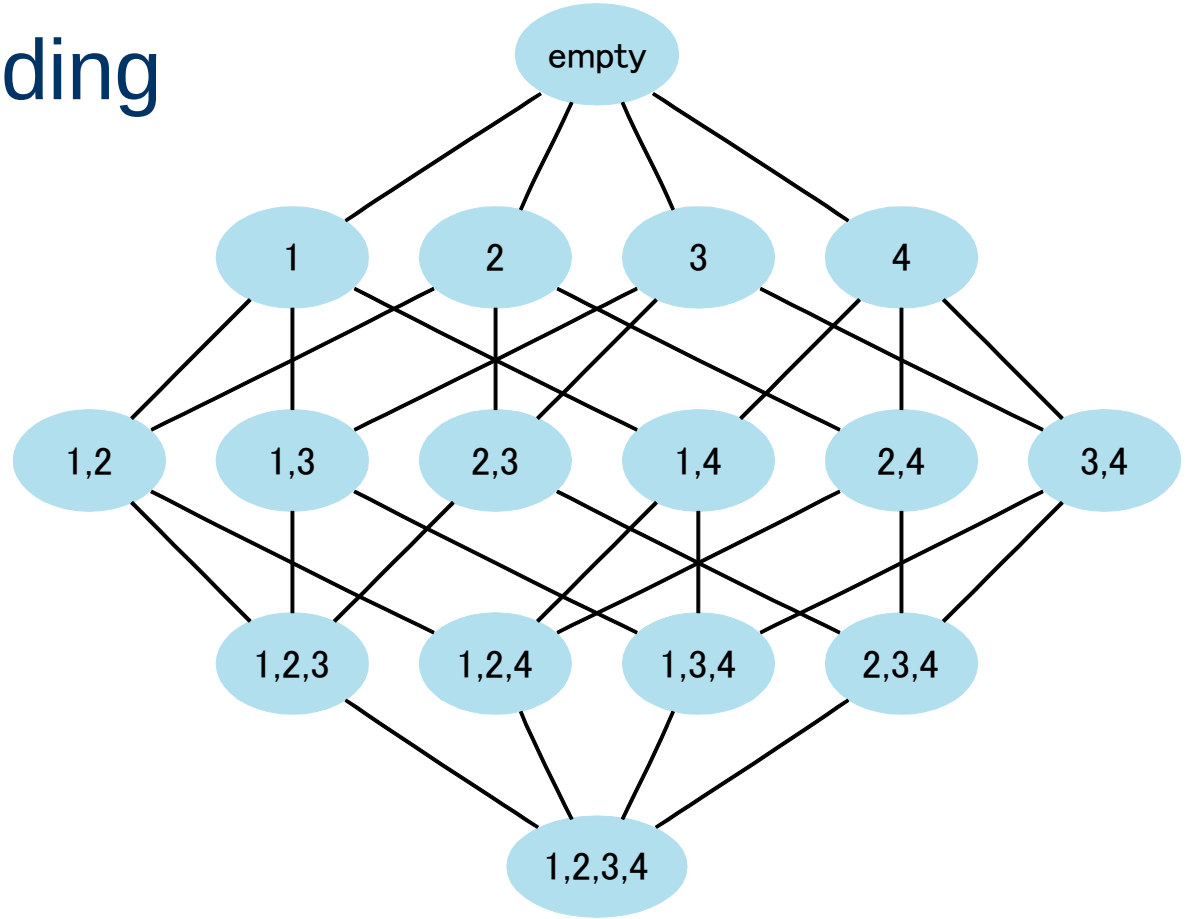
•Item

•Meaning of the transaction 003

"Custmer 003 bought Chips and Sausage together in his basket"

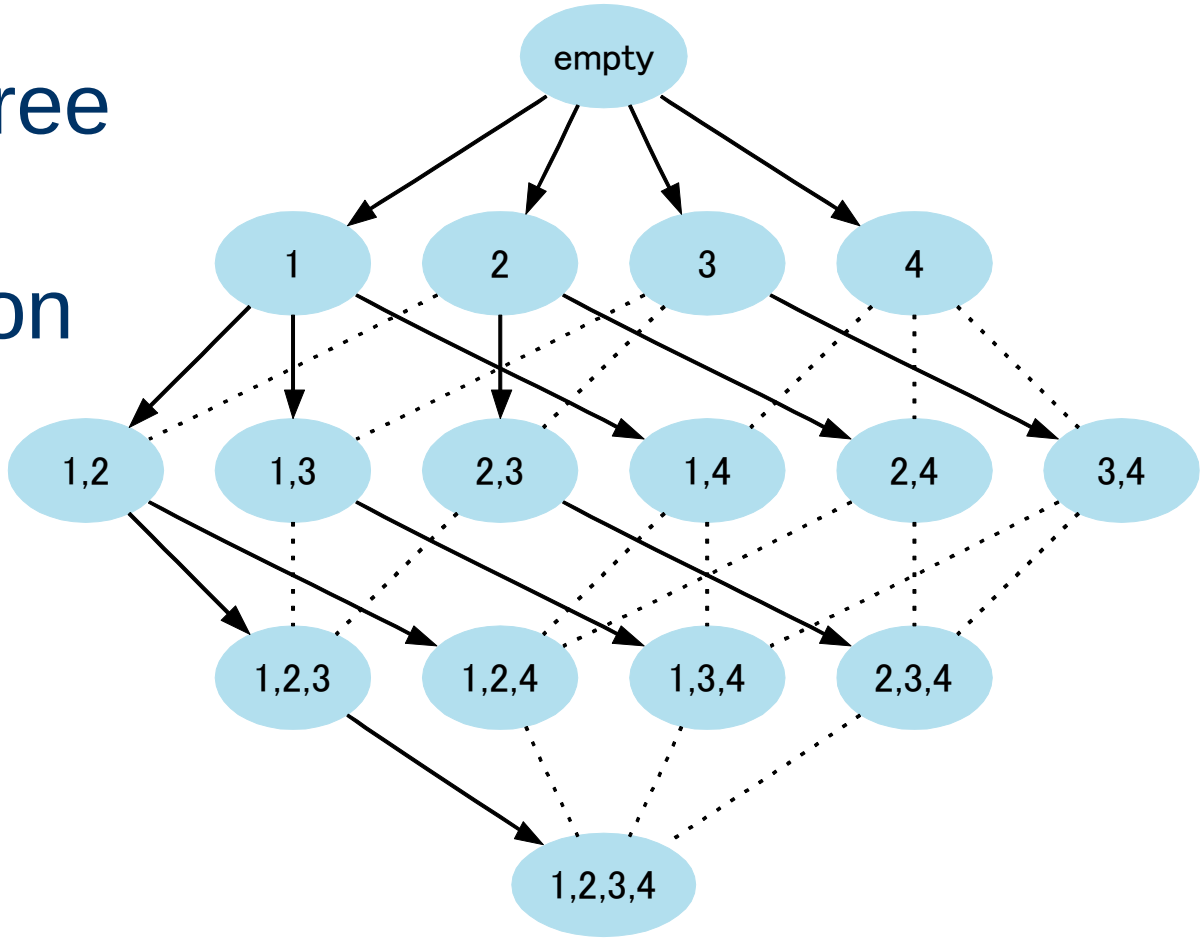
DAG of itemsets: Hasse diagram

- Edge: Adding one item



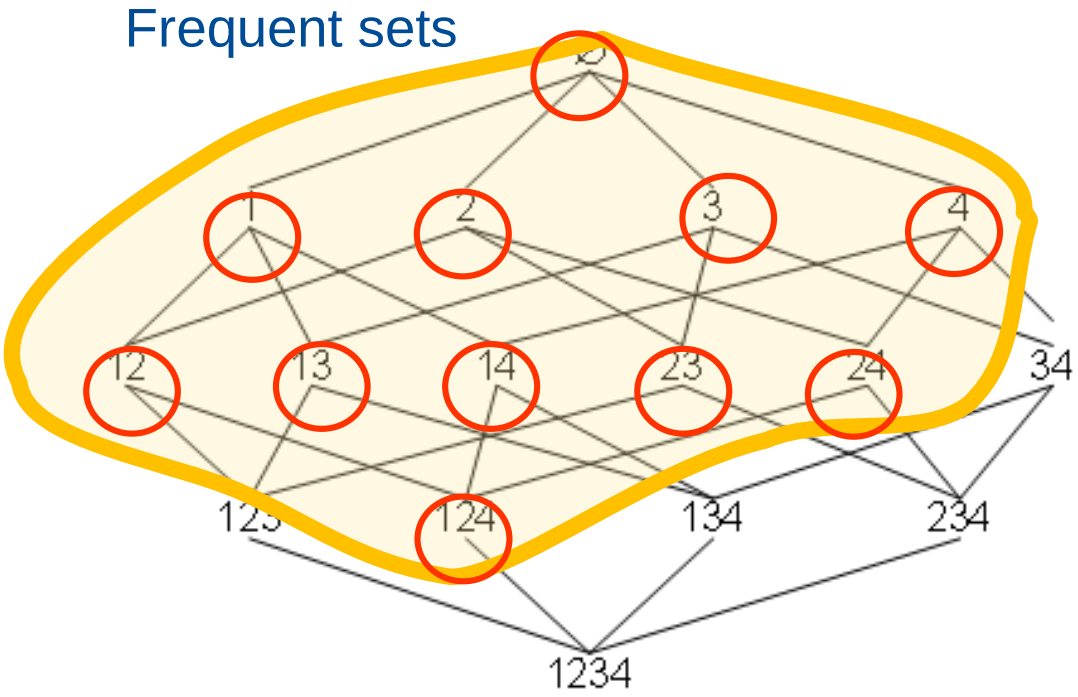
Enumeration Tree by Lexicographical Order

- Need a tree to avoid duplication



Backtracking Algorithm: FP Growth etc.

- Monotonicity: Support only decreases
- Depth First Traversal, Prune if support < σ



Association Rule Mining

- Confidence: Supp(A ∪ B)/ Supp(A)
  - Probability of B, Given A
- What item is likely to be bought when A is bought
- Search: large support, confidence large
- Post-processing of itemset mining

Summary: Itemset mining

- Itemset mining is the simplest of all mining algorithms
- Need to maintain occurrence of each pattern in database
- Tree by lexicographical order is (implicitly) used

Part 3: Closed Itemset mining

Problem in Frequent Pattern Mining

- **Huge Number of frequent itemsets**
- Hard to analyze
- Most of them are similar

Huge number of frequent itemsets discovered in **T**

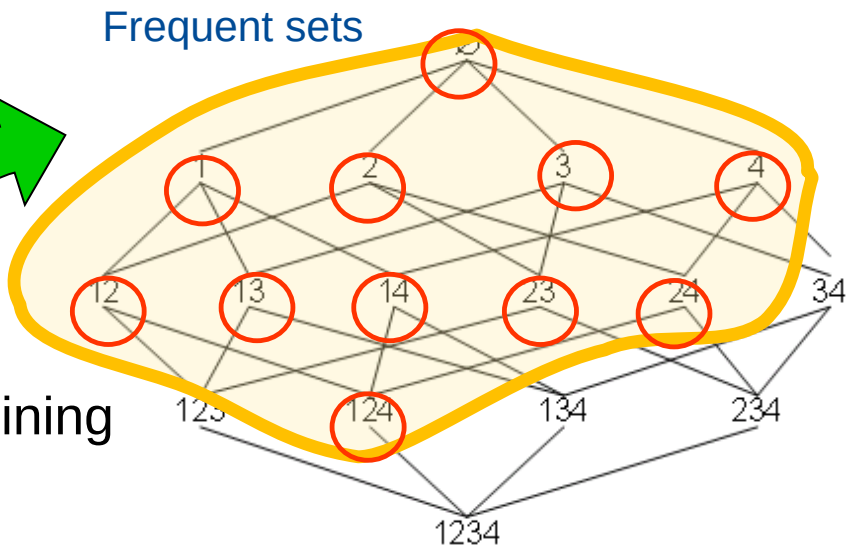
An input transaction database

	1	2	3	4	5
t1	○		○		
t2		○		○	
t3	○	○	○	○	
t4		○	○		○
t5	○	○		○	

database

minsup σ= 2

mining

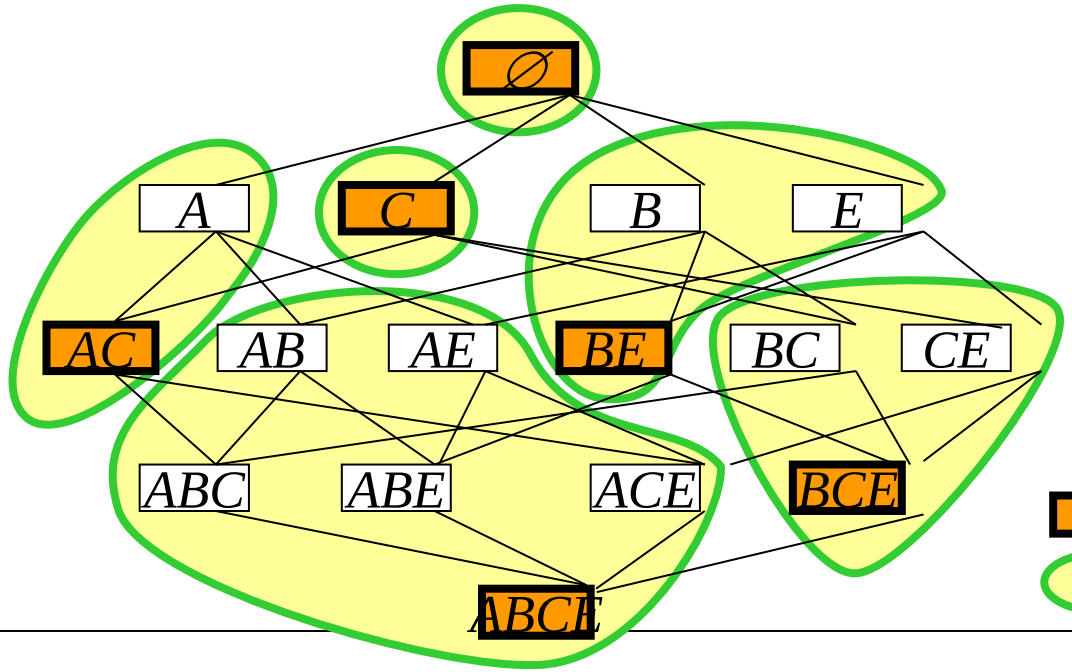


Solution: Closed Pattern Mining

- Find only closed **patterns**
- **Observation**: Most frequent itemset **X** can be extended without changing occurrence by adding new elements
- **def** ([Pasquier et al., ICDT'99]).
  An itemset **X** is a **closed set** if and only if there is **no proper superset** of **X** with the same frequency (thus the same occurrence set).

Closed Pattern Mining

- A closed itemset is **the maximal set** among all itemsets with the same occurrences.
- Equivalence class **[X] = {Y| Occ(X)=Occ(Y) }.**



Database

id	records
1	*A B C E*
2	*A C*
3	*B E*
4	*B C E*

Closed sets (maximal sets)

Equivalence class w.r.t. occurrences

Brute-force: Stupid Baseline

- **ALGORITHM** Bruteforce
  - First, generate all frequent itemsets
  - Check them one by one via maximality test
- **Maximality test** for each candidate frequent set **X**
  - Add some element **e** in Σ to **X**
  - If **Freq(X U {e})** is properly less than **Freq(X)** then reject **X**.

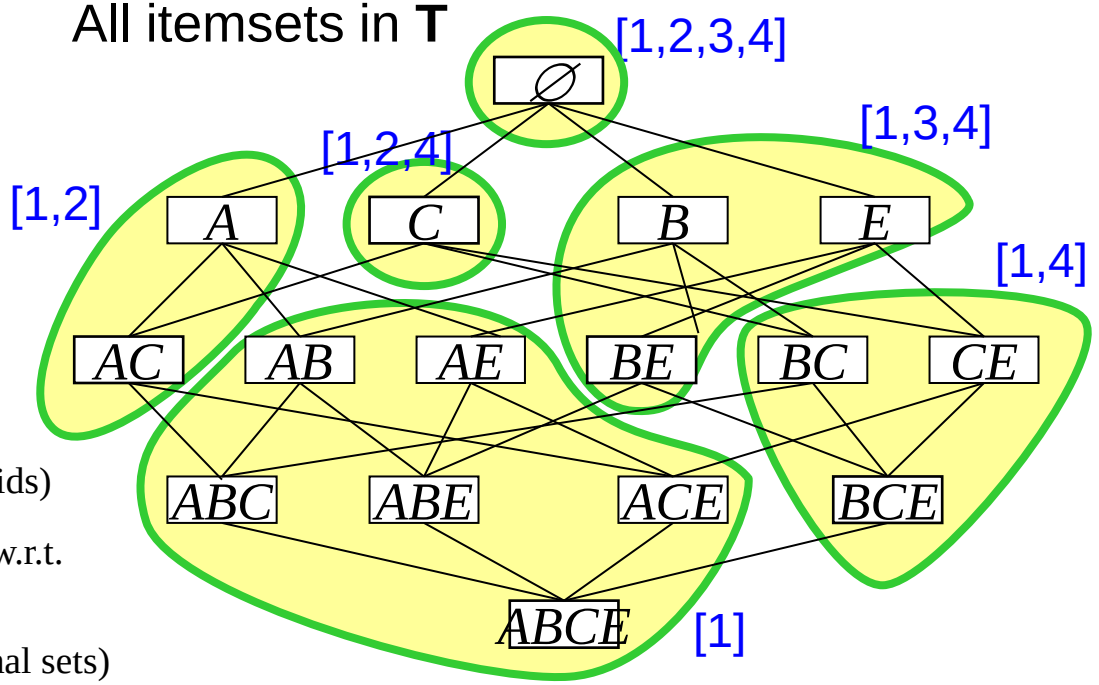
Bruteforce

- STEP1) first, generate all frequent sets

Database **T**

id	records
1	*A B C E*
2	*A C*
3	*B E*
4	*B C E*

All itemsets in **T**



[1,2] Occurrence (set of ids)

Equivalence class w.r.t. occurrences

Closed sets (maximal sets)

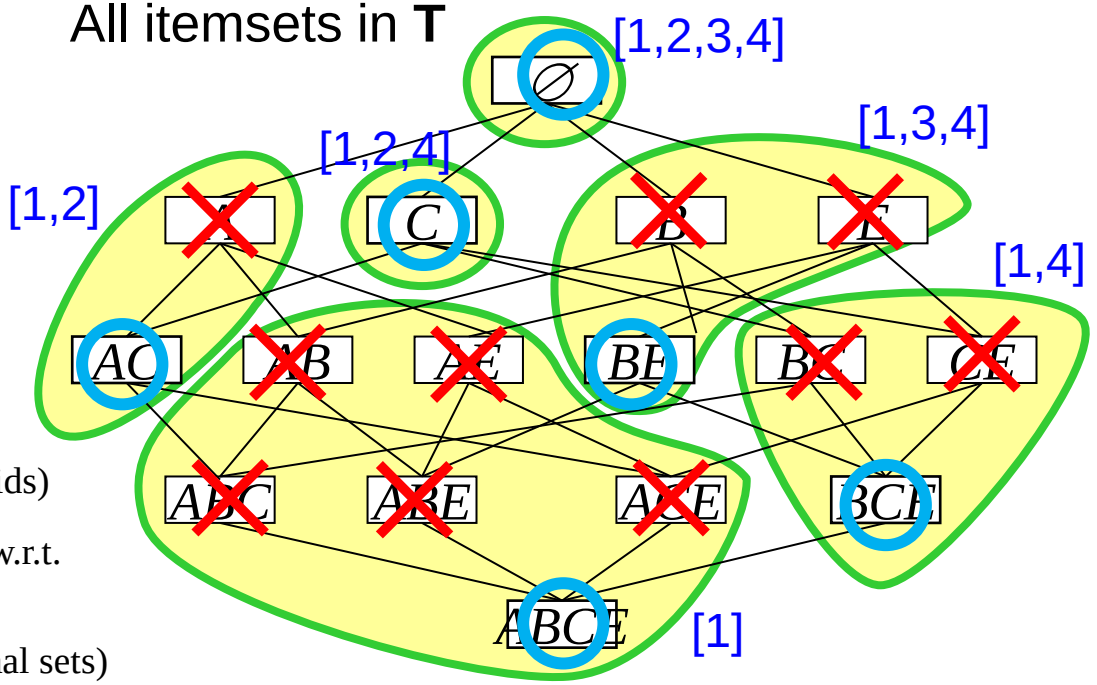
Bruteforce

- STEP1) first, generate all frequent sets
- STEP 2) make closedness test for each set

Database **T**

id	records
1	*A B C E*
2	*A C*
3	*B E*
4	*B C E*

All itemsets in **T**



[1,2] Occurrence (set of ids)

Equivalence class w.r.t. occurrences

Closed sets (maximal sets)

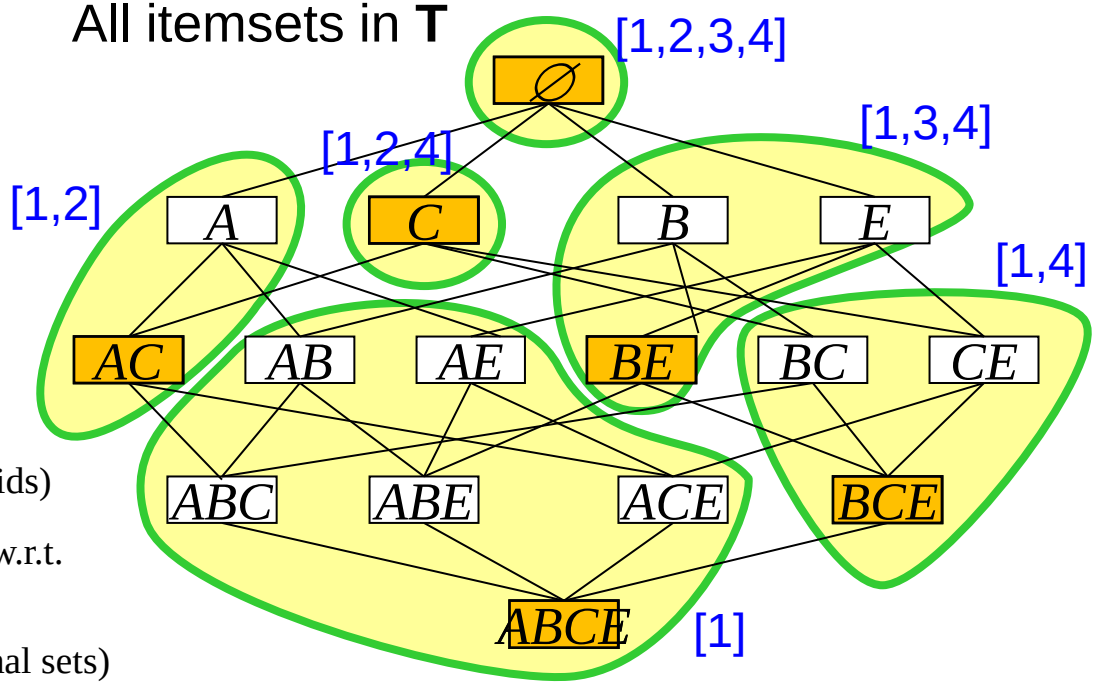
Bruteforce

- STEP1) first, generate all frequent sets
- STEP 2) make closedness test for each set
- STEP3) finally, extract all closed sets

Database **T**

id	records
1	*A B C E*
2	*A C*
3	*B E*
4	*B C E*

All itemsets in **T**



[1,2] Occurrence (set of ids)

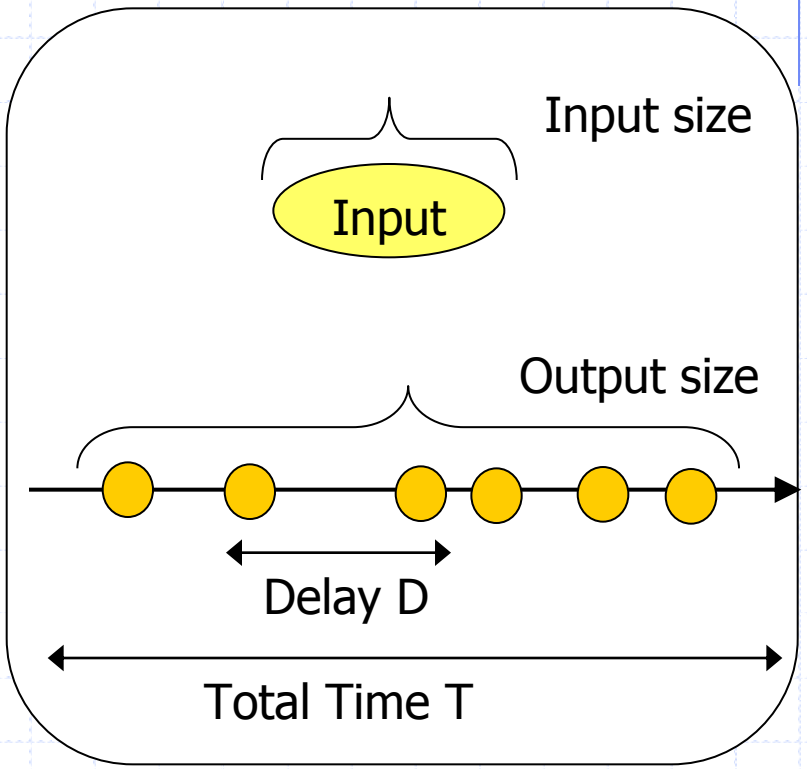
Equivalence class w.r.t. occurrences

Closed sets (maximal sets)

All closed sets are found!

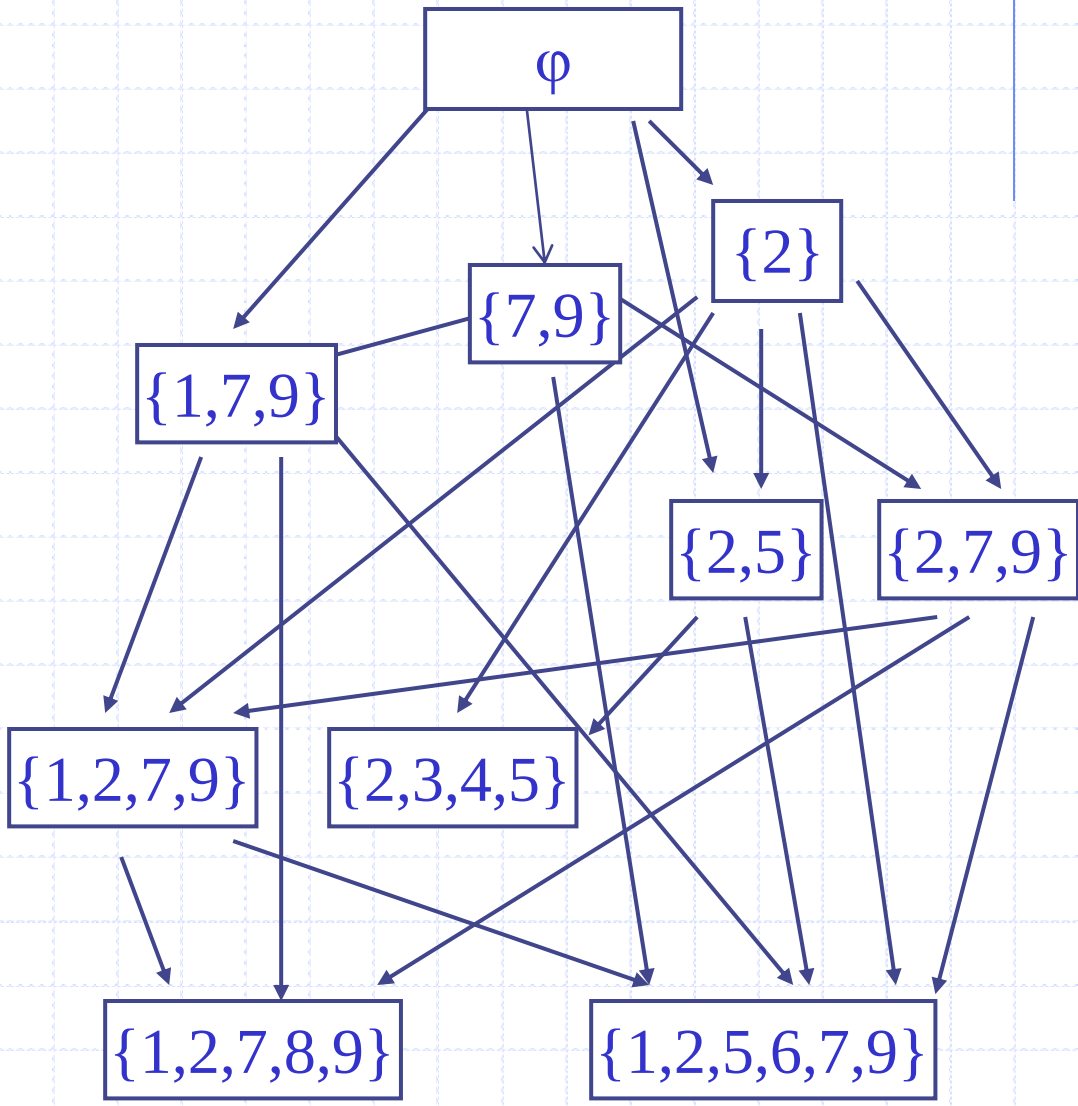
Complexity of Enumeration Algorithms

- Number of patterns usually exponential to input size
- Delay: Time between two pattern outputs
- Brute-force is **exponential** delay w.r.t. pattern size



To achieve *linear* delay,

- Must jump from closed set to closed set
- How to define the search tree?
- Reverse search!
  (Avis and Fukuda 1996)

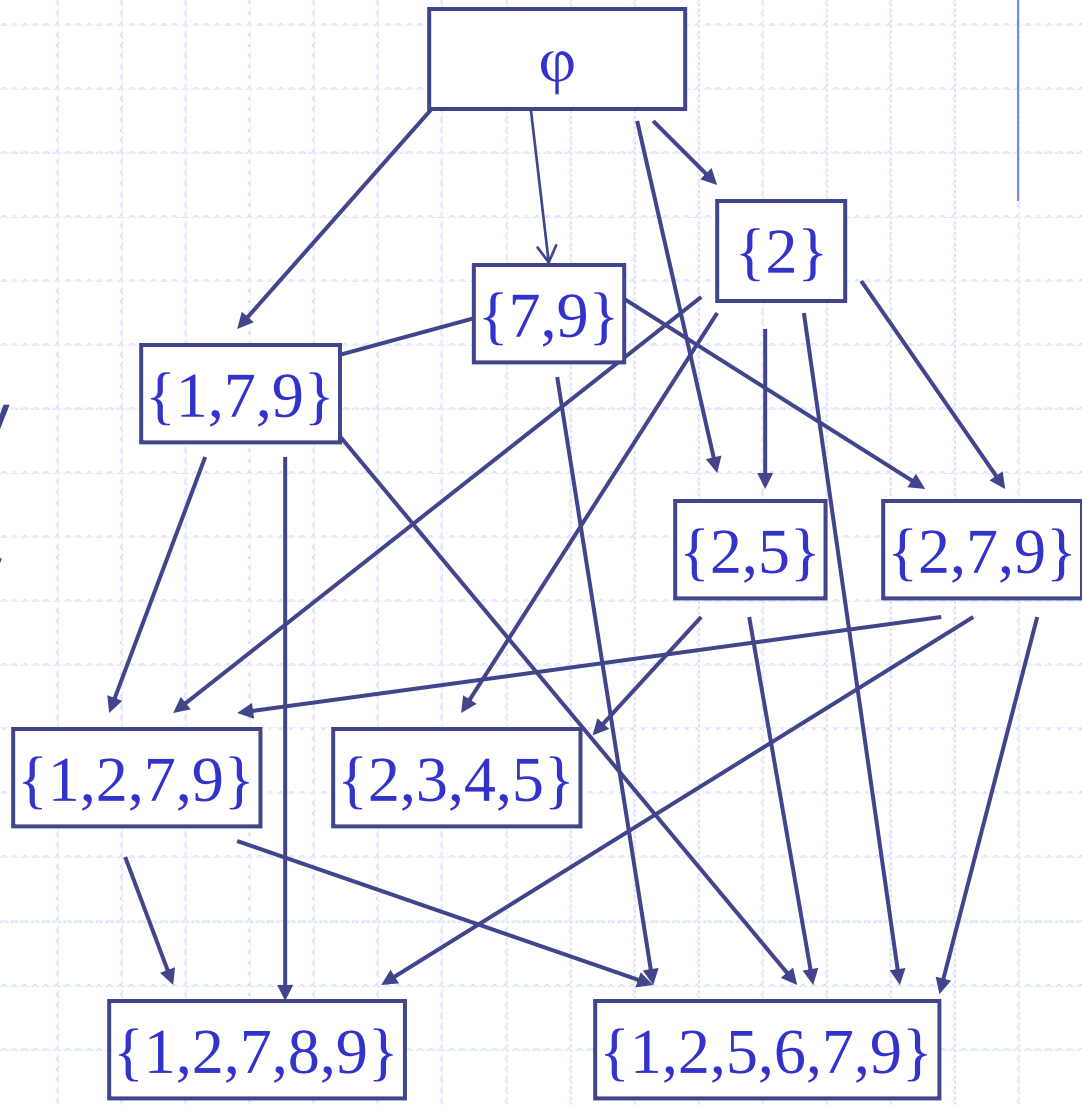


Reverse Search: It's a must

- A general mathematical framework to design enumeration algorithms
- Can be used to prove the correctness of the algorithm
- Popular in computational geometry
- Data mining algorithms can be explained in remarkable simplicity

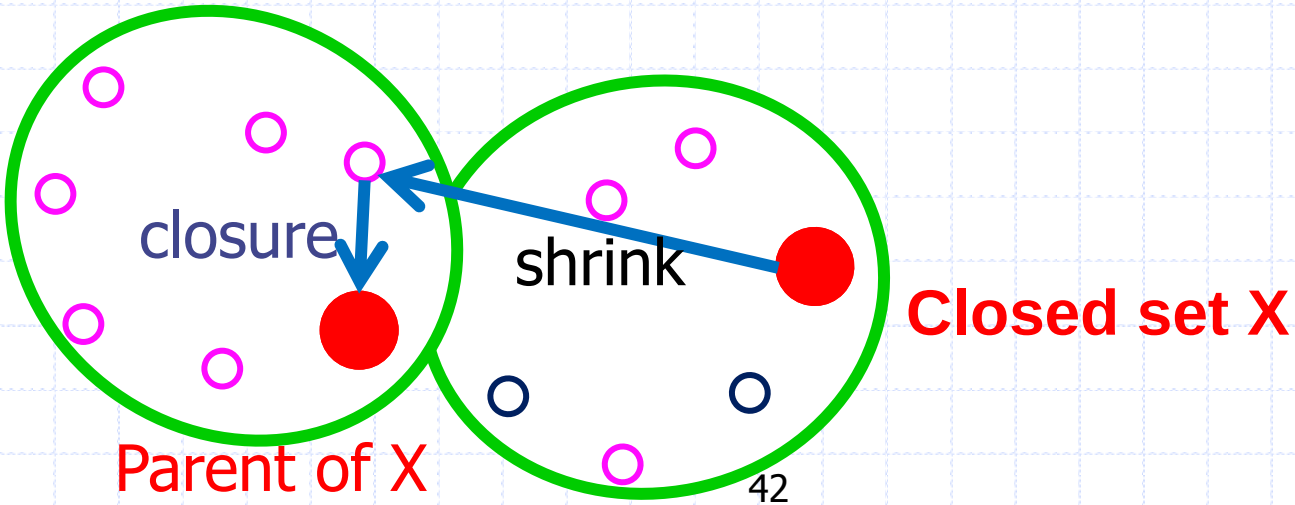
Often, search space comes as a DAG

- Naive Backtracking = Duplication
- Duplication check by Marking = Exponential Memory
- How to visit all nodes without duplication?



Reduction Map

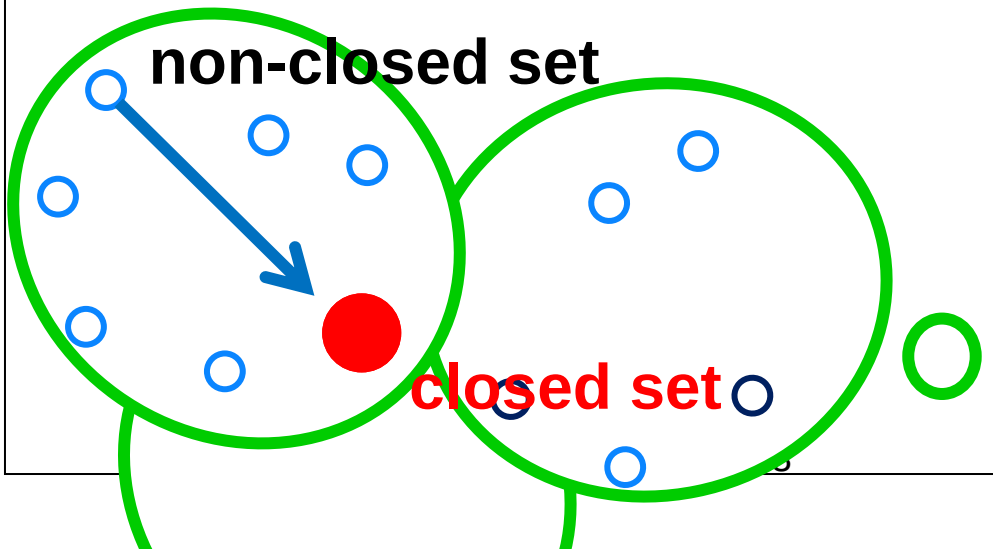
- Mapping from a children to the parent
- Reduction map for closed itemset
  - Shrink the itemset until occurrence changes
  - Take "closure operation"



42

Closure Operation

- **closure(X)** of a set **X:**
  - Closed set computed by **closure(X) = ∩ { t in T : X ⊆ t }.** **(taking the intersection** of all transactions in T that X occurs as subset)



Equivalence class of itemsets with same occurrence

Example of Closure Operation

Database **T**

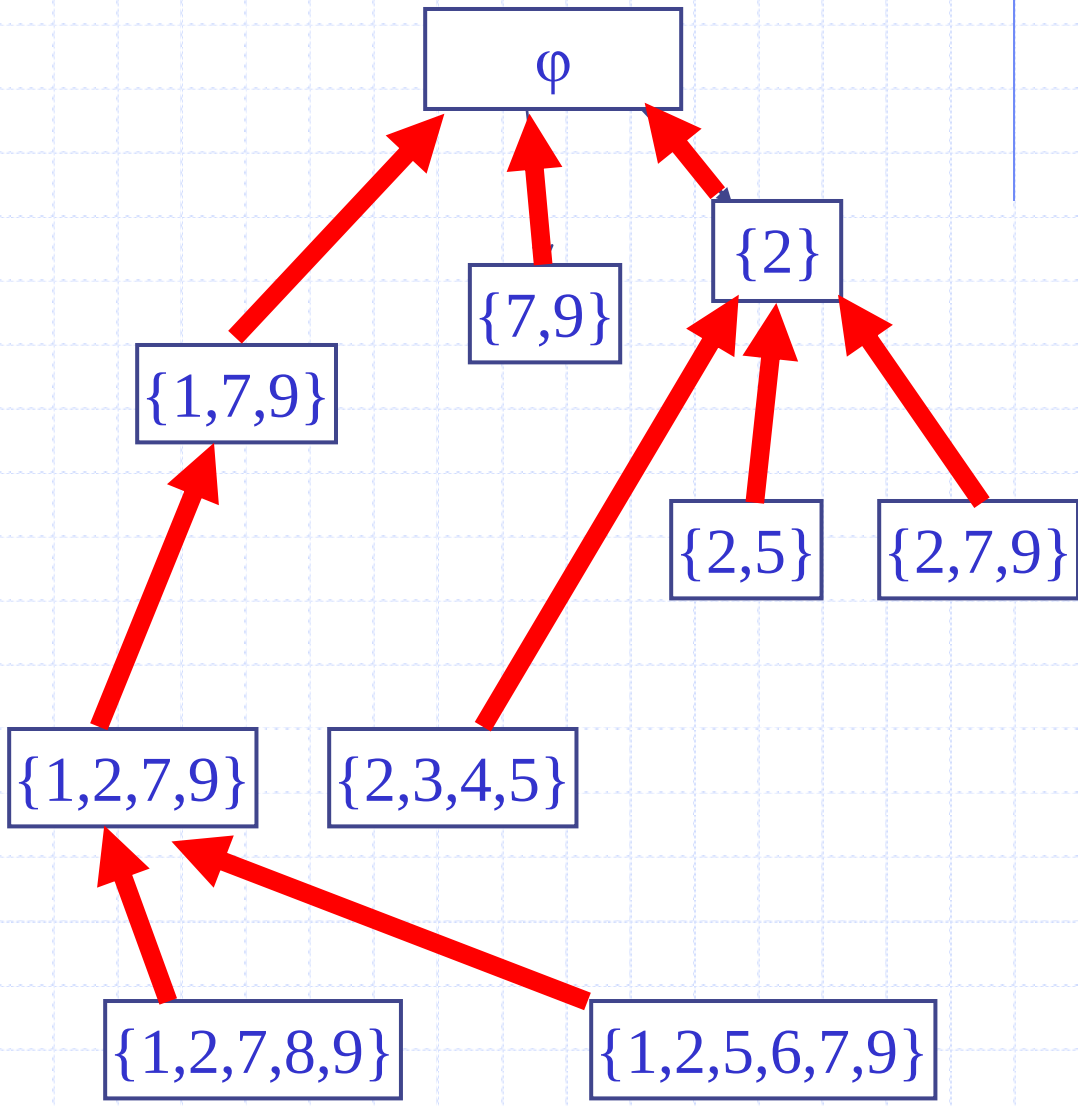
id	records
1	*A B C E*
2	*A C*
3	*B E*
4	*B C E*

- Non-closed itemset: (B,C)
- Occurrence: 1,4
- Take Intersection of 1 and 4

  (A,B,C,E) ∩ (B,C,E) = (B,C,E)
- This is closed itemset

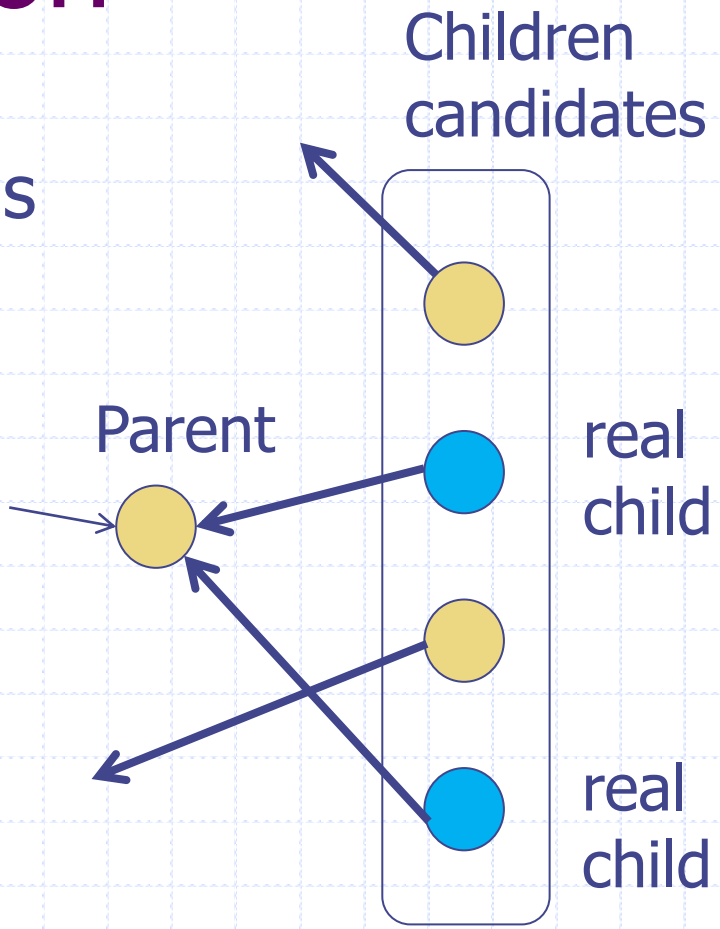
By applying the reduction map to all nodes, enumeration tree is defined.

- But arrows are in reverse direction..



Children generation

- In backtracking, one has to generate all children of the current node
- Inverse of reduction map
  - Generate all children candidates
  - Apply reduction map to them
  - Remove if not coming back



Reverse Search Theorem

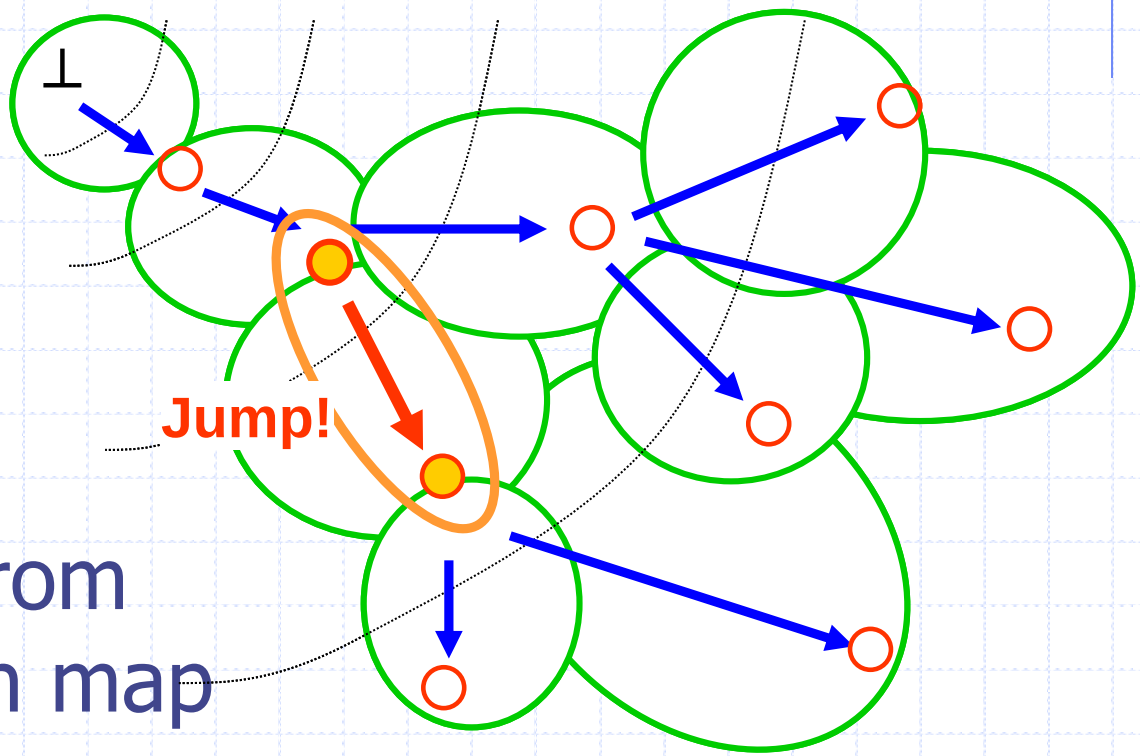
- To prove the correctness, prove the following
  - Reduction map is uniquely defined on all nodes
  - By applying the reduction map repeatedly, one can reach the root node from any node
  - Children generation is inverse of reduction map
- Easy to check !

