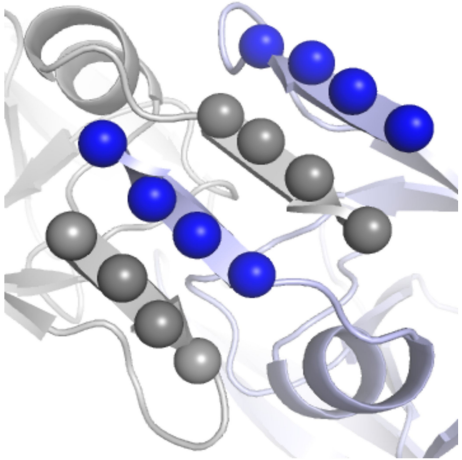
Discovery of Functional Motifs from the Interface Region of Oligomeric Proteins using Frequent Subgraph Mining Method

Tanay Kumar Saha, Ataur Katebi and Mohammad Al Hasan

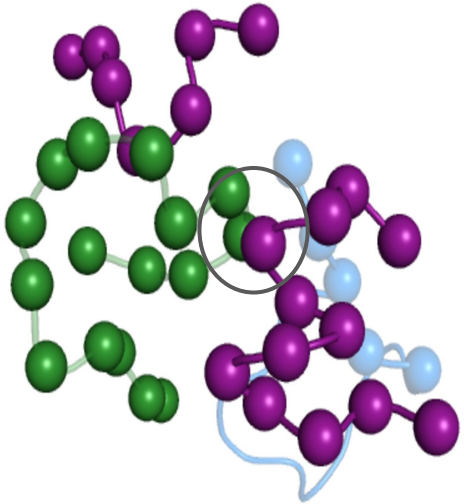
Presented by: Mohammad Hasan

Our Contribution

- Model the protein interface region as a network (Interfacial network)
  - Each conformation of a particular protein represented as a single network
- Use a scalable graph Mining approach to mine frequent subgraphs among conformations
  - Multiple conformations of a particular protein form a graph database
- Show that Interfacial network can be used to find
  - Lock structure in HIV
  - Hugging point in TIM structure



Lock Structure (HIV)



Hugging point (TIM)

