

# Extension and Empirical Comparison of Graph-Kernels for the Analysis of Protein Active Sites

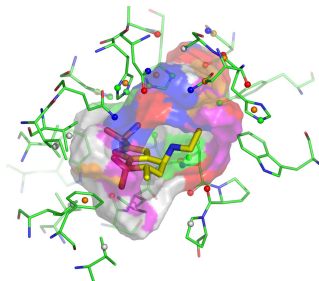
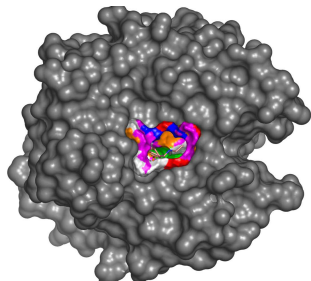
T. Fober, M. Mernberger, V. Melnikov, R. Moritz, E. Hüllermeier

Knowledge Engineering & Bioinformatics Group  
Mathematics and Computer Science Department



KDML, Darmstadt, 2009

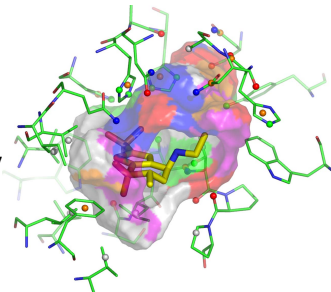
# Protein active sites



- small cavities on the surface of a protein
- protein consists of amino acids; surface of active site, too
- amino acids define physico-chemical properties
- abstraction: summarize patch into a spatial point

# Why are we interested in active sites?

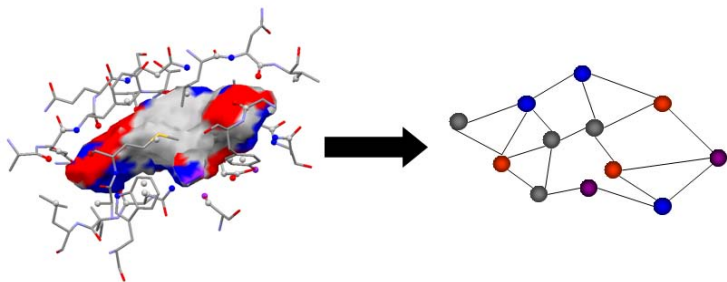
- small molecules bind to these sites and cause a reaction of the protein
- pharmaceutical chemistry is interested in active sites and similarity measures between them
  - ▶ inhibit active site to cause or suppress a reaction of the protein



# Applications

- 1 cross reactivities
  - ▶ active site of target is known
  - ▶ search for proteins with similar active site
  - ▶ these proteins may also be influenced by ligand
- 2 prediction of the function
  - ▶ protein with unknown function but known structure
  - ▶ search for proteins with known function and similar active site
- 3 ...

# Graph-representation of protein active sites



## Definition (graph)

$G = (V, E, l_V, l_E)$  is a node-labeled and edge weighted graph, where  $V$  is a finite set of nodes and  $E \subseteq V \times V$  a set of edges, and where  $l_V : V \rightarrow \mathcal{L}_V$  and  $l_E : E \rightarrow \mathbb{R}$  are functions that assign labels

- create discrete edge labels through binning

# Objective

- find appropriate similarity measures on graphs  $\text{sim} : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$
- in the following: compare graphs  $G = (V, E, I_V, I_E)$  and  $G' = (V', E', I'_V, I'_E)$
- existing approaches
  - ▶ graph isomorphism
  - ▶ subgraph isomorphism
    - ★ maximum common subgraph
    - ★ minimum common supergraph
  - ▶ graph edit distance
  - ▶ graph kernels

# Why kernels?

- enables the application of kernel-based methods for data analysis
- kernel trick:  $k(G, G') = \langle \phi(G), \phi(G') \rangle = \phi(G)^T \phi(G')$
- a function  $k$  is a kernel, iff
  - ▶  $k : \mathcal{G} \times \mathcal{G} \rightarrow \mathbb{R}$
  - ▶  $k$  is symmetric
  - ▶  $k$  is positive definite
- R-convolution framework

$$k(G, G') = \sum_{\substack{g \in R^{-1}(G) \\ g' \in R^{-1}(G')}} \kappa(g, g')$$