LCM = Linear Time Closed Sets Miner (Uno et al., 2003)

- **Prefix Preserving Closure Extension**

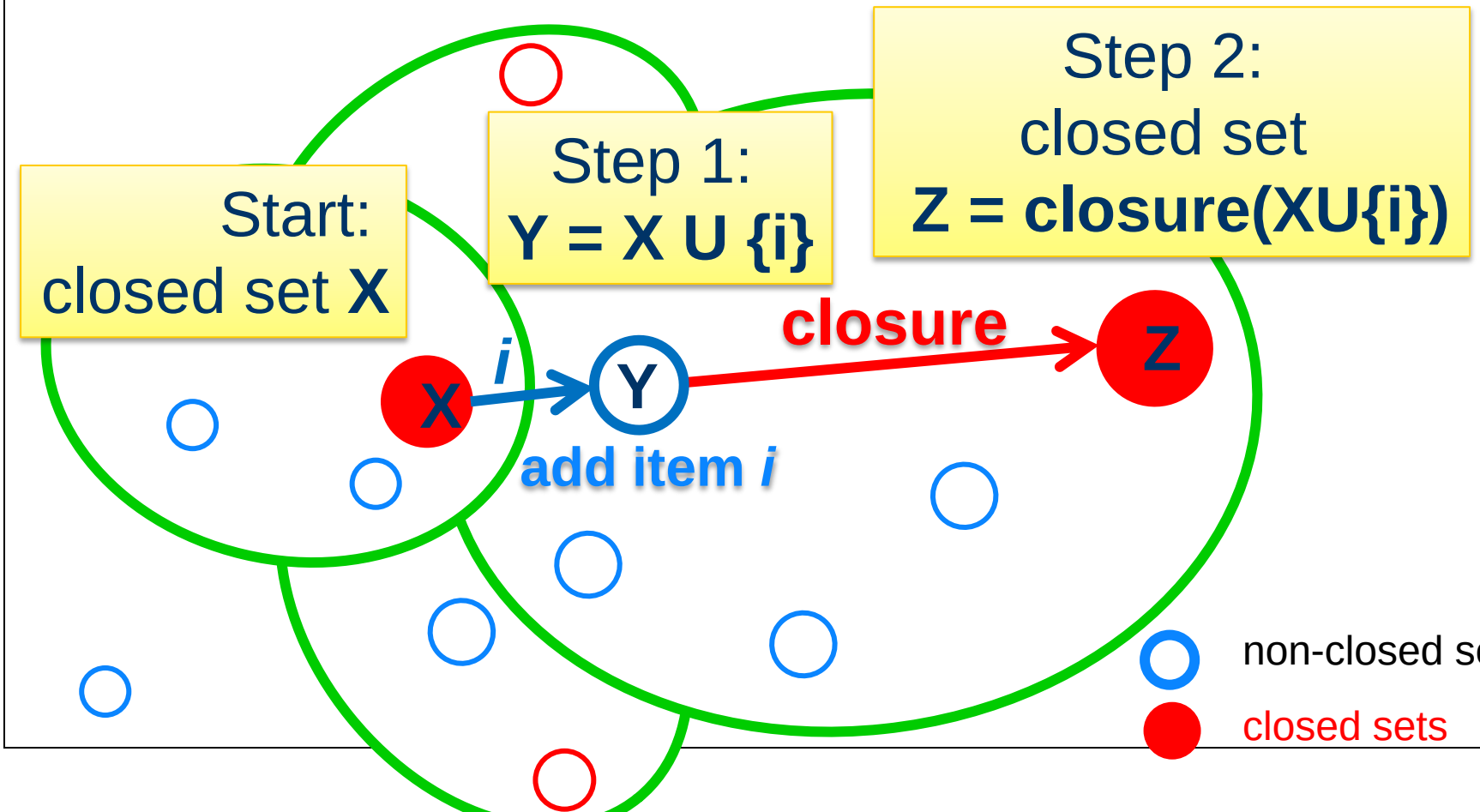
  = Children generation from the reduction map

- Linear Delay!



Closure Extension

- Repeat: Add an item and taking closure



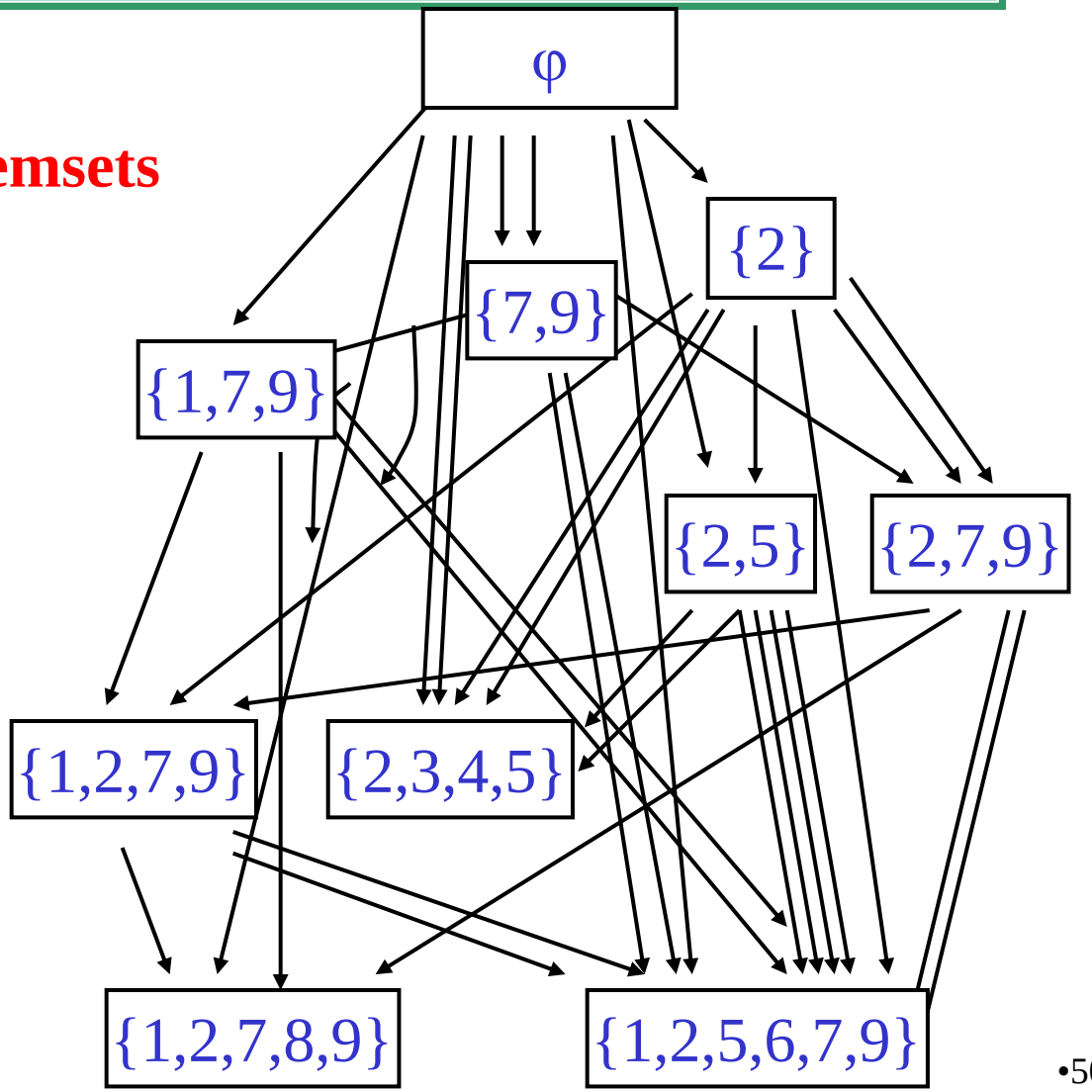
Naïve Closure Extension: Duplication!

- closure extension

➔ **DAG of closed itemsets**

T =
1,2,5,6,7,9
2,3,4,5
1,2,7,8,9
1,7,9
2,7,9
2

← closure extension



Prefix Preserving Closure Extension

- **Ensure any closed set is generated from a unique parent**

Def. **Closure tail** of a closed itemset P

$\Leftrightarrow$ the minimum j s.t. **closure** $(P \cap \{1,\ldots,j\}) = P$

Def. $H =$ **closure**$(P \cup \{i\})$ is a **PPC-extension** of P

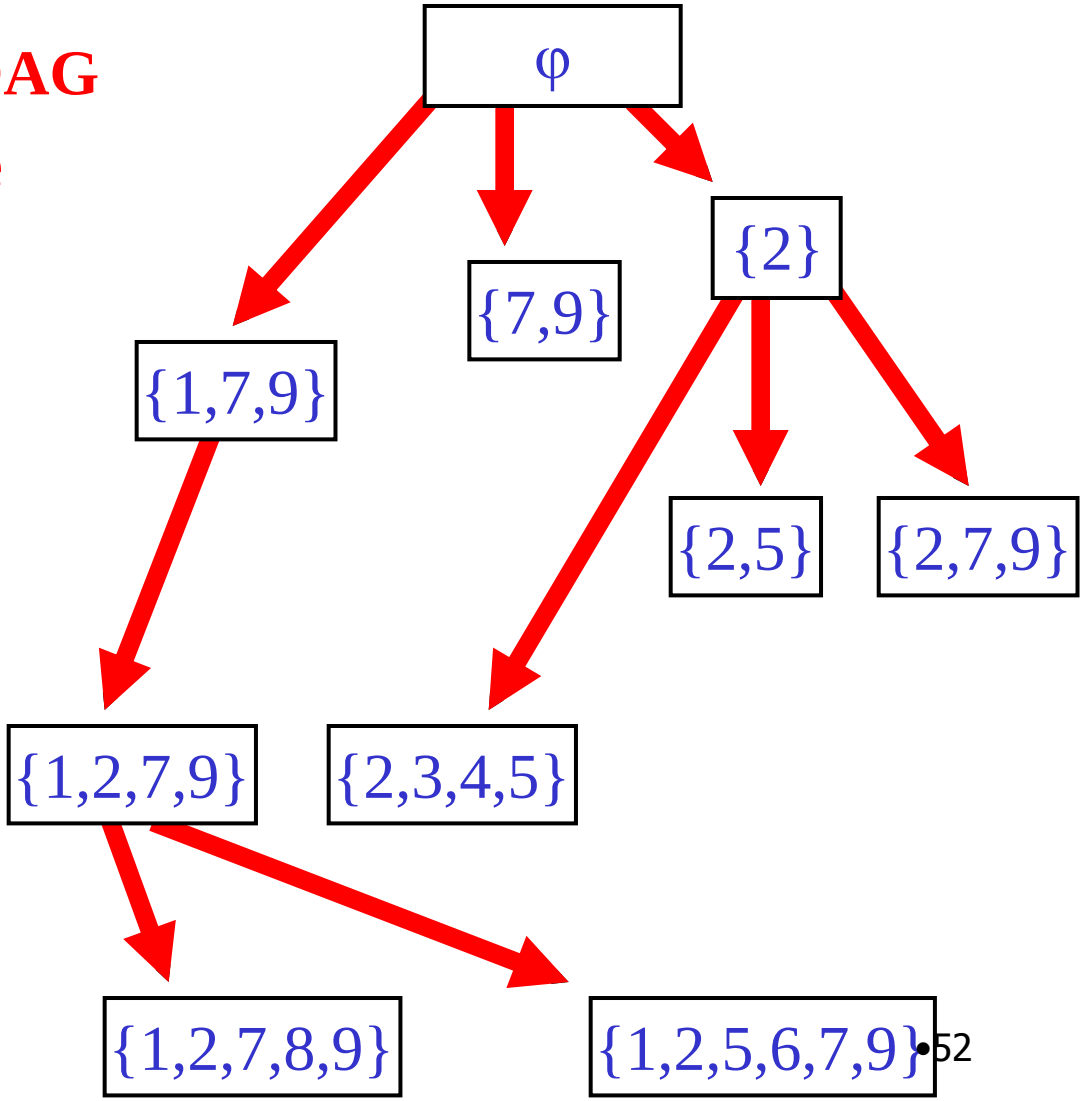
$\Leftrightarrow$ $i >$ **closure tail** and

$H \cap \{1,\ldots,i\text{-}1\} = P \cap \{1,\ldots,i\text{-}1\}$

Enumeration tree by PPC extension

- closure extension ➔ **DAG**
- ppc extension ➔ **tree**

T =
1,2,5,6,7,9
2,3,4,5
1,2,7,8,9
1,7,9
2,7,9
2



closure extension

ppc extension

Linear Delay in Pattern Size

(Uno, Uchida, Asai, Arimura, Discovery Science 2004)

Theorem : The algorithm **LCM** finds all frequent closed sets **X** appearing in a collection of a transaction database **D** in **O(lmn) time per closed set** in the total size of **D** without duplicates,

where ***l*** is the maximum length of transactions in **D**, and ***n*** is the total size of **D**, ***m*** is the size of pattern **X**.

Note: The output polynomial time complexity of Closed sets discovery is shown by [Makino et al. STACS2002]

Summary: Closed Itemset Mining

- Closure Extension: Jump from closed set to closed set
- LCM: Linear Delay
- Very fast in practice, too
  - Winner of FIMI'04 (Frequent Itemset Mining Implementation Workshop)
- Relation to clique enumeration (Arimura, Uno, SDM2009)

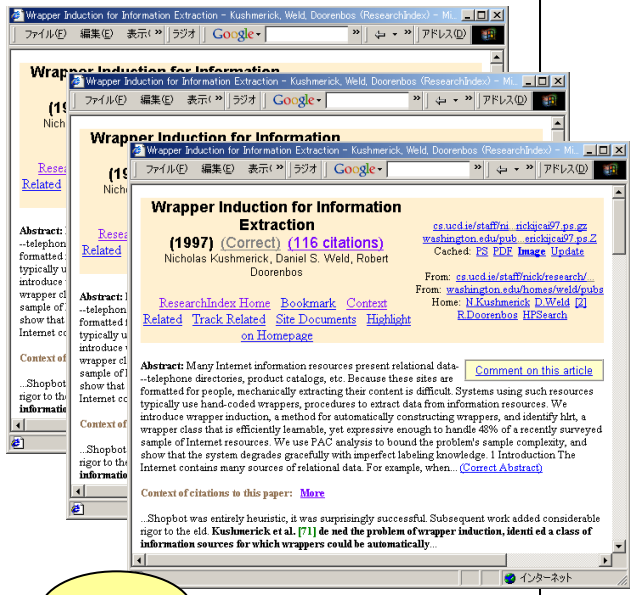
Part 4: Ordered Tree Mining

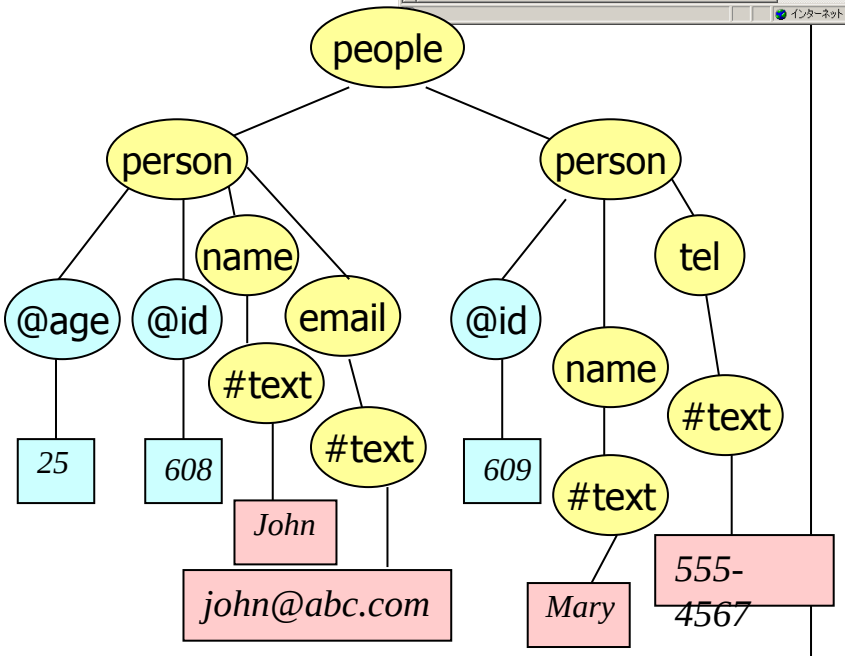
Frequent Ordered Tree Mining

- **Natural extension** of frequent itemset mining problem for trees
  - Finding all frequent substructure in a given collection of labeled trees
  - How to enumerate them without duplicates
- **Efficient DFS Algorithm**
  - FREQT [Asai, Arimura, SIAM DM2002]
  - TreeMiner [Zaki, ACM KDD2002]
  - Rightmost expansion technique

Labeled Ordered Trees

- Rooted:
- Ordered:
  - Siblings are ordered from left to right.
- Labeled
  - Each node has a label.
- A model of
  - HTML/XML
  - Hierarchical records
  - Dependency tree of natural language texts.



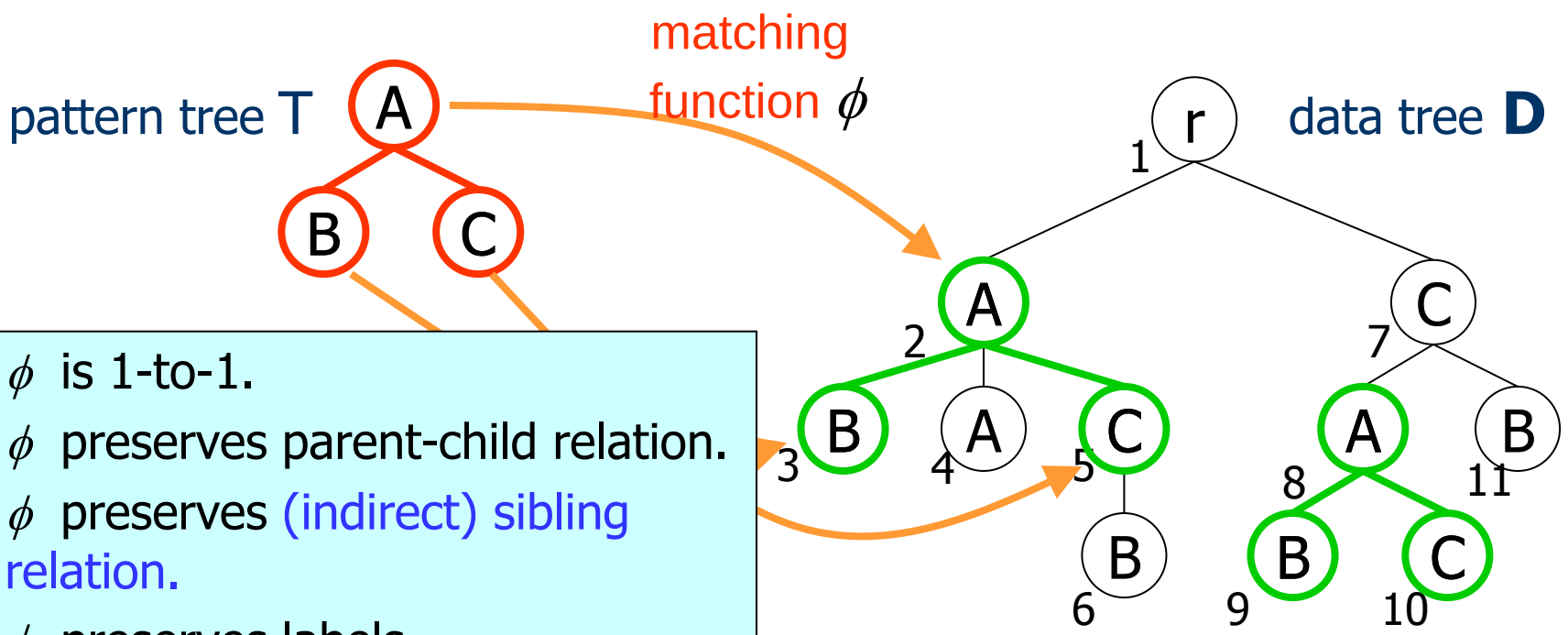


Matching between trees

Pattern tree ***T*** matches a data tree ***D***

(T occurs in D)

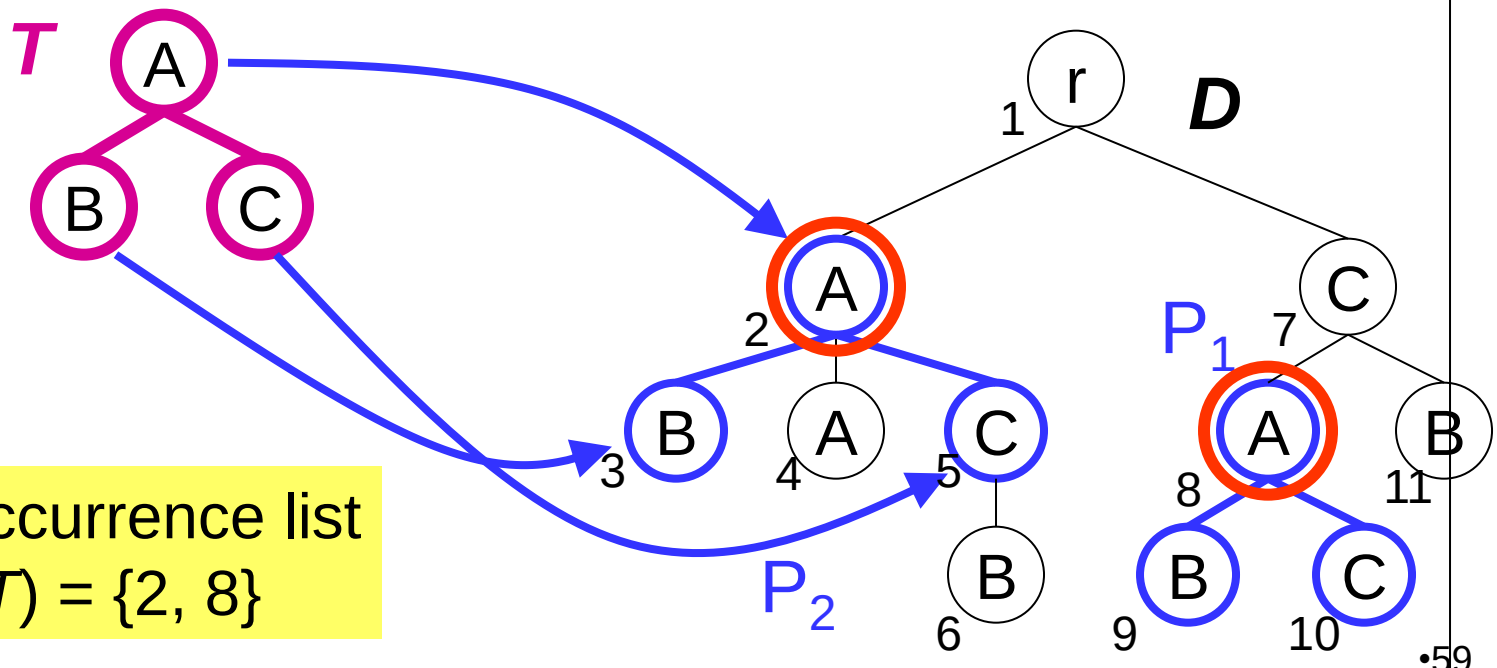
There is a matching function ϕ from ***T*** into ***D***.



- ϕ is 1-to-1.
- ϕ preserves parent-child relation.
- ϕ preserves (indirect) sibling relation.
- ϕ preserves labels.

Frequency of a pattern tree

- A root occurrence of pattern T:
  - The node to which the root of T maps by a matching function
- The frequency fr(T) = #root occurrences of T in D

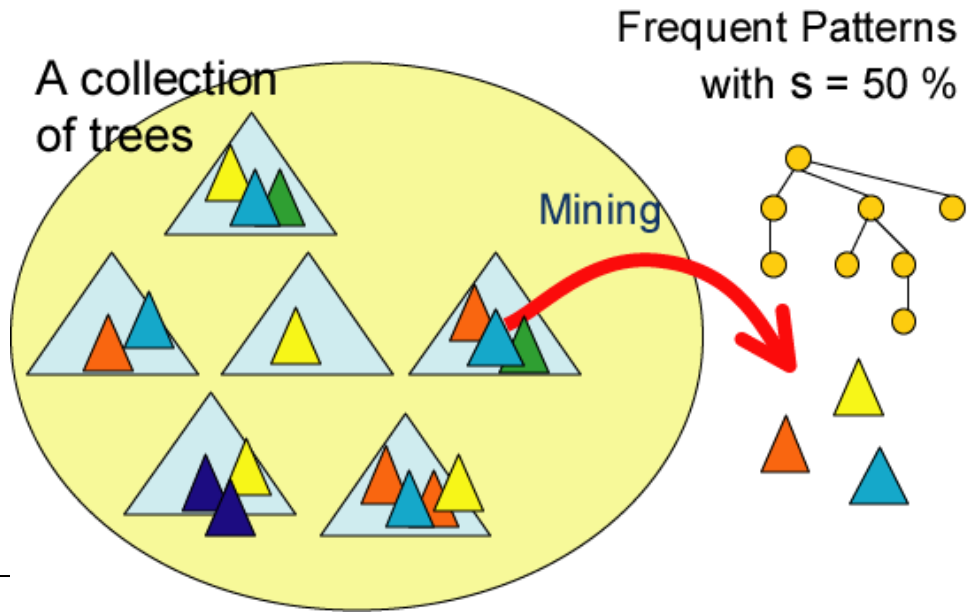


Root occurrence list
$Occ_D(T) = \{2, 8\}$

Frequent Tree Mining Problem

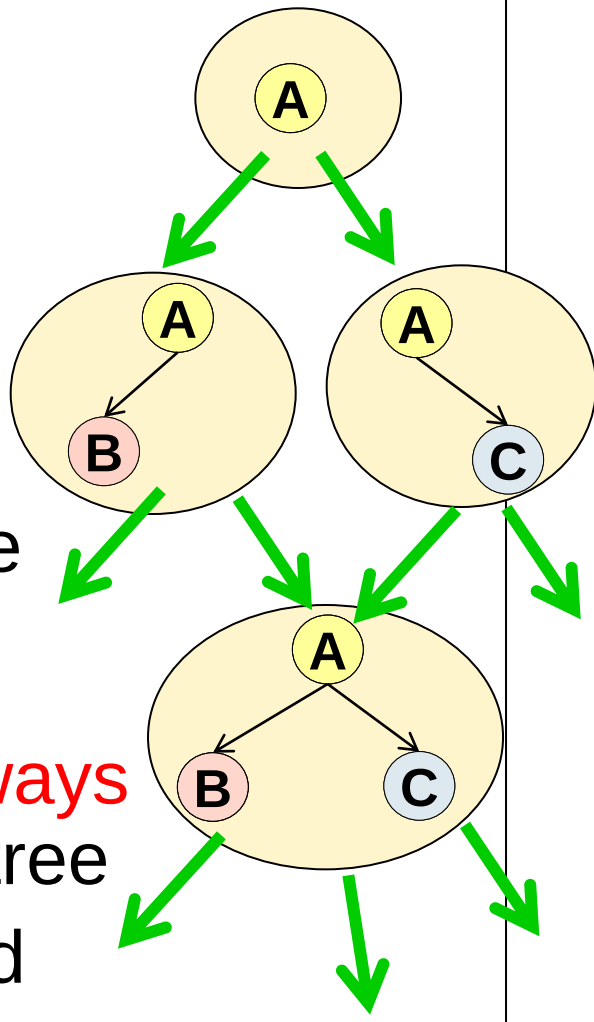
- **Given:** a colection **S** of labeled ordered trees and a minimum frequency threshold σ
- **Task:** Discover all frequent ordered trees in **S** (with frequency no less than σ) without duplicates

•A minimum frequency threshold (min-sup)
s = 50%



Key: How to enumerate ordered trees without duplicates?

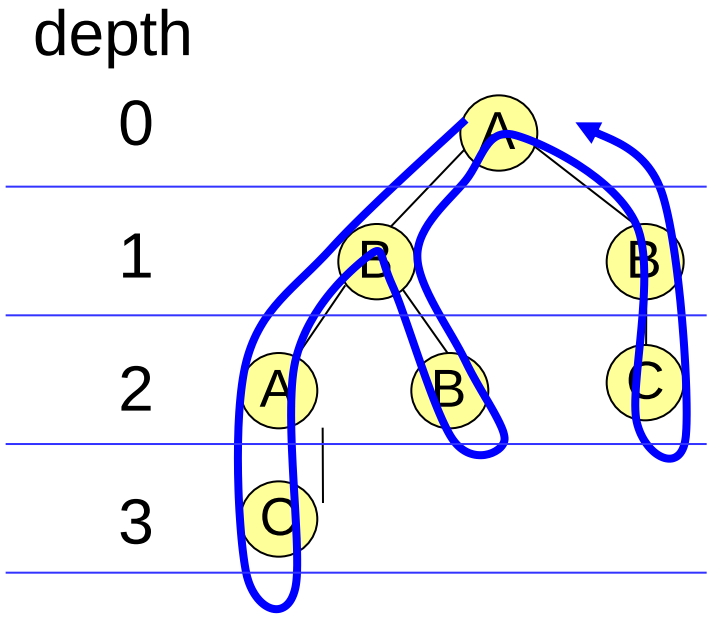
- **A naive algorithm**
  - Starting from the smallest tree
  - Grow a pattern tree by adding a new node one by one
- **Drawbacks**
  - Exponentially many different ways to generate the same pattern tree
  - Explicit duplication test needed
- **How to overcome this difficulty?**



An idea: DFS Code of Ordered Tree

- Depth-label sequence in the preorder traversal (depth first search)

$$S = ((d(v_1), l(v_1)) , \dots , (d(v_k), l(v_k))$$



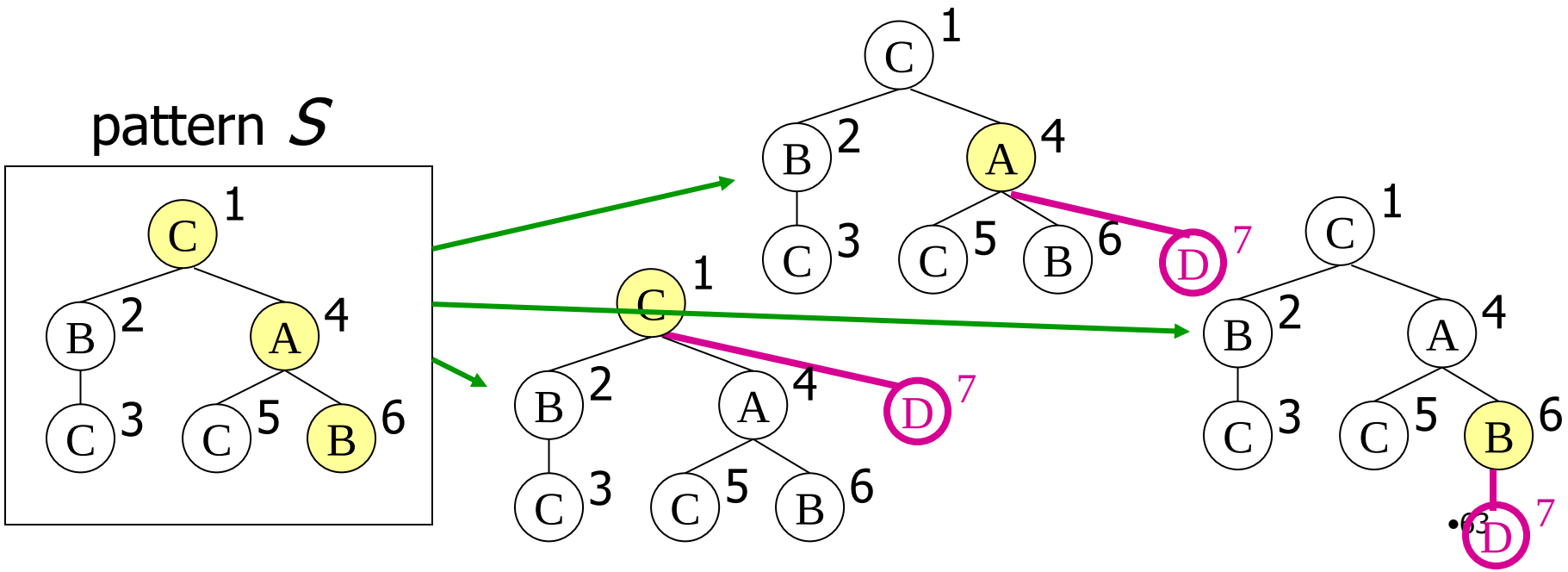
DFS code

id	1	2	3	4	5	6	7
seq	**0A**	**1B**	**2A**	**3C**	**2B**	**1B**	**2C**

Rightmost expansion

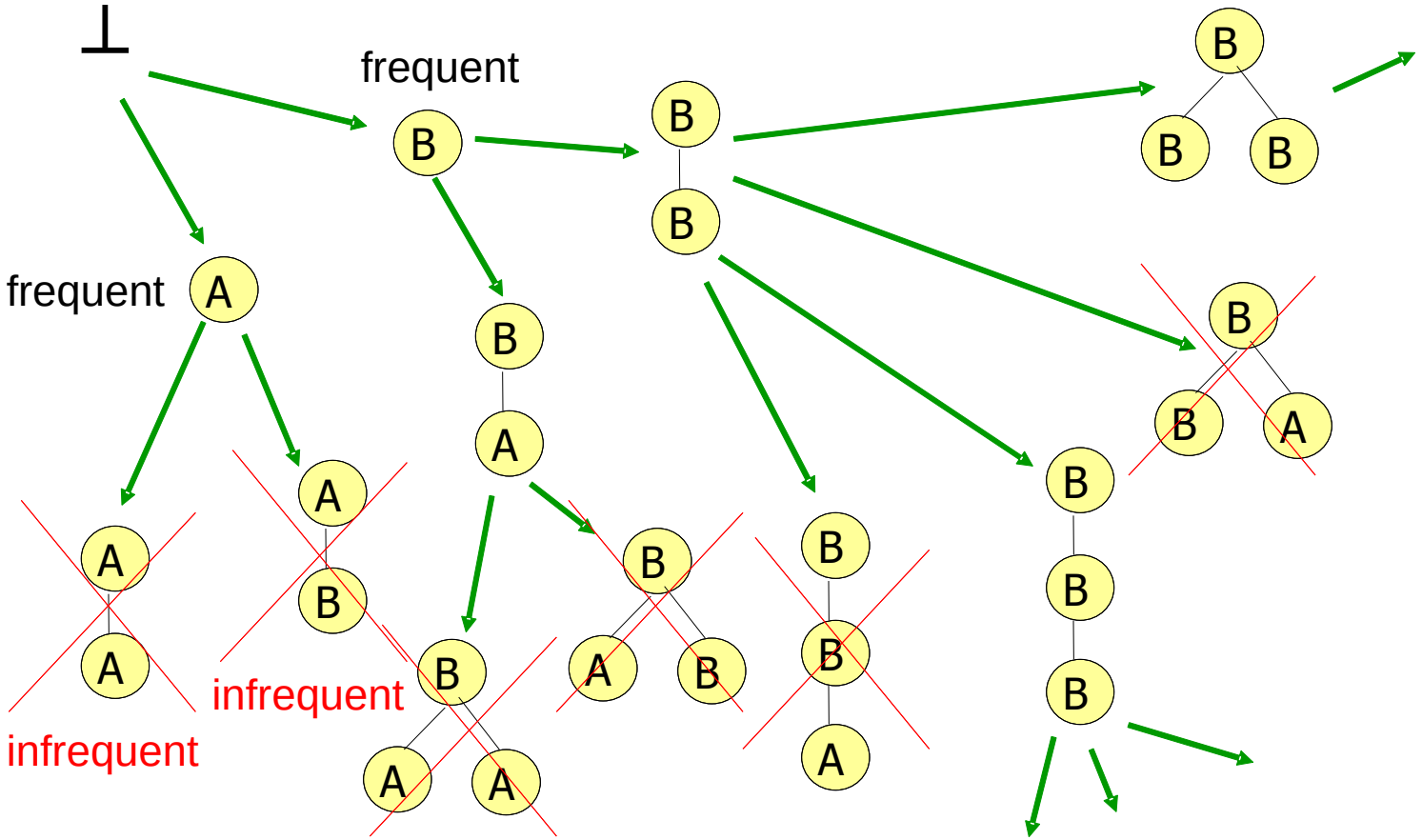
- Extending the DFS Code = Attaching a new node on the rightmost branch

  $(d_1,l_1),\ldots,(d_n,l_n)$, (d_{n+1},l_{n+1})



Searching frequent ordered trees

- Enumerate all frequent ordered trees by backtracking
- Tree extended only by rightmost extension = No duplication



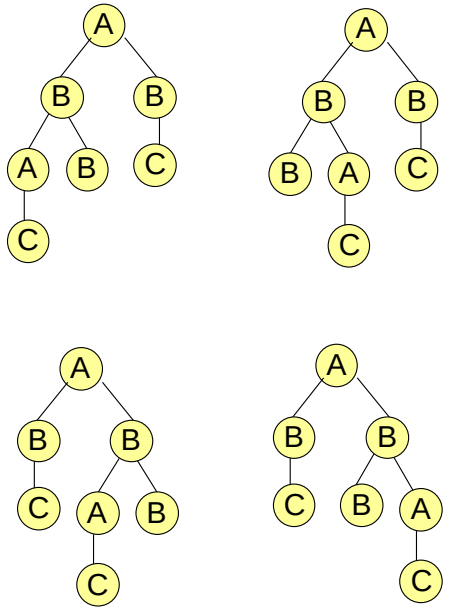
Summary: Ordered tree mining

- Convert tree to a string (DFS Code)
- Adding element to the code = Rightmost extension
- It was relatively easy because nodes are ordered
  - How about unordered case?

Part 5: Unordered Tree Mining

Frequent Unordered Tree Mining

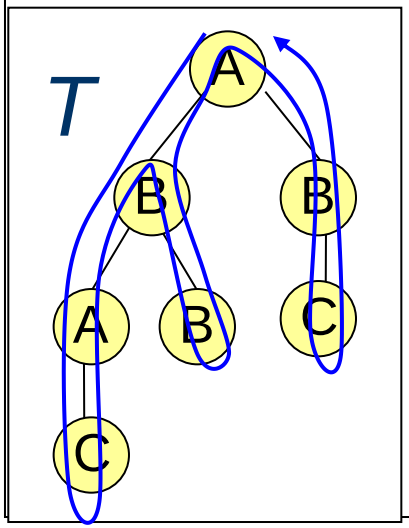
- Unordered trees: Non-trivial subclass of general graphs
- Problem: Exponentially many isomorphic trees
- Efficient DFS Algorithm
  - Unot [Asai, Arimura, DS'03]
  - NK [Nijssen, Kok, MGTS'03]



Canonical Ordered Representation

- Given ordering among siblings, depth-first search (DFS) code is defined

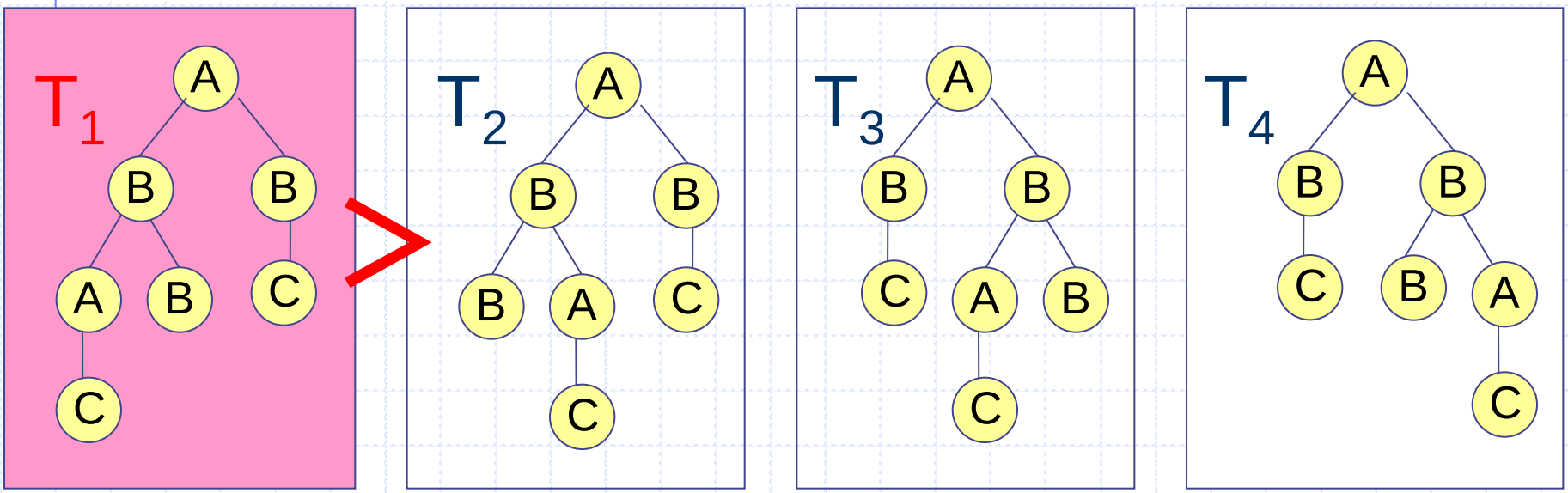
Code(T) = ((depth(v_1), label(v_1)) , … , (depth(v_k), label(v_k))



Code(T) = (0A,1B,2A,3C,2B,1B,2C)

Canonical representation

Ordered tree *T* with **lexicographically maximum code**



(0A,1B,2A,3C,2B,1B,2C) (0A,1B,2B,2A,3C,1B,2C) (0A,1B,2C,1B,2A,3C,2B) (0A,1B,2C,1B,2B,2A,3C)

Left Heavy Condition (Nakano and Uno, 2002)

- T(v): subtree rooted on v
- Ordered tree is canonical ***if and only if***

$$\mathrm{Code}(T(v1)) \geqq \mathrm{Code}(T(v2))$$

for any pair of sibling nodes v1 (left) and v2 (right)

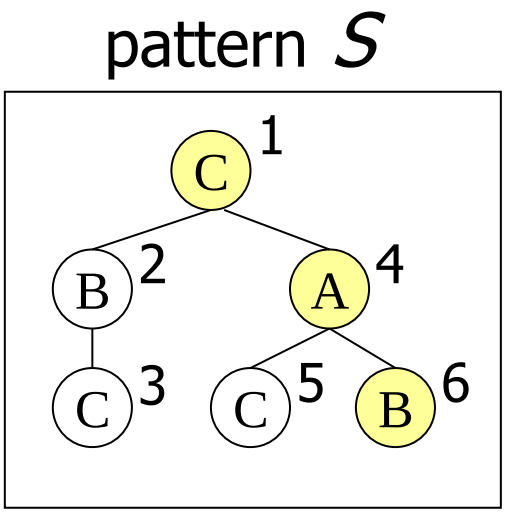
Reduction Map

- How to define parent from child in the enumeration tree
- Generate canonical tree of size k-1 from canonical tree of size k
- Remove the last element of DFS Code

Code(T) = (0A,1B,2A,3C,2B,1B,2C)

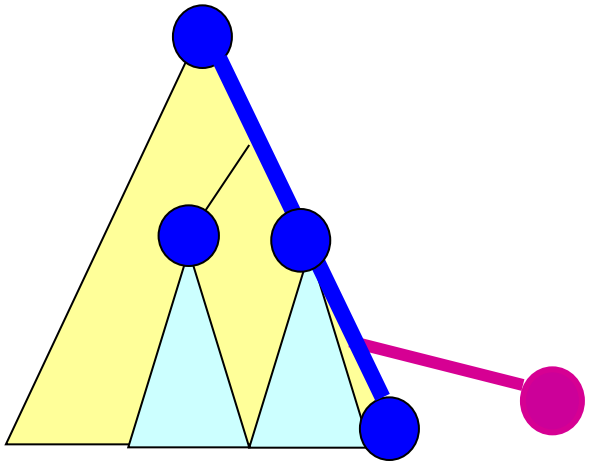
Children Generation

- Generate children candidates by rightmost extension
- Check the maximality of candidate based on left heavy property
- Discard if not maximal



Maximality Check by Left Heavy Property

- Code of left subtree must be larger than that of right subtree
- Check only rightmost sibling and second rightmost sibling



Complexity of UNOT

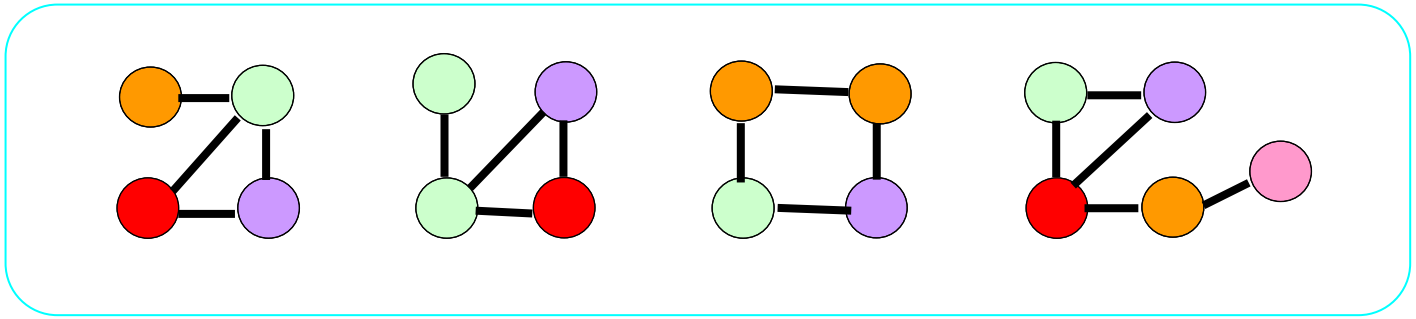
- Delay per pattern $O(kb^2\,mn)$
- k: pattern size
- b: branching factor of the data tree
- m: size of data tree
- n: database size

Summary: Mining Unordered Tree

- The following three elements are necessary for a mining algorithm
  - Canonical Representation
  - Reduction Map
  - Children Generation including Maximality Check
- Backtracking on the resulting enumeration tree

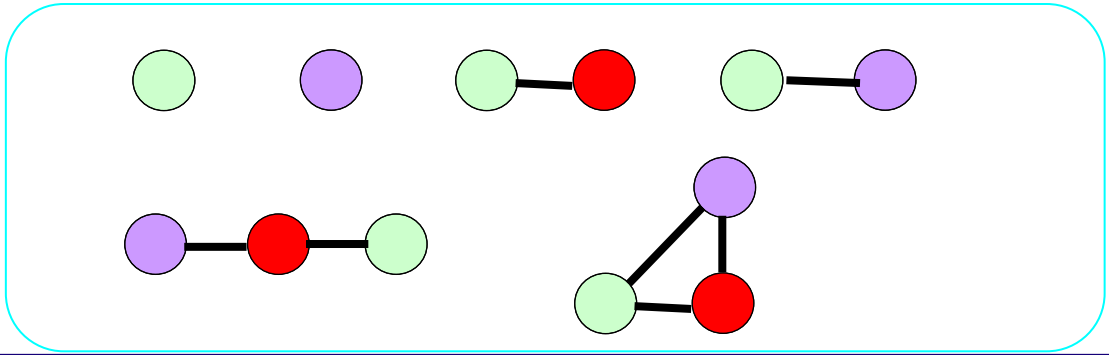
Part 6: Graph Mining

Frequent Subgraph Mining



Graph Database

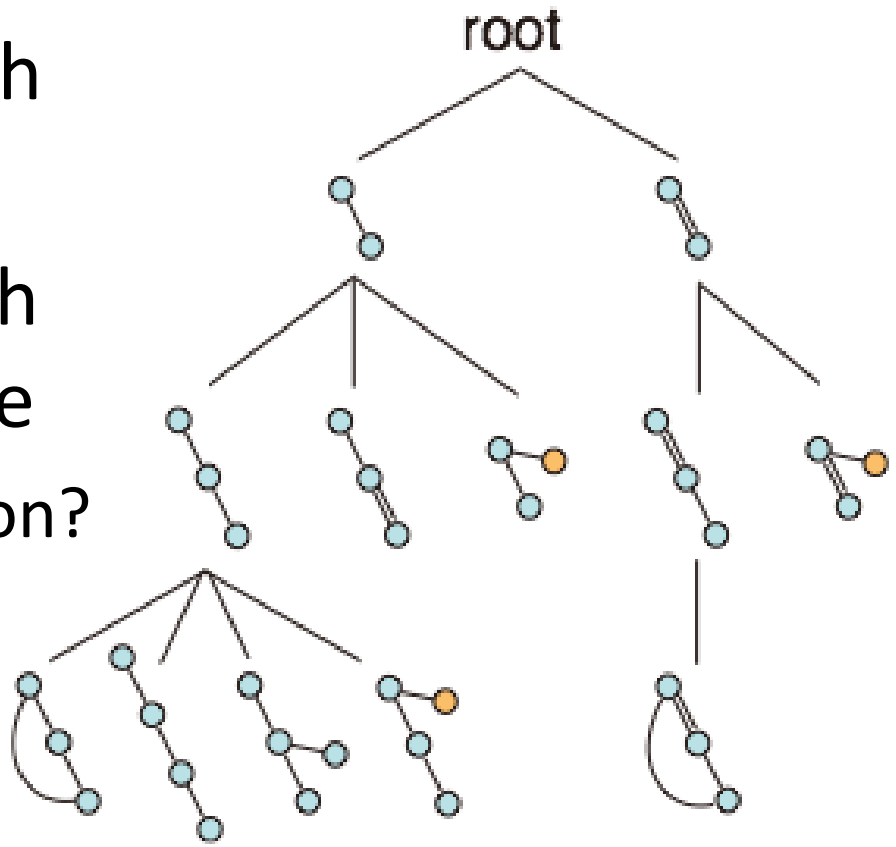
- Enumerate all subgraphs occurring more than 3 times



Patterns

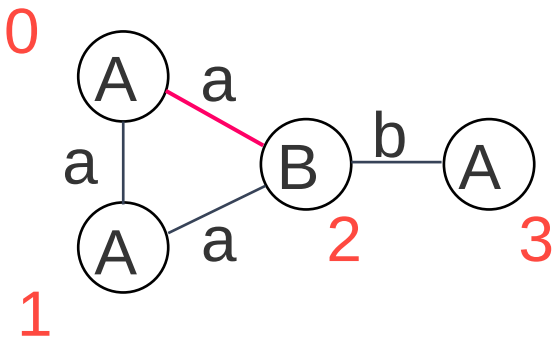
Gspan (Yan and Han, 2002)

- Most widely used graph mining algorithm
- Can be interpreted with reverse search principle
  - Canonical representation?
  - Reduction map?
  - Children generation?