# Random Walk Kernel

- decompose a graph into a set of random walks
- compare these walks with a kernel on sequences
- do this implicitly by using the product graph  $G_{\times} = G \times G'$

$$V_{\times} = \left\{ (v_i, v'_j) \mid v_i \in V, v'_j \in V', l_V(v_i) = l'_V(v'_j) \right\}$$

$$E_{\times} = \left\{ \left( (v_i, v'_j), (v_k, v'_l) \right) \in V_{\times} \times V_{\times} \mid l_E(v_i, v_k) = l'_E(v'_j, v'_l) \right\}$$

- $[A^n]_{i,j}$  gives the number of walks from  $i$  to  $j$  of length  $n$
- product graph:  $[A_{\times}^n]_{i,j}$  gives the number of walks from  $i$  to  $j$  of length  $n$  appearing in  $G$  and  $G'$

- consider all pairs of nodes  $(i, j)$  and sum-up  $[A_{\times}]_{i,j}$ 
  - ▶ count walks that appear in  $G$  and  $G'$
  - ▶ only walks of length 1
- consider  $[A_{\times}^k]$  for  $k = 0, \dots, \infty$ 
  - ▶ now walks of all length
  - ▶ limit is now becoming hard to determine
- use shrink factor  $\lambda_k$ 
  - ▶ extremely long path have lower weight
  - ▶ series can be calculated in a more easy way

$$k_{RW}(G, G') = \sum_{i,j=1}^{|V_{\times}|} \left[ \sum_{k=0}^{\infty} \lambda_k \cdot A_{\times}^k \right]_{i,j}$$

# Geometric Kernel

- use  $\lambda_k = \lambda^k = \left(\frac{1}{a}\right)^k$  with  $a \geq \max_{v \in V_X} \{\deg(v)\}$
- $k_{RW}$  is becoming geometrical series

$$k_{RW_{geo}}(G, G') = \sum_{i,j=1}^{|V_X|} \left[ (I - \lambda \cdot A_X)^{-1} \right]_{i,j}$$

- matrix inversion:  $\mathcal{O}(M^6)$  where  $M = \max\{|V|, |V'|\}$

# Exponential Kernel

- use  $\lambda_k = \lambda^k = \frac{\beta^k}{k!}$
- $k_{RW}$  is becoming exponential series

$$k_{RW_{exp}}(G, G') = \sum_{i,j=1}^{|V_x|} \left[ e^{\beta \cdot A_x} \right]_{i,j}$$

- matrix diagonalization:  $\mathcal{O}(M^6)$

# Drawbacks

- tottering: same node and edge can be counted repeatedly
- halting:  $\lambda$  down-weights longer walks; so short walks dominate the score, however they only represent very small parts of the graph
- runtime: graphs modeling proteins, or protein networks are large and contain up to  $\mathcal{O}(10^5)$  nodes

# Shortest Path Kernel

- instead of walks consider shortest paths (SP)
- number of SP is bounded in a graph  $G = (V, E)$  by  $\mathcal{O}(|V|^2)$
- calculation of all SP in graph  $G$  takes  $\mathcal{O}(|V|^3)$
- store all SP in a SP-graph  $G_{SP} = (V_{SP}, E_{SP})$ 
  - ▶  $V_{SP} = V$  (node labels remain)
  - ▶  $E_{SP} = V \times V$  (edge weight is giving length of shortest path)
- consider  $SP(v_i, v_j) = (\{l_V(v_i), l_V(v_j)\}, sp(v_i, v_j))$
- kernel on SP

$$\kappa(SP(v_i, v_j), SP(v'_k, v'_l)) = \begin{cases} 1 & \text{if } SP(v_i, v_j) = SP(v'_k, v'_l) \\ 0 & \text{else} \end{cases}$$

- shortest path kernel by summing up all kernels on paths

$$k_{SP}(G, G') = \sum_{v_i, v_j \in V} \sum_{v'_k, v'_l \in V'} \kappa_{path}(SP(v_i, v_j), SP(v'_k, v'_l))$$

- tottering is strongly reduced
- reduced runtime  $\mathcal{O}(M^4)$

# Fingerprint kernels

- explicit construction of a feature vector of fixed length
- why graph kernels?
  - ▶ feature vectors lead to a loss of information
  - ▶ vectors must have equal length but graphs can differ strongly in size
- define vector representation of *certain length* for graphs
- capture as much structure as possible