DFS Code for Graph

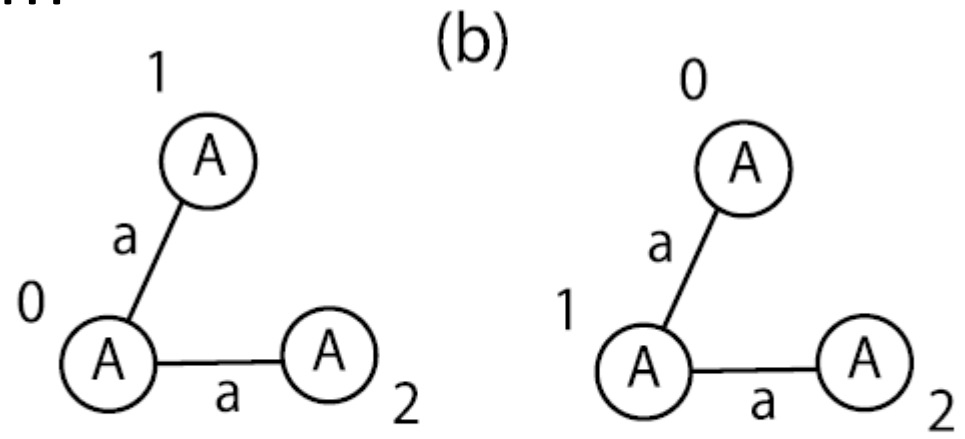
- Depth first search and preorder node labeling
- (src, dest, src_label, edge_label, dest_label)
- Some edges not traversed
  - backward edge (dest < src)
- Elements sorted in the code



{(0,1,A,a,A), (1,2,A,a,B),
(2,0,B,a,A), (2,3,B,b,A)}

Canonical Representation

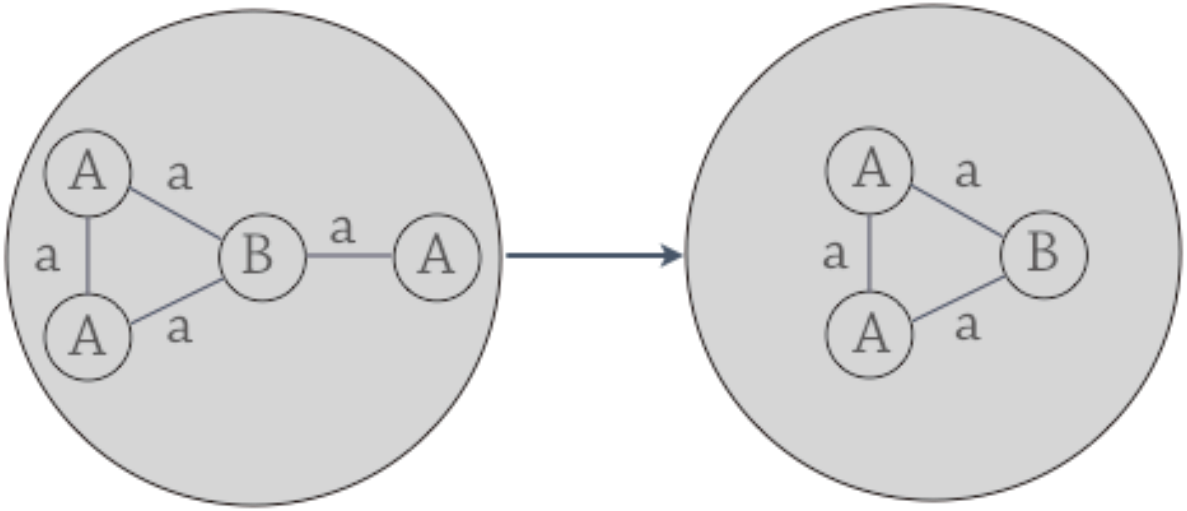
- Multiple DFS codes: different starting point and children ordering
- Minimum DFS Code: Lexicographically Minimum



((0,1),A,a,A), ((0,2),A,a,A) ((0,1),A,a,A), ((1,2),A,a,A)

Reduction Map

- Removing the tail of minimum DFS code preserves minimality

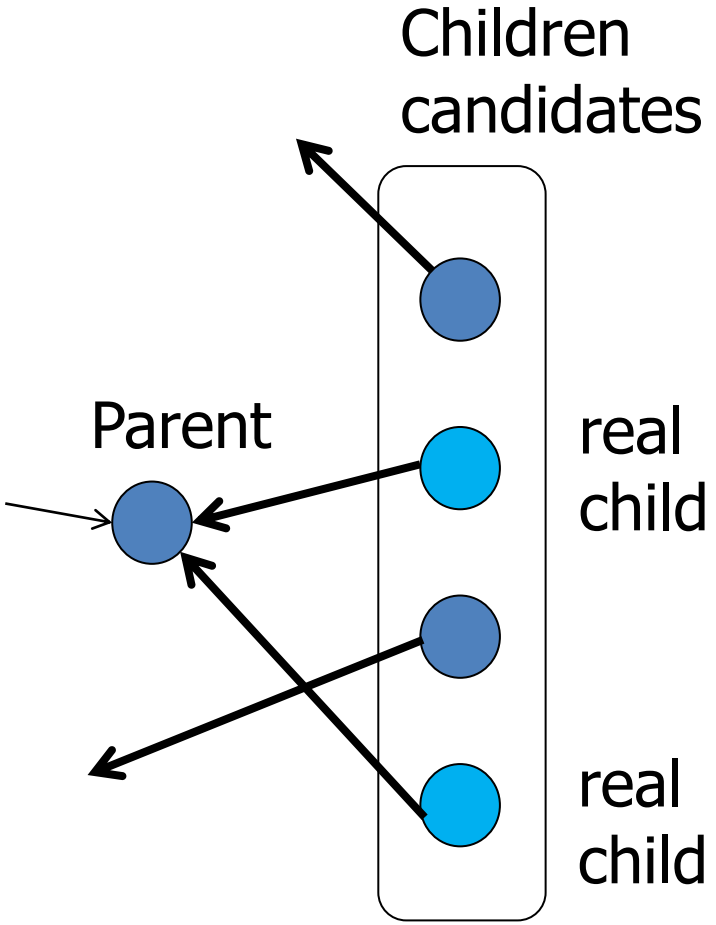


min DFS:
{e1, e2, e3, e4}

{e1, e2, e3}

Children Generation

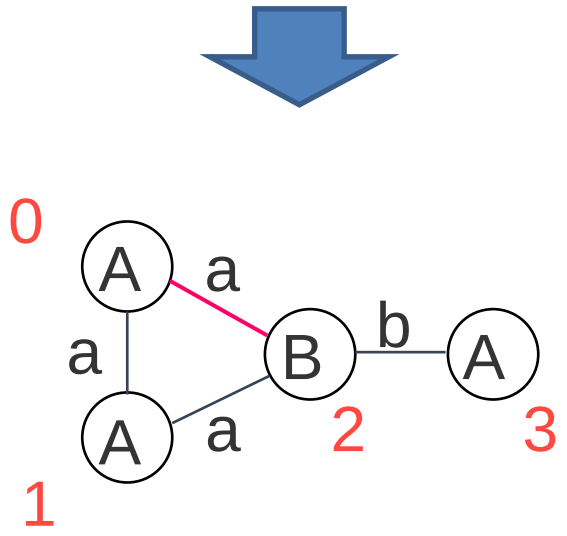
- Create candidates by adding an element to DFS code
- Check if each candidate is minimum
- If not, remove it



Minimality Check

- Reconstruct the graph from DFS Code
- Derive the minimum DFS Code by trying all DFSs
  - Speed up by traversing minimal label only
- If the minimal code is not identical to the original, prune it

{(0,1,A,a,A), (1,2,A,a,B), (2,0,B,a,A), (2,3,B,b,A)}



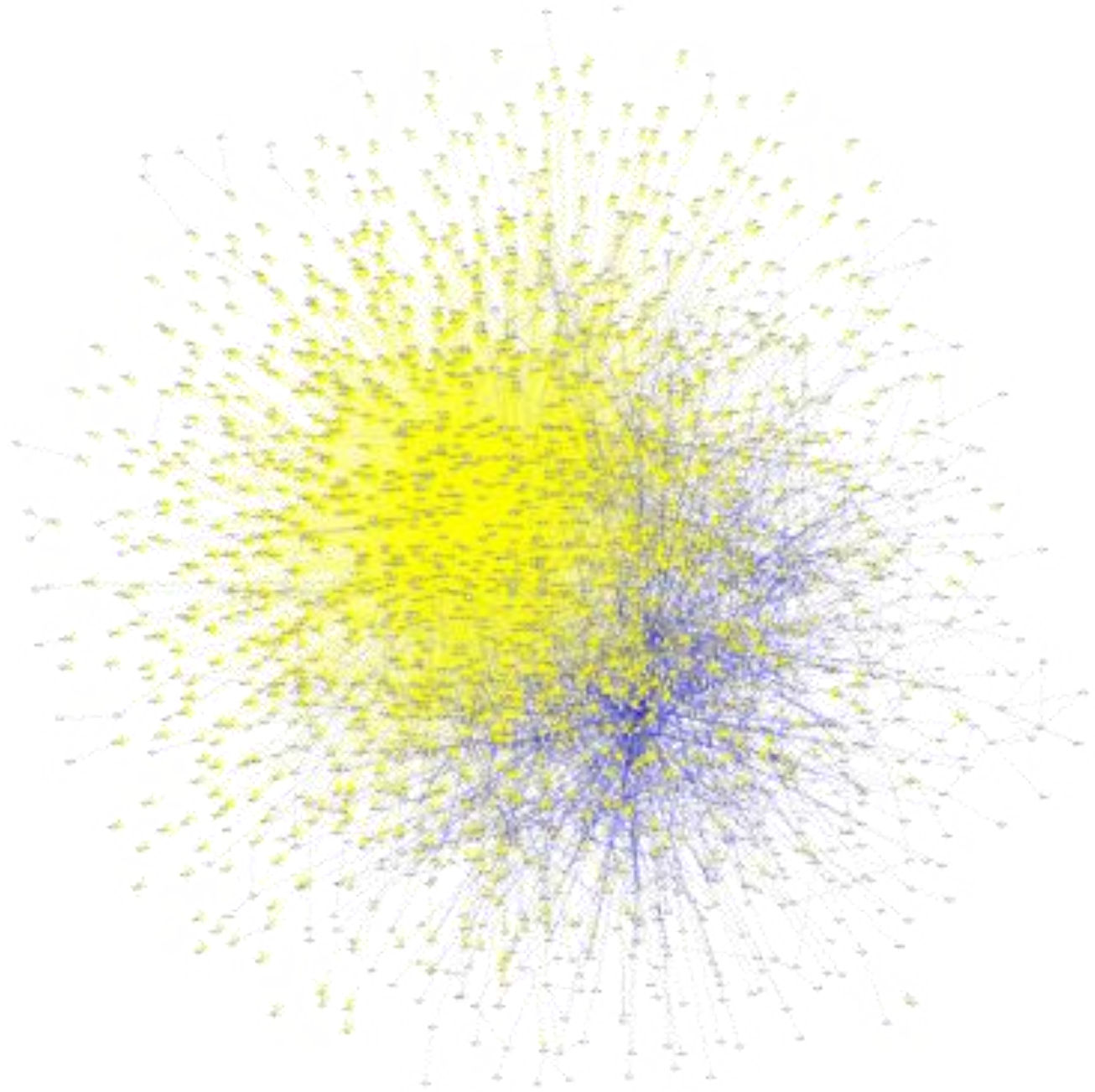
Summary: Graph Mining

- gSpan is a typical example of reverse search
- Not explained: Closed tree mining, Closed Graph mining
- Delay exponential to pattern size
  - It cannot be avoided due to NP-hardness of graph isomorphism
  - Yet it scales to millions for sparse molecular graphs
- Applications covered in next chapter

Part7: Dense Module Enumeration

Biological Motivation

- Most cellular processes performed by multi-component protein complexes
- Increasing amount of experimental protein interaction data available
- Our approach
  - Predict complexes (modules) from protein interaction network
  - Exploit additional information given by gene expression data, evolutionary conservation, phenotypic profiles etc.



Protein interaction networks

- Node: Proteins
- Edge: Physical interaction of two proteins

- Challenge 1: False negative edges
  - Go beyond clique search!
- Challenge 2: False positive edges
  - Assign confidence scores to edges

- Find node sets with high density of high confidence edges

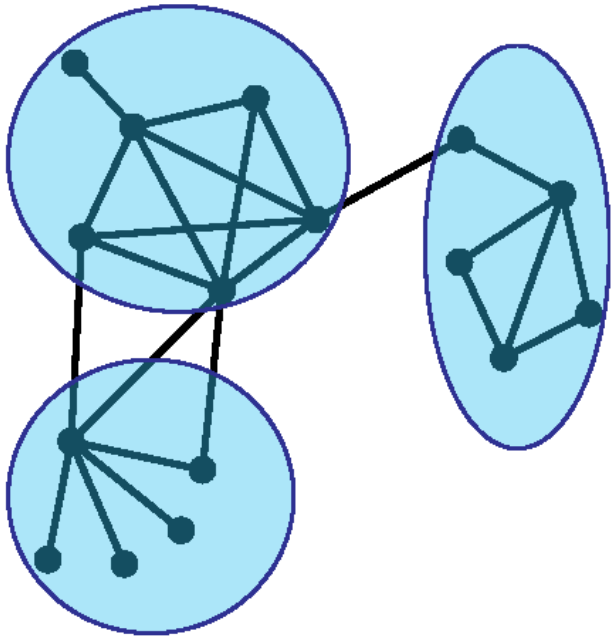
Module Discovery

- Previous work
  - Clique percolation [Palla et al., 2005]
  - Partitioning
    - Hierarchical clustering [Girvan and Newman, 2001]
    - Flow Simulation [Krogan et al., 2006]
    - Spectral methods [Newman, 2006]
  - Heuristic Local Search [Bader and Hogue, 2003]

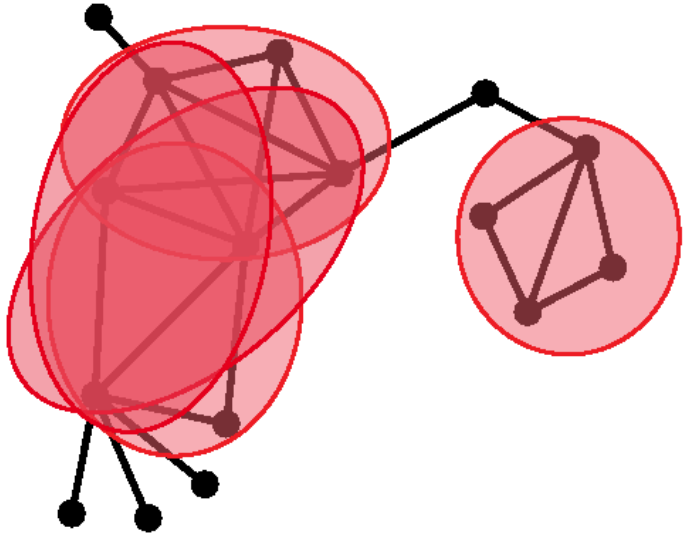
- Our approach
  - Exhaustively enumerate all dense subgraphs efficiently

Motivation for Enumeration Approach

- Detects overlapping modules
- Allows to specify minimum density for outcoming modules
- Outputs all modules satisfying the density threshold



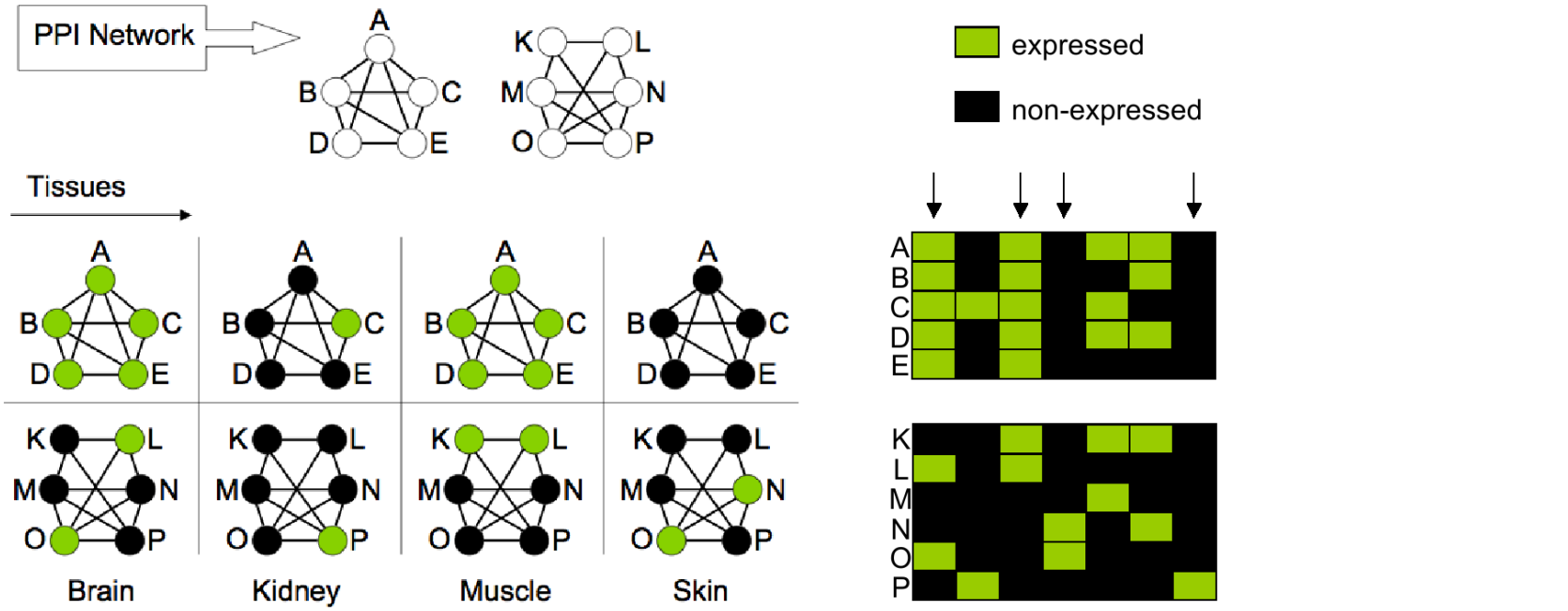
Partitioning



Enumeration

Differential Expression Criterion

- Incorporation of gene expression
  - Presence of proteins depends on cell type
- Additional Criterion for modules
  - e_1: Num of conditions where whole module expressed
  - e_0: Num of conditions where whole module not expressed
  - Fix minimum values for both quantities



Problem Formalization

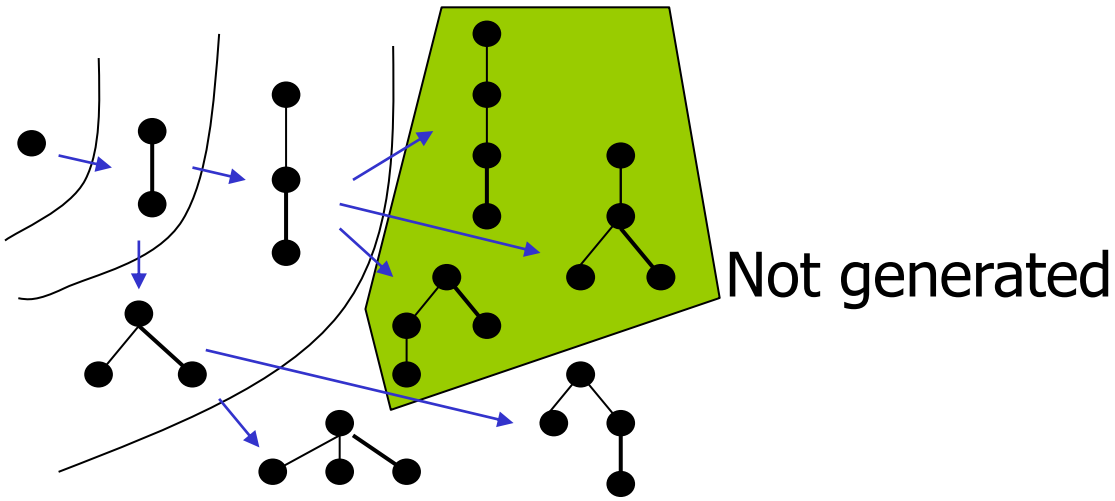
- Interaction network: $G = (V, E(V))$
- Edge weights: $0 \leq w(\{u, v\}) \leq 1$
- Density of $U \subset V$:

$$d(U) = \frac{\sum_{\{u,v\} \in E(U)} w(\{u, v\})}{|U|(|U| - 1)/2}$$

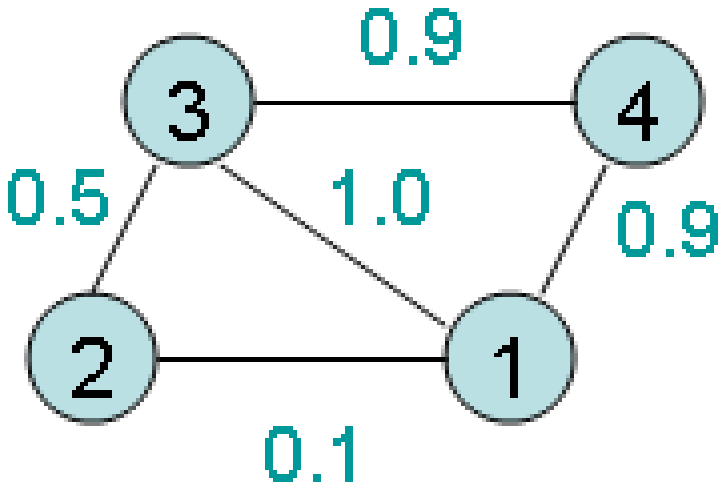
- Find all $U \subset V$ with $d(U) \geq \theta$, $e_1(U) \geq n_1$, and $e_0(U) \geq n_0$

Typical Enumeration Algorithms

- Itemset mining, graph mining etc.
- Enumerate all entities whose frequency >= 10
- Set up a search tree
- Tree Pruning by anti-monotonicity
  - An entity's frequency is always smaller than that of sub-entity



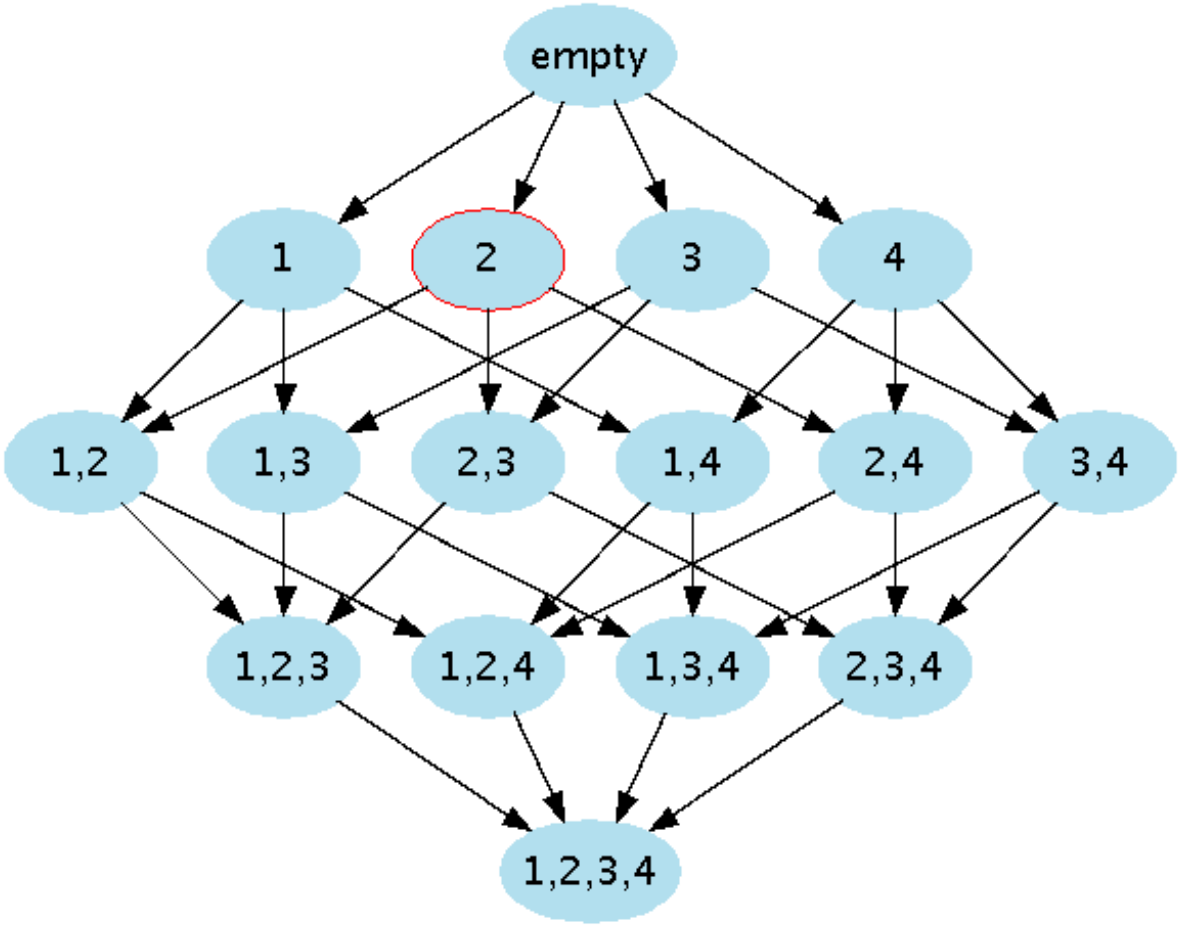
Network Example



Density of Modules

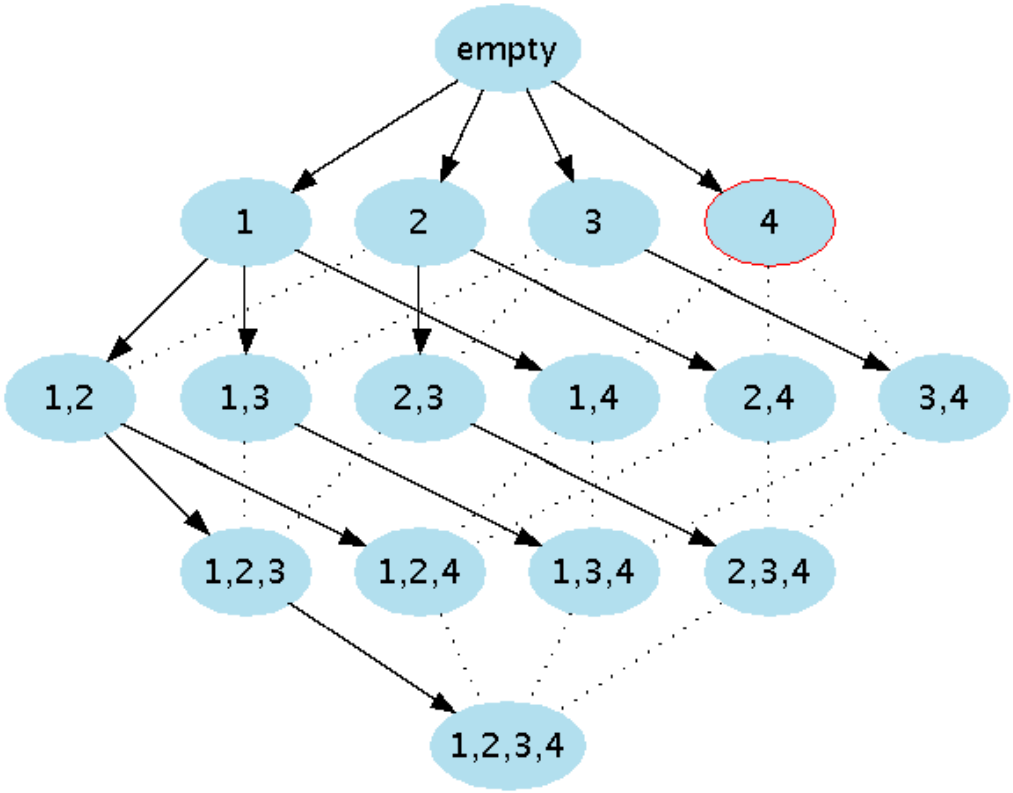
1,2,3	0.5
1,3,4	0.9
2,3,4	0.5
1,2,4	0.3

Graph-shaped Search Space of Modules



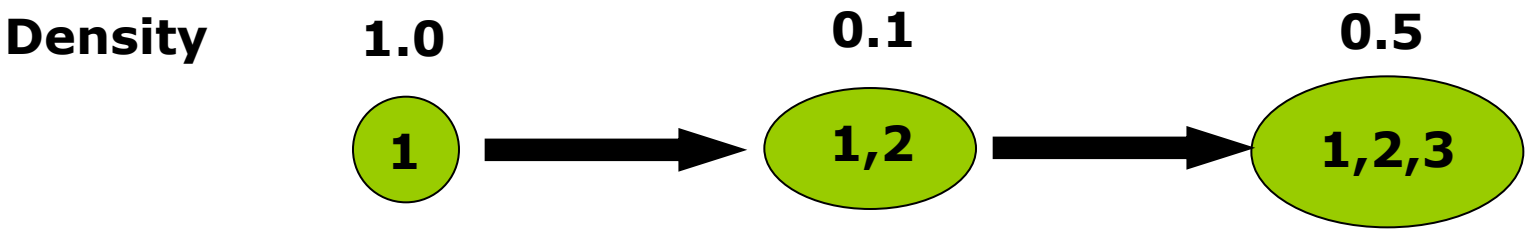
Choosing a search tree

- For efficient search, a search tree is needed
- There are many possible search trees
- Default: Lexicographical ordering



Density is not a monotonic criterion

- Subset of dense set is not necessarily dense
- Density does not decrease monotonically on a path
  - Pruning Impossible



Question

- Is it possible to make a search tree such that density decreases monotonically?

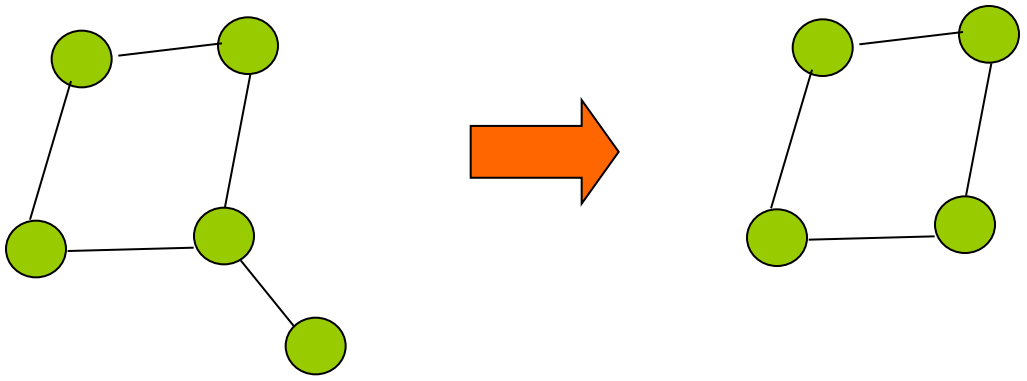
Question

- Is it possible to make a search tree such that density decreases monotonically?

- YES!
  - Use Reverse Search (Avis and Fukuda 1993)

Reverse search (Avis and Fukuda, 1993)

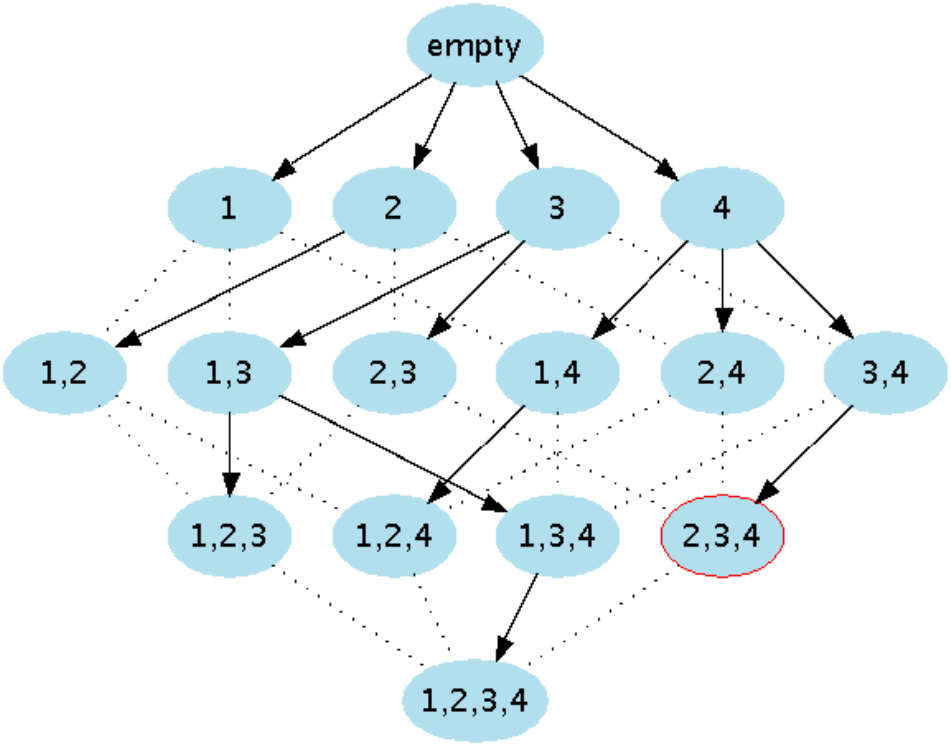
- Specify a search tree in the graph-shaped search space
- Reduction Mapping
  - Rule to generate a parent from a child
  - **Remove the node with the smallest degree**
  - Density always increase by the removal



CHILD **PARENT**

Search Tree is uniquely specified by the reduction mapping

- Condition: Every node should converge to the root node by applying the reduction mapping repeatedly



Enumeration algorithm by reverse search

- A set of children is generated from a parent node
- Try every possible children, and choose the ones satisfying the reduction mapping
- Prune if no children exist

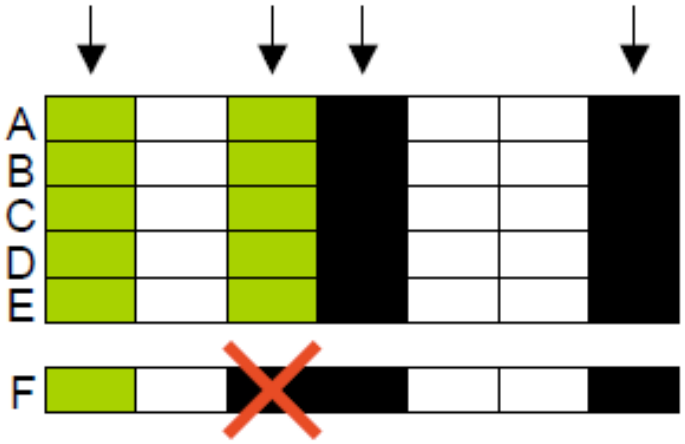
Constraint Integration

- Differential expression constraint

$$e_1(U) \geq n_1,\ e_0(U) \geq n_0$$

- Monotonicity: e_0 and e_1 decrease with extension of U

Module	e_1	e_0
ABCDE	2	2
ABCDEF	1	2



- Can be used for extra pruning without difficulty

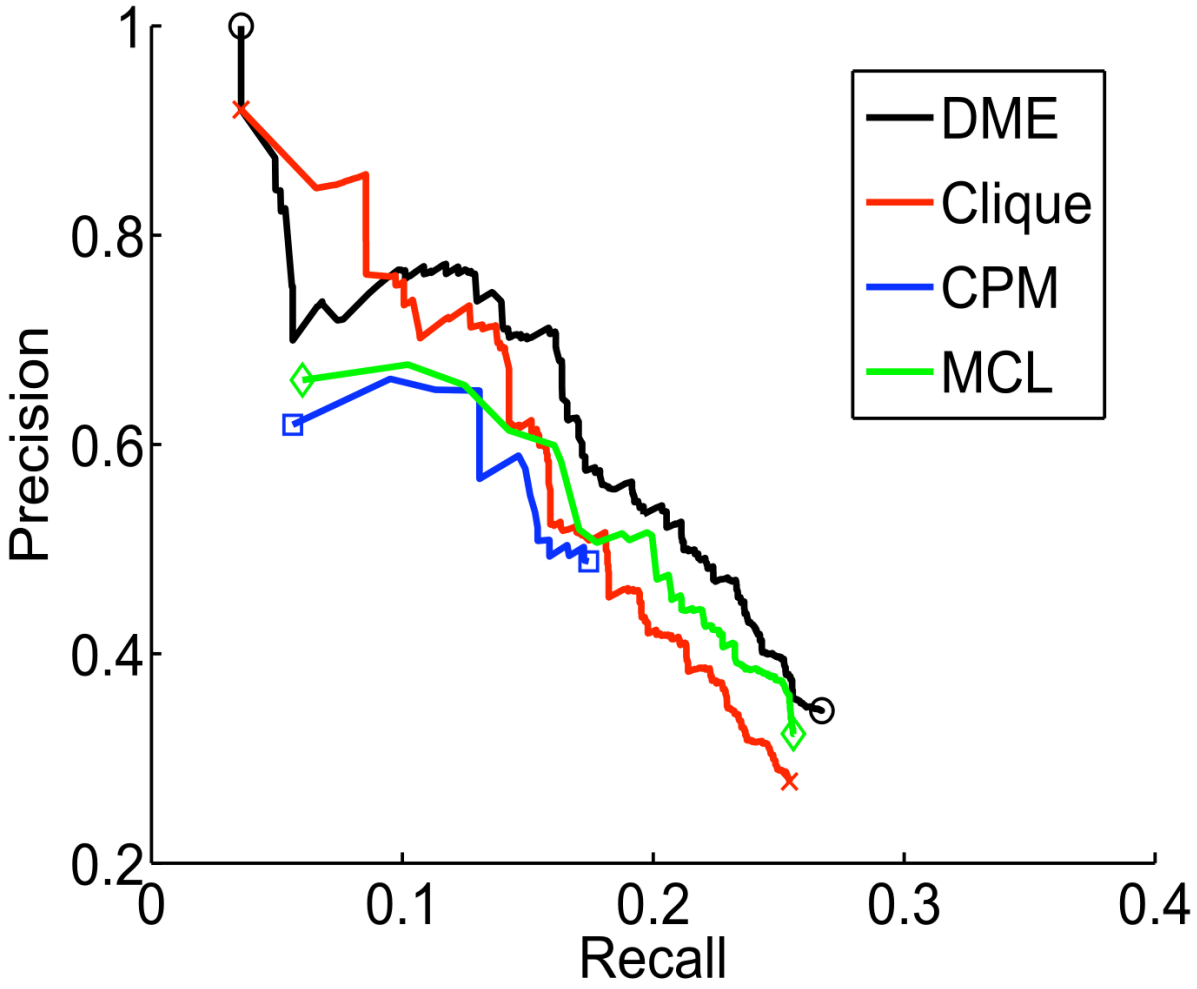
Statistical Significance of a module

- k : The number of nodes in the module
- ρ : Density of the module
- $m_k(\rho)$: The number of modules of size k with density at least ρ
- Probability of random selection making a denser module (p-value)

$$p = m_k(\rho) / \binom{n}{k}$$

Benchmarking in yeast complex discovery

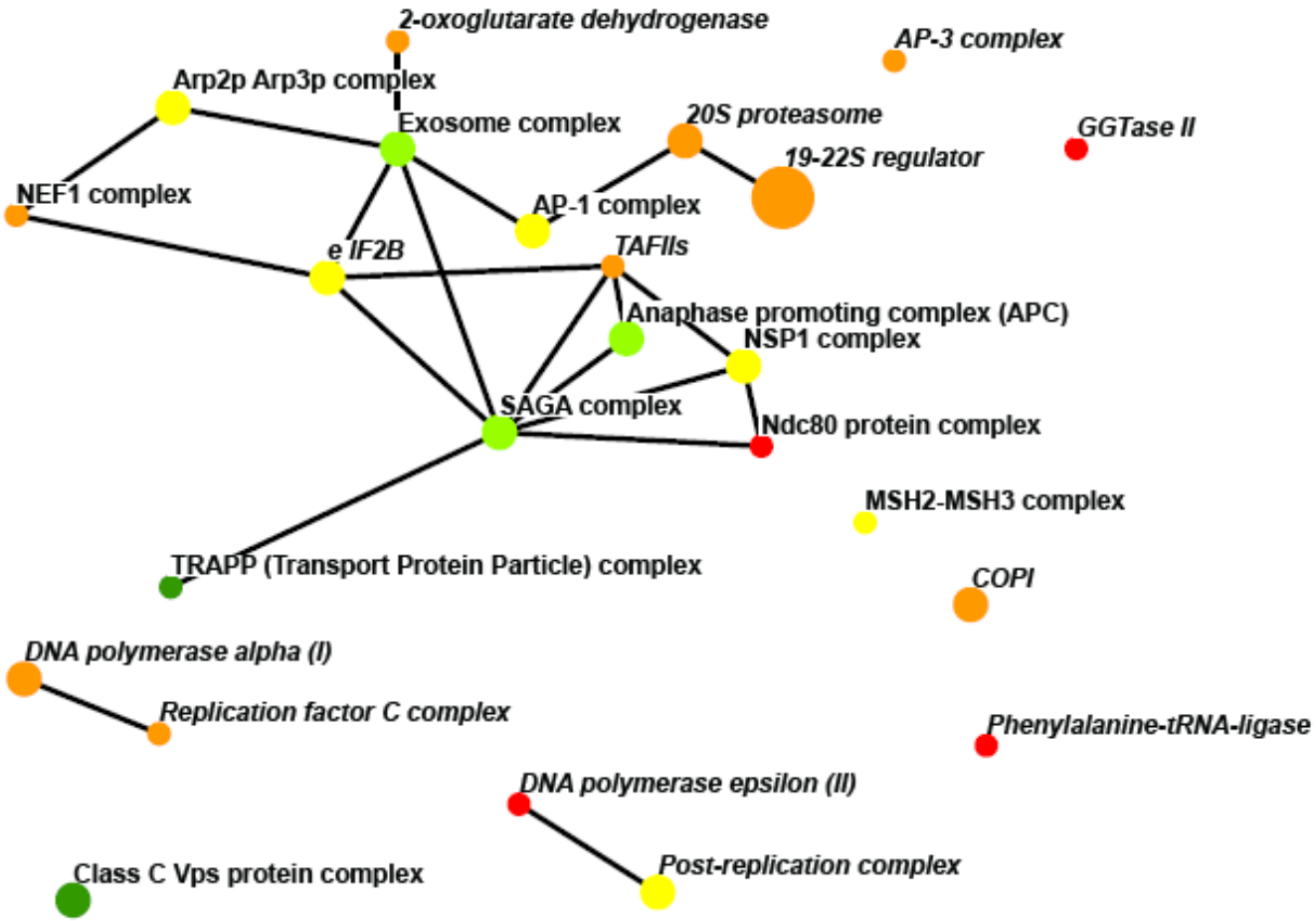
- Combined interactions from CYGD-Mpact and DIP
- Interactions among 3559 nodes
- Confidence weights on edges due to (Jansen, 2003)
- Methods in comparison
  - Clique detection (Clique)
  - Clique Parcolation Method (CPM)
  - Markov Clustering
- Modules compared with MIPS complexes

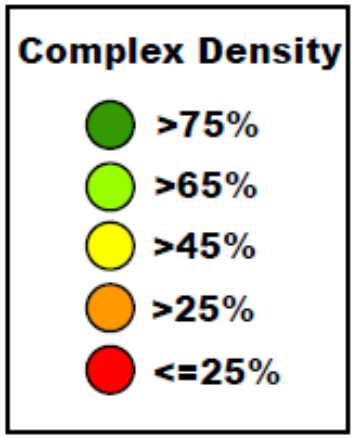

1
0.8
0.6
0.4
0.2
Precision
0
0.1
0.2
0.3
0.4
Recall
DME
Clique
CPM
MCL

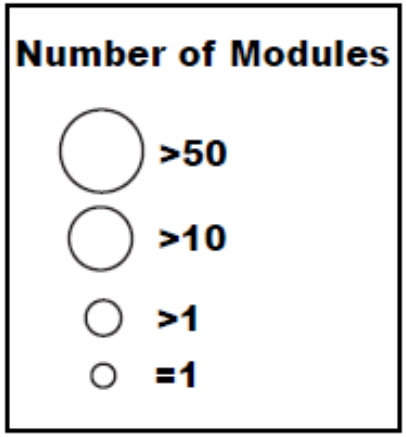
Evolutionary Conserved Yeast Modules

- Use ortholog profiles (10 species, InParanoid)
- Density >= 50%, at least three orthologs
- 1917 modules in 30 minutes
- Recovered evolutionary conserved complexes
  - 20S proteasome
  - 19S/22S regulator
  - COPI vesicle coat complex
  - DNA polymerase I and II subunits
  - Translation initiation factor eIF2B complex
- They could not be recovered by simple DME due to low density

MIPS Complexes discovered by DME (Conserved in Evolution)



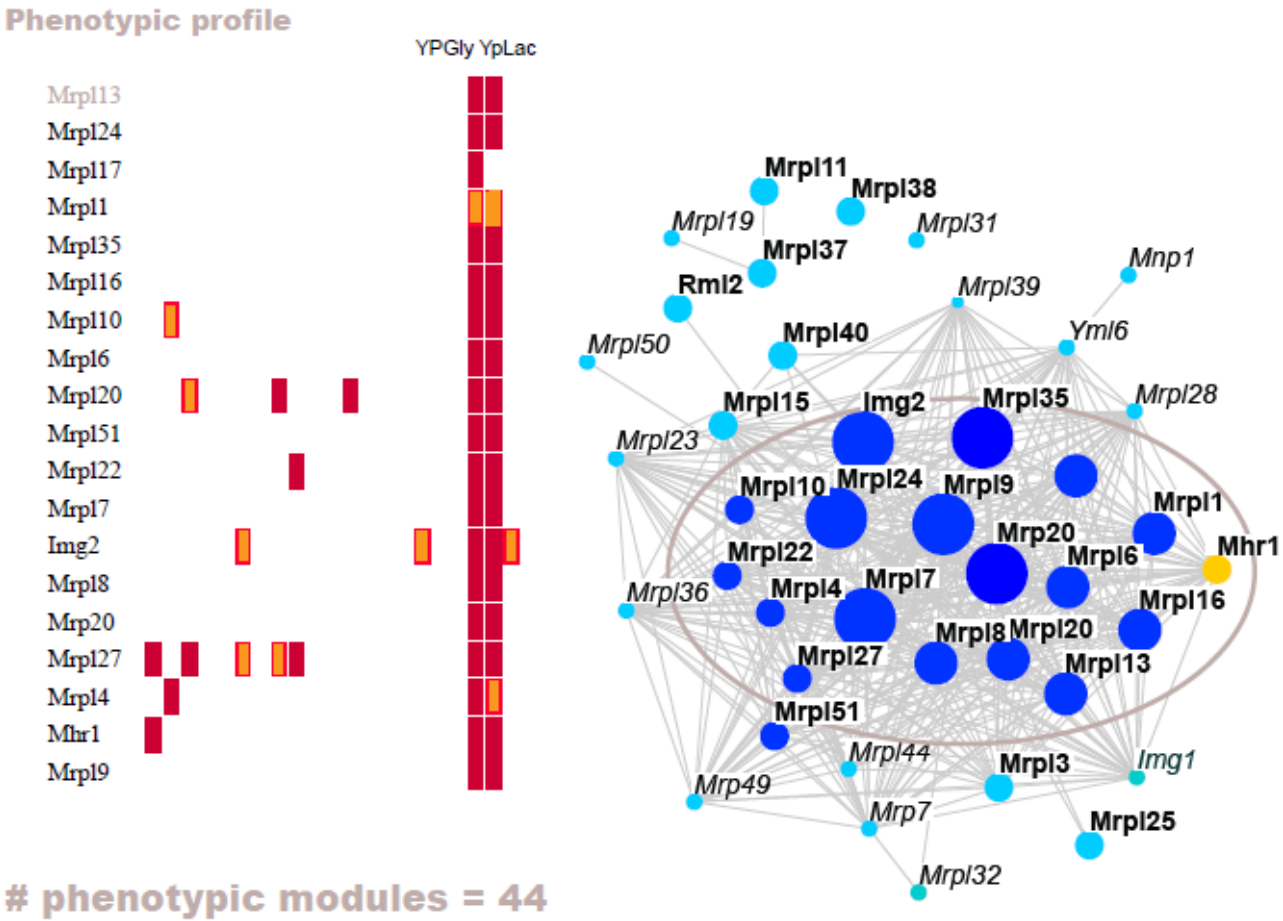




Phenotype-associated yeast modules

- Use growth phenotypic profiles (21 conditions, Dudley et al, 2005)
- Growth defect in at least one condition
- Each of the 13 highest ranking modules covers the large subunit of mitochondrial ribosome
  - Found additional protein, Mhr1
- Exactly recovered the nucleoplasmic THO complex (Hpr1, Mft1, Rlr1, Thp2)
  - Transcription elongation, hyperrecombination
  - Growth defect under ethanol

Phenotype Associated Modules
Large subunit of mitochondrial ribosome



Mhr1: involved in homologous recombination of the mitochondrial genome

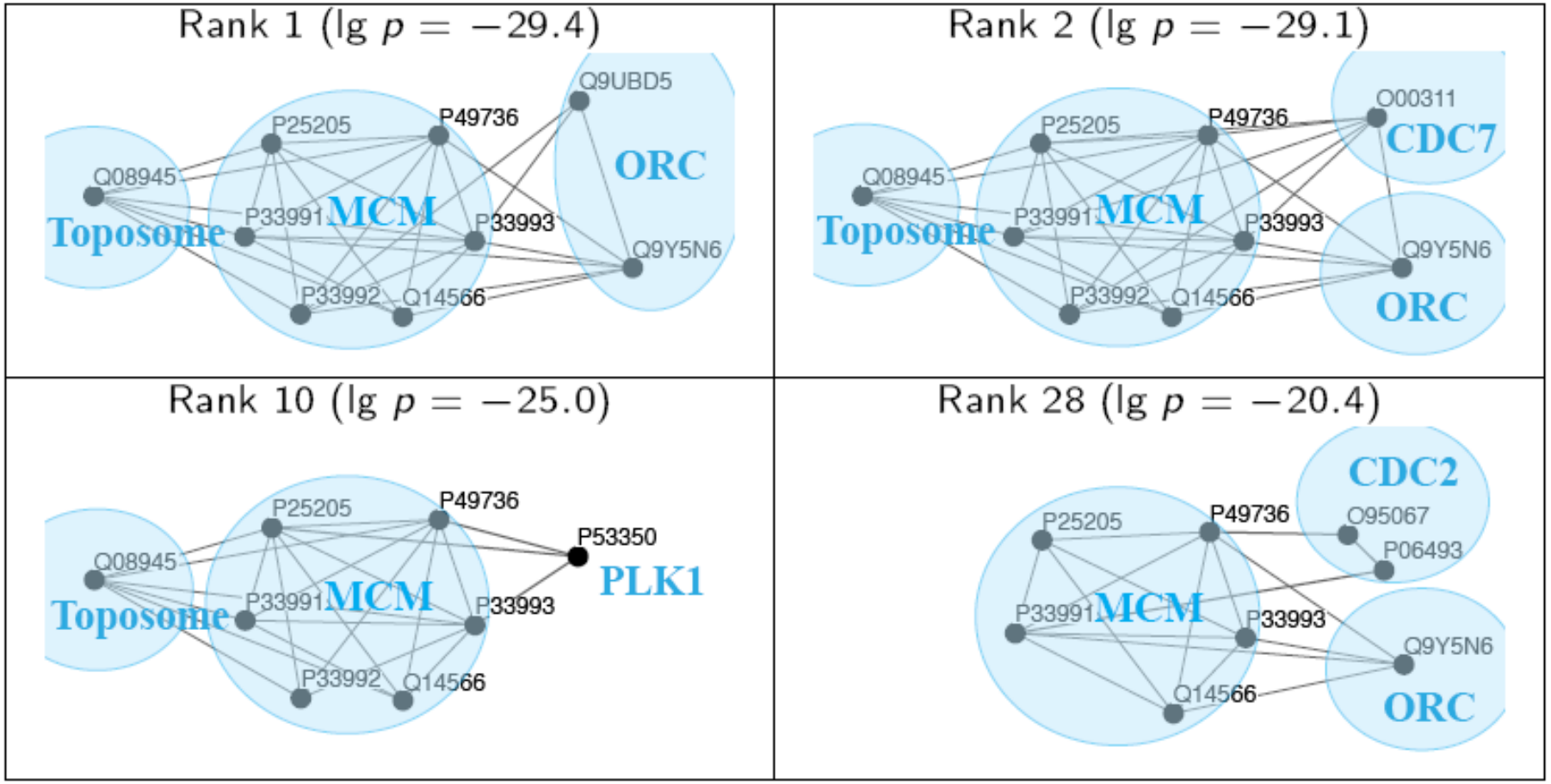
Human Settings

- Tissue-specific gene expression data (Su et al., 2004)
  - 79 different tissues
- Consistently expressed in 3 tissues, not in 10 tissues
- 7763 proteins, density >= 35%, 5 minutes
- 1021 maximal modules
- MIPS human complex database (Ruepp et al., to appear)

Human-expression result

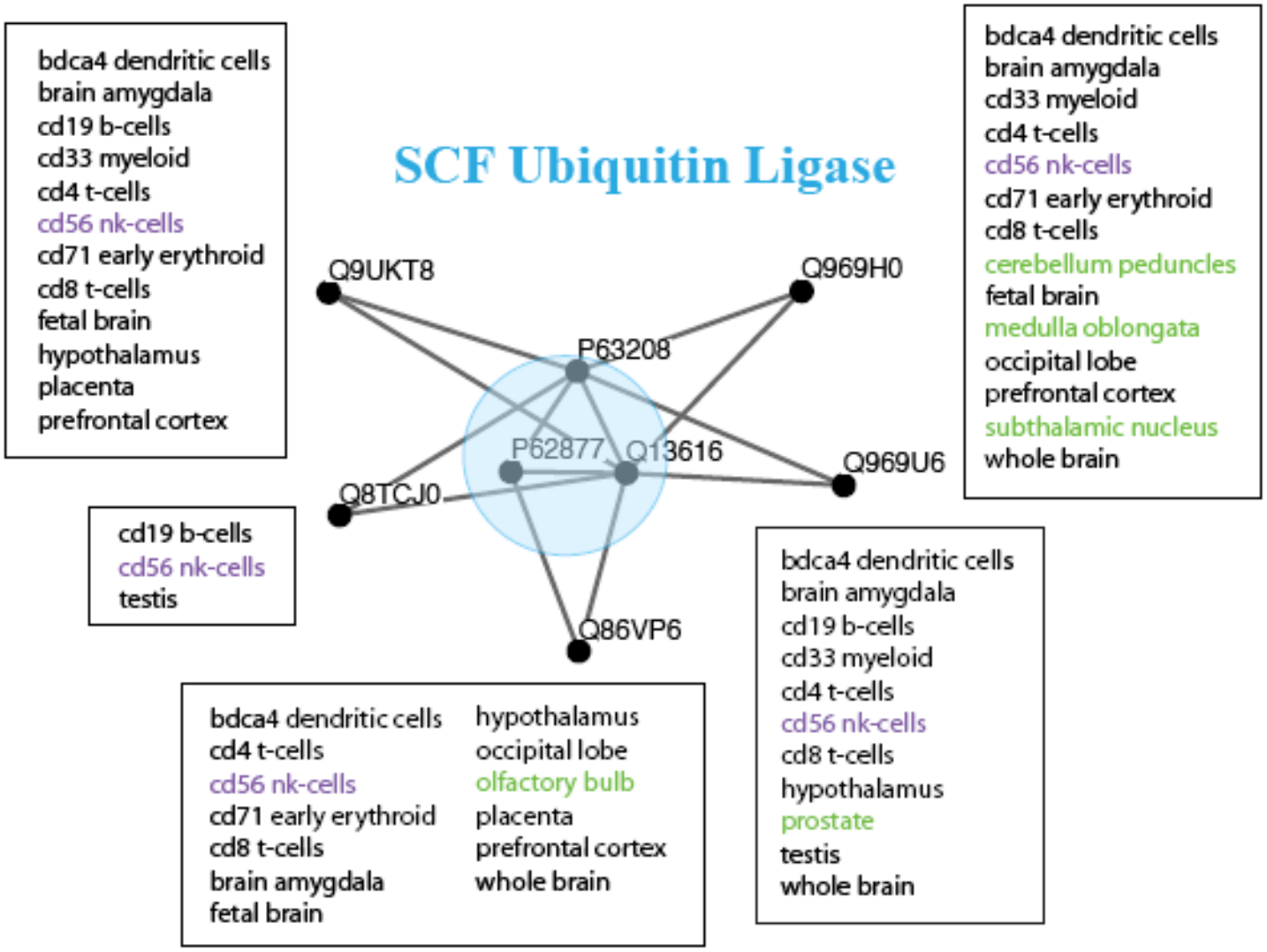
- Around MCM complex, we found inter-complex relationships with ORC, CDC7, Toposome, PLK1 protein
- Module Uqcrc1, Uqcrc2, Uqcrb, Cyc1 (lg p = -13)
  - No overlap with MIPS
  - Ubiquinol-cytochrome c reductase complex
- SCF E3 ubiquitin ligase complex: Mark protein for degradation
  - 5 different modules with different tissue specificity
  - Peripheral proteins: Substrate recognition particles
  - Target proteins are selected in a tissue specific manner!
  - Natural Killer cells have all particles

High ranking modules around the MCM complex



Expressed in bone mallow cells
Not expressed in brain, liver, kidney etc.

Tissue Specific organization of the SCF ligase complex



Summary: Dense Module Enumeration

- Novel module enumeration algorithm based on reverse search
- Combination with other information sources
- Statistical significance of dense modules
- Successfully applied to
  - yeast/human protein interaction networks

Reference

- [1] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” in *Proc. 20th Int. Conf. Very Large Data Bases*, 1994.
- [2] T. Asai, H. Arimura, T. Uno, and S. Nakano, “Discovering frequent substructures in large unordered trees,” *Discovery science*, 2003.
- [3] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, and S. Arikawa, “Efficient Substructure Discovery from Large Semi-structured Data.,” in *SDM*, 2002.
- [4] D. Avis and K. Fukuda, “Reverse search for enumeration,” *Discrete Applied Mathematics*, vol. 65, no. 1–3, pp. 21–46, 1996.
- [5] E. Georgii, S. Dietmann, T. Uno, P. Pagel, and K. Tsuda, “Enumeration of condition-dependent dense modules in protein interaction networks.,” *Bioinformatics (Oxford, England)*, vol. 25, no. 7, pp. 933–40, Apr. 2009.
- [6] S. Nijssen and J. Kok, “Efficient discovery of frequent unordered trees,” *MGTS*, 2003.
- [7] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, “Discovering frequent closed itemsets for association rules,” *Database Theory—ICDT’99*, 1999.
- [8] I. Takigawa and H. Mamitsuka, “Efficiently mining δ-tolerance closed frequent subgraphs,” *Machine Learning*, vol. 82, no. 2, pp. 95–121, Sep. 2010.
- [9] X. Yan and J. Han, “gSpan: graph-based substructure pattern mining,” in *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pp. 721–724.
- [10] M. Zaki, “Efficiently mining frequent trees in a forest: Algorithms and applications,” *IEEE Transaction on Knowledge and Data Engineering*, 2005.

Machine Learning Summer School Kyoto 2012

Graph Mining

Chapter 2: Learning from Structured Data

National Institute of Advanced Industrial Science and Technology

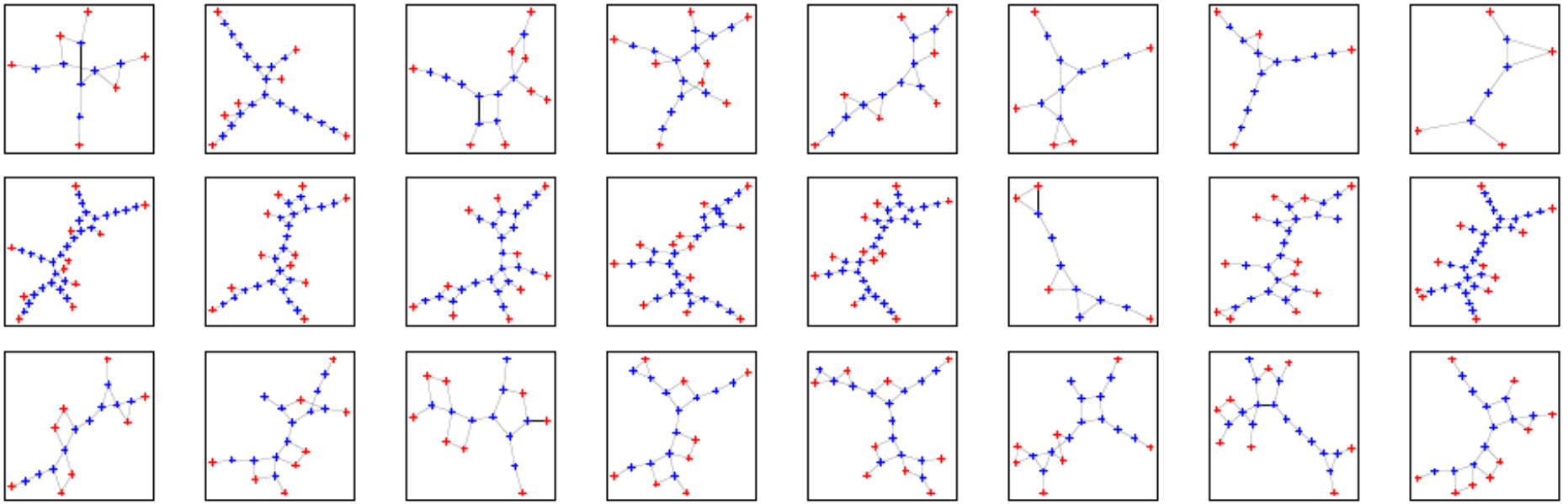
Koji Tsuda

Agenda 2 (Learning from Structured Data)

- 1. Preliminaries
- 2. Graph Clustering by EM
- 3. Graph Boosting
- 4. Regularization Paths in Graph Classification
- 5. Itemset Boosting for predicting HIV drug resistance

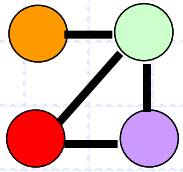
Part 1: Preliminaries

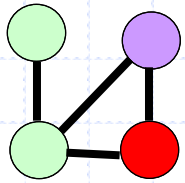
Clustering Graphs

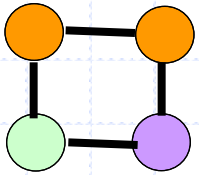


Graph Regression

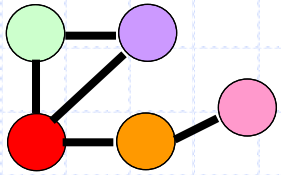
Training

( ,-0.2)

( , 0.7)

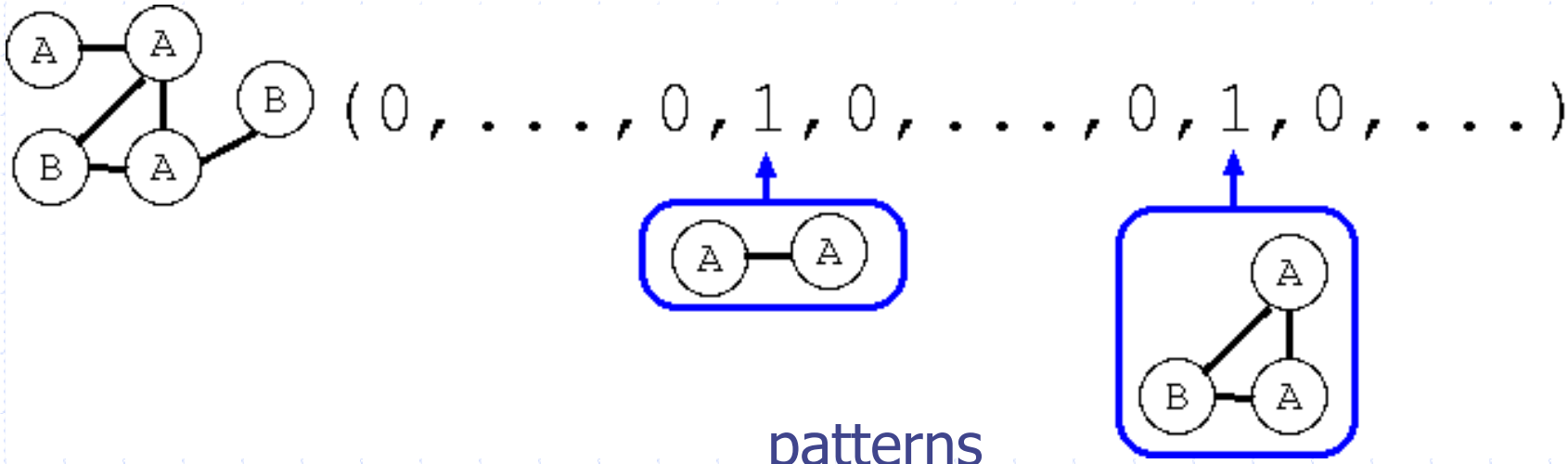
( ,-0.5)

Test

( , ?)

Substructure Representation

- 0/1 vector of *pattern* indicators
- Huge dimensionality!
- Need Graph Mining for selecting features



patterns

Graph Mining

- Frequent Substructure Mining
  - Enumerate all patterns occurred in at least m graphs

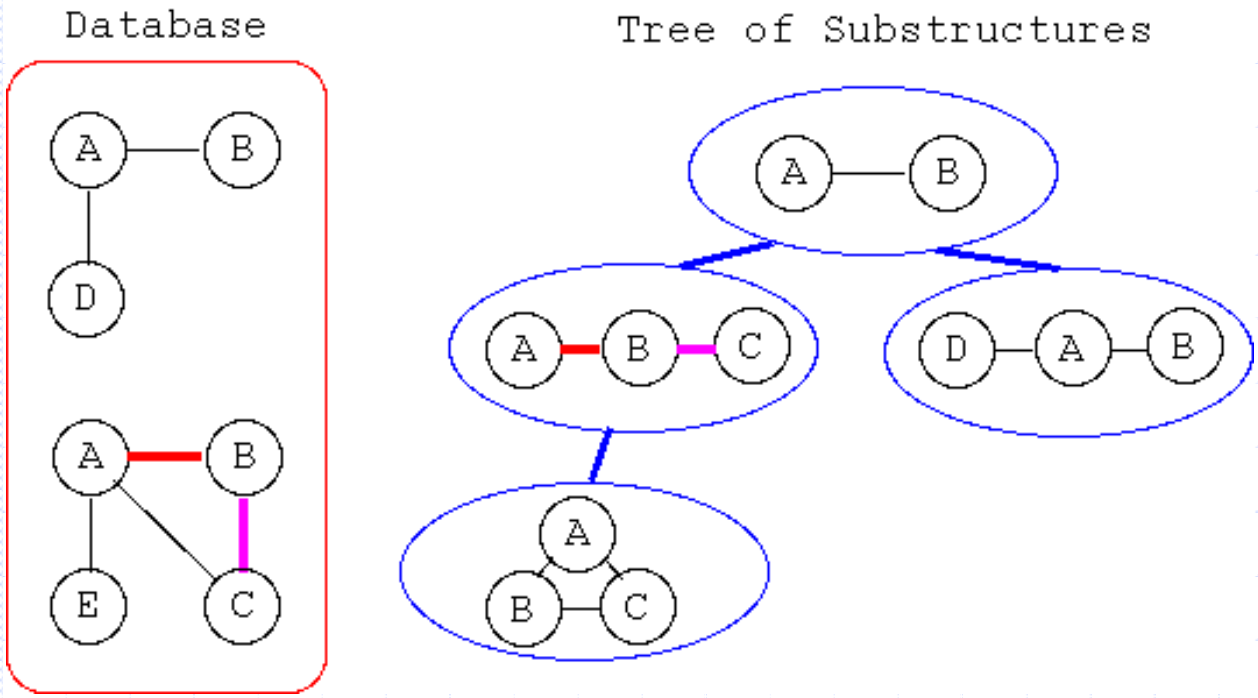
$$S_{freq} = \{k \mid \sum_{i=1}^{n} x_{ik} \geq m\}.$$

$x_{ik} \in \{0, 1\}$:Indicator of pattern k in graph i

Support(k): # of occurrence of pattern k

Enumeration on Tree-shaped Search Space

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



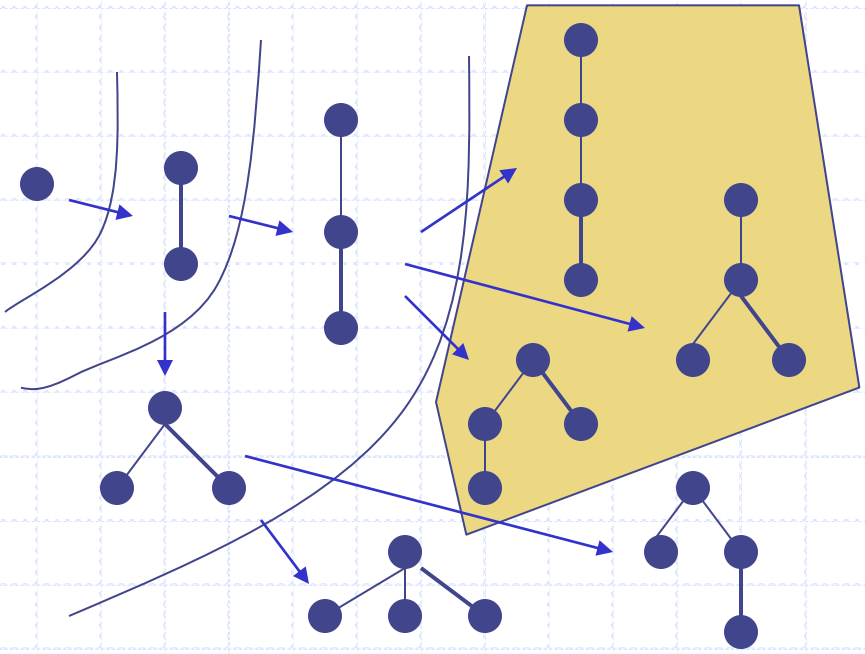
Support(g): # of occurrence of pattern g

Tree Pruning

- Anti-monotonicity:

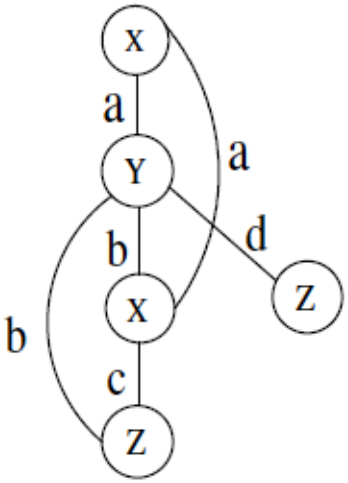
$$g \subseteq g' \Rightarrow support(g) \geq support(g')$$

- If support(g) < m, stop exploring!



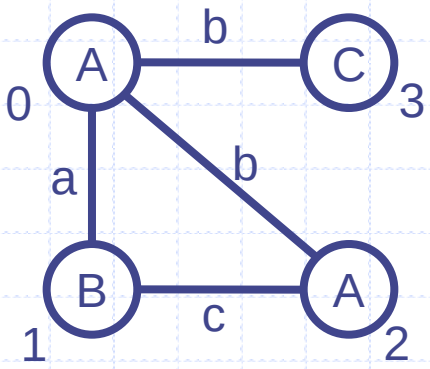
Gspan (Yan and Han, 2002)

- Efficient Frequent Substructure Mining Method
- DFS Code
  - Efficient detection of isomorphic patterns

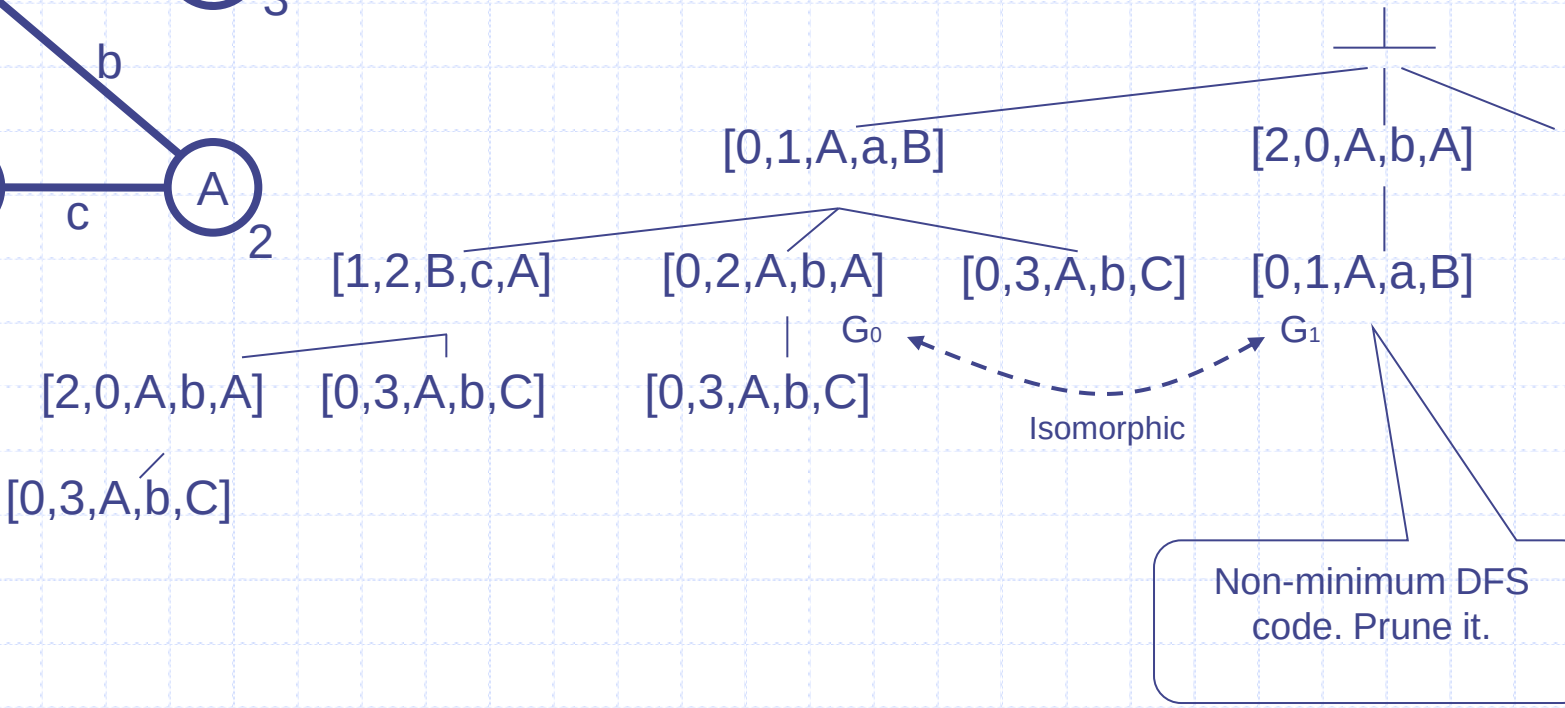


Depth First Search (DFS) Code

A labeled graph G



DFS Code Tree on G



Discriminative patterns

- w_i > 0: positive class
- w_i < 0: negative class
- Weighted Substructure Mining

$$S_w = \{k \mid \left| \sum_{i=1}^{n} w_i (2x_{ik} - 1) \right| \geq \tau \},$$

- Patterns with large frequency difference
- Not Anti-Monotonic: Use a bound

Multiclass version

- Multiple weight vectors
  - $w_{\ell i} > 0$ (graph i belongs to class ℓ)
  - $w_{\ell i} < 0$ (otherwise)

- Search patterns overrepresented in a class

$$S_W = \{k \mid \max_{\ell=1,\ldots,c} \left| \sum_{i=1}^{n} w_{\ell i}(2x_{ik} - 1) \right| - \tau_\ell \geq 0\}.$$

Basic Bound

- x_{ij} : Occurrence of pattern j
- If k is supergraph of pattern j,

$$\left| \sum_{i=1}^{n} w_{\ell i}(2x_{ik} - 1) \right| \leq \gamma_\ell \qquad \gamma_\ell = \max(\gamma_\ell^+, \gamma_\ell^-)$$

$$\gamma_\ell^+ = 2 \sum_{\{i | w_{\ell i} \geq 0, x_{ij}=1\}} |w_{\ell i}| - \sum_{i=1}^{n} w_{\ell i}$$

$$\gamma_\ell^- = 2 \sum_{\{i | w_{\ell i} < 0, x_{ij}=1\}} |w_{\ell i}| + \sum_{i=1}^{n} w_{\ell i}$$

Pruning Condition

- Summarizing the bound for all classes,

$$\max_{\ell} \left| \sum_{i=1}^{n} w_{\ell i}(2x_{ik} - 1) \right| - \tau_{\ell} \leq \max_{\ell}(\gamma_{\ell} - \tau_{\ell})$$

- If it is negative, the search tree can be pruned safely

Summary: Preliminaries

- Various graph learning problems
  - Supervised/Unsupervised
- Discovery of salient features by graph mining
- Actual speed depends on the data
  - Faster for..
    - Sparse graphs
    - Diverse kinds of labels

Part 2: EM-based clustering of graphs

EM-based graph clustering

- Motivation
  - Learning a mixture model in the feature space of patterns
  - Basis for more complex probabilistic inference
- L1 regularization & Graph Mining
- E-step -> Mining -> M-step

Probabilistic Model

- Binomial Mixture

$$p(\boldsymbol{x}|\Theta) = \sum_{\ell=1}^{c} \alpha_\ell p_\ell(\boldsymbol{x}|\boldsymbol{\theta}_\ell)$$

- Each Component

$$p_\ell(\boldsymbol{x}|\boldsymbol{\theta}_\ell) = \prod_{k=1}^{d} \frac{\exp(\theta_{\ell k} x_k)}{1+\exp(\theta_{\ell k})}.$$

$\boldsymbol{x}$:Feature vector of a graph (0 or 1)

α_ℓ :Mixing weight for cluster ℓ

$\boldsymbol{\theta}_\ell$:Parameter vector for cluster ℓ

Ordinary EM algorithm

- Maximizing the log likelihood

$$\underset{\Theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \sum_{\ell=1}^{c} \alpha_\ell p_\ell(\boldsymbol{x}_i|\boldsymbol{\theta}_\ell).$$

- E-step: Get posterior $r_{\ell i} = p(y = \ell|\boldsymbol{x}_i)$
- M-step: Estimate $\boldsymbol{\theta}_\ell$ using posterior probs.
- Both are computationally prohibitive (!)

Regularization

- L1-Regularized log likelihood

$$\frac{1}{n}\sum_{i=1}^{n}\log\sum_{\ell=1}^{c}\alpha_\ell p_\ell(\boldsymbol{x}_i|\boldsymbol{\theta}_\ell) - \lambda\sum_{\ell=1}^{c}\sum_{k=1}^{d}|\theta_{\ell k} - \theta_{0k}|$$

- Baseline constant $\boldsymbol{\theta}_0$
  - ML parameter estimate using single binomial distribution

$$\theta_{0k} = \log\eta_{0k} - \log(1-\theta_{0k}) \qquad \eta_{0k} = \frac{1}{n}\sum_i x_{ik}$$

- In solution, most parameters exactly equal to constants

E-step

- Active pattern

$$F = \{k \mid \text{there exists } \ell \text{ such that } \theta_{\ell k} \neq \theta_{0k}\}.$$

- E-step computed only with active patterns (computable!)

$$p(y = \ell | \boldsymbol{x}) = \frac{\alpha_\ell \prod_{k \in F} p_{\ell k}(x_k | \theta_{\ell k})}{\sum_\ell \alpha_\ell \prod_{k \in F} p_{\ell k}(x_k | \theta_{\ell k})}.$$

M-step

- Putative cluster assignment $r_{\ell i}$
- Each parameter is solved separately

$$\min_{\theta_{\ell k}} -\frac{1}{n}\sum_i r_{\ell i} \log p(x_{ik}|\theta_{\ell k}) + \lambda|\theta_{\ell k} - \theta_{0k}|.$$

- Naïve way:
  - solve it for all params and identify active patterns
- Use graph mining to find active patterns

Solution

$$\lambda_\ell = \frac{\lambda n}{\sum_j r_{\ell j}}$$

$$\theta_{\ell k} = \begin{cases} \log \frac{\eta_{\ell k} - \lambda_\ell}{1-(\eta_{\ell k} - \lambda_\ell)} & (\eta_{\ell k} \geq \eta_{0k} + \lambda_\ell). \\ \theta_{0k} & (\eta_{0k} - \lambda_\ell \leq \eta_{\ell k} \leq \eta_{0k} + \lambda_\ell) \\ \log \frac{\eta_{\ell k} + \lambda_\ell}{1-(\eta_{\ell k} + \lambda_\ell)} & (\eta_{\ell k} \leq \eta_{0k} - \lambda_\ell). \end{cases}$$

- Occurrence probability in a cluster

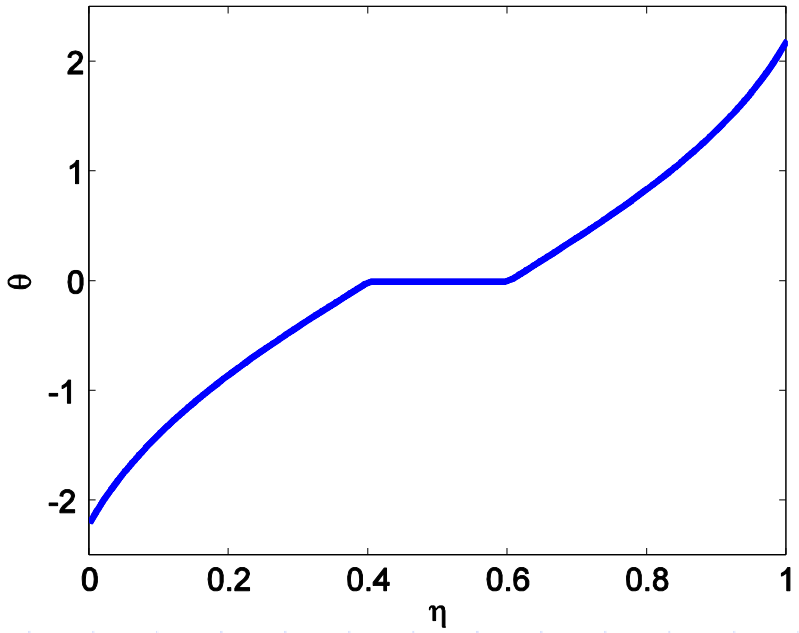
$$\eta_{\ell k} = \sum_i r_{\ell i} x_{ik} / \sum_j r_{\ell j}$$

- Overall occurrence probability

$$\eta_{0k} = \frac{1}{n} \sum_i x_{ik}$$

Solution

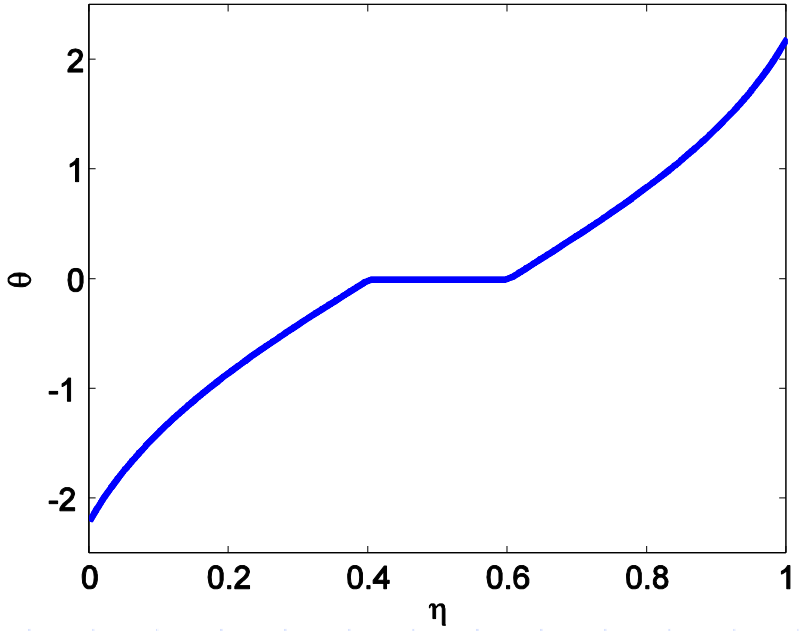
$$\theta_{\ell k} = \begin{cases} \log \frac{\eta_{\ell k} - \lambda_\ell}{1-(\eta_{\ell k} - \lambda_\ell)} & (\eta_{\ell k} \geq \eta_{0k} + \lambda_\ell). \\ \theta_{0k} & (\eta_{0k} - \lambda_\ell \leq \eta_{\ell k} \leq \eta_{0k} + \lambda_\ell) \\ \log \frac{\eta_{\ell k} + \lambda_\ell}{1-(\eta_{\ell k} + \lambda_\ell)} & (\eta_{\ell k} \leq \eta_{0k} - \lambda_\ell). \end{cases}$$



Important Observation

For active pattern k, the occurrence probability in a graph cluster is significantly different from the average

$$\theta_{\ell k} \neq \theta_{0k} \Leftrightarrow \left|\eta_{\ell k} - \eta_{0k}\right| \geq \lambda_{\ell}$$



Mining for Active Patterns

- Active pattern

$$F = \{k \mid \text{there exists } \ell \text{ such that } \theta_{\ell k} \neq \theta_{0k}\}.$$

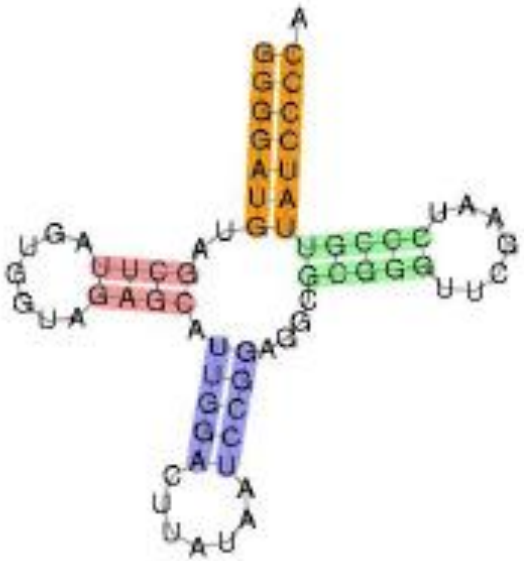
- Equivalently written as

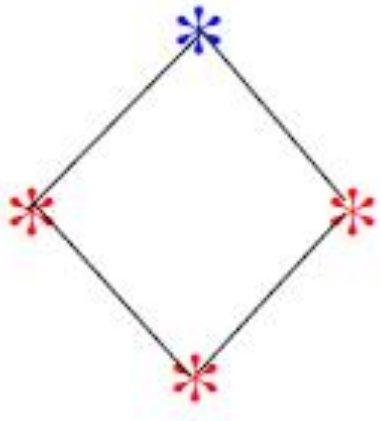
$$F = \{k \mid \max_{\ell=1,\ldots,c} \left| \sum_{i=1}^{n} w_{\ell i}(2x_{ik} - 1) \right| - 2\lambda_\ell \geq 0\}.$$

- F can be found by graph mining! (multiclass)

Experiments: RNA graphs

- Stem as a node
- Secondary structure by RNAfold
- 0/1 Vertex label (self loop or not)

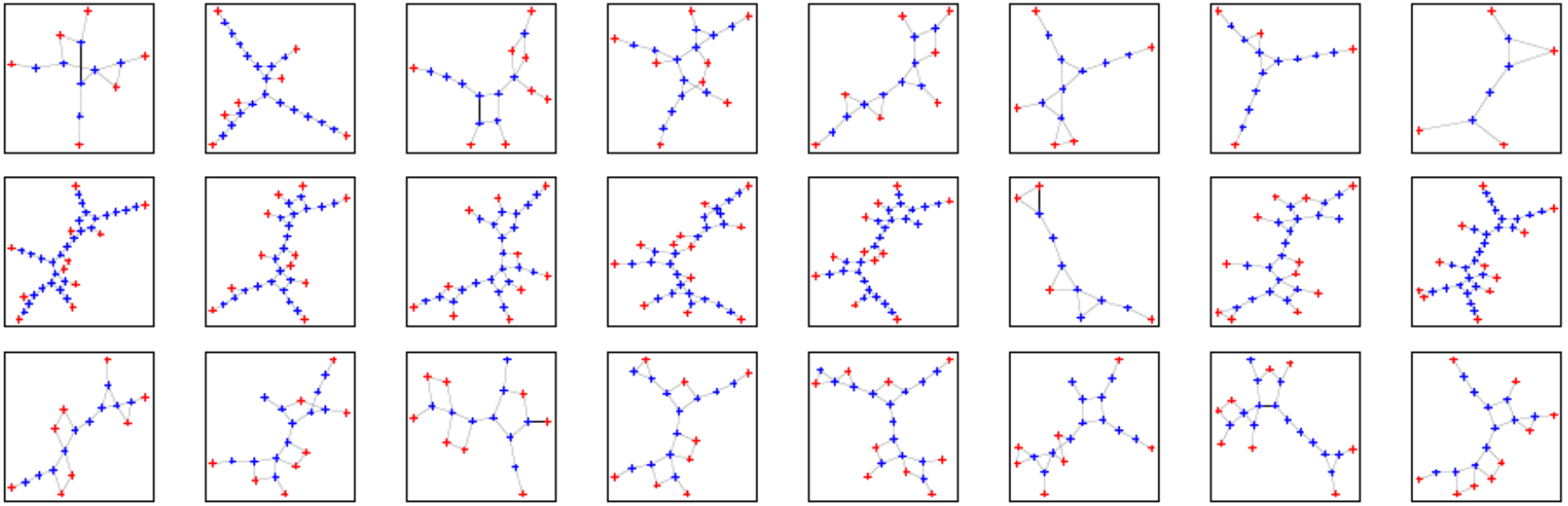




Clustering RNA graphs

- Three Rfam families
  - Intron GP I (Int, 30 graphs)
  - SSU rRNA 5 (SSU, 50 graphs)
  - RNase bact a (RNase, 50 graphs)
- Three bipartition problems
  - Results evaluated by ROC scores (Area under the ROC curve)