# Triangles

- consider all non-isomorphic graphs of size 3
- assume  $n$  distinct node- and  $k$  distinct edge-labels ( $n = 7, k = 11$ )

$$N(n, k) = \binom{n}{3} \cdot k^3 + n(n-1) \cdot k \cdot \binom{k+1}{2} + n \cdot \binom{k+2}{3}$$

- fingerprint vector

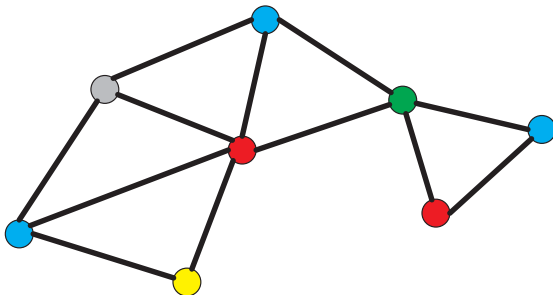
$$f_G = (G \supseteq t_1, G \supseteq t_2, \dots, G \supseteq t_{N(n,k)}) \in \{0, 1\}^{N(n,k)}$$

- kernel on fingerprints

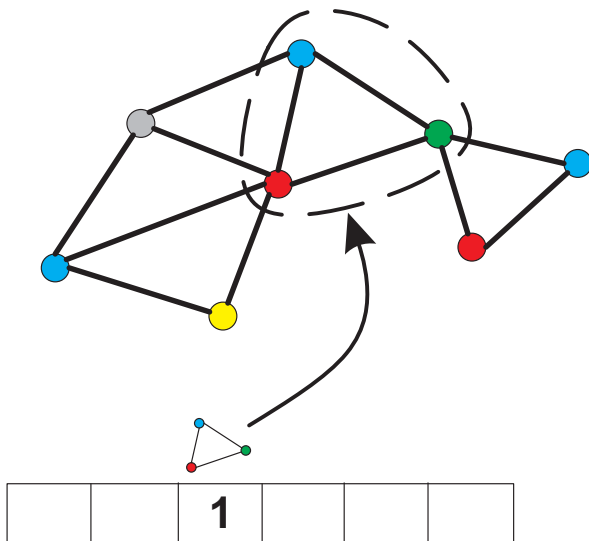
$$k_{FPH}(G, G') = \frac{1}{N(n, k)} \sum_{i=1}^{N(n, k)} \kappa_{\delta}([f_G]_i, [f_{G'}]_i)$$

$$k_{FPJ}(G, G') = \frac{\sum_{i=1}^{N(n, k)} \min([f_G]_i, [f_{G'}]_i)}{\sum_{i=1}^{N(n, k)} \max([f_G]_i, [f_{G'}]_i)}$$

# Example

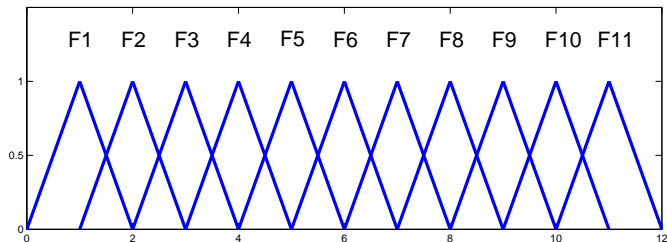


# Example



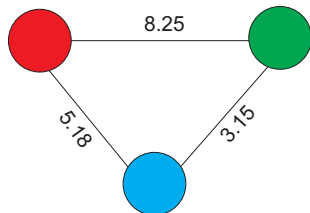
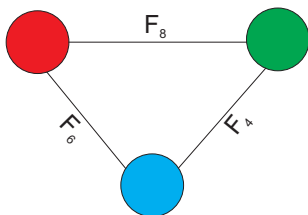
# Fuzzy Fingerprints

- drawback of crisp fingerprints is their discontinuity on boundaries
- use fuzzy discretization:
  - family of fuzzy subsets  $F_1, F_2, \dots, F_k$  of  $\mathbb{R}_+$  so that  $\sum_{i=1}^k F_i(x) > 0$
  - $F_i(x) = \max\{0, 1 - |x - i|\}$  ("approx. equal i")