Examples of RNA Graphs



ROC Scores

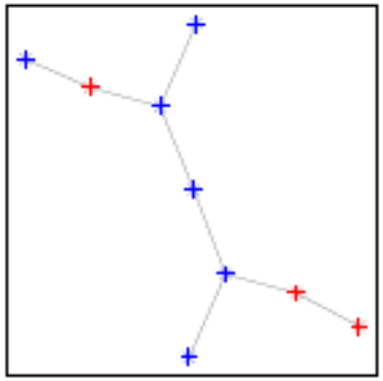
	Int-SSU	Int-RNase	SSU-RNase
MGK	0.748	0.531	**0.878**
Spec	0.550	0.573	0.848
$\lambda = 0.01$	0.824	0.921	0.863
$\lambda = 0.02$	0.821	0.920	0.862
$\lambda = 0.03$	0.825	**0.948**	0.843
$\lambda = 0.04$	0.832	0.947	0.825
$\lambda = 0.06$	0.831	0.941	0.782
$\lambda = 0.08$	**0.845**	0.941	0.787
$\lambda = 0.10$	0.815	0.927	0.786

No of Patterns & Time

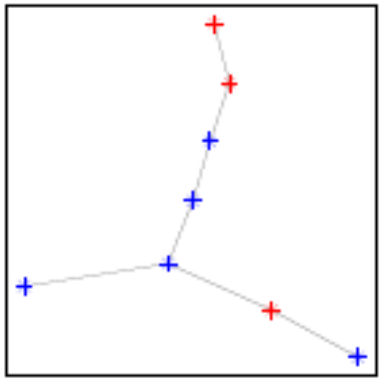
	Int-SSU	Int-RNase	SSU-RNase
$\lambda = 0.01$	12505 (71s)	14366 (77s)	17934 (102s)
$\lambda = 0.02$	12596 (75s)	10988 (65s)	11025 (76s)
$\lambda = 0.03$	9799 (66s)	7632 (52s)	8875 (73s)
$\lambda = 0.04$	6904 (57s)	5924 (45s)	6925 (67s)
$\lambda = 0.06$	5093 (47s)	4305 (37s)	5230 (58s)
$\lambda = 0.08$	4065 (42s)	3001 (32s)	3896 (50s)
$\lambda = 0.10$	3245 (37s)	2074 (26s)	2923 (44s)

Found Patterns

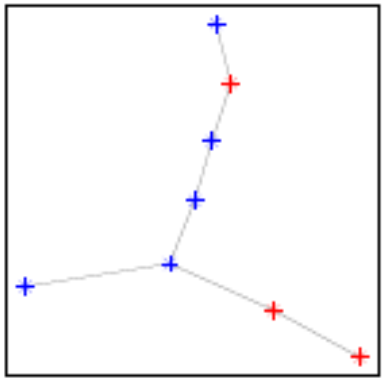
1.150



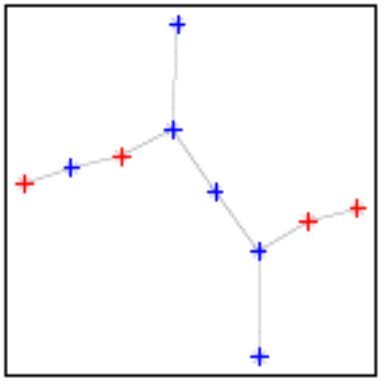
1.150



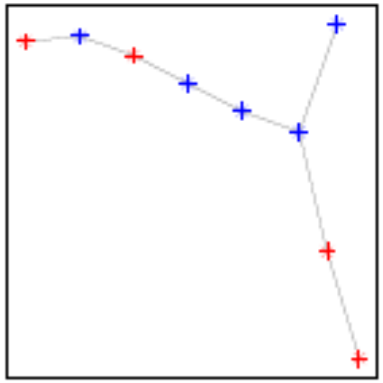
1.125



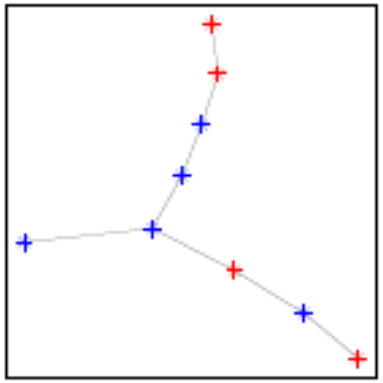
1.117



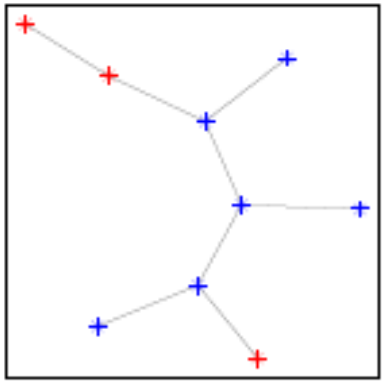
1.117



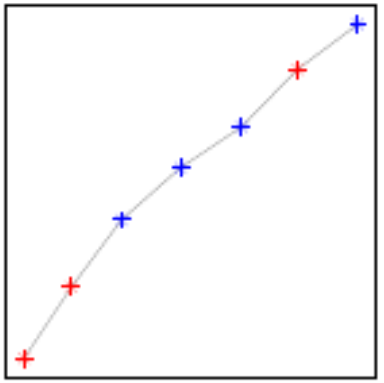
1.117



1.100



1.100



Summary (graph EM)

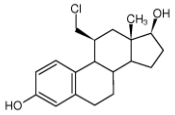
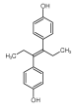
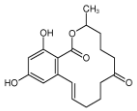
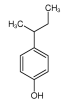
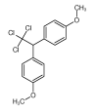
- Substructure representation is better than paths
- Probabilistic inference helped by graph mining
- Extension to Dirichlet mixture model
  - Reported in Tsuda et al., SDM 2008
- Possible extension
  - Graph PCA, LFD, CCA
  - Semi-supervised learning

Part 3: Graph Boosting

Graph classification problem in chemoinformatics

- Known as SAR problem in chemical informatics
  - (Quantitative) Structure-Activity Analysis
- Given a graph, predict a class-label (+1 or -1)
  - Typically, features (descriptors) are given
  - e.g., Dragon Descriptors, JOELIB2

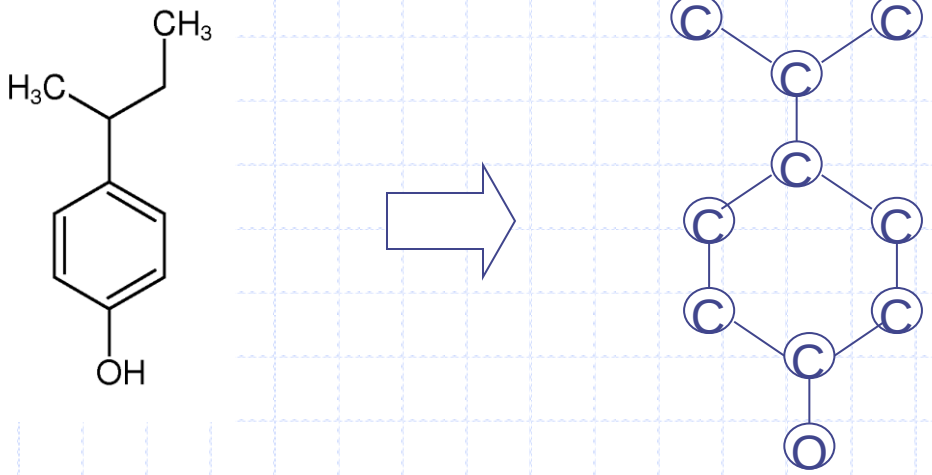
SAR with conventional descriptors

	#atoms	#bonds	#rings	…	Activity
	22	25			+1
	20	21			+1
	23	24			+1
	11	11			-1
	21	22			-1

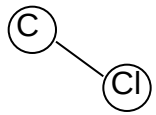
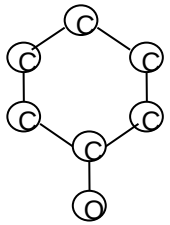
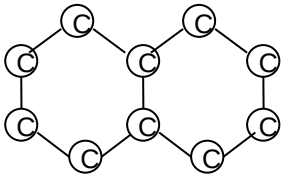
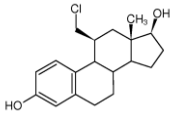
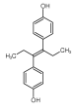
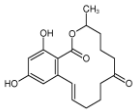
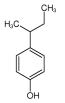
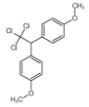
Motivation of Graph Boosting

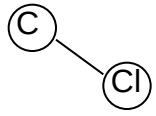
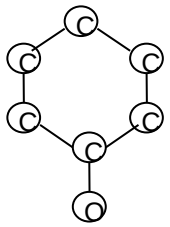
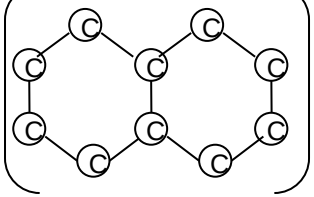
- Descriptors are not always available
- New features by obtaining informative patterns (i.e., subgraphs)
- Greedy pattern discovery by Boosting + gSpan
- Linear Programming (LP) Boosting
  - Reduce the number of graph mining calls
  - Faster than AdaBoost
- Accurate prediction & interpretable results

Molecule as a labeled graph



SAR with patterns

				…	Activity
	1	1	1		+1
	-1	1	-1		+1
	-1	1	-1		+1
	-1	1	-1		-1
	1	1	-1		-1

$$f = \alpha_1 \left(\text{  } \right) + \alpha_2 \left(\text{  } \right) + \alpha_3 \left(\text{  } \right) + \ldots$$

Sparse classification in a very high dimensional space

- **G**: all possible patterns (intractably large)
- |G|-dimensional feature vector **x** for a molecule
- Linear Classifier

$$f(\mathbf{x}) = \sum_{j=1}^{d} \alpha_j x_j$$

- Use L1 regularizer to have sparse α
- Select a tractable number of patterns

Problem formulation

$$\min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad ||\boldsymbol{\alpha}||_1 + C\sum_{n=1}^{\ell}\xi_n$$

$$s.t. \quad y_n\boldsymbol{\alpha}^\top\mathbf{x}_n \geq 1-\xi_n, \quad \xi_n \geq 0, \quad n=1,\ldots,\ell$$

Sum of hinge loss and L1 regularizer

$\{\boldsymbol{x}_n, y_n\}_{n=1}^{\ell}$: Training examples

ξ: slack variables

Dual LP

- Primal: Huge number of weight variables
- Dual: Huge number of constraints

Dual problem

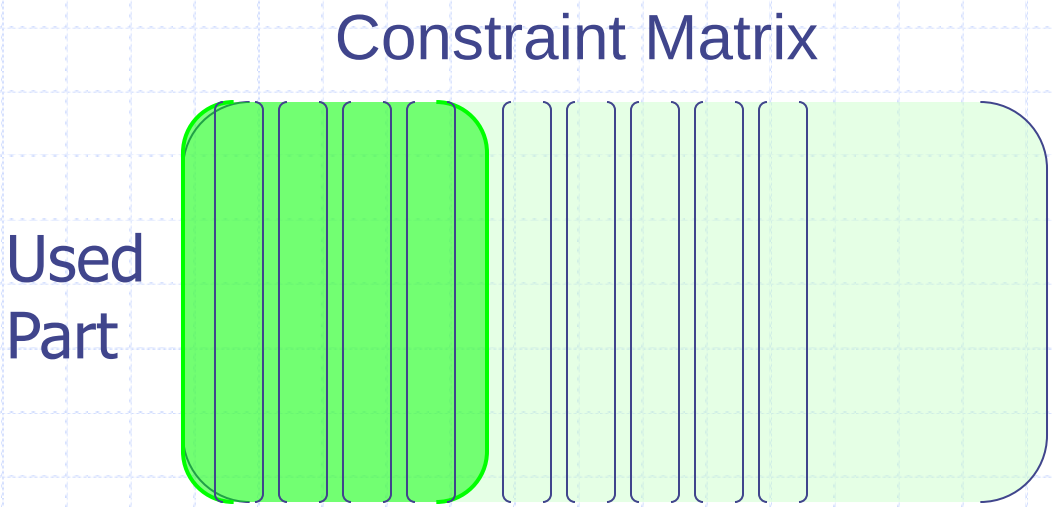
$$\max_{\mathbf{u}} \quad \sum_{n=1}^{\ell} u_i$$

$$s.t. \quad \sum_{n=1}^{\ell} u_n y_n x_{ni} \leq 1, \; i = 1, \ldots, d$$

$$0 \leq u_n \leq C, \; n = 1, \ldots, \ell$$

Column Generation Algorithm for LP Boost (Demiriz et al., 2002)

- Start from the dual with no constraints
- Add the most violated constraint each time
- Guaranteed to converge



Finding the most violated constraint

- Constraint for a pattern (shown again)

$$\sum_{n=1}^{\ell} u_n y_n x_{ni} \leq 1$$

- Finding the most violated one

$$\mathrm{argmax}_i \sum_{n=1}^{\ell} u_n y_n x_{ni}$$

- Searched by weighted substructure mining

Algorithm Overview

- Iteration
  - Find a new pattern by graph mining with weight $\mathbf{u}$
  - If all constraints are satisfied, break
  - Add a new constraint
  - Update $\mathbf{u}$ by solving the dual problem
- Return
  - Convert dual solution to obtain primal solution $\mathbf{a}$

Experimental Settings

- Classification and Regression Datasets

	# data	# positives	# negatives	avg. atoms	avg. bonds
CPDB	684	341	343	14.1	14.6
CAS	4337	2401	1936	29.9	30.9

	# data	avg. atoms	avg. bonds
AR	146	19.5	21.1
ER	131	19.2	20.7
ES	59	18.2	19.7

Classification Results

Table 2 Classification performance obtained by 10-fold cross validation in two datasets measured by the accuracy (ACC) and the area under the ROC curve (AUC). We obtained the results of MGK and gBoost from our implementations, but the other results are quoted from literature. The best results are highlighted in bold fonts.

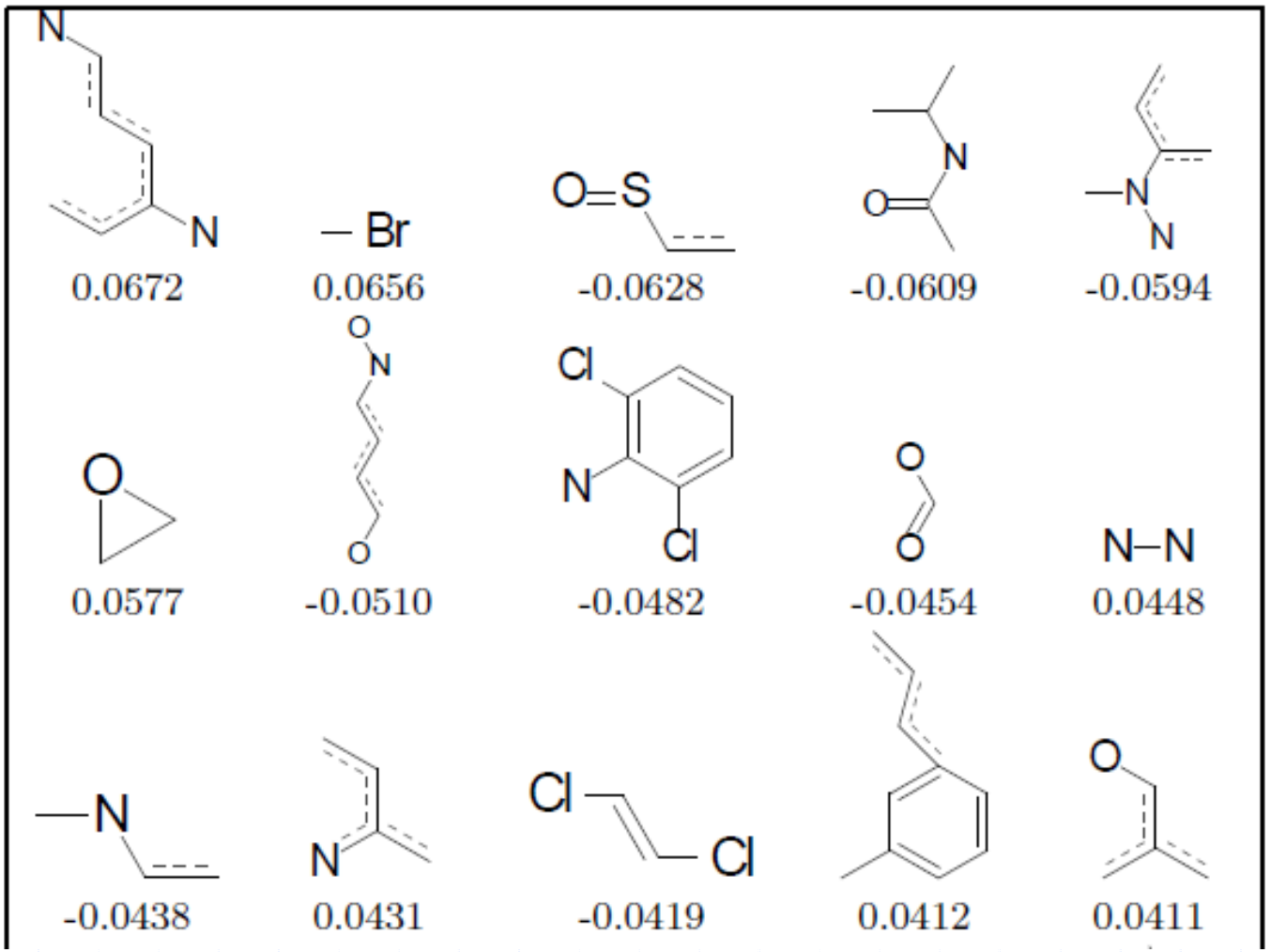
	Measure	MOLFEA[13]	Gaston[19]	CPM[2]	MGK[18]	gBoost
CPDB	ACC	-	78	75.96	76.5	**78.8**
	AUC	-	-	-	0.756	**0.854**
CAS	ACC	79	-	80.14	77.1	**82.5**
	AUC	-	-	-	0.763	**0.889**

Regression Results

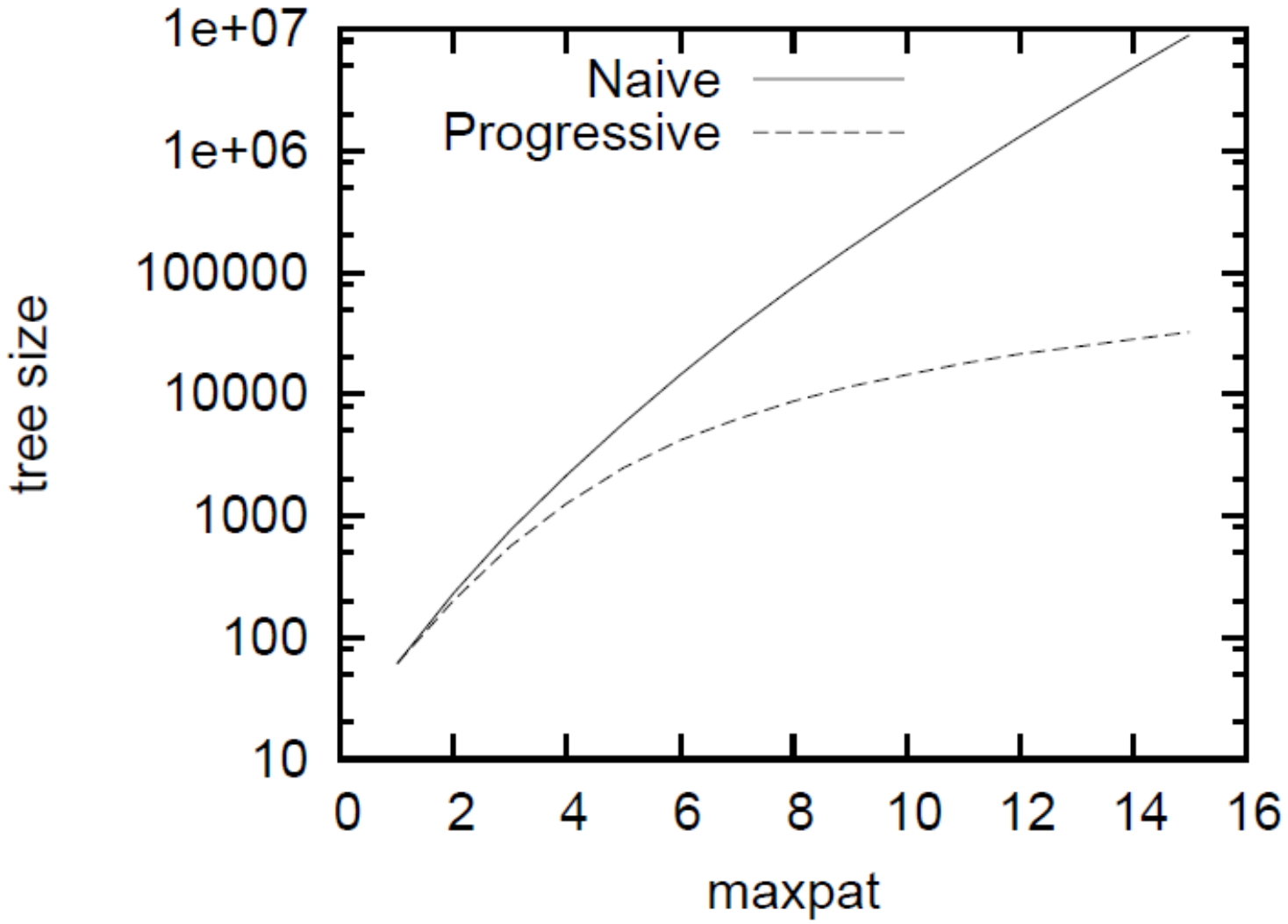
Table 3 Regression performance obtained by leave-one-out cross validation in three assays from the EDKB evaluated by mean absolute error (MAE), root mean squared error (RMSE), and Q^2. Note that for MAE and RMSE, lower values indicate better prediction, which is vice versa for Q^2. We obtained the results of MGK and gBoost from our implementations, but the other results are quoted from literature. The best results are highlighted in bold fonts.

	Measure	CoMFA[14,37]	MGK[18]	gBoost
AR	MAE	-	0.229	**0.176**
	RMSE	-	0.335	**0.232**
	Q^2	0.571	0.346	**0.682**
ER	MAE	-	0.320	0.307
	RMSE	-	0.427	0.393
	Q^2	**0.660**	0.267	0.378
ES	MAE	-	0.322	**0.249**
	RMSE	-	0.413	**0.362**
	Q^2	-	0.522	**0.632**

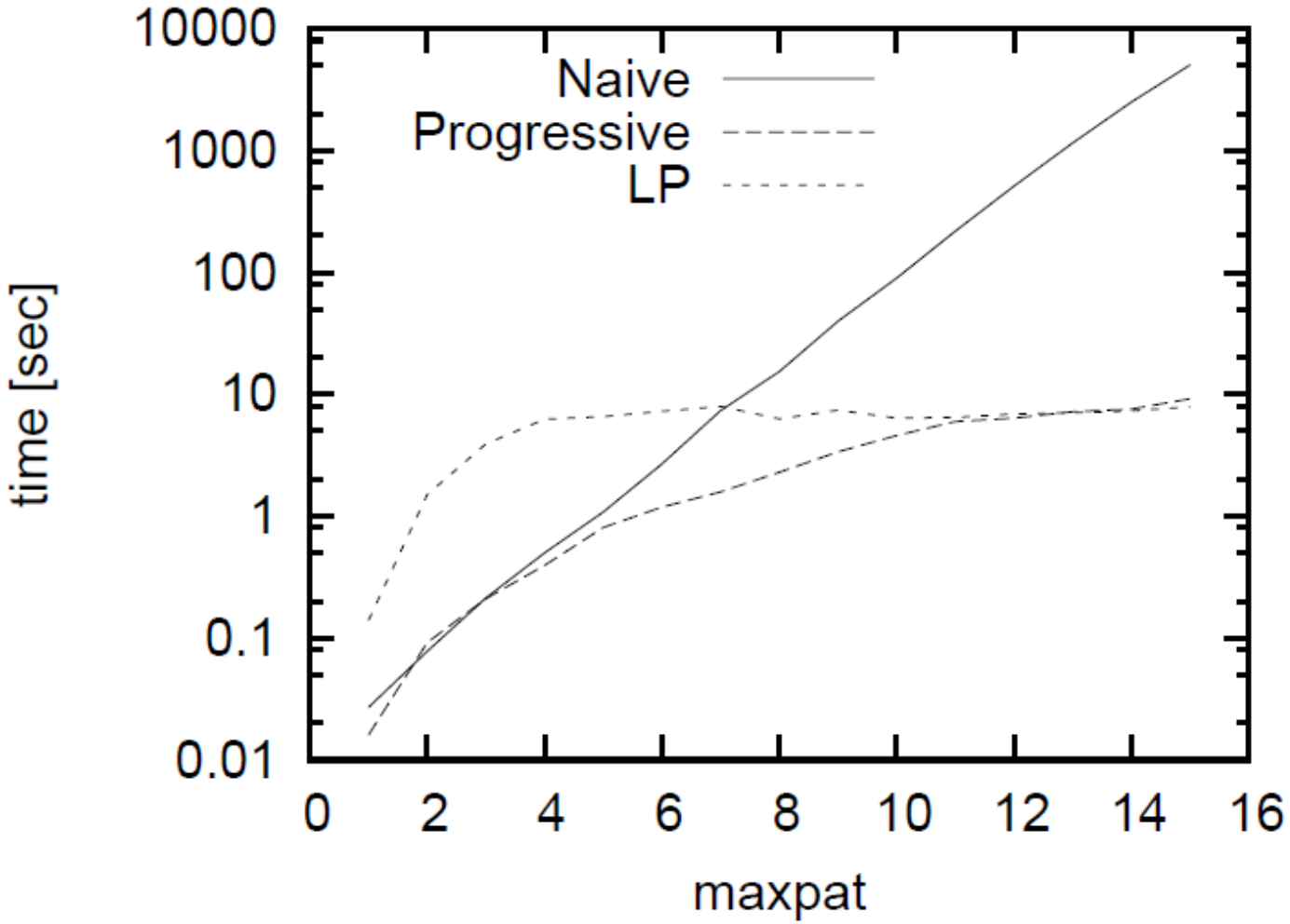
Extracted patterns from CPDB



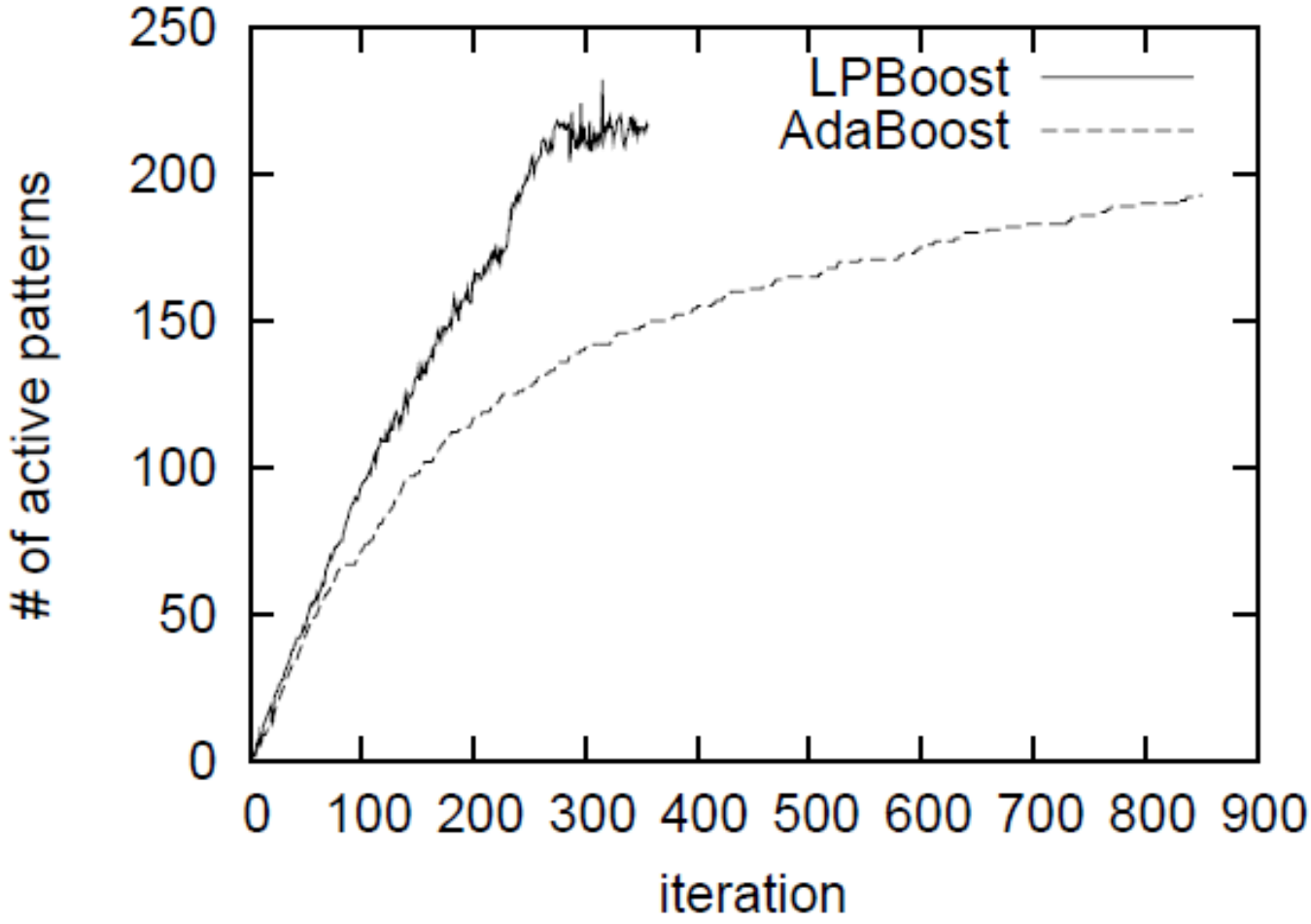
Memory Usage



Runtime


10000
1000
100
10
1
0.1
0.01
time [sec]
Naive
Progressive
LP
0
2
4
6
8
10
12
14
16
maxpat

Comparison with AdaBoost



Summary (Graph Boosting)

- Graph Boosting simultaneously generate patterns and learn their weights
- Finite convergence by column generation
- Interpretable by chemists.
- Flexible constraints and speed-up by LP.

Part 4: Entire Regularization Path

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

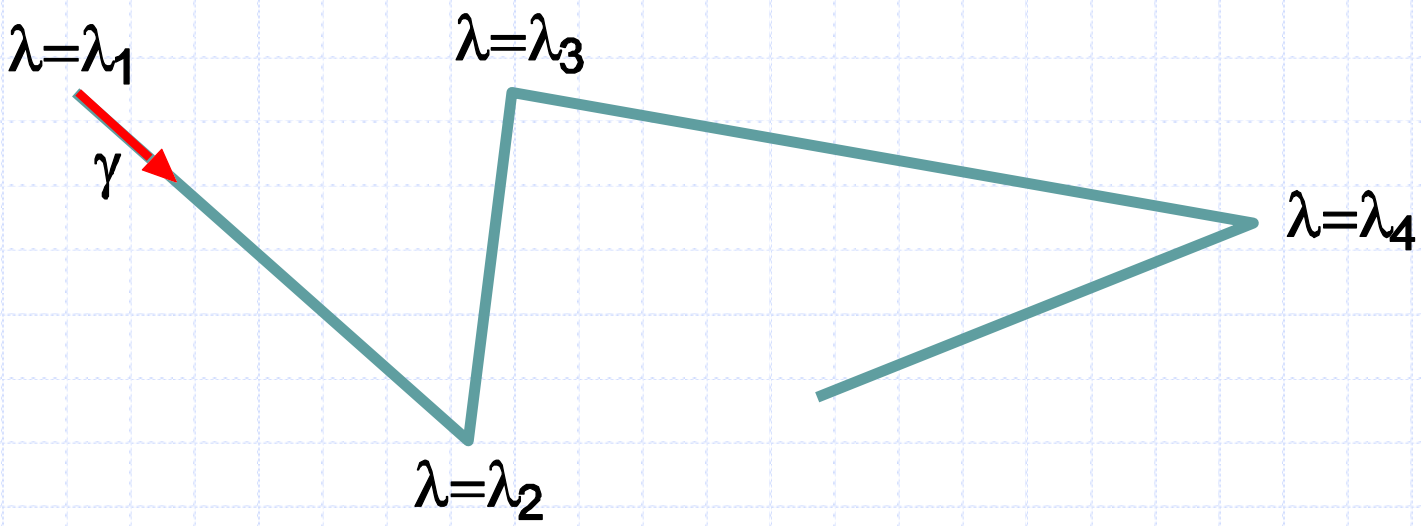
$$\boldsymbol{\beta}(\lambda) = \underset{\boldsymbol{\beta}}{\operatorname{argmin}}\, L(\boldsymbol{y}, X\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1.$$

- 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

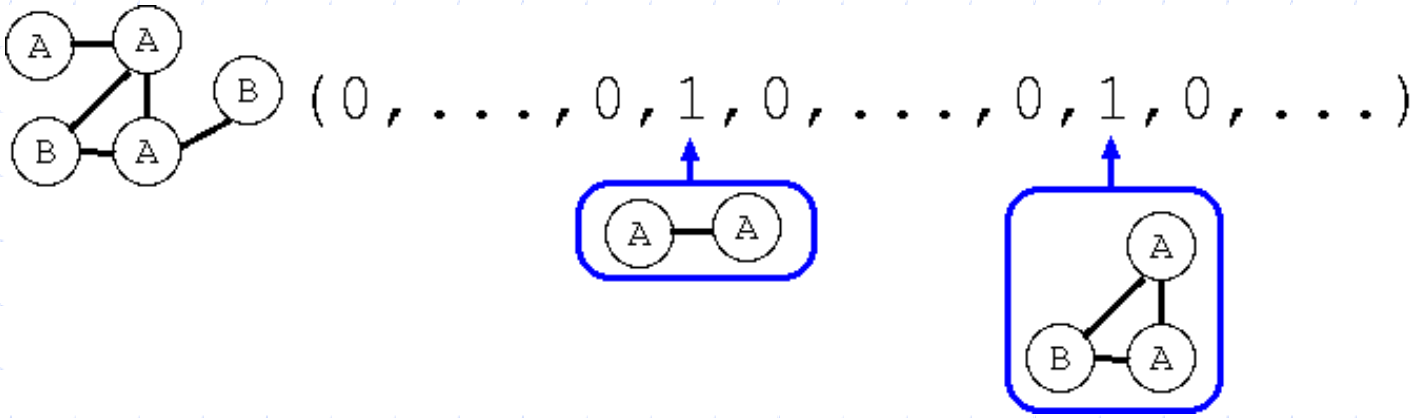
- Set initial point β and direction γ
- Do
  - d1 = Step size if next event is inclusion
  - d2 = Step size if next event is exclusion
  - d = min(d1,d2)
  - $\beta = \beta + d\gamma$
  - Update the active feature set
  - Set the next direction γ
- Until all features are included

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

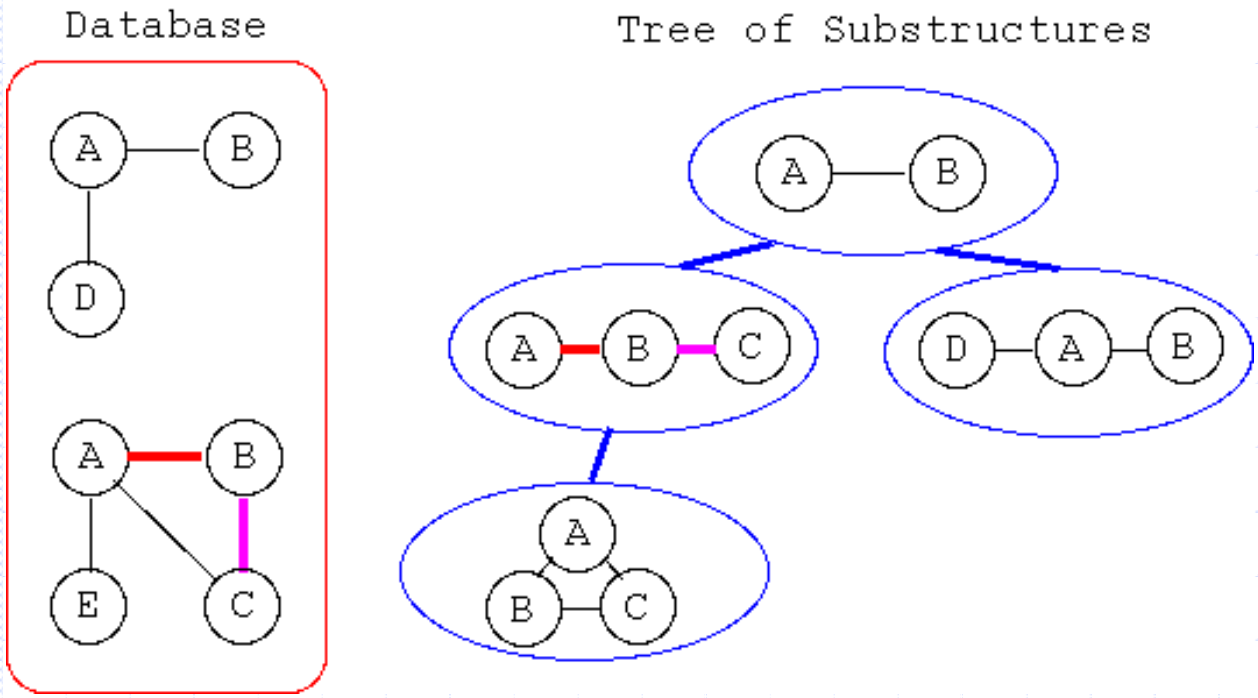
$$d_t = \min_{+} \left\{ \frac{\rho_0 - \sum_i w_i x_{it}}{\eta_0 - \sum_i v_i x_{it}}, \quad \frac{\rho_0 + \sum_i w_i x_{it}}{\eta_0 + \sum_i v_i x_{it}} \right\}.$$

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

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

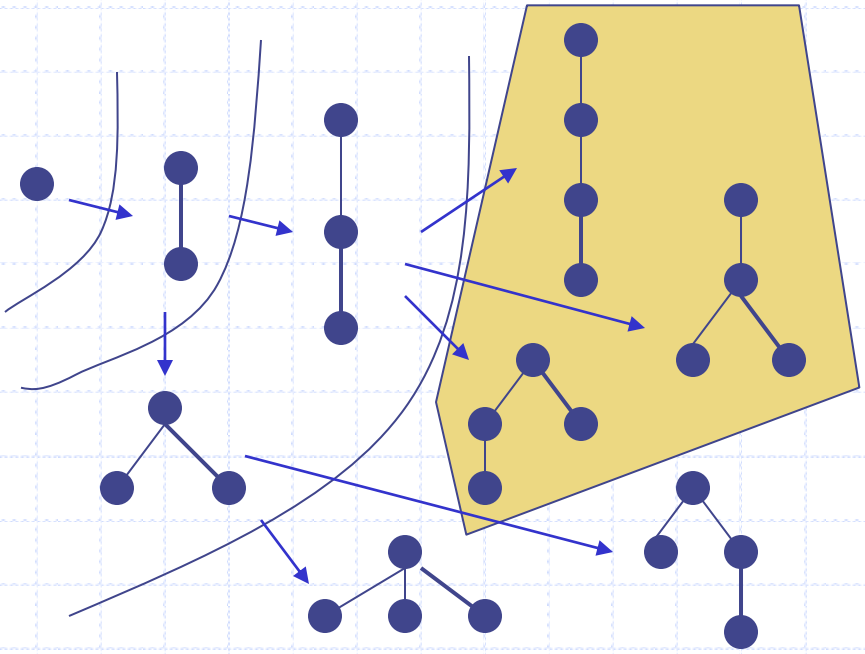
Tree-shaped Search Space

- 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

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

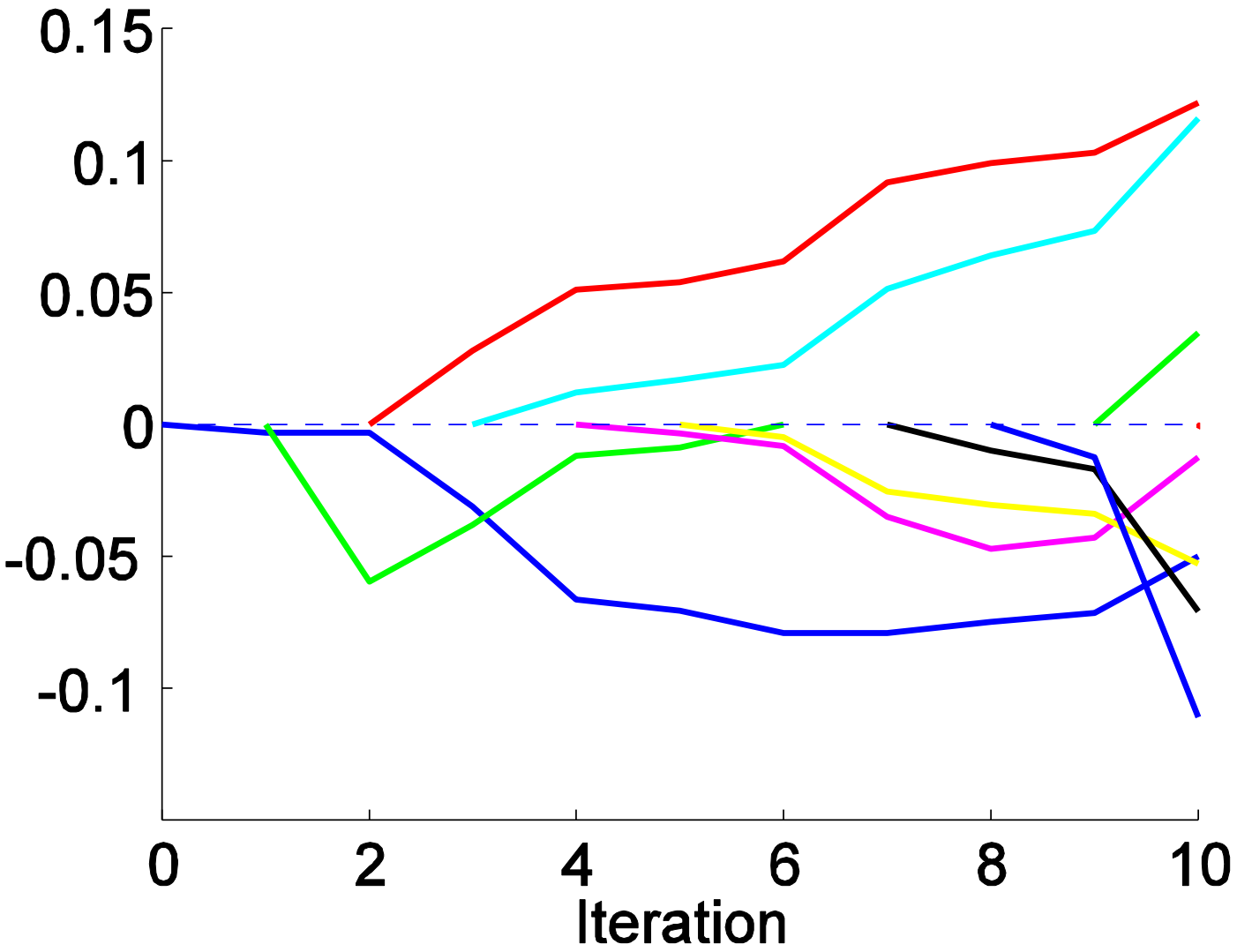
where

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

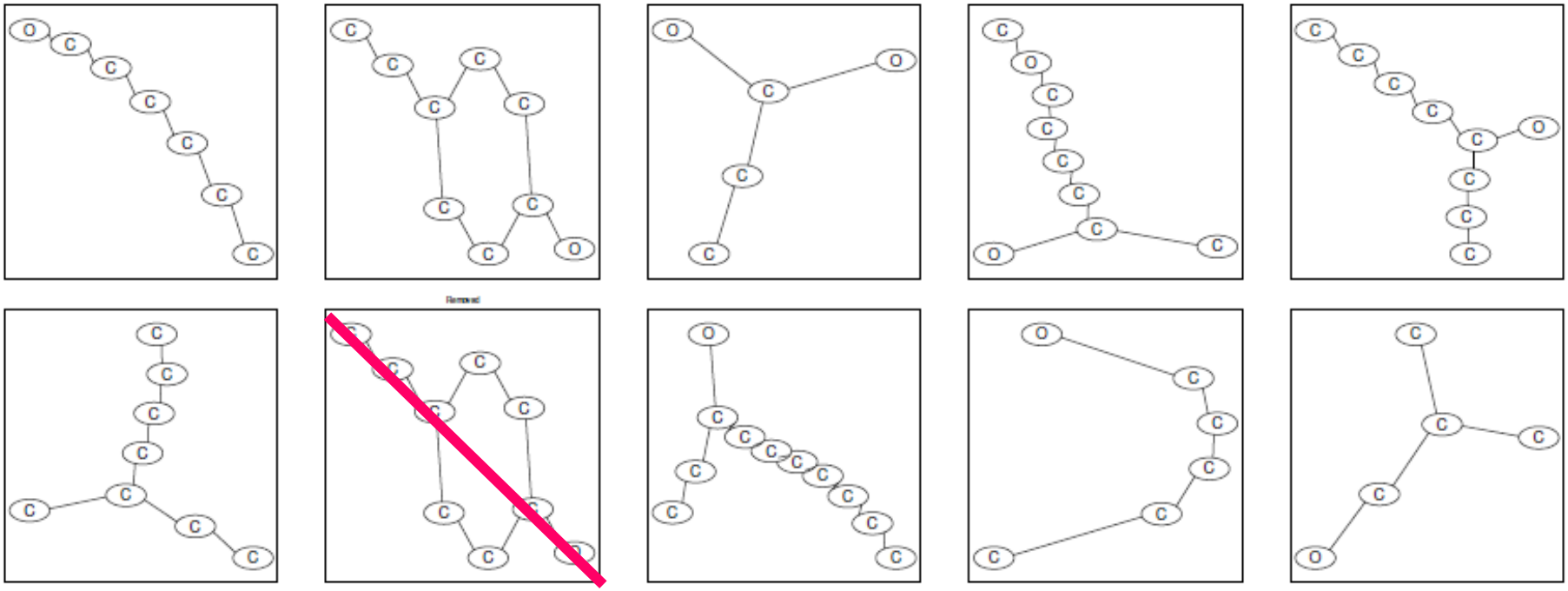
Initial Experiments

- EDKB Estrogen receptor database
  - 131 training graphs (chemical compounds)
- Computation Time: 4 sec/search
  - Pattern size limitation: 10 edges

Solution Path



Events

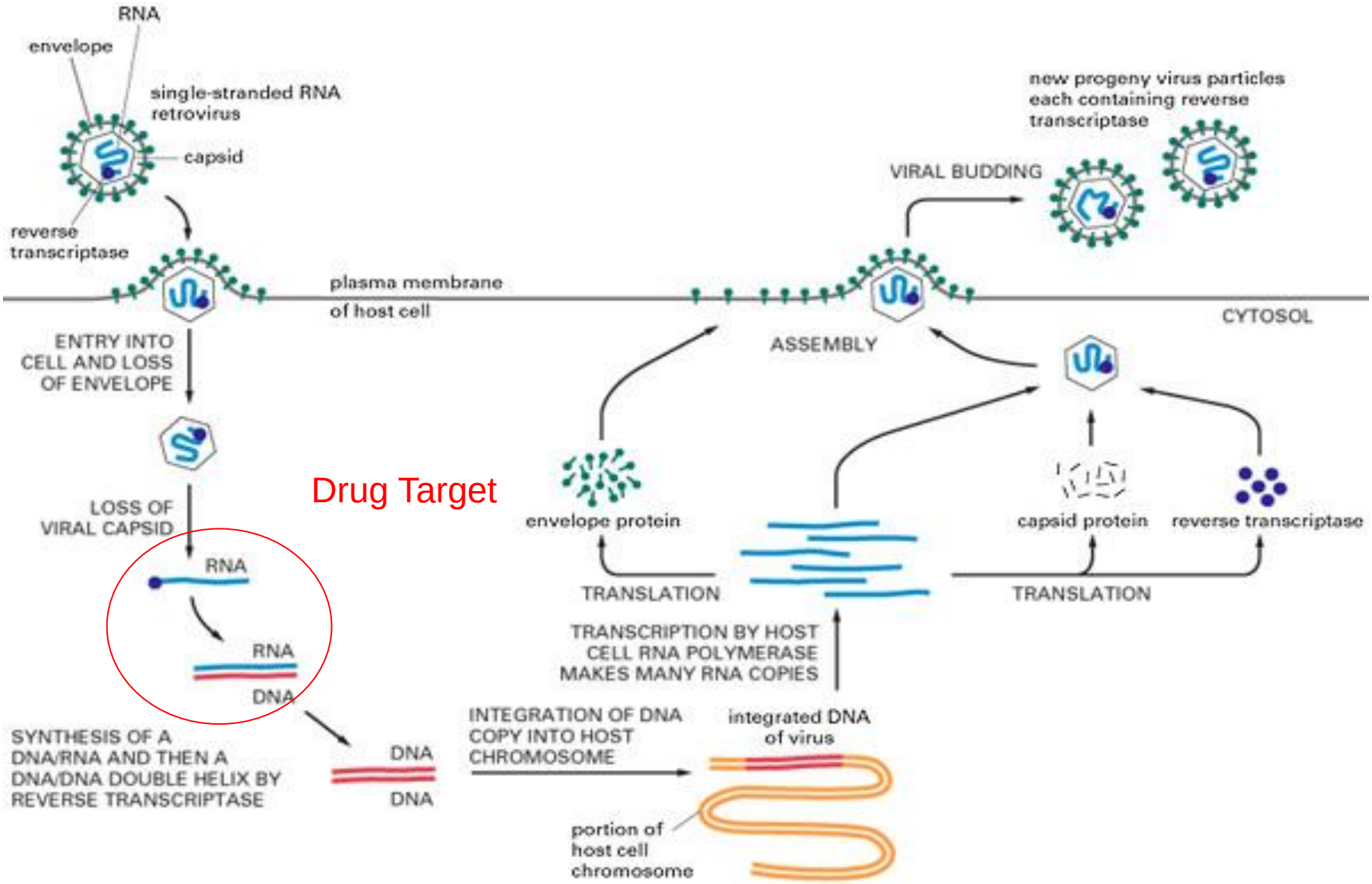


Summary: Regularization Path

- Path following implemented for graph data
- Pattern search by graph mining
- Classification: To do
- Combination with item set mining

Part 5: . Itemset Boosting for predicting HIV drug resistance

Life cycle of an HIV Virus



Approved HIV Drugs

- 8 Protease inhibitors (PI)
- 8 Nucleoside/nucleotide reverse transcriptase inhibitors (NRTI)
- 3 Non-nucleoside reverse transcriptase inhibitors (NNRTI)
- 1 Fusion inhibitor

Drug resistance of HIV

- Exposure to a drug causes mutations in HIV's genes
- As a result, HIV gains resistance against the drug

- Cost of identifying the genotypes of HIV in a patient is relatively cheap
- Predict the drug resistance from HIV's genotypes !
  - Effective Pharmacotherapy for individuals

Drug resistance prediction problem as regression

- Input: Set of mutations in a target protein
  - 41L: Amino acid at position 41 changed to L
- Output: Resistance against a drug (fold change)

(40F,41L,43E,210W,211K,215Y) ⇒ 0.8
(43Q,75M,122E,210W,211K) ⇒ 12.8
(75I, 77L, 116Y,151M,184V) ⇒ ?

Simple vs Complex Genotypic Features

- Simple genotypic features

(0, 1, 0, 1, 0, 0, 0, 1,…)

41L 62V F116Y

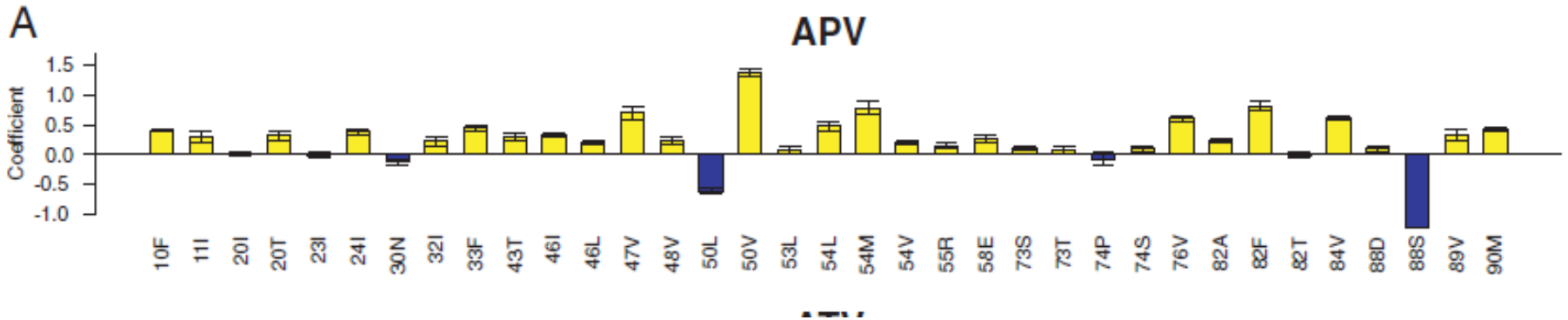
- Complex genotypic features

(0, 1, 0, 1, 0, 0, 0, 1,…)

77L,116Y 103N,210W,215Y

Linear Regression on Simple Genotypic Features (Rhee et al., PNAS 2006)

- Mutation associations not discovered



Complex Genotypic Features

- 0/1 vector of *pattern* indicators
- Huge dimensionality!
- Need itemset mining for selecting features
- Selection of salient features

(0, 1, 0, 1, 0, 0, 0, 1,…)

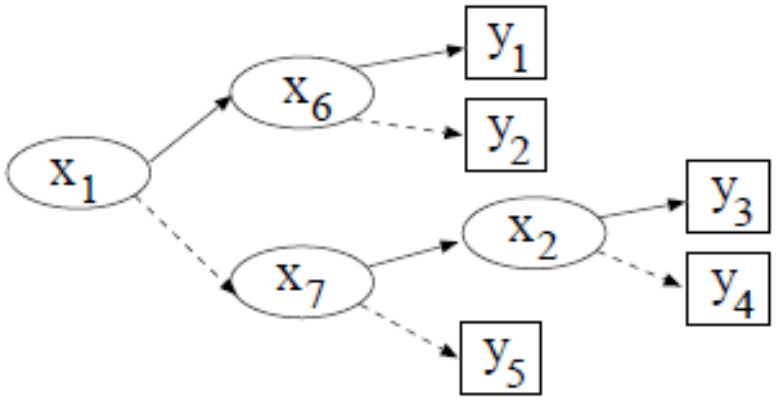
77L,116Y

103N,210W,215Y

patterns

Other methods

- Nonlinear SVM
  - Not interpretable
  - High accuracy
- Decision trees
  - Interpretable
  - Low accuracy



$$x_1 \wedge x_6 \Rightarrow y_1,\ x_1 \wedge \neg x_6 \Rightarrow y_2,\ \neg x_1 \wedge \neg x_7 \Rightarrow y_5$$

$$\neg x_1 \wedge x_7 \wedge x_2 \Rightarrow y_3,\ \neg x_1 \wedge x_7 \wedge \neg x_2 \Rightarrow y_4.$$

Motivation of Itemset Boosting

- Impossible to maintain all complex features
- Greedy feature discovery by Boosting + itemset mining
- Quadratic Programming (QP) Boosting
  - Reduce the number of itemset mining calls
  - Faster than AdaBoost
- Accurate prediction & interpretable results

Sparse classification in a very high dimensional space

- **G**: all possible patterns (intractably large)
- |G|-dimensional feature vector **x**
- Linear Classifier

$$f(\mathbf{x}) = \sum_{j=1}^{d} \alpha_j x_j$$

- Use L1 regularizer to have sparse α (LASSO)
- Select a tractable number of patterns

Problem formulation: Quadratic Programming

$$\min_{\boldsymbol{\alpha},\boldsymbol{\xi}} \quad ||\boldsymbol{\alpha}||_1 + \frac{C}{2}\sum_{n=1}^{\ell}\xi_n^2$$

$$s.t. \quad |y_n - \boldsymbol{\alpha}^\top \mathbf{x}_n| \leq \xi_n, \quad \xi_n \geq 0, \quad n = 1, \ldots, \ell$$

Sum of squared loss and L1 regularizer

$\{\boldsymbol{x}_n, y_n\}_{n=1}^{\ell}$: Training examples

ξ: slack variables

Dual QP

- Primal: Huge number of weight variables
- Dual: Huge number of constraints

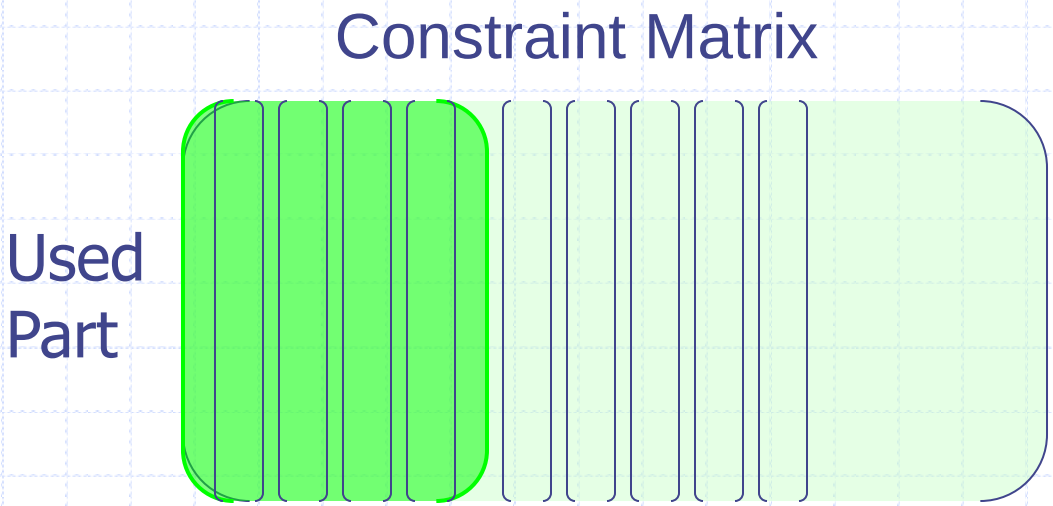
$$\min_{\mathbf{u}} \quad \frac{1}{2C}\sum_{n=1}^{\ell} u_n^2 - \sum_{n=1}^{\ell} y_n u_n$$

$$s.t. \quad -1 \le \sum_{n=1}^{\ell} u_n x_{ni} \le 1,\ i = 1, \ldots, d$$

$$\sum_{n=1}^{\ell} u_i = 0$$

Column Generation Algorithm for QP Boost (Demiriz et al., 2002)

- Start from the dual with no constraints
- Add the most violated constraint each time
- Guaranteed to converge



Finding the most violated constraint

- Constraint for a pattern (shown again)

$$\left| \sum_{n=1}^{\ell} u_n x_{ni} \right| \leq 1$$

- Finding the most violated one

$$\mathrm{argmax}_i \left| \sum_{n=1}^{\ell} u_n x_{ni} \right|$$

- Searched by weighted itemset mining

Algorithm Overview

- Iteration
  - Find a new pattern by graph mining with weight $\mathbf{u}$
  - If all constraints are satisfied, break
  - Add a new constraint
  - Update $\mathbf{u}$ by solving the dual problem
- Return
  - Convert dual solution to obtain primal solution $\mathbf{a}$

Experimental settings

- Three classes of drugs
  - NRTI (Nucleotide Reverse Transcriptase Inhibitors)
  - PI (Protease Inhibitors)
  - NNRTI (Nonnucleotide Reverse Transcriptase Inhibitors)
- 5fold cross validation
  - Linear SVM, Ridge regression, Lars
  - Nonlinear SVM, iBoost

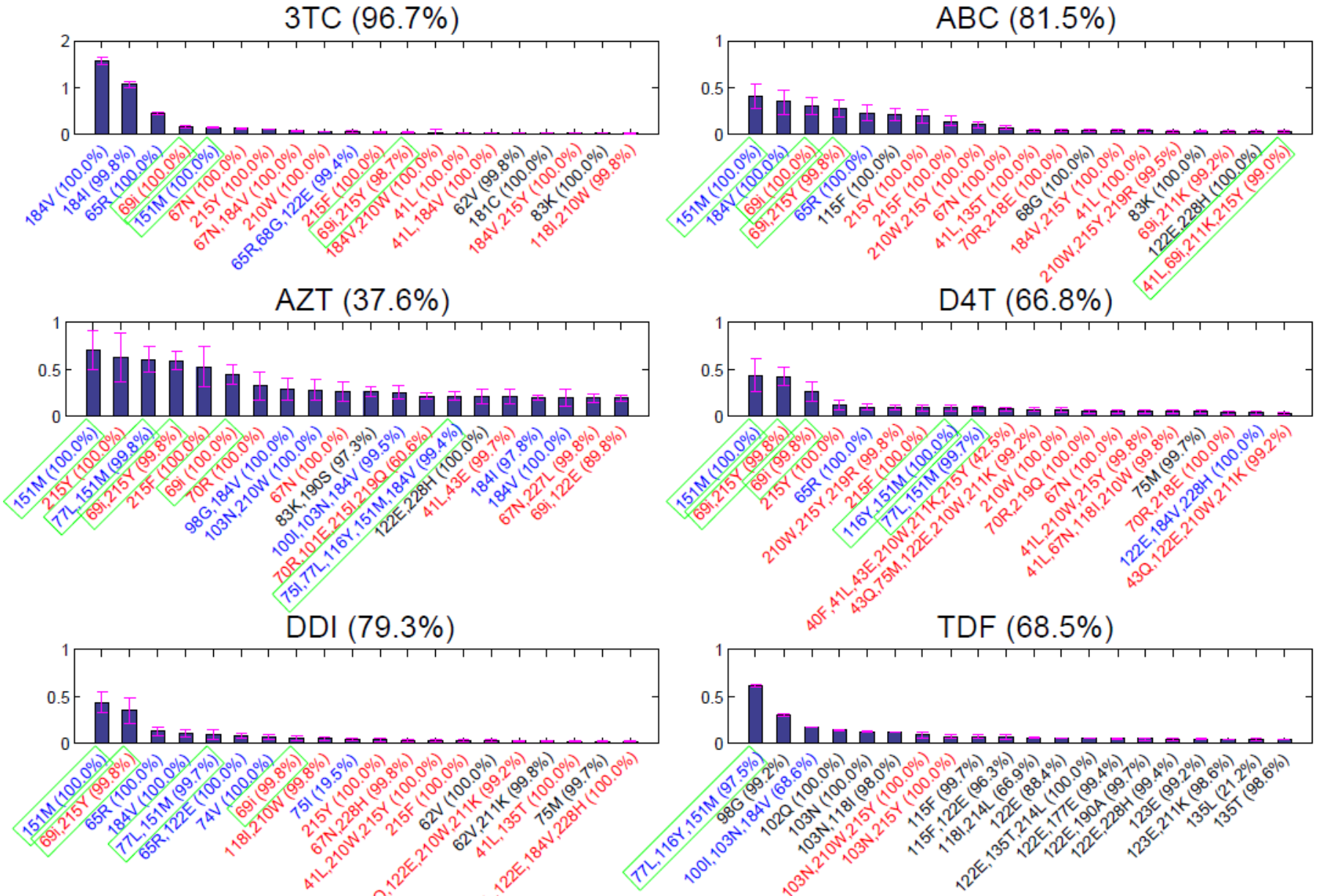
Regression results

Drug	# isolates	Linear Methods			Nonlinear Methods					
		SVR linear	Ridge	LARS	SVR p2	SVR p3	SVR r1	SVR r10	SVR r100	iBoost
NRTI										
Lamivudine (3TC)	633	0.913	0.753	0.93	0.927	0.876	0.306	0.934	0.608	**0.940**
Abacavir (ABC)	628	0.731	0.585	0.79	0.772	0.720	0.256	0.745	0. 614	**0.801**
Zidovudine (AZT)	630	0.751	0.72	0.65	**0.797**	0.760	0.240	0.754	0.515	**0.797**
Stavudine (D4T)	630	0.729	0.662	0.76	0.729	0.676	0.184	0.732	0.534	**0.789**
Didanosine (DDI)	632	0.695	0.591	0.72	0.690	0.653	0.170	0.705	0.372	**0.737**
Tenofovir (TDF)	353	0.603	0.568	0.40	0.554	0.465	0.107	**0.612**	0.366	0.552
Average		0.737	0.647	0.708	0.744	0.692	0.211	0.747	0.502	**0.769**
NNRTI										
Delavirdine (DLV)	732	0.799	0.794	0.79	0.729	0.684	0.189	**0.802**	0.323	0.771
Efavirenz (EFV)	734	0.793	0.772	**0.85**	0.730	0.640	0.170	0.797	0.205	0.771
Nevirapine (NVP)	746	0.757	0.719	**0.79**	0.704	0.592	0.166	0.765	0.181	0.781
Average		0.783	0.762	**0.810**	0.721	0.639	0.175	0.788	0.236	0.774
PI										
Amprenavir (APV)	768	0.819	0.749	0.81	0.756	0.697	0.483	**0.82**	0.576	0.802
Atazanavir (ATV)	329	0.724	0.594	**0.76**	0.731	0.654	0.297	0.739	0.450	0.701
Indinavir (IDV)	827	0.830	0.710	0.81	0.819	0.775	0.574	**0.844**	0.710	0.816
Lopinavir (LPV)	517	0.845	0.715	0.85	0.821	0.757	0.496	**0.857**	0.697	0.827
Nelfinavir (NFV)	844	0.845	0.697	0.84	0.800	0.728	0.615	**0.858**	0.638	0.838
Ritonavir (RTV)	795	0.893	0.794	0.89	0.875	0.820	0.598	**0.903**	0.788	0.892
Saquinavir (SQV)	826	0.814	0.738	0.81	0.782	0.733	0.468	**0.829**	0.559	0.791
Average		0.824	0.712	0.824	0.798	0.738	0.504	**0.836**	0.631	0.810

Accuracy: $r^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f(x_i))^2}{\sum_{i=1}^{n}(y_i - \frac{1}{n}\sum_{i=1}^{n} y_i)^2}.$

Accuracy Summary

- NRTIs: iBoost performed best
- PIs:
  - Nonlinear methods were better than linear
  - SVMs were slightly better (non-significant)
- NNRTIs
  - Linear methods were better
  - Combination is not necessary


3TC (96.7%)
184V (100.0%)
184I (99.8%)
65R (100.0%)
69i (100.0%)
151M (100.0%)
67N (100.0%)
215Y (100.0%)
67N,184V (100.0%)
210W (100.0%)
65R,68G,122E (99.4%)
215F (100.0%)
69i,215Y (98.7%)
184V,210W (100.0%)
41L (100.0%)
41L,184V (100.0%)
62V (99.8%)
181C (100.0%)
184V,215Y (100.0%)
83K (100.0%)
118I,210W (99.8%)
ABC (81.5%)
151M (100.0%)
184V (100.0%)
69i (100.0%)
69i,215Y (99.8%)
65R (100.0%)
115F (100.0%)
215Y (100.0%)
215F (100.0%)
210W,215Y (100.0%)
67N (100.0%)
41L,135T (100.0%)
70R,218E (100.0%)
68G (100.0%)
184V,215Y (100.0%)
41L (100.0%)
210W,215Y,219R (99.5%)
83K (100.0%)
69i,211K (99.2%)
122E,228H (100.0%)
41L,69i,211K,215Y (99.0%)
AZT (37.6%)
151M (100.0%)
215Y (100.0%)
77L,151M (99.8%)
69i,215Y (99.8%)
215F (100.0%)
69i (100.0%)
70R (100.0%)
98G,184V (100.0%)
103N,210W (100.0%)
67N (100.0%)
83K,190S (97.3%)
100I,103N,184V (99.5%)
70R,101E,215I,219Q (60.6%)
75I,77L,116Y,151M,184V (99.4%)
122E,228H (100.0%)
41L,43E (99.7%)
184I (97.8%)
184V (100.0%)
67N,227L (99.8%)
69i,122E (89.8%)
D4T (66.8%)
151M (100.0%)
69i,215Y (99.8%)
69i (99.8%)
215Y (100.0%)
65R (100.0%)
210W,215Y,219R (99.8%)
215F (100.0%)
116Y,151M (100.0%)
77L,151M (99.7%)
40F,41L,43E,210W,211K,215Y (42.5%)
43Q,75M,122E,210W,211K (99.2%)
210W (100.0%)
70R,219Q (100.0%)
67N (100.0%)
41L,210W,215Y (99.8%)
41L,67N,118I,210W (99.8%)
75M (99.7%)
70R,218E (100.0%)
122E,184V,228H (100.0%)
43Q,122E,210W,211K (99.2%)
DDI (79.3%)
151M (100.0%)
69i,215Y (99.8%)
65R (100.0%)
184V (100.0%)
77L,151M (99.7%)
65R,122E (100.0%)
74V (100.0%)
69i (99.8%)
118I,210W (99.8%)
75I (19.5%)
215Y (100.0%)
67N,228H (99.8%)
41L,210W,215Y (100.0%)
215F (100.0%)
43Q,122E,210W,211K (99.2%)
62V (100.0%)
62V,211K (99.8%)
41L,135T (100.0%)
75M (99.7%)
41L,122E,184V,228H (100.0%)
TDF (68.5%)
77L,116Y,151M (97.5%)
98G (99.2%)
100I,103N,184V (68.6%)
102Q (100.0%)
103N (100.0%)
103N,118I (98.0%)
103N,210W,215Y (100.0%)
103N,215Y (100.0%)
115F (99.7%)
115F,122E (96.3%)
118I,214L (66.9%)
122E (88.4%)
122E,135T,214L (100.0%)
122E,177E (99.4%)
122E,190A (99.7%)
122E,228H (99.4%)
123E (99.2%)
123E,211K (98.6%)
135L (21.2%)
135T (98.6%)

NRTI Drugs

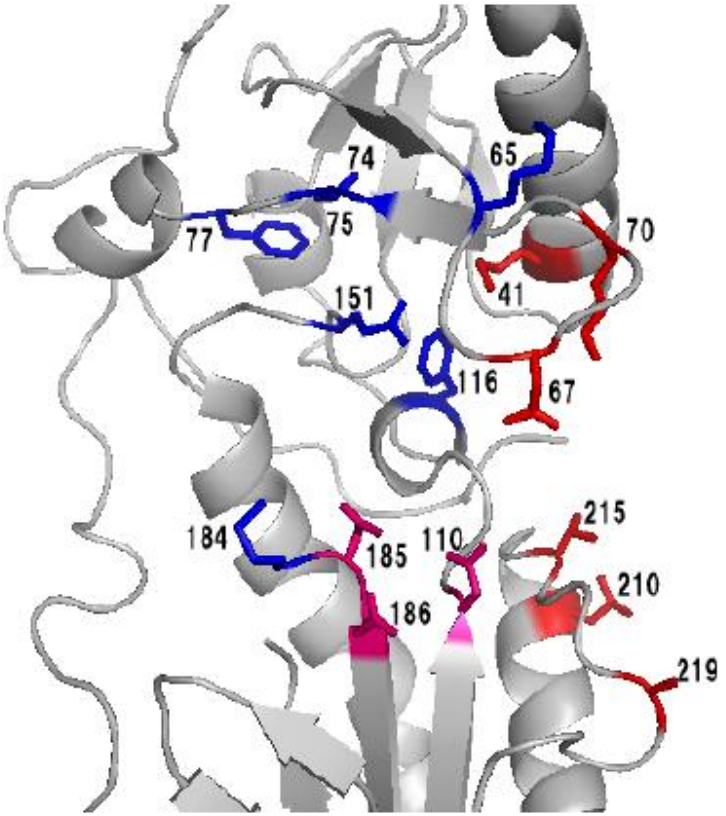
Known mutation associations in RT

Red: Thymidine-associated Mutations (TAM)

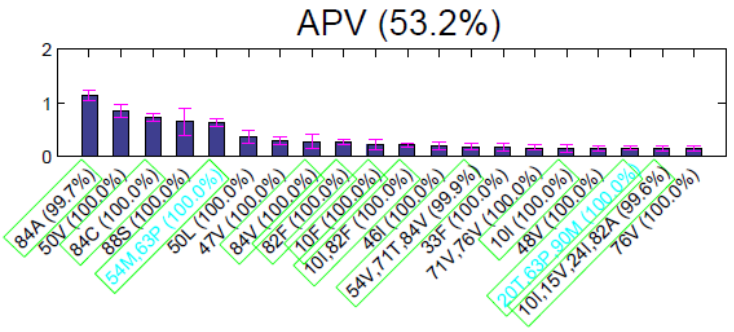
- 41L, 67N, 70R, 210W, 215Y/F, 219Q, 69i

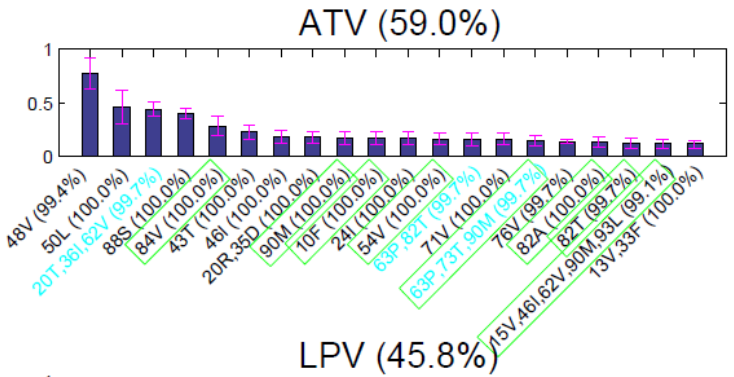
Blue: Q151M Complex

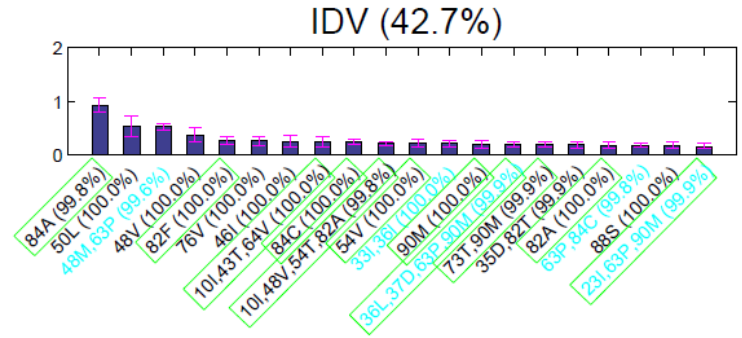
- 75I, 77L, 116Y, 151M, 65R, 74V, 184I/V

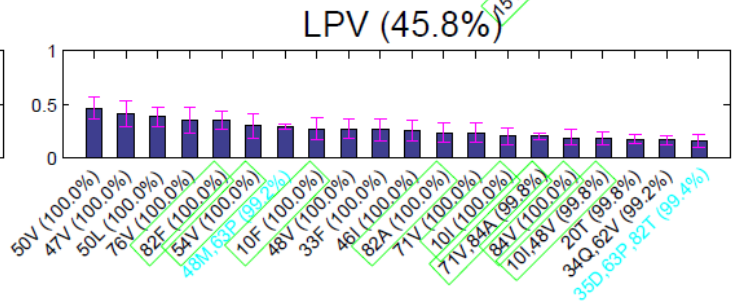


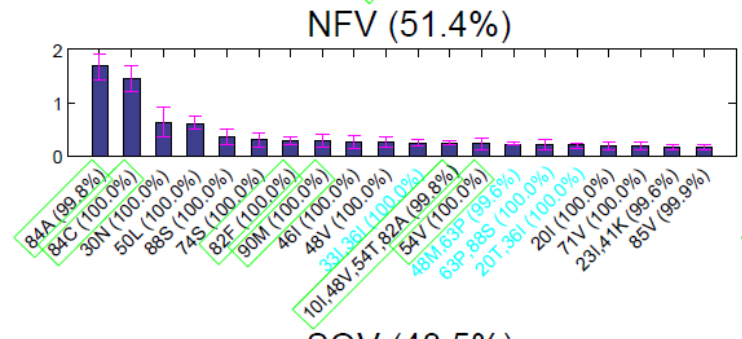
RT of HIV-1

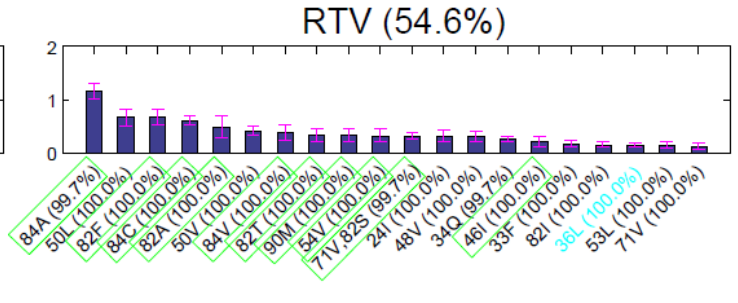

APV (53.2%)
84A (99.7%)
50V (100.0%)
84C (100.0%)
88S (100.0%)
54M,63P (100.0%)
50L (100.0%)
47V (100.0%)
84V (100.0%)
82F (100.0%)
10F (100.0%)
10I,82F (100.0%)
46I (100.0%)
54V,71T,84V (99.9%)
33F (100.0%)
71V,76V (100.0%)
10I (100.0%)
48V (100.0%)
20T,63P,90M (100.0%)
10I,15V,24I,82A (99.6%)
76V (100.0%)

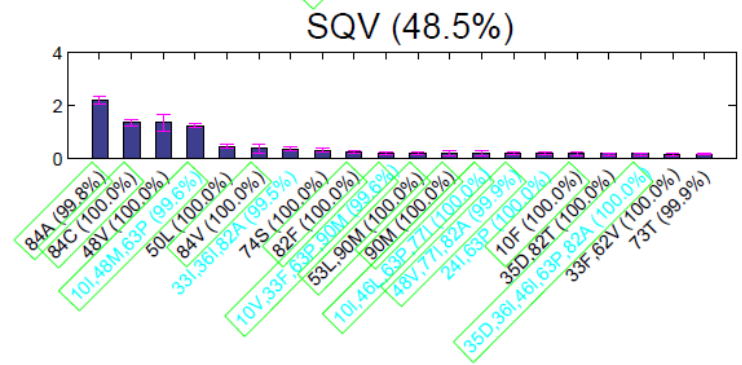

ATV (59.0%)
48V (99.4%)
50L (100.0%)
20T,36I,62V (99.7%)
88S (100.0%)
84V (100.0%)
43T (100.0%)
46I (100.0%)
20R,35D (100.0%)
90M (100.0%)
10F (100.0%)
24I (100.0%)
54V (100.0%)
63P,82T (99.7%)
71V (100.0%)
63P,73T,90M (99.7%)
76V (99.7%)
82A (100.0%)
82T (99.7%)
15V,46I,62V,90M,93L (99.1%)
13V,33F (100.0%)


IDV (42.7%)
84A (99.8%)
50L (100.0%)
48M,63P (99.6%)
48V (100.0%)
82F (100.0%)
76V (100.0%)
46I (100.0%)
10I,43T,64V (100.0%)
84C (100.0%)
10I,48V,54T,82A (99.8%)
54V (100.0%)
33I,36I (100.0%)
36L,37D,63P,90M (99.9%)
90M (100.0%)
73T,90M (99.9%)
35D,82T (99.9%)
82A (100.0%)
63P,84C (99.8%)
88S (100.0%)
23I,63P,90M (99.9%)

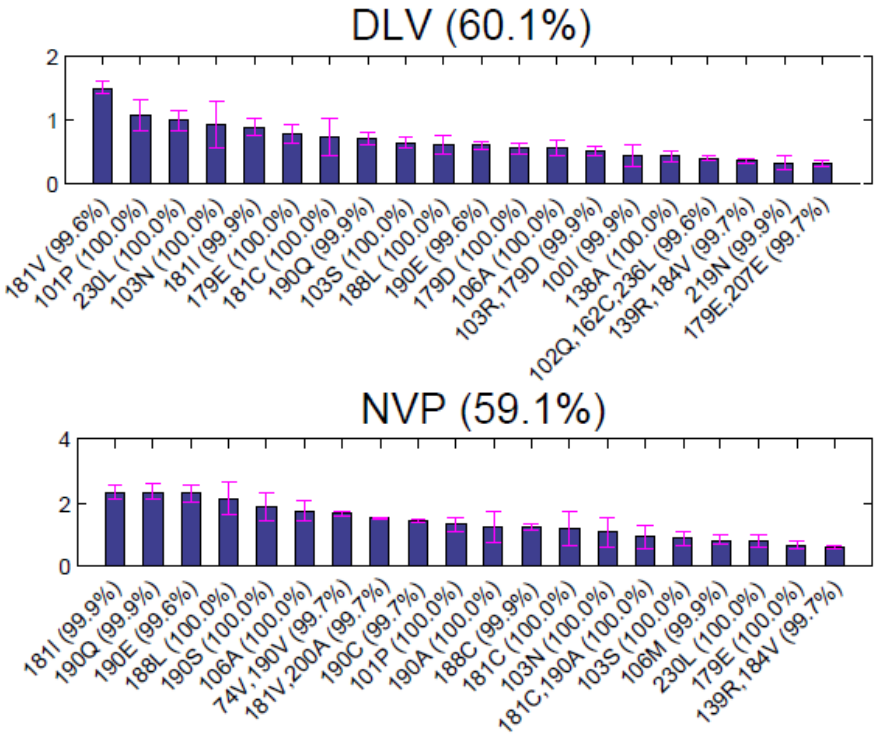

LPV (45.8%)
50V (100.0%)
47V (100.0%)
50L (100.0%)
76V (100.0%)
82F (100.0%)
54V (100.0%)
48M,63P (99.2%)
10F (100.0%)
48V (100.0%)
33F (100.0%)
46I (100.0%)
82A (100.0%)
71V (100.0%)
10I (100.0%)
71V,84A (99.8%)
84V (100.0%)
10I,48V (99.8%)
20T (99.8%)
34Q,62V (99.2%)
35D,63P,82T (99.4%)

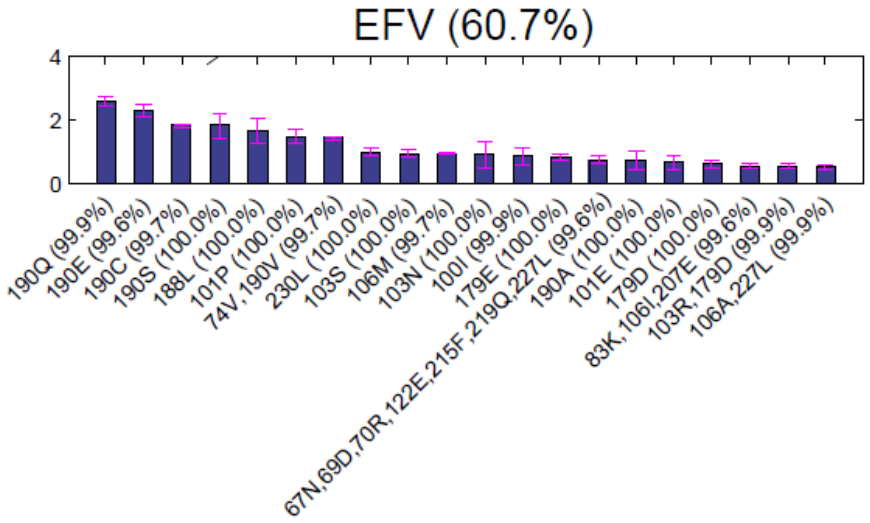

NFV (51.4%)
84A (99.8%)
84C (100.0%)
30N (100.0%)
50L (100.0%)
88S (100.0%)
74S (100.0%)
82F (100.0%)
90M (100.0%)
46I (100.0%)
48V (100.0%)
33I,36I (100.0%)
10I,48V,54T,82A (99.8%)
54V (100.0%)
46M,63P (99.6%)
63P,88S (100.0%)
20T,36I (100.0%)
20I (100.0%)
71V (100.0%)
23I,41K (99.6%)
85V (99.9%)


RTV (54.6%)
84A (99.7%)
50L (100.0%)
82F (100.0%)
84C (100.0%)
82A (100.0%)
50V (100.0%)
84V (100.0%)
82T (100.0%)
90M (100.0%)
54V (100.0%)
71V,82S (99.7%)
24I (100.0%)
48V (100.0%)
34Q (99.7%)
46I (100.0%)
33F (100.0%)
82I (100.0%)
36L (100.0%)
53L (100.0%)
71V (100.0%)


SQV (48.5%)
84A (99.8%)
84C (100.0%)
48V (100.0%)
10I,48M,63P (99.6%)
50L (100.0%)
84V (100.0%)
33I,36I,82A (99.5%)
74S (100.0%)
82F (100.0%)
10V,33F,63P,90M (99.6%)
53L,90M (100.0%)
90M (100.0%)
10I,46I,63P,77I (100.0%)
48V,77I,82A (99.9%)
24I,63P (100.0%)
10F (100.0%)
35D,82T (100.0%)
35D,36I,46I,63P,82A (100.0%)
33F,62V (100.0%)
73T (99.9%)

PI Drugs

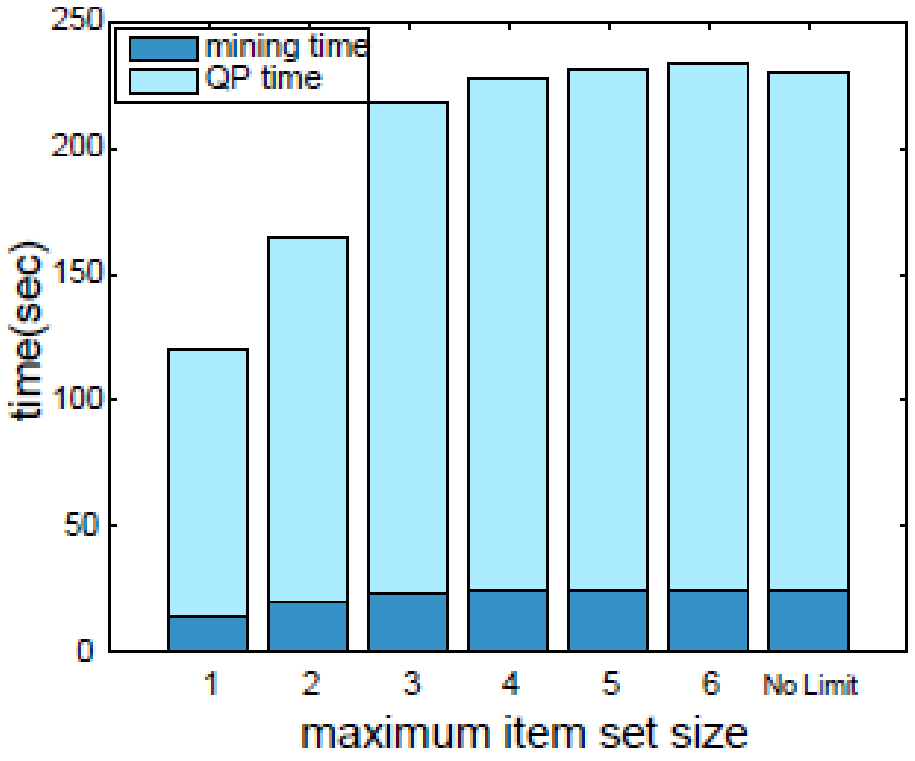




NNRTI Drugs
(almost no combination found)

Computation Time of iBoost

- Training time for 3TC
- 507 isolates with 371 mutations on average
- QP time longer than mining time



Summary (HIV)

- Itemset Boosting for finding mutation associations
- Good accuracy for NRTIs
- Our complex features re-discover known mutation clusters
- Broad applications
  - Multiple SNP analysis, RNAi efficacy prediction
  - Motif combination, Flu mutation analysis
  - P53 mutation analysis

Reference

- [1] K. Tsuda and T. Kudo, “Clustering graphs by weighted substructure mining,” in *Proceedings of the 23rd international conference on Machine learning - ICML ’06*, 2006, pp. 953–960.
- [2] H. Saigo, N. Krämer, and K. Tsuda, “Partial least squares regression for graph mining,” in *Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’08*, 2008, p. 578.
- [3] H. Saigo, S. Nowozin, T. Kadowaki, T. Kudo, and K. Tsuda, “gBoost: a mathematical programming approach to graph classification and regression,” *Machine Learning*, vol. 75, no. 1, pp. 69–89, Nov. 2008.
- [4] H. Saigo, T. Uno, and K. Tsuda, “Mining complex genotypic features for predicting HIV-1 drug resistance.,” *Bioinformatics (Oxford, England)*, vol. 23, no. 18, pp. 2455–62, Sep. 2007.
- [5] K. Tsuda, “Entire regularization paths for graph data,” in *Proceedings of the 24th international conference on Machine learning - ICML ’07*, 2007, pp. 919–926.