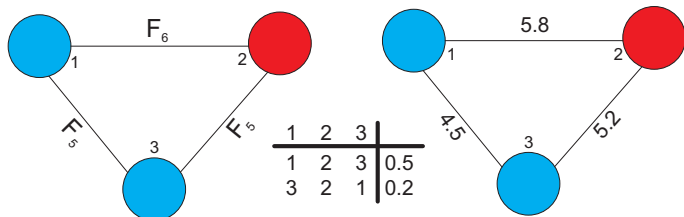
- for patterns  $t$  edges are labeled with fuzzy-numbers  $F_i$

- $G$  has real-valued edge lengths
- consider all subgraphs  $s$  of  $G$
- degree of isomorphism  $[t \sim s]$ :
  - ▶ find all permutations so that node labels match
  - ▶ take that permutation that maximizes the minimum of the  $\{F_i\}$

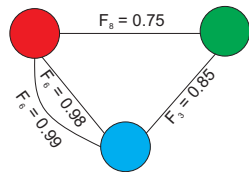
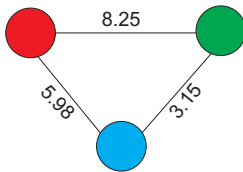
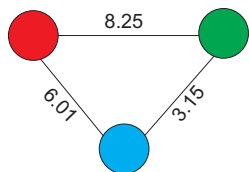


- degree of inclusion  $t$  in  $G$  is given by  $[G \supseteq t] = \max_s [t \sim s]$

- $G$  has real-valued edge lengths
- consider all subgraphs  $s$  of  $G$
- degree of isomorphism  $[t \sim s]$ :
  - ▶ find all permutations so that node labels match
  - ▶ take that permutation that maximizes the minimum of the  $\{F_i\}$

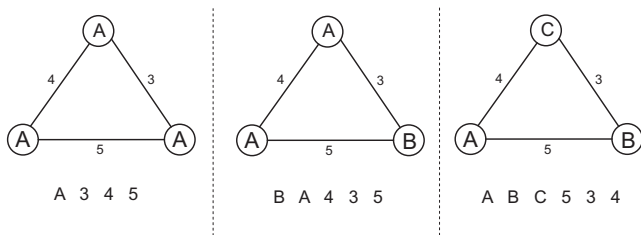


- degree of inclusion  $t$  in  $G$  is given by  $[G \supseteq t] = \max_s [t \sim s]$



# Efficient Implementation

- non-fuzzy: stop search as soon as pattern has been found
- not possible for fuzzy approach, instead
  - ▶ canonical form  $\rightarrow$  3 cases



- $i : \Sigma \rightarrow \{1, \dots, N(n, k)\} \subset \mathbb{N}$  assigns unique number to each form

- take subgraph of size 3
- there exist  $\binom{|V|}{3} = \mathcal{O}(|V|^3)$  such subgraphs
- transform each subgraph of size 3 into canonical form in time  $\mathcal{O}(1)$
- determine its position using  $i$  in time  $\mathcal{O}(1)$  and update vector
- runtime is  $\mathcal{O}(|V|^3)$ , naive approach has  $\mathcal{O}(|V|^3) \cdot \underbrace{N(n, k)}_{36729}$

# Classification

Dataset: NADH/ATP

- data set consists of 355 protein active sites
- 241 pockets bind to ATP ligands
- 141 pockets bind to NADH ligands
- binary classification problem

## How to measure quality of a similarity measure?

- higher classification rates  $\Leftrightarrow$  better similarity
- two-class ATP/NADH data set
- predict to which ligand a pocket binds
- SVM and  $k$ -NN classifier
- leave-one-out cross validation

method	RW	SP	FPH	FPJ	FFP
SVM	0.606	0.625	0.916	0.907	<b>0.916</b>
$k = 1$	0.597	0.606	0.828	0.842	0.879
$k = 3$	0.597	0.628	0.839	0.882	0.887
$k = 5$	0.597	0.634	0.839	0.873	0.887
$k = 7$	0.608	0.625	0.819	0.859	0.854
$k = 9$	0.608	0.634	0.814	0.839	0.839
runtime	$65 \pm 121$	$9.8 \pm 98$	$2.05 \pm 3.6$	$2.05 \pm 3.6$	$18 \pm 66$

# Conclusions

- fingerprint kernels are very efficient and effective
  - ▶ they are of high interest in protein structure comparison
- R-convolution framework is problematic
  - ▶ all-against-all comparison
  - ▶ large objects consisting of many heterogeneous substructures
  - ▶ averaging effect
- extension by considering subgraphs of size larger 3, or/and counting occurrences