Machine Learning Summer School Kyoto 2012

Graph Mining

Chapter 3: Kernels for Structured Data

National Institute of Advanced Industrial Science and Technology

Koji Tsuda

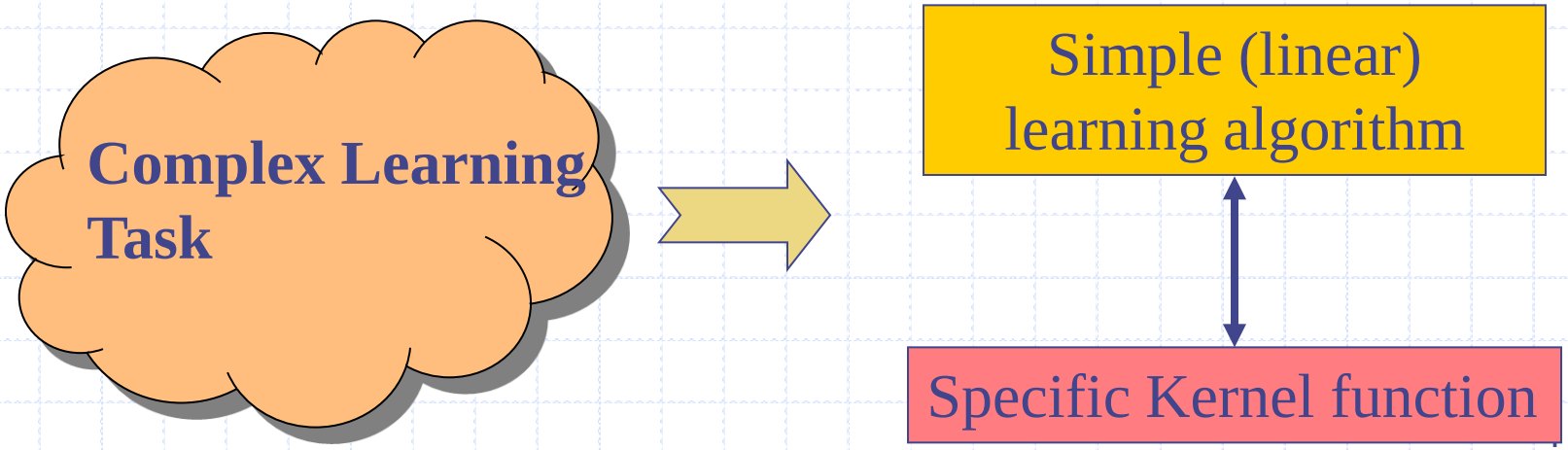
Agenda 3 (Kernel)

- 1. Kernel Method Revisited
- 2. Marginalized Kernels (Fisher Kernels)
- 3. Marginalized Graph Kernels
- 4. Weisfeiler-Lehman kernels
- 5. Reaction Graph kernels
- 6. Concluding Remark

Part 1: Kernel Method Revisited

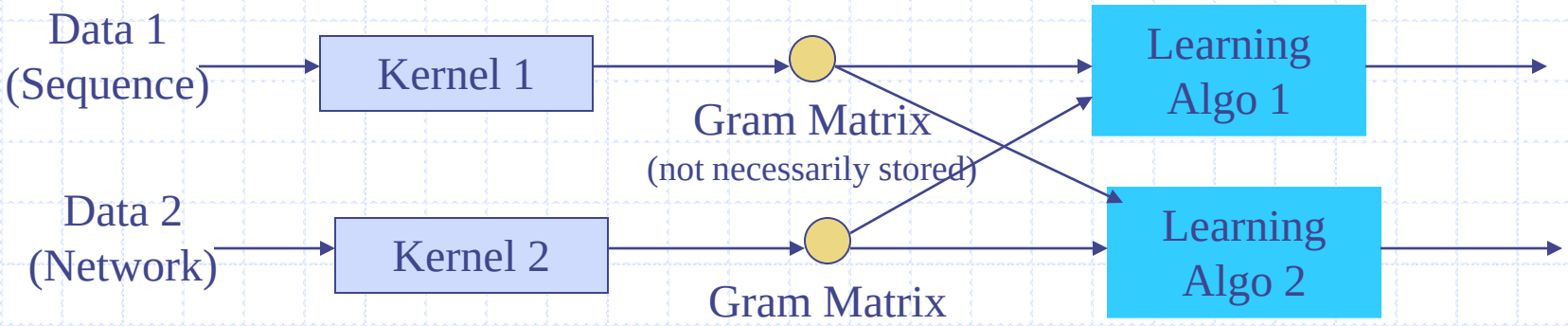
Kernels and Learning

- In Kernel-based learning algorithms, problem solving is now decoupled into:
  - A general purpose learning algorithm (e.g. SVM, PCA, …) – Often linear algorithm
  - A problem specific kernel



Current Synthesis

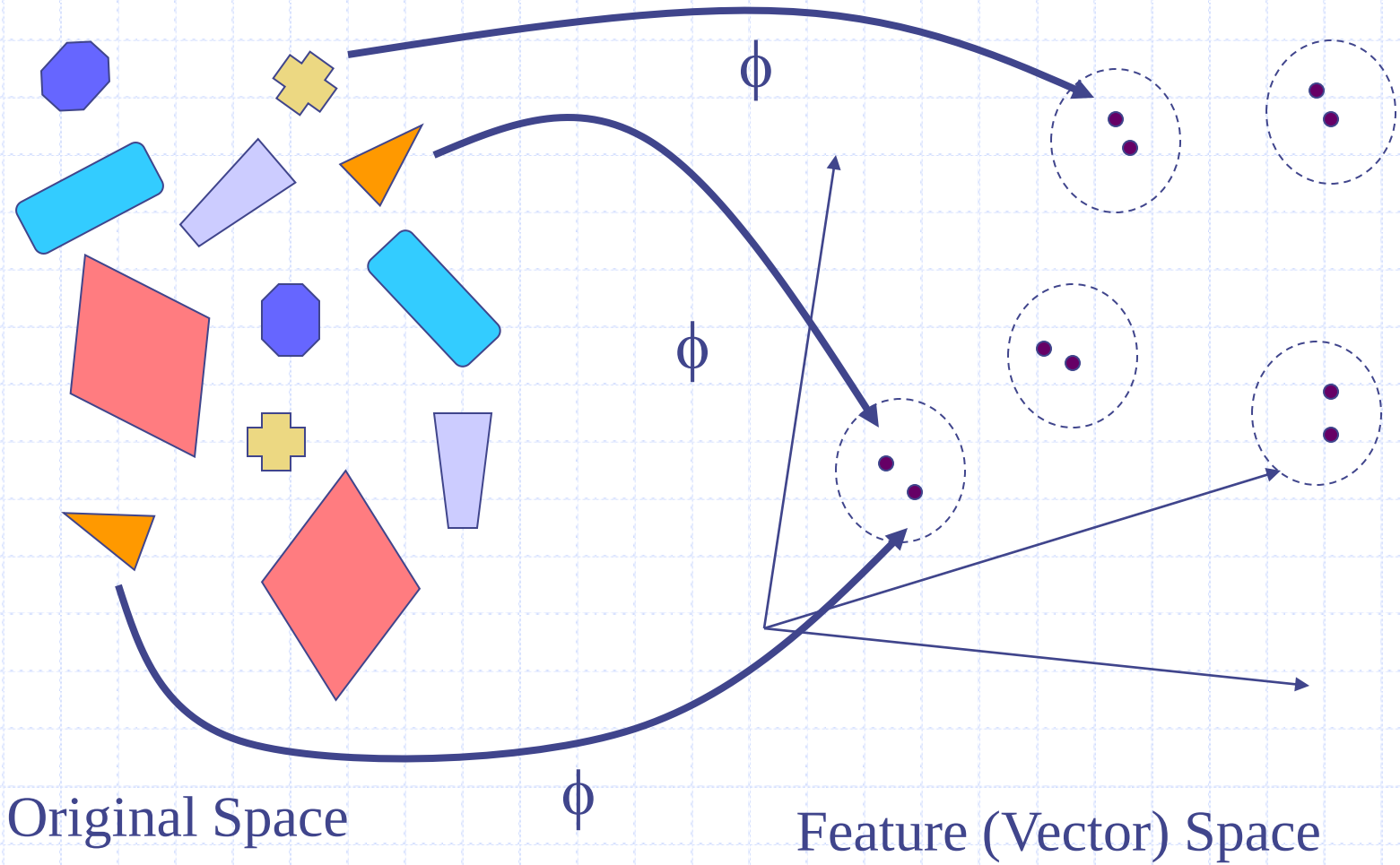
- Modularity and re-usability
  - Same kernel ,different learning algorithms
  - Different kernels, same learning algorithms



Kernel Methods : intuitive idea

- Find a mapping ϕ such that, in the new space, problem solving is linear
- Kernel represents the similarity between two objects, defined as the dot-product in this new vector space
- But the mapping is left implicit
- Easy generalization of a lot of dot-product-based learning algorithms

Kernel Methods : the mapping



Kernel : more formal definition

- A kernel k(x,y)
  - is a similarity measure
  - defined by an implicit mapping ϕ,
  - such that: $k(x,y)=\phi(x)\bullet\phi(y)$
- This similarity measure implies:
  - Invariance or other a priori knowledge
  - The class of functions the solution is taken from
  - Possibly infinite dimension (hypothesis space for learning)
  - … but still computational efficiency when computing $k(x,y)$

Kernel Trick

- Generalizes (nonlinearly) algorithms in clustering, classification, density estimation ..
  - When these algorithms are dot-product based, by replacing the dot product $(x \cdot y)$ by $k(x,y)=\phi(x)\cdot\phi(y)$
  - When these algorithms are distance-based, by replacing $d(x,y)$ by $k(x,x)+k(y,y)-2k(x,y)$
- Freedom of choosing ϕ implies a large variety of learning algorithms

Valid Kernels

- Theorem: *k*(x,y) is a valid kernel if *k* is positive definite and symmetric (Mercer Kernel)
  - A function is P.D. if $\int K(\mathbf{x},\mathbf{y}) f(\mathbf{x}) f(\mathbf{y}) d\mathbf{x} d\mathbf{y} \geq 0 \quad \forall f \in L_2$
  - In other words, the Gram matrix **K** (whose elements are $k(x_i,x_j)$) must be positive definite for all x_i, x_j of the input space
  - One possible choice of $\phi(x)$: $k(\bullet,x)$ (maps a point x to a function $k(\bullet,x)$ → feature space with infinite dimension!)

How to build new kernels

- Kernel combinations, preserving *validity*:

$$K(\mathbf{x},\mathbf{y}) = \lambda K_1(\mathbf{x},\mathbf{y}) + (1-\lambda)K_2(\mathbf{x},\mathbf{y}) \quad 0 \le \lambda \le 1$$

$$K(\mathbf{x},\mathbf{y}) = a.K_1(\mathbf{x},\mathbf{y}) \quad a > 0$$

$$K(\mathbf{x},\mathbf{y}) = K_1(\mathbf{x},\mathbf{y}).K_2(\mathbf{x},\mathbf{y})$$

$$K(\mathbf{x},\mathbf{y}) = f(x).f(y) \quad f \text{ is real-valued function}$$

$$K(\mathbf{x},\mathbf{y}) = K_3(\boldsymbol{\varphi}(\mathbf{x}),\boldsymbol{\varphi}(\mathbf{y}))$$

$$K(\mathbf{x},\mathbf{y}) = \mathbf{x}'P\mathbf{y} \quad P \text{ symmetric definite positive}$$

$$K(\mathbf{x},\mathbf{y}) = \frac{K_1(\mathbf{x},\mathbf{y})}{\sqrt{K_1(\mathbf{x},\mathbf{x})}\sqrt{K_1(\mathbf{y},\mathbf{y})}}$$

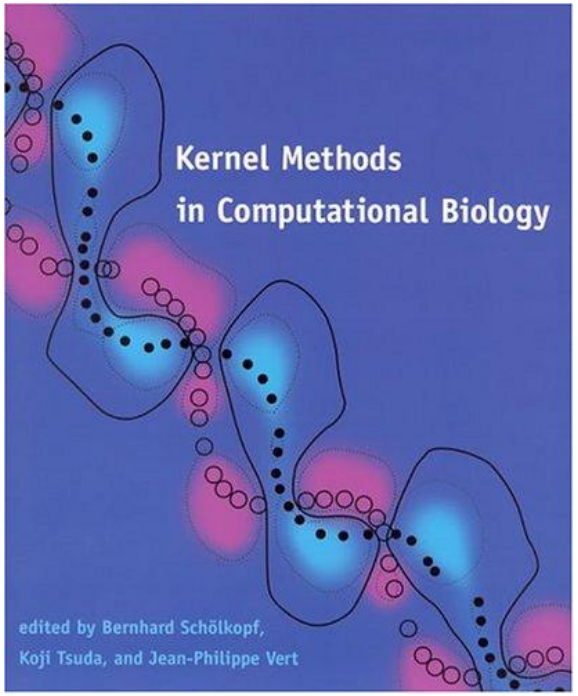
Strategies of Design

- Convolution Kernels: text is a recursively-defined data structure. How to build "global" kernels form local (atomic level) kernels?
- Generative model-based kernels: the "topology" of the problem will be translated into a kernel function

Family of kernels

- **Kernels for biological sequences**
  - Spectrum kernel
  - Marginalized kernel
  - Profile kernel
  - Local alignment kernel

- **Tree Kernels**
  - Kernel for phylogenetic profiles
  - Kernel for natural language
  - Kernel for RNA sequences



Kernel Methods in Computational Biology, MIT Press

Family of kernels

- **Kernels for nodes in a network**
  - Diffusion kernel
  - Locally constrained diffusion kernel

- **Graph Kernels**
  - Marginalized Graph Kernels
  - MGK without tottering
  - Acyclic Pattern Kernels
  - Shortest Path Kernel
  - Weisfeiler-Lehman Kernel

Weak points of kernel methods

- Not Interpretable
  - Not sure which features are used
  - -> Graph Mining, Boosting
- Dense kernel matrices: Slow
  - Take O(n^3) time for manipulation
  - -> Semi-supervised learning

Part 2. Marginalized kernels

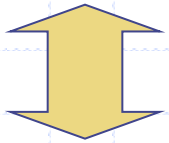
Biological Sequences: Classification Tasks

- DNA sequences (A,C,G,T)
  - Gene Finding, Splice Sites
- RNA sequences (A,C,G,U)
  - MicroRNA discovery, Classification into Rfam families
- Amino Acid Sequences (20 symbols)
  - Remote Homolog Detection, Fold recognition

Kernels for Sequences

- Similarity between sequences of different lengths

ACGGTTCAA



ATATCGCGGGAA

- Later combined with SVMs and other kernel methods

Count Kernel

- Inner product between symbol counts

	A	C	G	T
ACGGTTCAA	3	2	2	2
ATATCGCGGGAA	4	2	4	2

- Extension: Spectrum kernels (Leslie et al., 2002)
  - Counts the number of k-mers (k-grams) efficiently

- Not good for sequences with frequent context change
  - E.g., coding/non-coding regions in DNA

Hidden Markov Models for Estimating Context

- Visible Variable $\boldsymbol{x} = (x_1, \ldots, x_m)$: Symbol Sequence
- Hidden Variable $\boldsymbol{h} = (h_1, \ldots, h_m)$: Context
- HMM can estimate the posterior probability of hidden variables $p(\boldsymbol{h}|\boldsymbol{x})$ from data

h: 1 2 2 1 2 2 1 2 2

x: A C G G T T C A A

Marginalized kernels

- Design a joint kernel $K_z(\boldsymbol{z}, \boldsymbol{z}')$ for combined $\boldsymbol{z} = (\boldsymbol{x}, \boldsymbol{h})$
  - Hidden variable is not usually available
  - Take expectation with respect to the hidden variable
- The marginalized kernel for visible variables

$$K(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h} \in \mathcal{H}} \sum_{\boldsymbol{h}' \in \mathcal{H}} p(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{h}'|\boldsymbol{x}') K_z(\boldsymbol{z}, \boldsymbol{z}')$$

Designing a joint kernel for sequences

- Symbols are counted separately in each context

h: 1 2 2 1 2 2 1 2 2

x: A C G G T T C A A

(A,1) = 1 (C,1) = 1 (G,1) = 1 (T,1) = 0

(A,2) = 2 (C,2) = 1 (G,2) = 1 (T,2) = 2

- $c_{k\ell}(\boldsymbol{z})$:count of a combined symbol (k,l)

- Joint kernel: count kernel with context information

$$K_z(\boldsymbol{z}, \boldsymbol{z}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} c_{k\ell}(\boldsymbol{z}) c_{k\ell}(\boldsymbol{z}')$$

Marginalization of the joint kernel

- Joint kernel

$$K_z(\boldsymbol{z}, \boldsymbol{z}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} c_{k\ell}(\boldsymbol{z}) c_{k\ell}(\boldsymbol{z}')$$

- Marginalized count kernel

$$\begin{aligned} K(\boldsymbol{x}, \boldsymbol{x}') &= \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} p(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{h}'|\boldsymbol{x}') K_z(\boldsymbol{z}, \boldsymbol{z}') \\ &= \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \gamma_{kl}(\boldsymbol{x}) \gamma_{kl}(\boldsymbol{x}') \end{aligned}$$

γ_{kl} is a marginalized count $\gamma_{kl}(\boldsymbol{x}) = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{x}) c_{k\ell}(\boldsymbol{z})$

Computing Marginalized Counts from HMM

- Marginalized count is described as

$$\gamma_{kl}(\boldsymbol{x}) = \frac{1}{m} \sum_{i=1}^{m} \sum_{h_i=1}^{n_h} p(h_i|\boldsymbol{x}) I(x_i = k, h_i = \ell).$$

- Posterior probability of i-th hidden variable is efficiently computed as

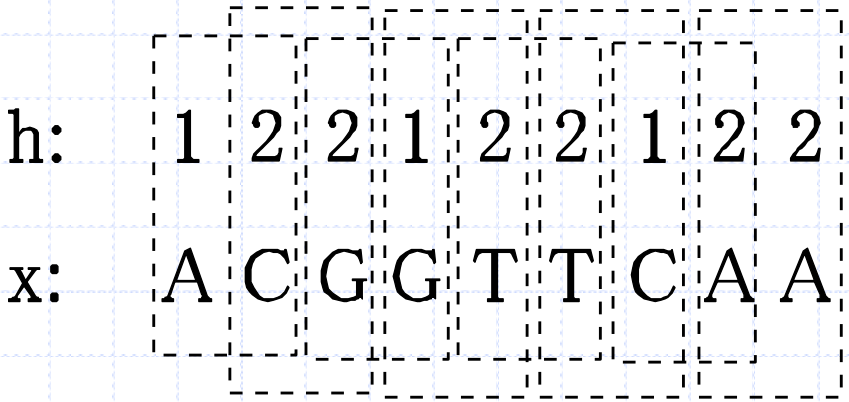
$$p(h_i = k|\boldsymbol{x}) = f_k(i) b_k(i) / p(\boldsymbol{x})$$

$f_k(i)$: forward variable, $b_k(i)$: backward variable

2nd order marginalized count kernel

- If adjacent relations between symbols have essential meanings, the count kernel is obviously not sufficient

- 2nd order marginalized count kernel
  - 4 neighboring symbols (i.e. 2 visible and 2 hidden) are combined and counted



Fisher Kernel

- Probabilistic mode $p(\boldsymbol{x}|\hat{\boldsymbol{\theta}}),\ \boldsymbol{x} \in \mathcal{X}$

  $\hat{\boldsymbol{\theta}} \in \Re^r$ is a parameter vector obtained from training samples

- Fisher Kernel
  - Mapping to a feature vector (Fisher score vector)

$$\boldsymbol{s}(\boldsymbol{x}, \hat{\boldsymbol{\theta}}) = \left(\frac{\partial \log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})}{\partial \theta_1}, \ldots, \frac{\partial \log p(\boldsymbol{x}|\hat{\boldsymbol{\theta}})}{\partial \theta_p}\right)$$

  - Inner product of Fisher scores

$$K_f(\boldsymbol{x}, \boldsymbol{x}') = \boldsymbol{s}(\boldsymbol{x}, \hat{\boldsymbol{\theta}})^\top Z^{-1}(\hat{\boldsymbol{\theta}}) \boldsymbol{s}(\boldsymbol{x}', \hat{\boldsymbol{\theta}})$$

Z: Fisher information matrix

Fisher Kernel from HMM

- Derive FK from HMM (Jaakkola et al. 2000)
  - Derivative for emission probabilities only
  - No Fisher information matrix
- FK from HMM is a special case of marginalized kernels
  - Counts are centralized and weighted

$$K_f(\boldsymbol{x}, \boldsymbol{x}') = \sum_{\boldsymbol{h}} \sum_{\boldsymbol{h}'} p(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{h}'|\boldsymbol{x}') K_{fz}(\boldsymbol{z}, \boldsymbol{z}')$$

$$K_{fz}(\boldsymbol{z}, \boldsymbol{z}') = \sum_{k=1}^{n_x} \sum_{\ell=1}^{n_h} \frac{1}{e_{\ell k}^2} (c_{k\ell}(\boldsymbol{z}) - \bar{c}_{k\ell}(\boldsymbol{z}))(c_{k\ell}(\boldsymbol{z}') - \bar{c}_{k\ell}(\boldsymbol{z}'))$$

Difference between FK and Marginalized Kernels

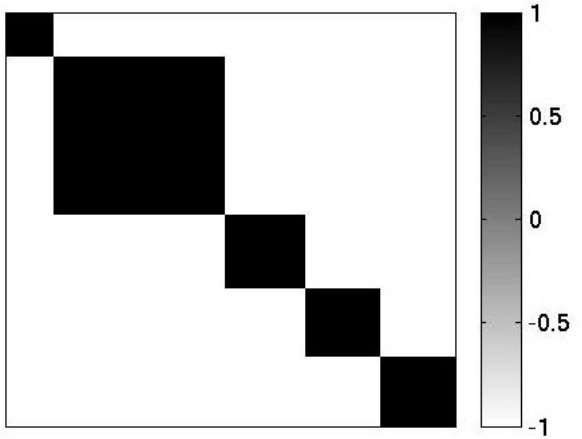
- FK: Probabilistic model determines the joint kernel and the posterior probabilities
- MK: You can determine them separately
  - More flexible design!

Protein clustering experiment

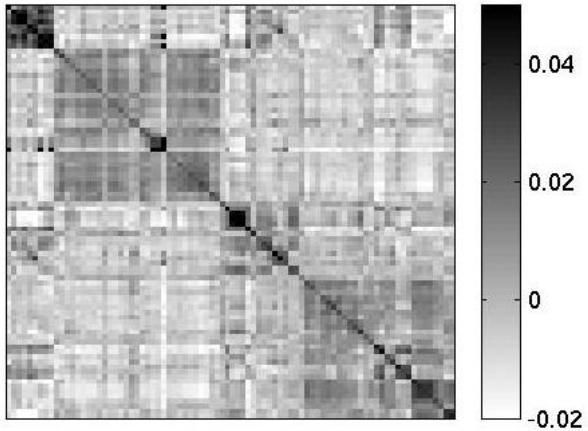
- 84 proteins containing five classes
  - gyrB proteins from five bacteria species
- Clustering methods
  - HMM + {FK,MCK1,MCK2}+K-Means
- Evaluation
  - Adjusted Rand Index (ARI)

Kernel Matrices

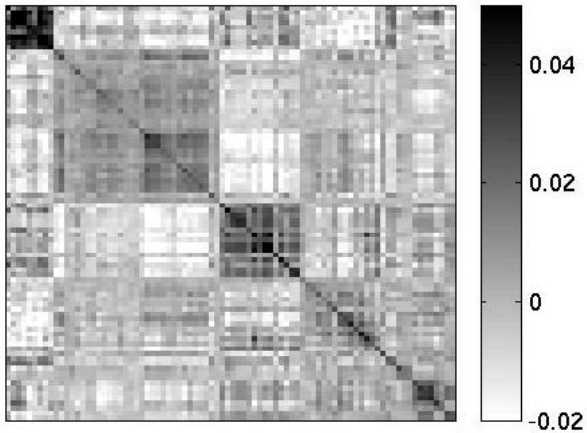
Ideal



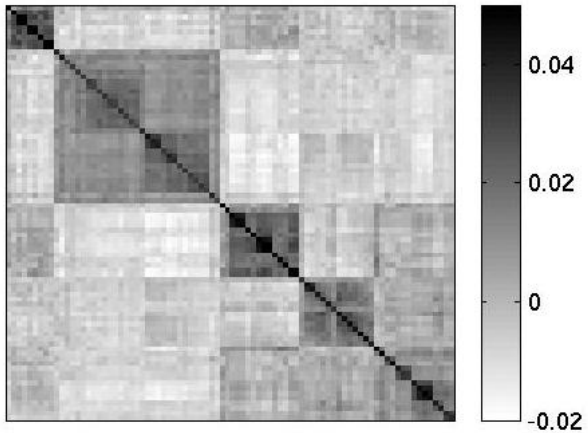
FK



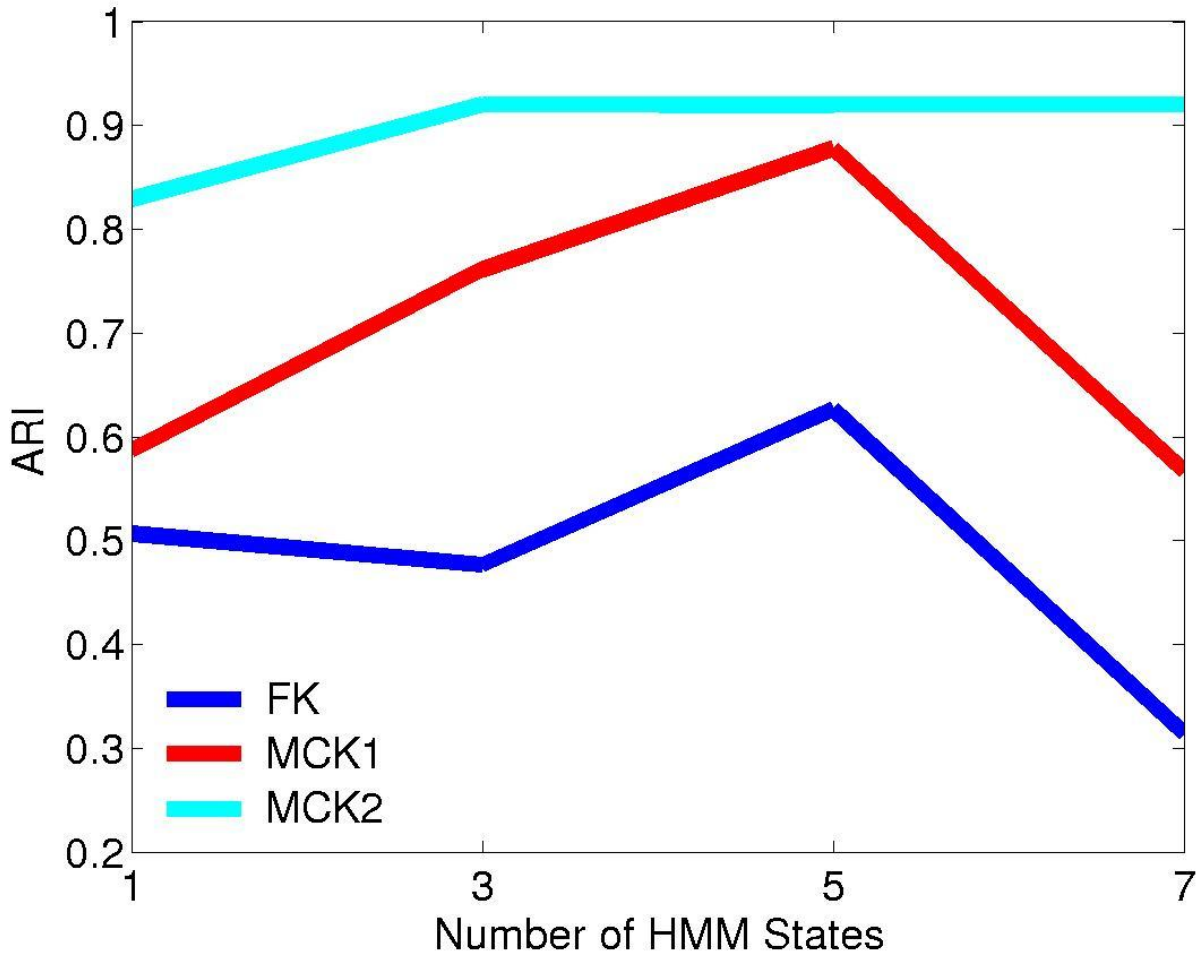
MCK1



MCK2



Clustering Evaluation



Applications since then..

- Marginalized Graph Kernels (Kashima et al., ICML 2003)
- Sensor networks (Nyugen et al., ICML 2004)
- Labeling of structured data (Kashima et al., ICML 2004)
- Robotics (Shimosaka et al., ICRA 2005)
- Kernels for Promoter Regions (Vert et al., NIPS 2005)
- Web data (Zhao et al., WWW 2006)

Summary (Marginalized Kernels)

- General Framework for using generative model for defining kernels
- Fisher kernel as a special case
- Broad applications

Part 3 Marginalized Graph Kernels

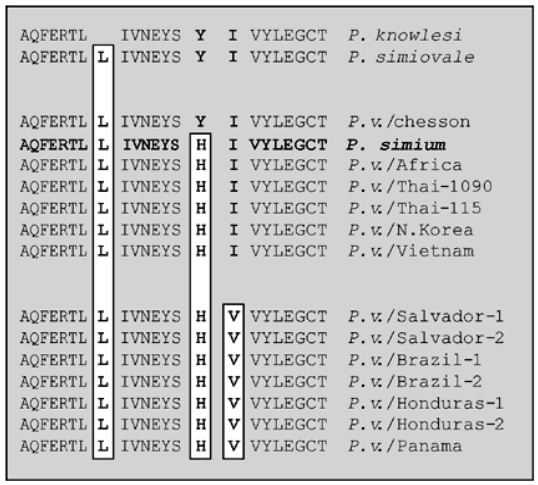
Motivations for graph analysis

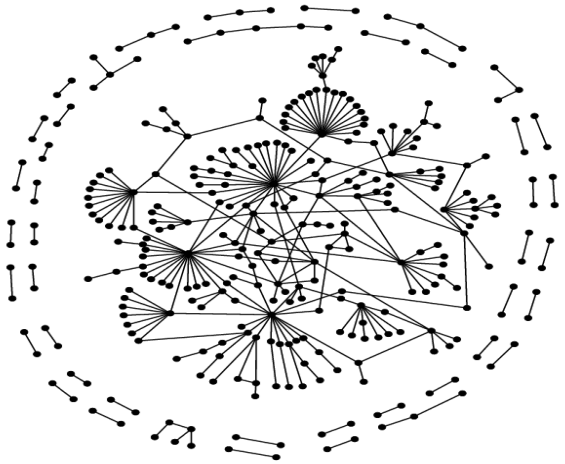
- Existing methods assume ” tables”

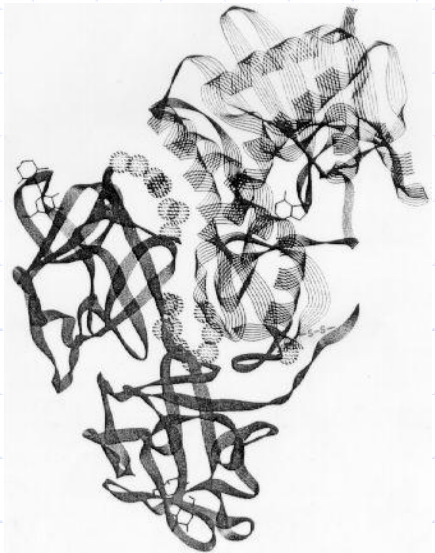
Serial Num	Name	Age	Sex	Address	…
0001	○○	40	Male	Tokyo	…
0002	× ×	31	Female	Osaka	…

- Structured data beyond this framework

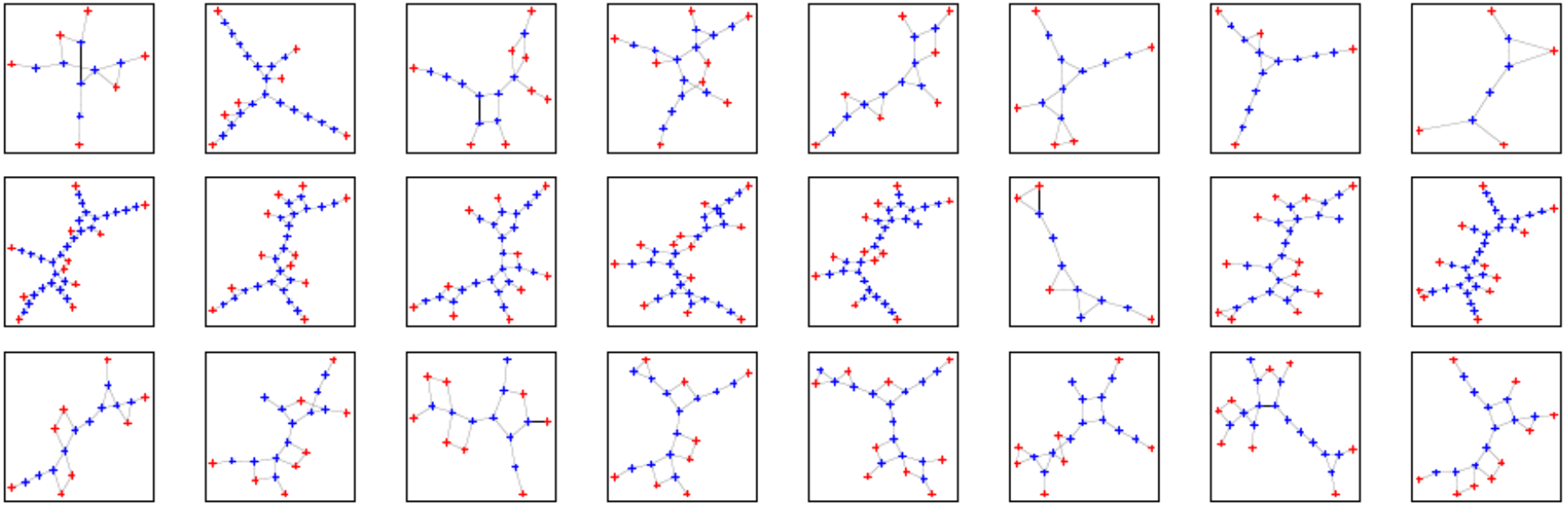
→ New methods for analysis







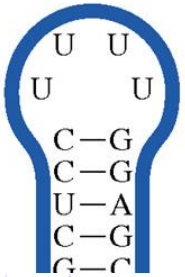
Graphs..

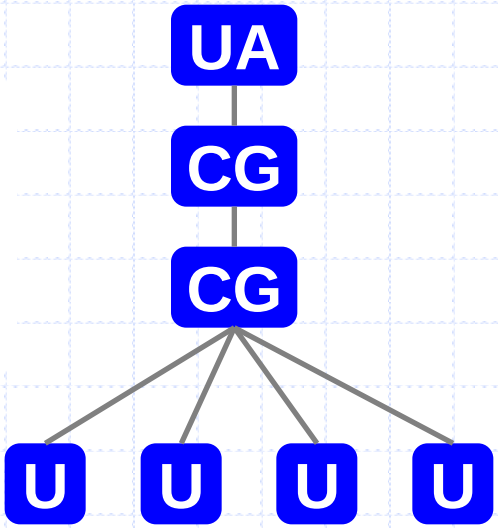


Graph Structures in Biology

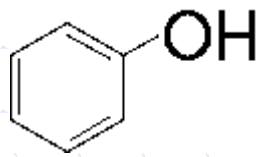
- DNA Sequence

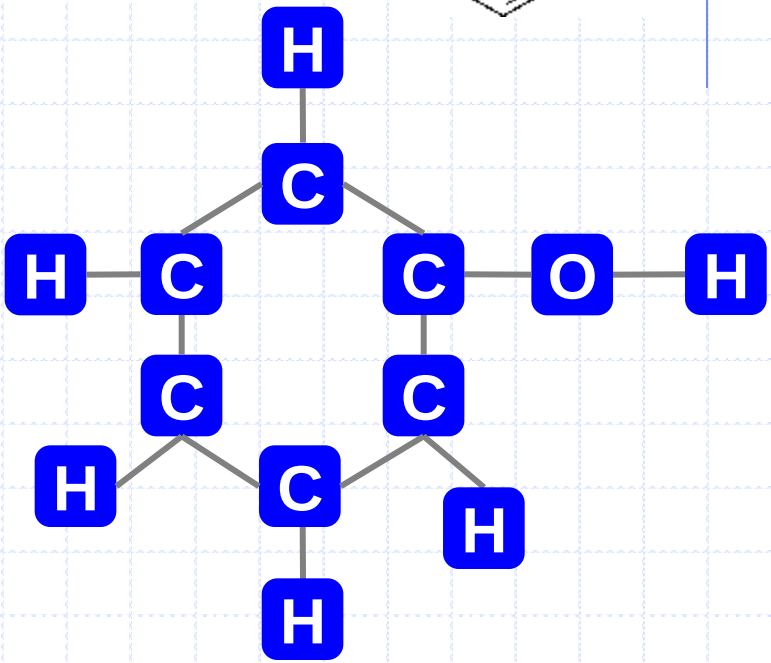
- RNA



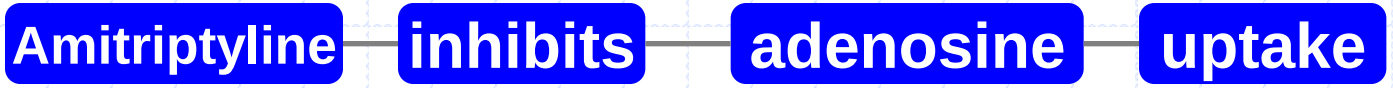


- Compounds





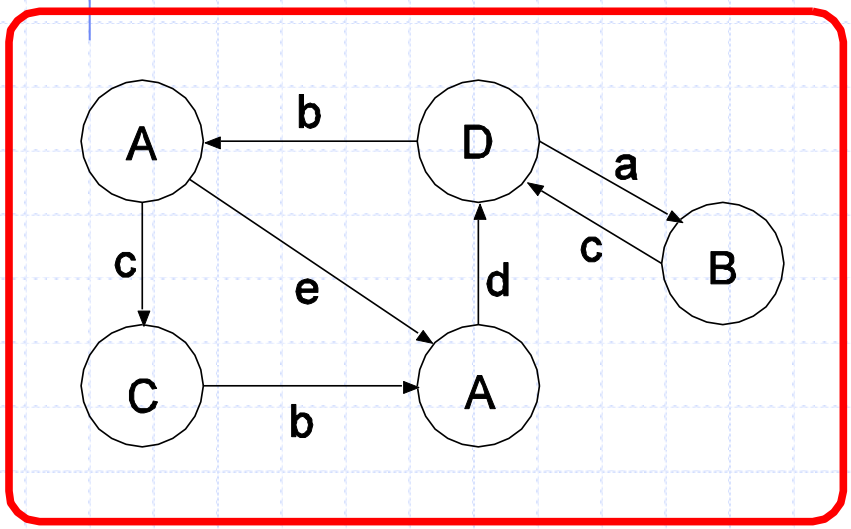
- Texts in literature



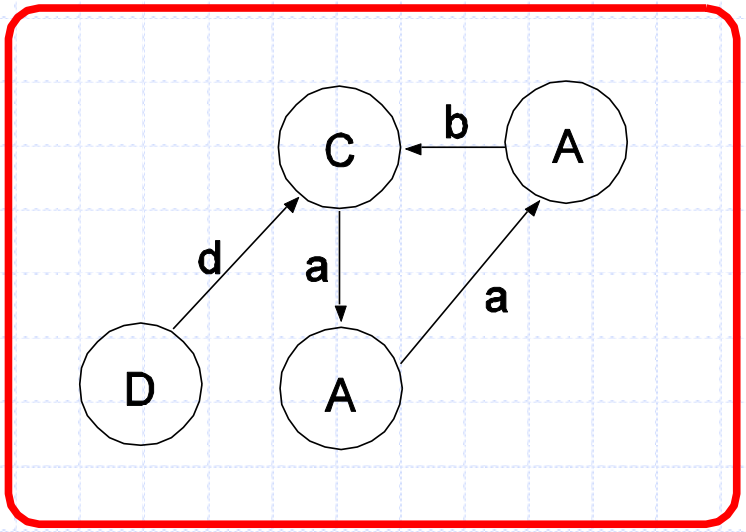
Marginalized Graph Kernels

(Kashima, Tsuda, Inokuchi, ICML 2003)

- Going to define the kernel function $K(G, G')$
- Both vertex and edges are labeled



G



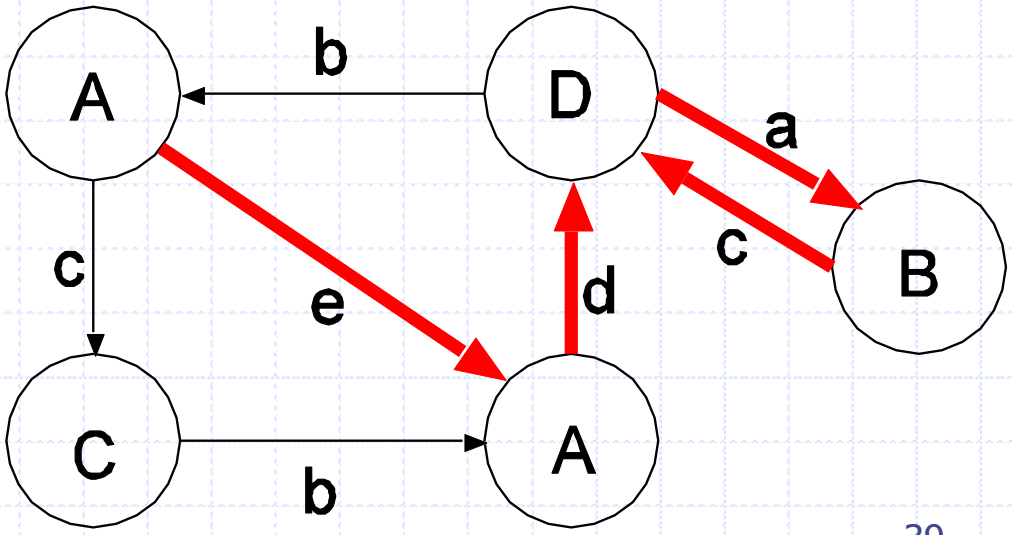
G'

Label path

- Sequence of vertex and edge labels

$$\boldsymbol{h} = (A, e, A, d, D, a, B, c, D)$$

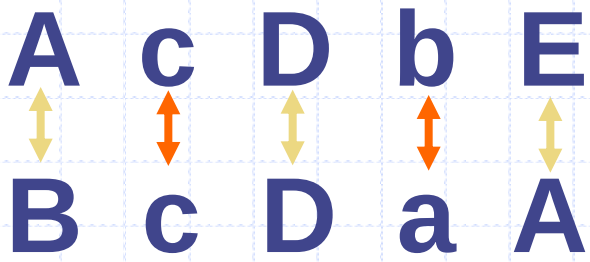
- Generated by random walking
- Uniform initial, transition, terminal probabilities



Path-probability vector

Label path $\boldsymbol{h}$	Probability $p(\boldsymbol{h}\|G)$
AaA	0.001
⋮	⋮
AcDbE	0.000003
⋮	⋮
AeAdDaBcD	0.00000007
⋮	⋮

Kernel definition



- Kernels for paths

$$K(\boldsymbol{h},\boldsymbol{h}') = \begin{cases} 0 & (|\boldsymbol{h}| \neq |\boldsymbol{h}'|) \\ k_v(h_1,h_1')k_e(h_2,h_2')\cdots k_v(h_\ell,h_\ell') & (|\boldsymbol{h}| = |\boldsymbol{h}'|) \end{cases}$$

- Take expectation over all possible paths!
- Marginalized kernels for graphs

$$K(G,G') = \sum_{\boldsymbol{h}}\sum_{\boldsymbol{h}'} p(\boldsymbol{h}|G)p(\boldsymbol{h}'|G')K(\boldsymbol{h},\boldsymbol{h}')$$

Computation

Transition probability : λ

Initial and terminal : omitted

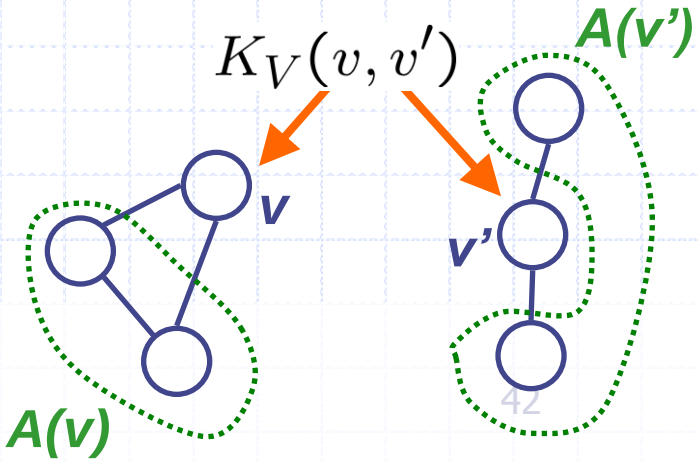
- $S_v(x)$: Set of paths ending at v
- K_V: Kernel computed from the paths ending at (v, v')

$$K_V(v,v') = \sum_{s\in S_v(x)} \sum_{s'\in S_{v'}(x')} \lambda^{|s|}\lambda^{|s'|} K_S(s,s')$$

- K_V is written recursively

$$K_V(v,v') = \lambda^2 I(v=v')\Big(1 + \sum_{\tilde{v}\in A(v)} \sum_{\tilde{v}'\in A(v')} \lambda^2 K_V(\tilde{v},\tilde{v}')\Big)$$

- Kernel computed by solving linear equations
  （polynomial time）



Graph Kernel Applications

- Chemical Compounds (Mahe et al., 2005)
- Protein 3D structures (Borgwardt et al, 2005)
- RNA graphs (Karklin et al., 2005)
- Pedestrian detection
- Signal Processing

Predicting Mutagenicity

- MUTAG benchmark dataset
  - Mutation of Salmonella typhimurium
  - 125 positive data (effective for mutations)
  - 63 negative data (not effective for mutations)

Table 5: Accuracy Results Obtained for the Leave-One-Out Classification of the "Unfriendly Part" of the Mutag Data Set[a]

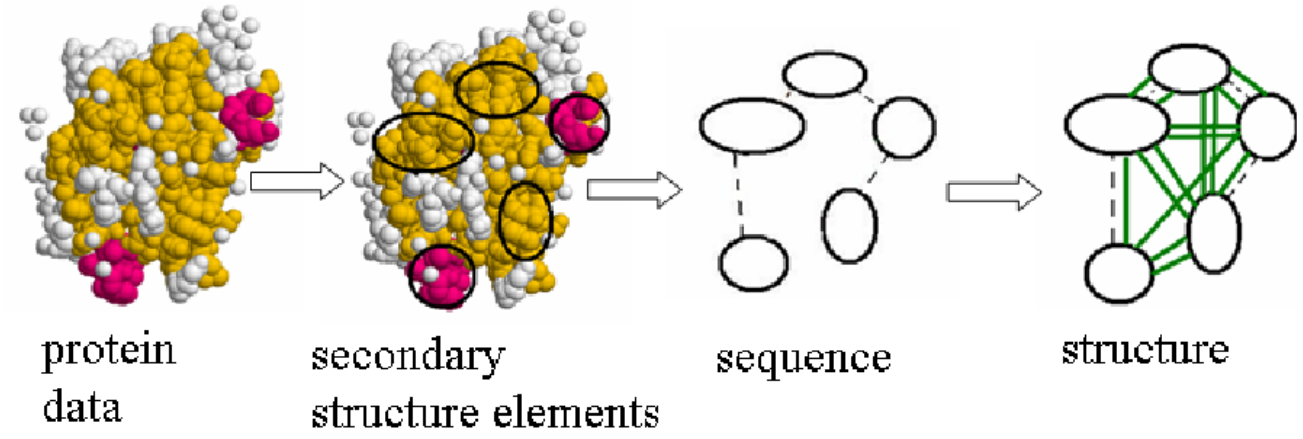
Lin.Reg	*Lin.Reg+*	*DT*	*NN*	*Progol1*	*Progol2*	graph kernels
66.7%	71.8%	83.3%	69.0%	85.7%	83.3%	88.1%

[a] *Lin.Reg* (Linear Regression), *DT* (Decision Tree), *NN* (Neural Network), and *Progol1/2* (Inductive Logic Programming): ref 19.

Mahe et al. J. Chem. Inf. Model., 2005

Classification of Protein 3D structures

- Graphs for protein 3D structures
  - Node: Secondary structure elements
  - Edge: Distance of two elements
- Calculate the similarity by graph kernels



Borgwardt et al. "*Protein function prediction via graph kernels*", ISMB2005

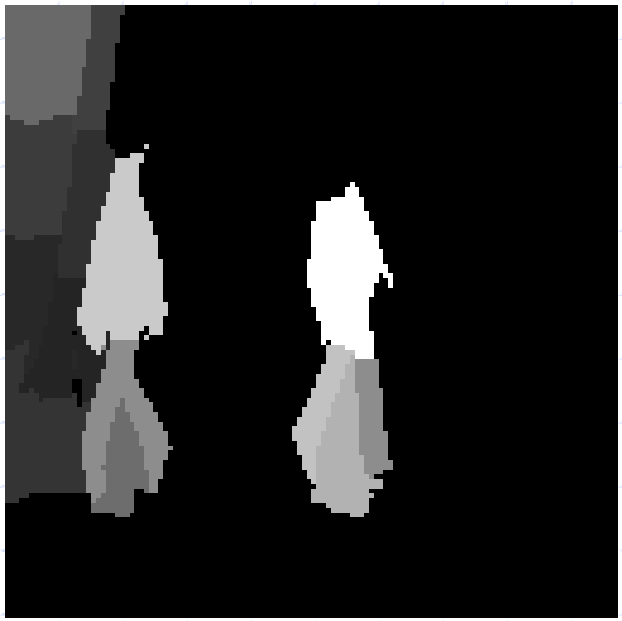
Classification of proteins: Accuracy

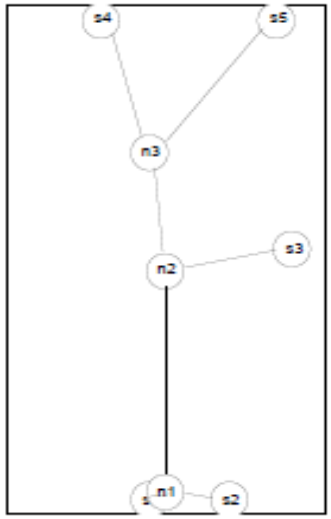
Kernel type	Accuracy	St. dev.
Vector kernel	76.86	1.23
Optimized vector kernel	80.17	1.24
Graph kernel	77.30	1.20
Graph kernel without structure	72.33	5.32
Graph kernel with global info	84.04	3.33
DALI classifier	75.07	4.58

Table 1. Accuracy of prediction of functional class of enzymes and non-enzymes in 10-fold cross-validation with C-SVM. The first two results are the results obtained by Dobson and Doig (2003). "Graph kernel" is our protein kernel defined as in Section 2.3, "Graph kernel without structure" is the same kernel but on protein models without structural edges, "Graph kernel with global info" is our protein graph kernel plus additional global node labels. "DALI classifier" is a Nearest Neighbor Classifier on DALI Z-scores.

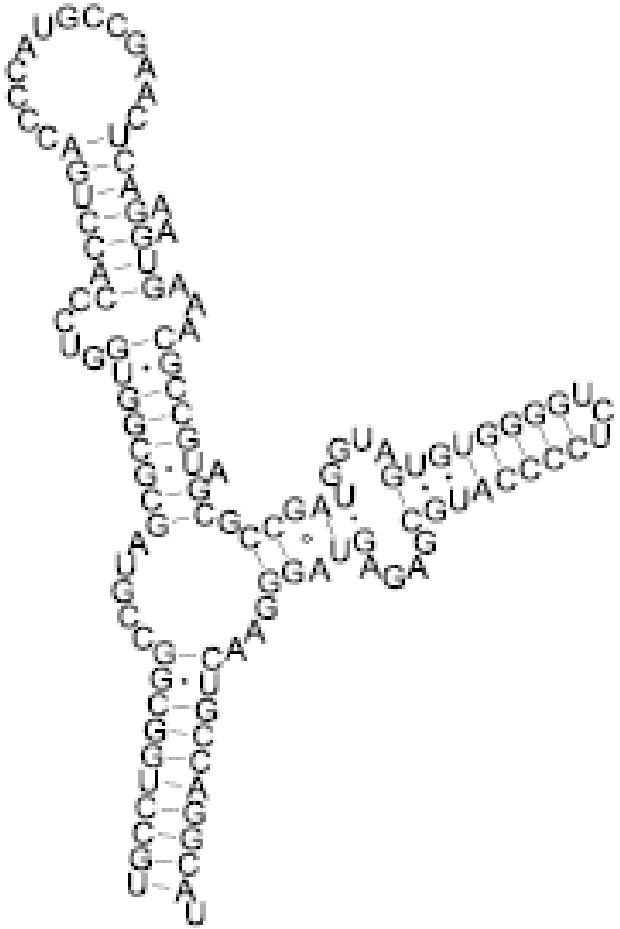
Borgwardt et al. "*Protein function prediction via graph kernels*", ISMB2005

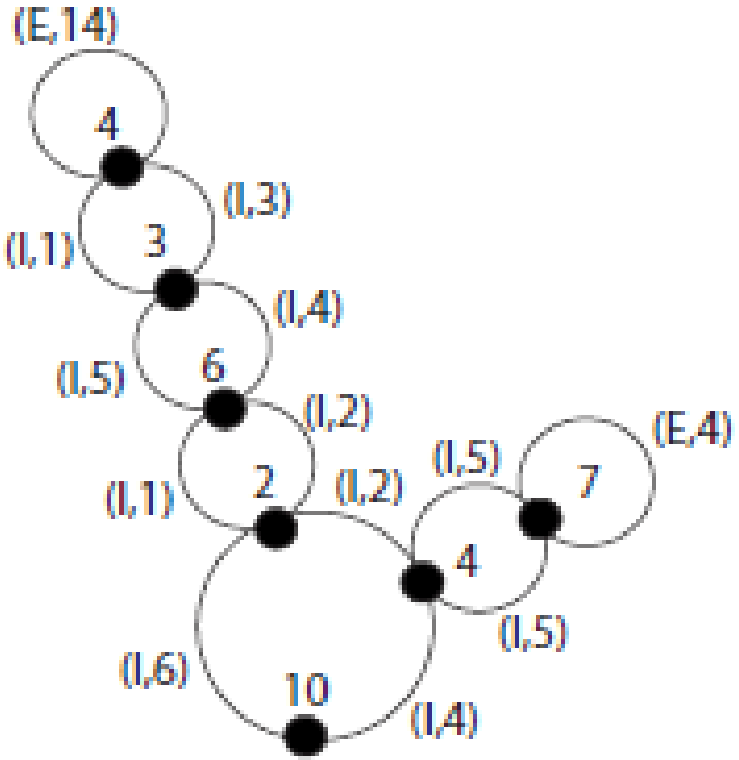
Pedestrian detection in images (F. Suard et al., 2005)





Classifying RNA graphs (Y. Karklin et al.,, 2005)





Strong points of MGK

- Polynomial time computation O(n^3)
- Positive definite kernel
  - Support Vector Machines
  - Kernel PCA
  - Kernel CCA
  - And so on…

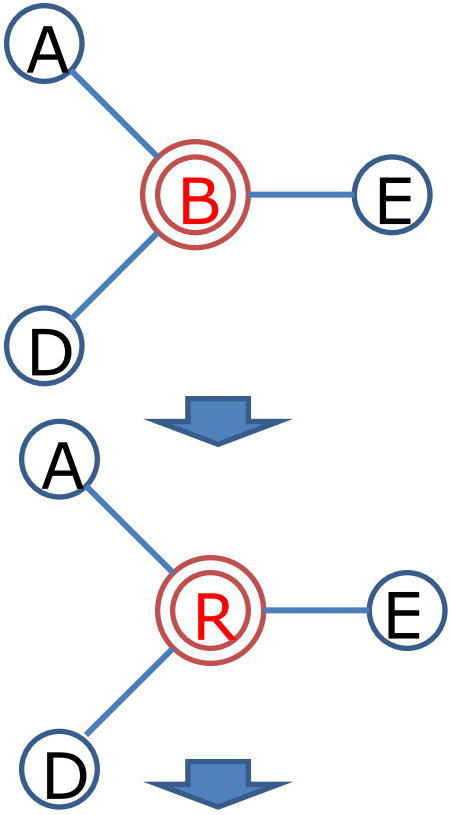
Drawbacks of graph kernels

- Global similarity measure
  - Fails to capture subtle differences
  - Long paths suppressed
- Results not interpretable

- Structural features ignored (e.g. loops)
  - No labels -> kernel always 1

Part 4. Weisfeiler Lehman kernel

Convert a graph into a set of words



Bag-of-words

{A,B,D,E,…,R,…}

i) Make a label set of adjacent vertices ex) {E,A,D}

ii) Sort ex) A,D,E

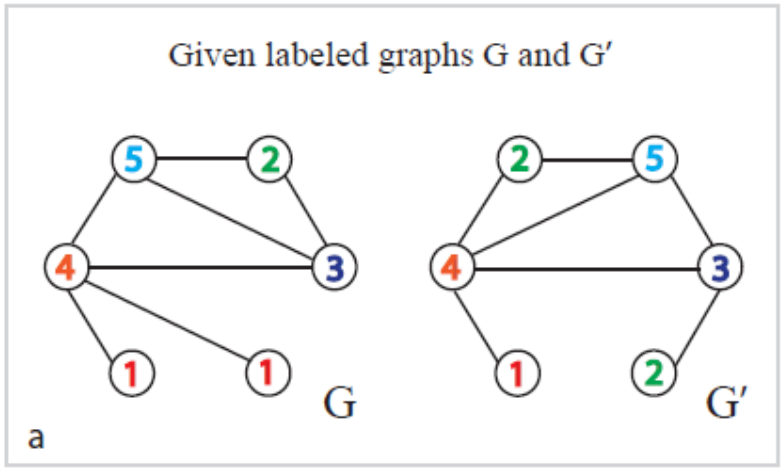
iii) Add the vertex label as a prefix

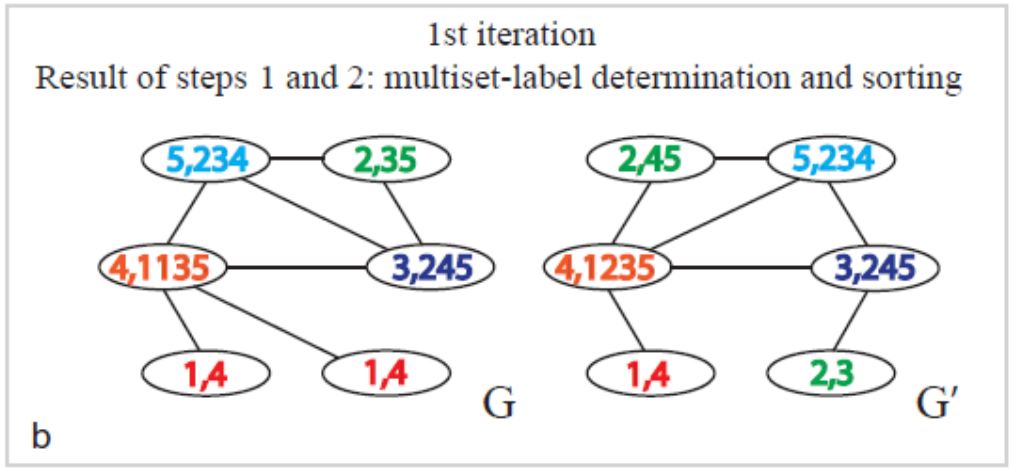
ex) B,A,D,E

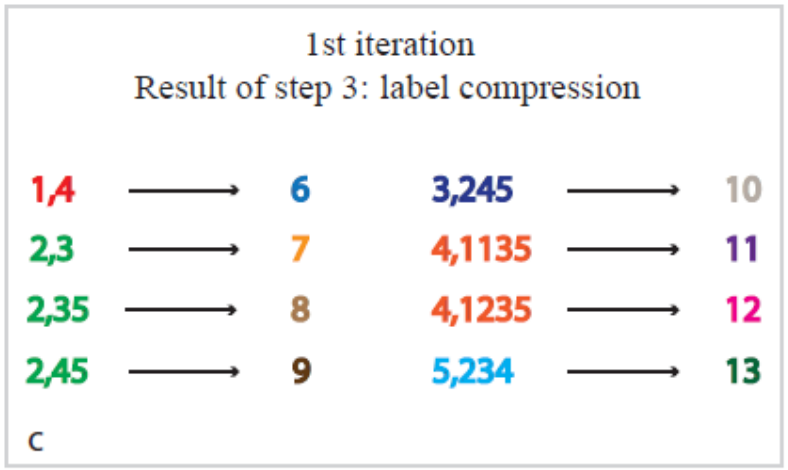
iv) Map the label sequence to a unique value

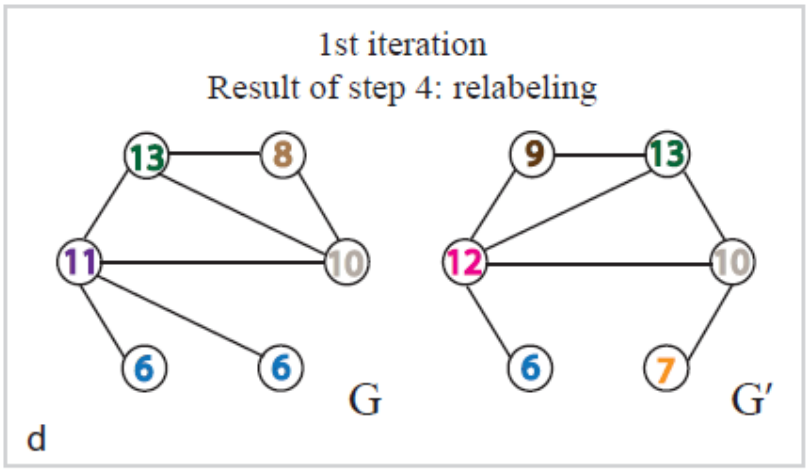
ex) B,A,D,E→R

v) Assign the value as the new vertex label









Courtesy K. Borgwardt

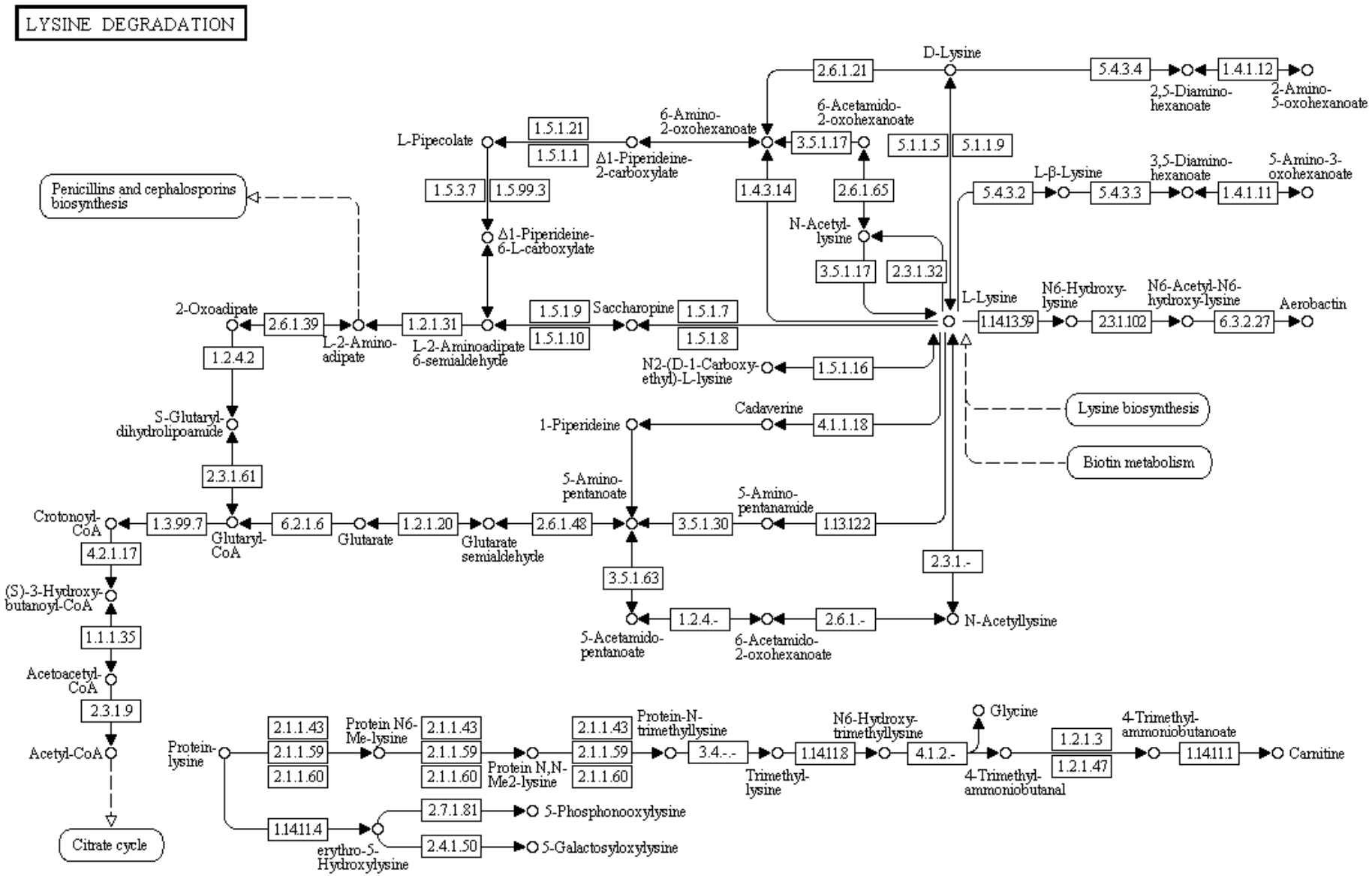
Dataset	MUTAG	NCI1		NCI109		D & D	
Maximum # nodes	28	111		111		5748	
Average # nodes	17.93	29.87		29.68		284.32	
# labels	7	37		54		89	
Number of graphs	188	100	4110	100	4127	100	1178
Weisfeiler-Lehman	6”	5”	7’20”	5”	7’21”	58”	11’
Ramon & Gärtner	40’6”	25’9”	29 days*	26’40”	31 days*	—-	—-
Graphlet count	3”	2”	1’27”	2”	1’27”	2’40”	30’21”
Random walk	12”	58’30”	68 days*	2h 9’41”	153 days*	—-	—-
Shortest path	2”	3”	4’38”	3”	4’39”	58’45”	23h 17’2”

—-: did not finish in 2 days, * = extrapolated.

Table 2: CPU runtime for kernel computation on graph classification benchmark datasets

Part 5. Reaction Graph Kernels

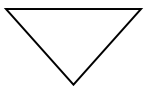
KEGG lysine degradation pathway



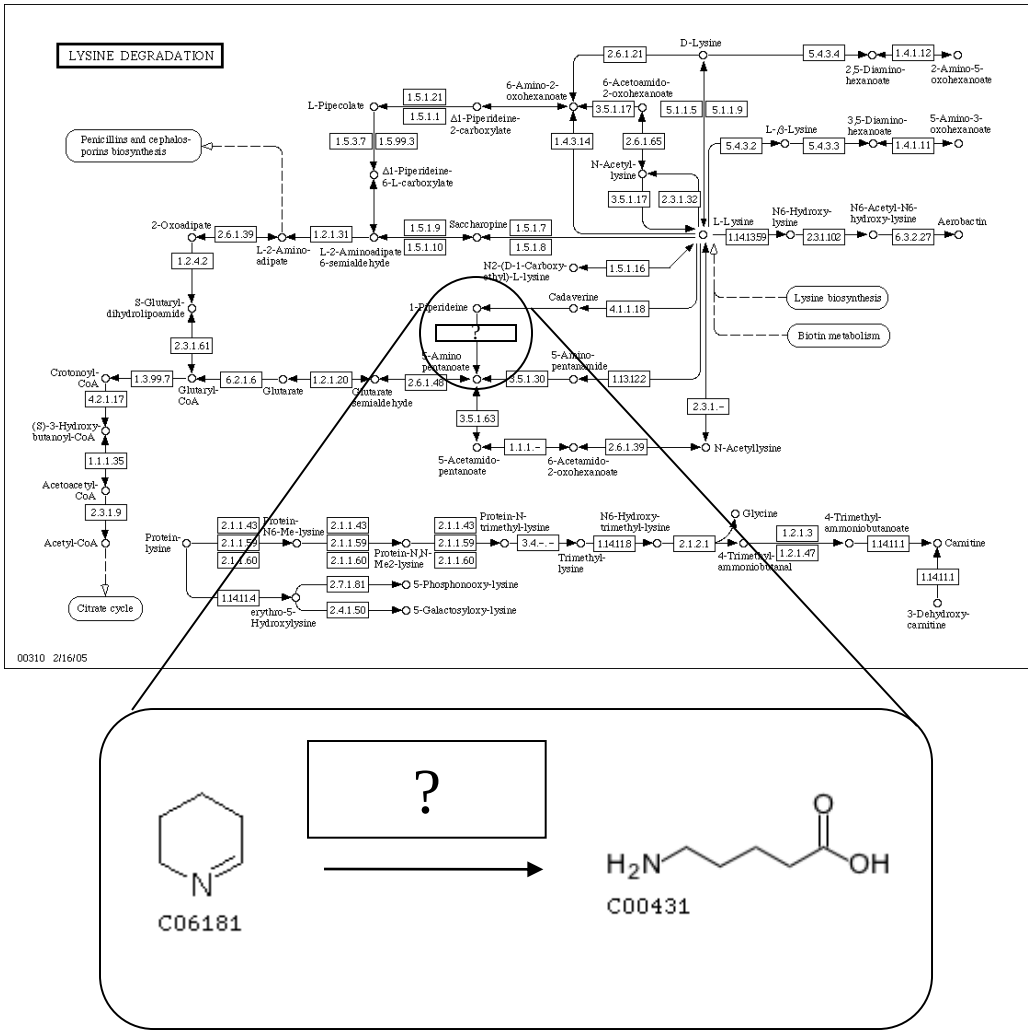
00310 8/28/09
(c) Kanehisa Laboratories

Missing enzymes in metabolic networks

- Many enzymatic reactions whose substrate and product are known, but the enzyme involved is unknown.



- Need to **assign Enzymatic Classification numbers**.



EC number

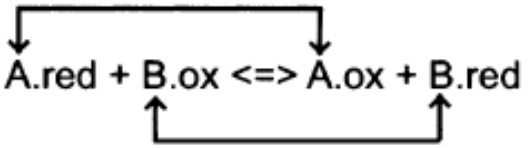
- EC (Enzymatic Classification) number is a **hierarchical categorization of**
  - **Enzymes**
  - **Enzymatic reactions**

class ↓ ↓ subsubclass

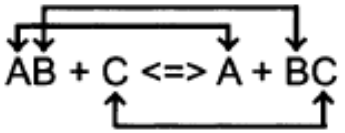
EC 1.3.3.-

↑ subclass

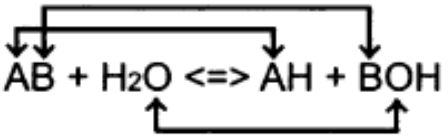
EC1, Oxidoreductases



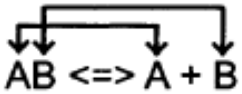
EC2, Transferases



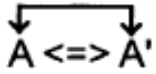
EC3, Hydrolases



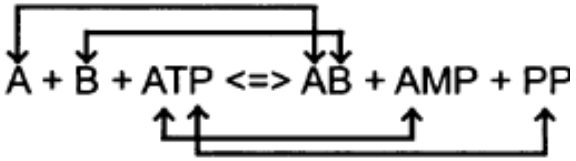
EC4, Lyases



EC5, Isomerases



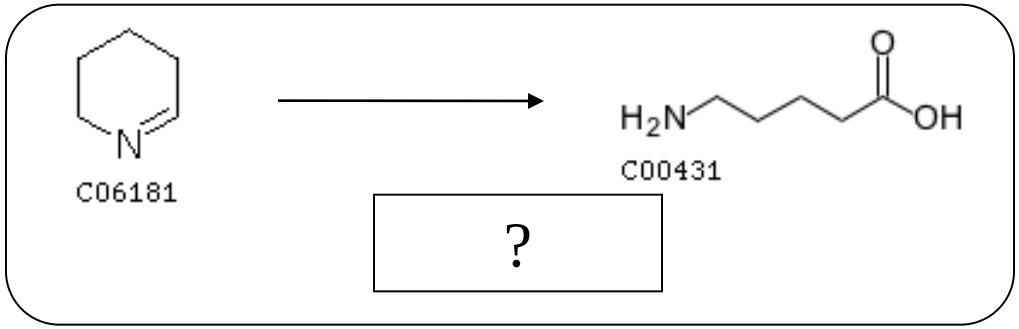
EC6, Ligases



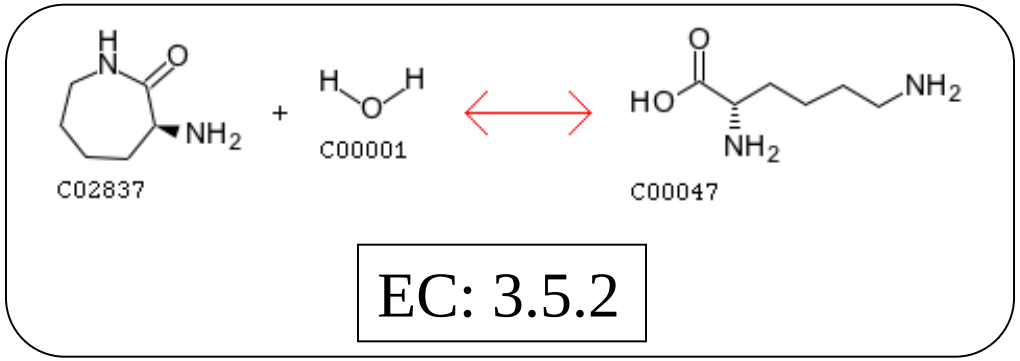
Task

Given a pair of substrate and product as a query, find similar reactions in the database

Query



Result of Retrieval



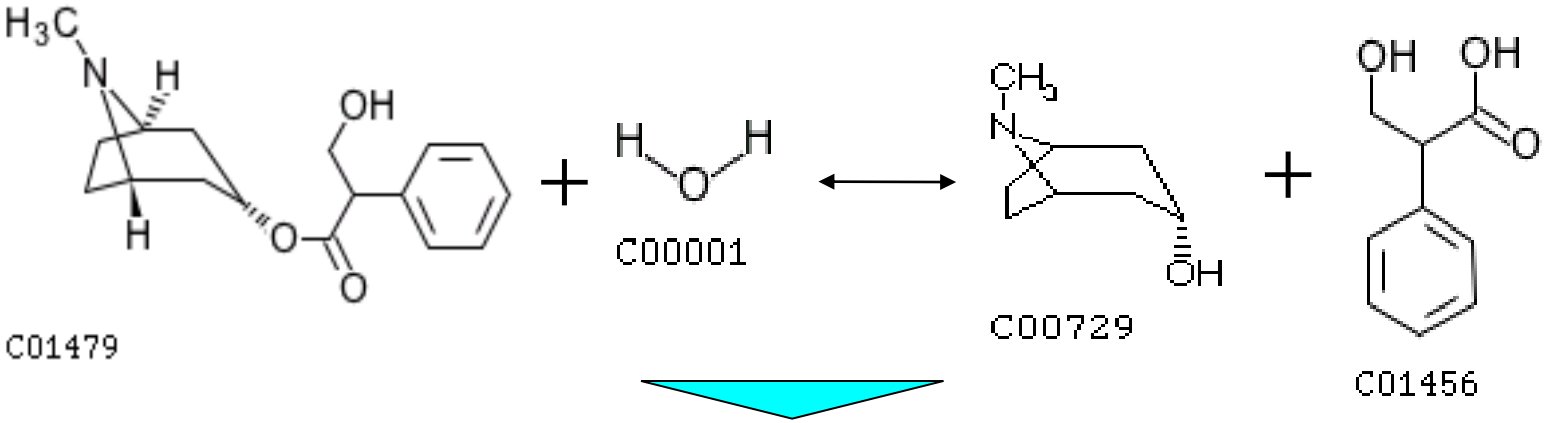
Similarity measure of reactions is necessary

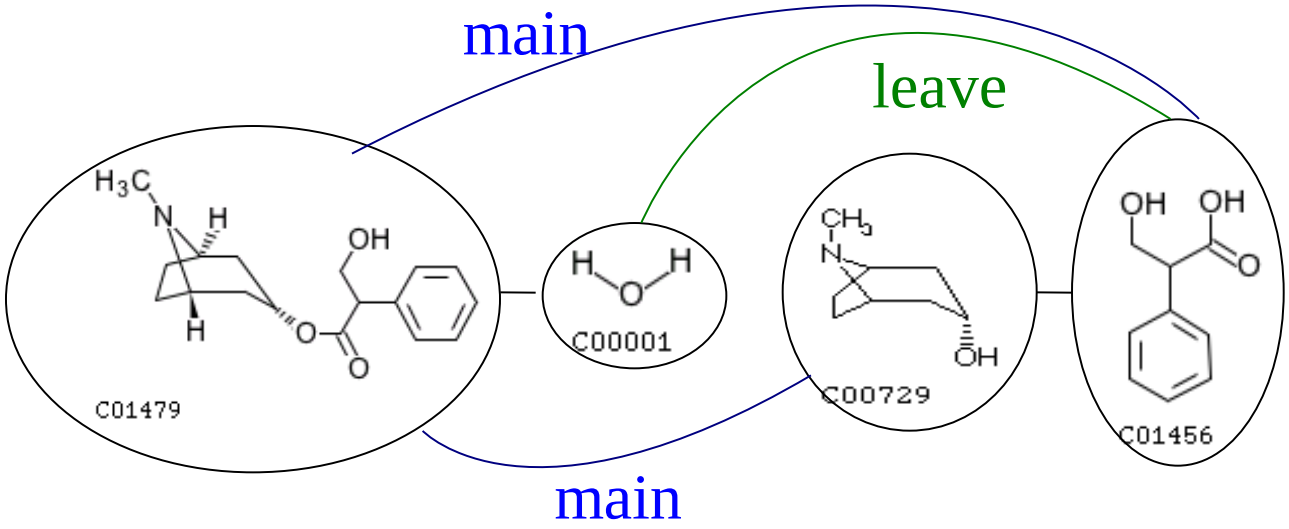
Reaction Graph

- Represent enzymatic reaction as reaction graph
  - Node: Molecules
  - Edge: chemical relation of molecules (main, leave, co-factor, transferase, ligase)
- Reaction graph kernel: Similarity measure of reaction graphs
  - Molecule = Graph of atoms
  - Reaction graph has ‘graph of graphs’ structure
  - Extension of existing walk-based graph kernel (Kashima et al., 2003)

Main Substrate

2 Main Products

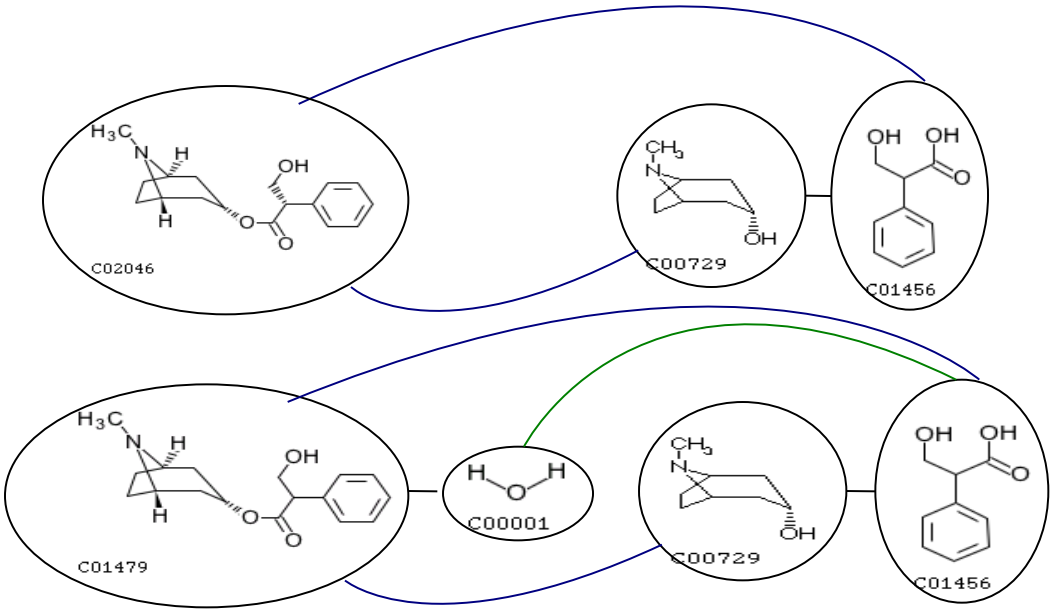




Reaction graph kernels (RGK)

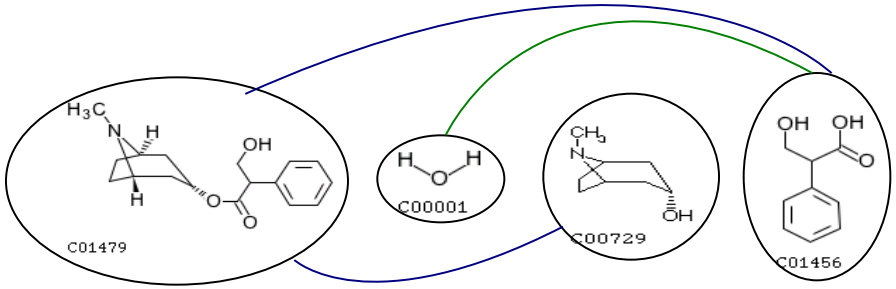
- Two-layered **kernels on graphs of graphs**
  - Node kernel = walk-based graph kernel of molecules
  - Edge kernel = delta kernel of labels
    - "main", "leave", "cofactor", "transferase", "ligase"

$K_r(R, R')$



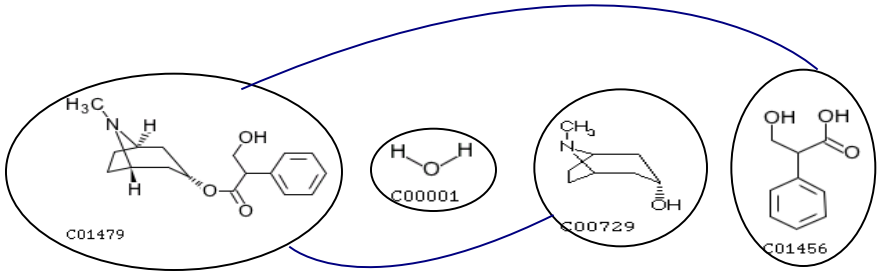
Simplified Settings

- Query might not come in the complete form
- Remove some edges in the database entries



RPAIR

Use only reactant edges (main, leave)



main-only

Use “main” only

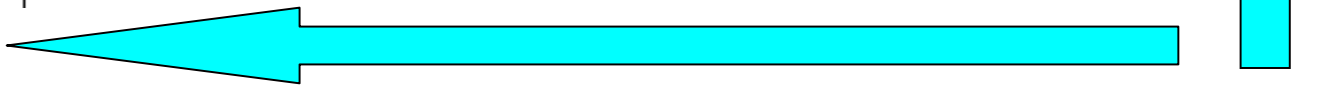
Automatic classification of enzymatic reactions in KEGG

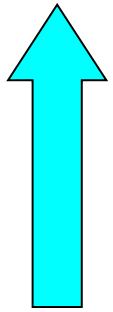
- KEGG/REACTION database
  - 4610 reactions with known EC number
  - 6 classes, 50 subclasses, 124 subsubclasses

- Construct nearest neighbor classifier based on the reaction graph kernel
  - Three different levels: class, subclass,subsubclass
  - Three kernel versions: full, RPAIR, main-only
- Measure leave-one-out classification error

Prediction Accuracy

	EC class	EC subclass	EC subsubclass
full-edge	94.8%	86.0%	82.5%
RPAIR	92.3%	81.4%	78.1%
main-pair	77.8%	69.8%	66.2%





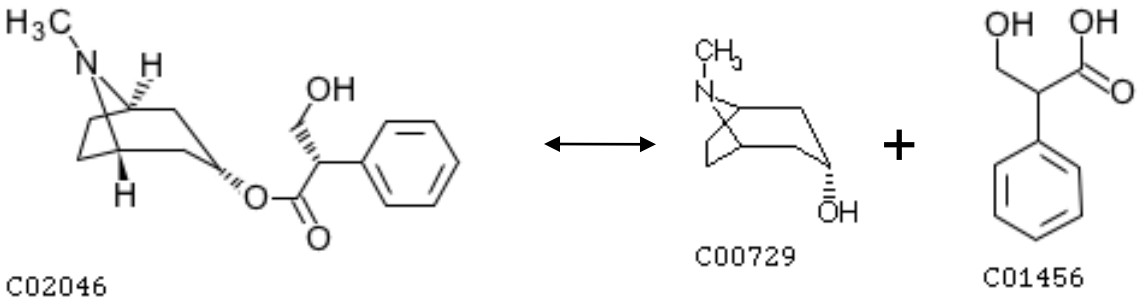
- As expected, classification is easier for upper categories, but difficult for lower categories such as subsubclass.
- The order of accuracy (full-edge > RPAIR > main-pair) suggests that detailed edge information contributes to further accuracy.

Predicting unannotated reactions in plant secondary metabolism

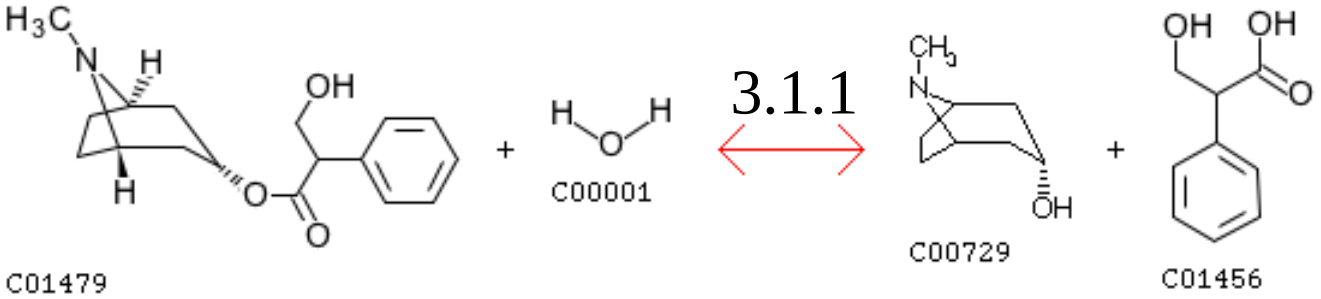
- KEGG pathway "Biosynthesis of Secondary Metabolites"
- Out of unannotated 56 reactions, we have manually assigned ECs of 36 reactions under chemists' guidance
- Comparison with an existing rule-based method: e-zyme (Kotera et al., J. Am. Chem. Soc, 2004).
- RGK's accuracy was better than e-zyme. (50% improvement for the top candidate)

Case 1: EC 3.1.1

query



Manual annotation

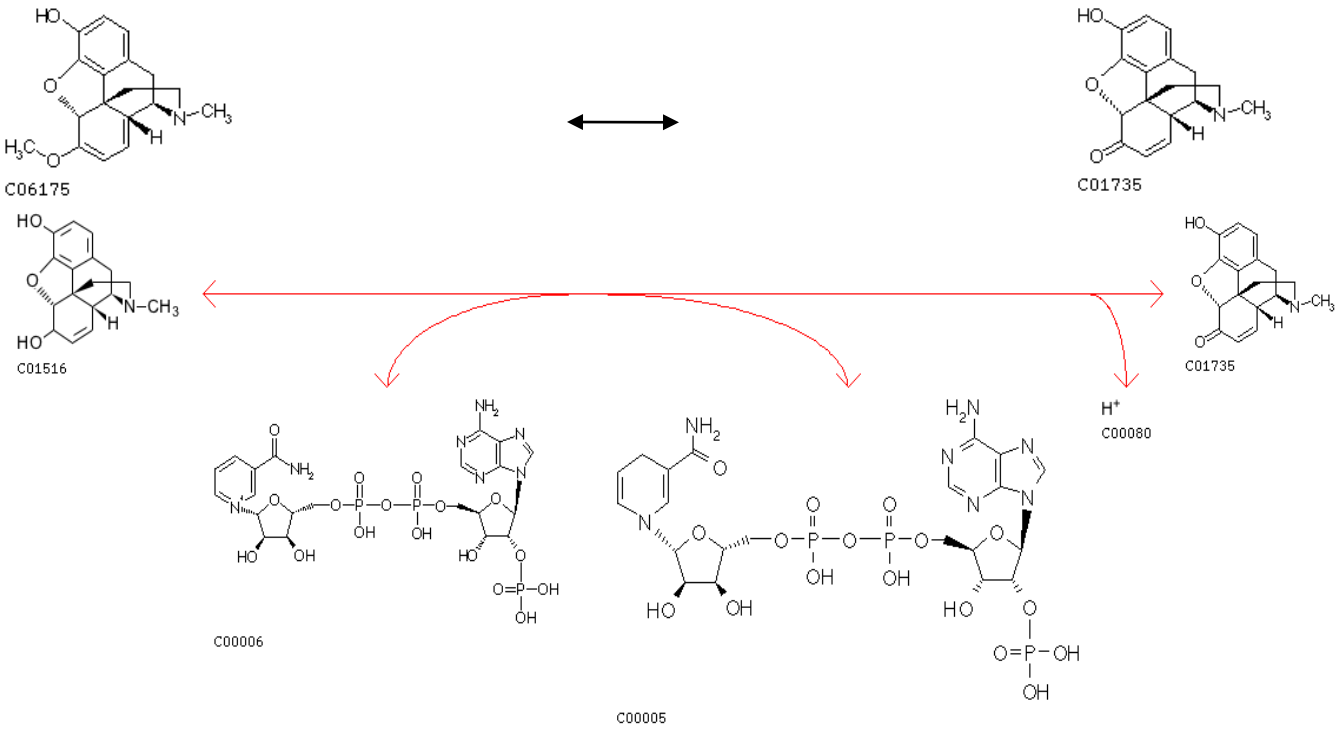


Structures of C02046 and C01479 are almost same except structural isomerism. A hydrolysis occur at a carboxylic-esther bond.

method	rank1	rank2	rank3
RGK	3.1.1	1.14.11	1.14.11
e-zyme	6.1.1	3.1.1	NA

Case 2: EC 2.1.1+1.1.1

query



Manual

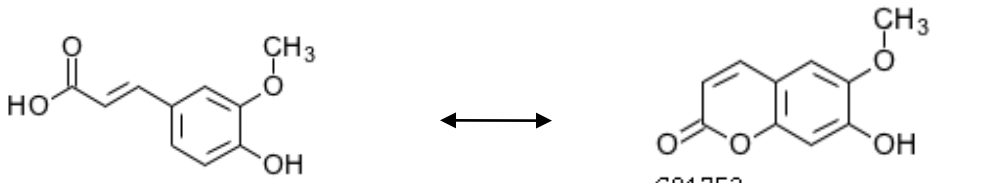
annotation

After removing a methyl group of C06175 (2.1.1), C01735 will be produced by oxidation of CH-OH group (1.1.1). In the second reactions, enzymes usually use NAD+/NADP+ as an acceptor.

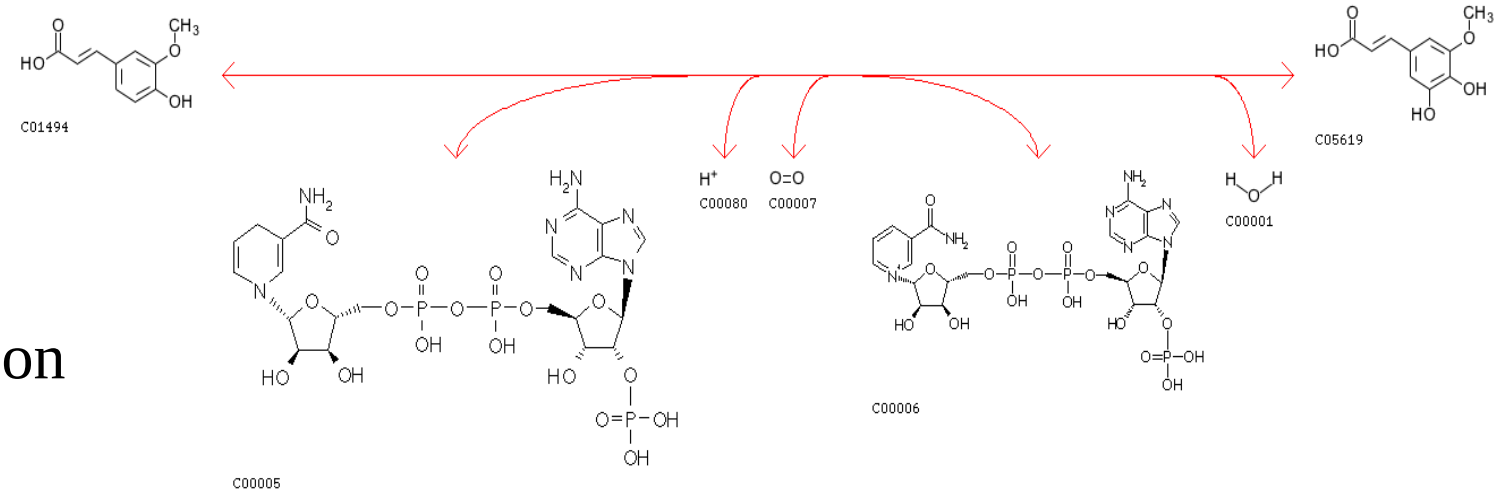
method	rank1	rank2	rank3
RGK	1.1.1	1.1.1	1.1.1
e-zyme	2.1.1	1.13.12	1.14.14

Case 3: EC1.14+2.4.1+5.2.1+3.2.1

Query



Manual Annotation

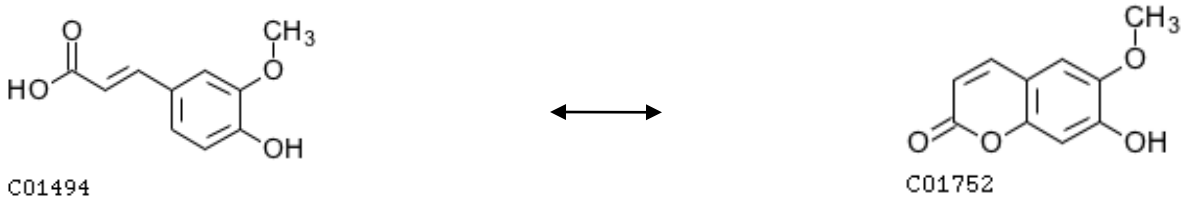


This reaction is thought to be a set of 5 reactions by analogy with the pathway from C00423 => C01772 => C05158 => C05839 => C05838, and to C05851. The last reaction is spontaneous and not enzymatic.

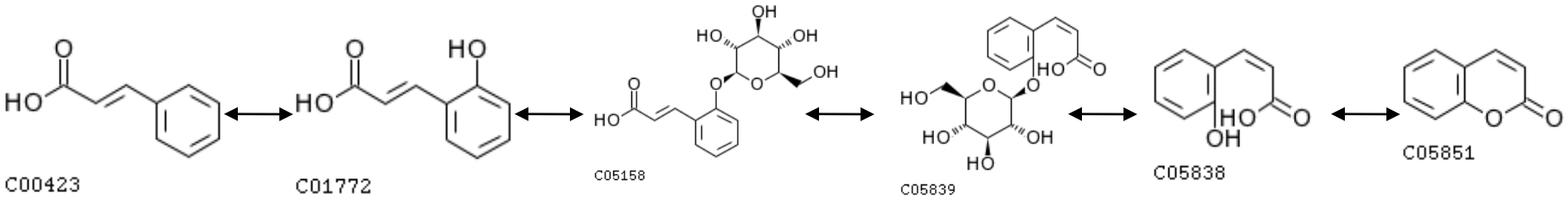
method	rank1	rank2	rank3
RGK	1.14.13	1.2.3	1.2.1
e-zyme	NA	NA-	NA-

A difficult case

No annotation



A very similar reaction (below) is found by manual inspection



Multi-step reaction is difficult to analyze. We don't know how many steps are hidden between given substrate and product.

Concluding Remarks: New Frontier

- Developing Algorithms for Learning from Graphs
- Taming Combinatorial Explosion
  - Recursive Fixed Point Iteration: Graph Kernels
  - Statistical Pruning in Search Tree: Graph Boosting
  - Hashing to a set of words: WL Kernel
- New ideas are necessary to get beyond the current level of speed and prediction accuracy
- Deeper integration of learning and mining

- THANK YOU VERY MUCH!!

Reference

- [1] H. Kashima, K. Tsuda, and A. Inokuchi, "Marginalized kernels between labeled graphs," *ICML*, pp. 321–328, 2003.
- [2] H. Saigo, M. Hattori, H. Kashima, and K. Tsuda, "Reaction graph kernels predict EC numbers of unknown enzymatic reactions in plant secondary metabolism.," *BMC bioinformatics*, vol. 11 Suppl 1, no. Suppl 1, p. S31, Jan. 2010.
- [3] N. Shervashidze, P. Schweitzer, V. Leeuwen, E. Jan, K. Mehlhorn, and K. Borgwardt, "Weisfeiler-Lehman graph kernels," *Journal of Machine Learning Research*, vol. 12, pp. 2539–2561, 2011.
- [4] K. Tsuda, T. Kin, and K. Asai, "Marginalized kernels for biological sequences," *Bioinformatics*, vol. 18, no. Suppl 1, pp. S268–S275, Jul. 2002.