Method

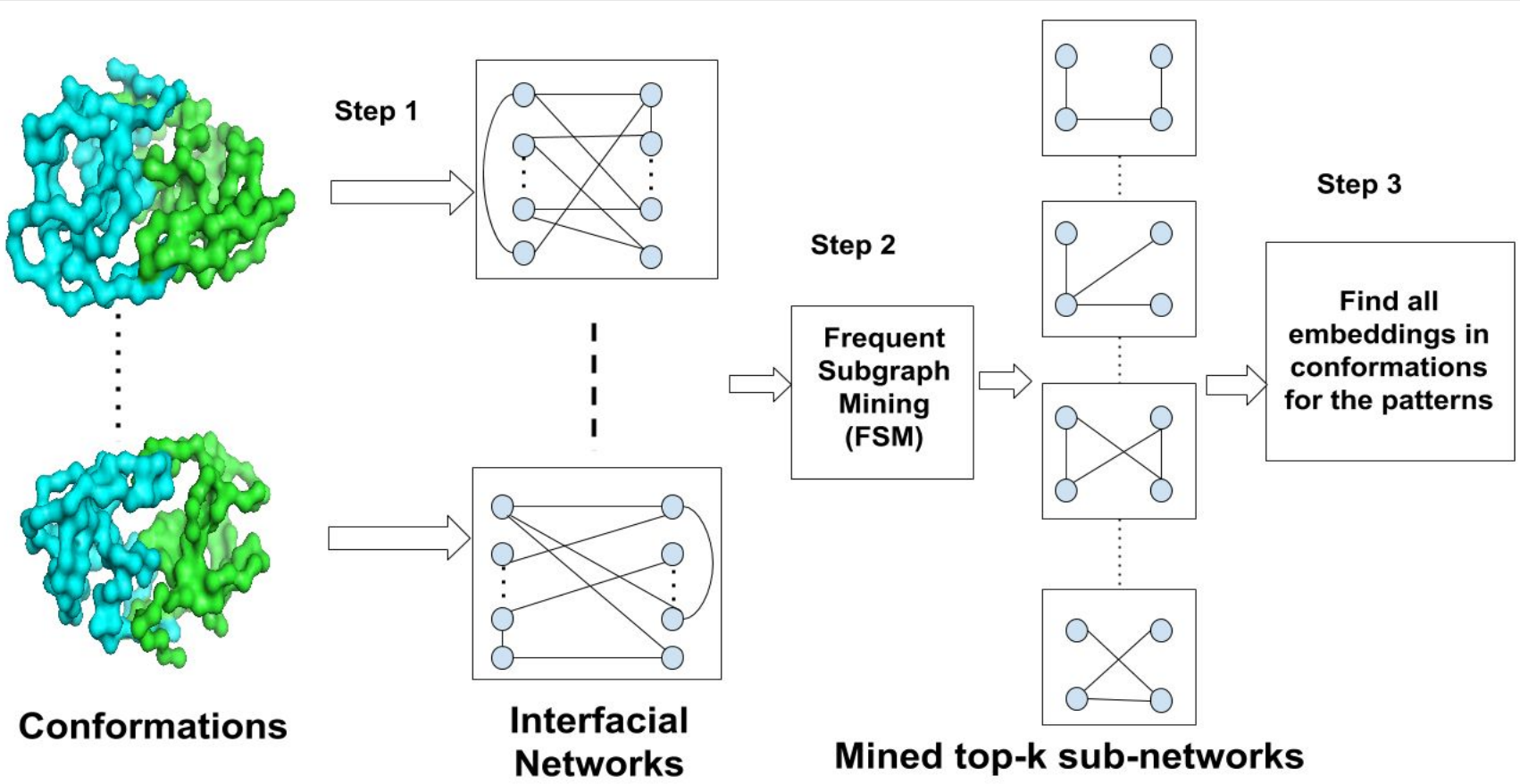
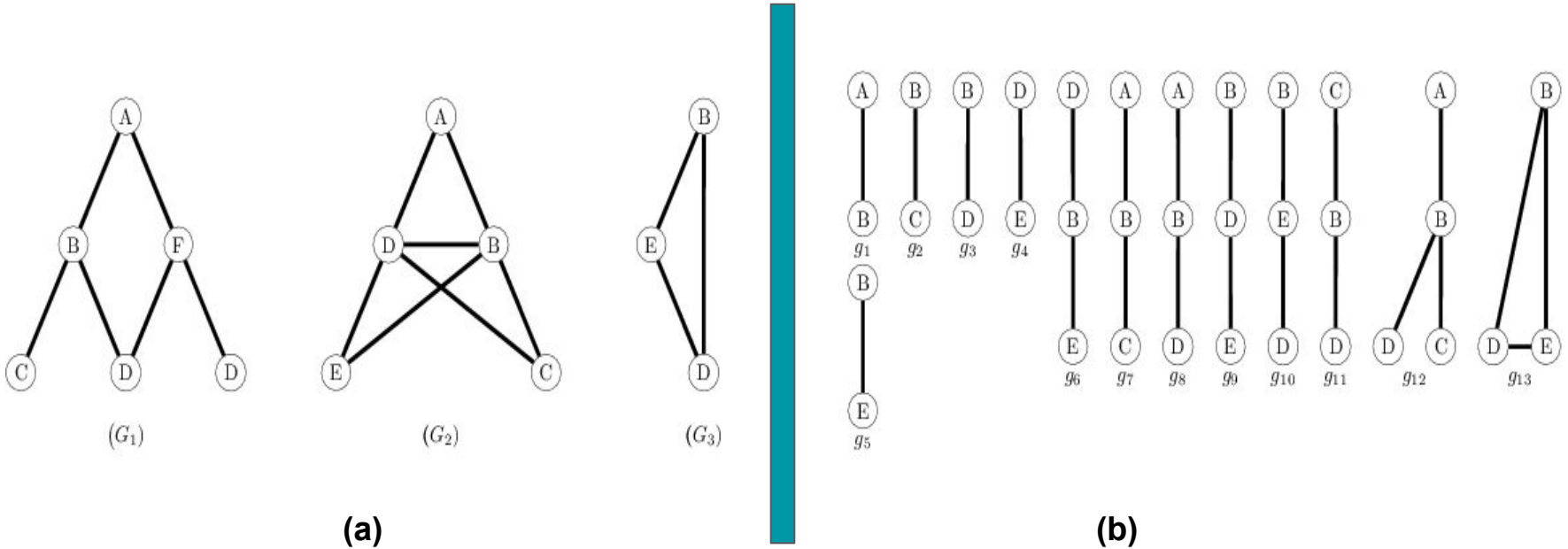


Fig: Method

Modeling interface region as a Network

- Retrieve the backbone carbons along with their 3D coordinates
- Select the subset of C_alpha residues that are in the interface region of either of the chains
- Consider a residue in a chain to be at the interface region if it is within 8A distance of any C_alpha residue in the other chain
- Represent those residues as nodes
- Connect two nodes in the inter-chain if they are within 8A distance from each other
- For intra-chain, that distance is 4A
- Obtain a set of undirected, vertex-labeled graphs---each corresponding to one of the conformations

Mining Frequent Subgraphs (traditional definition)



(a) A graph database with 3 graphs (b) All frequent subgraph of the graph database in (a) using minimum support value of 2

Issues with frequent subgraph mining techniques

- Not Scalable for dense and large graphs

Dataset Statistics: # graphs: 90, avg. # vertices: 67, avg. # edges: 268
 # node labels: 20, # edge labels: 3

Time vs Max. subgraph size (min-sup is fixed at 40%)		Time vs different minsup (Max-size is fixed at 8)		Search Space vs subgraph size	
Max-size	Time	Support (%)	Time	Size	Induced Subgraph Count
8	6 minutes	28	1.1 hours	6	26 millions
9	2.8 hours	22	3.5 hours	7	157 millions
10	> 1.5 days	17	9 hours	8	947 millions
		11	>16 hours	9	5 billions

- Distributed methods solve the scalability issue for the case of many graphs in the database, but not for the case of large and dense graphs

Other Issues

- User needs to choose minimum support threshold
  - When mining for patterns in graph database, due to protein dynamics identical patterns does not appear in many conformations

- Frequent subgraphs have substantial overlap
  - Functional motifs can be fragmented in many overlapping frequent subgraphs

FS^3 Algorithm: An alternative approach of frequent subgraph Mining

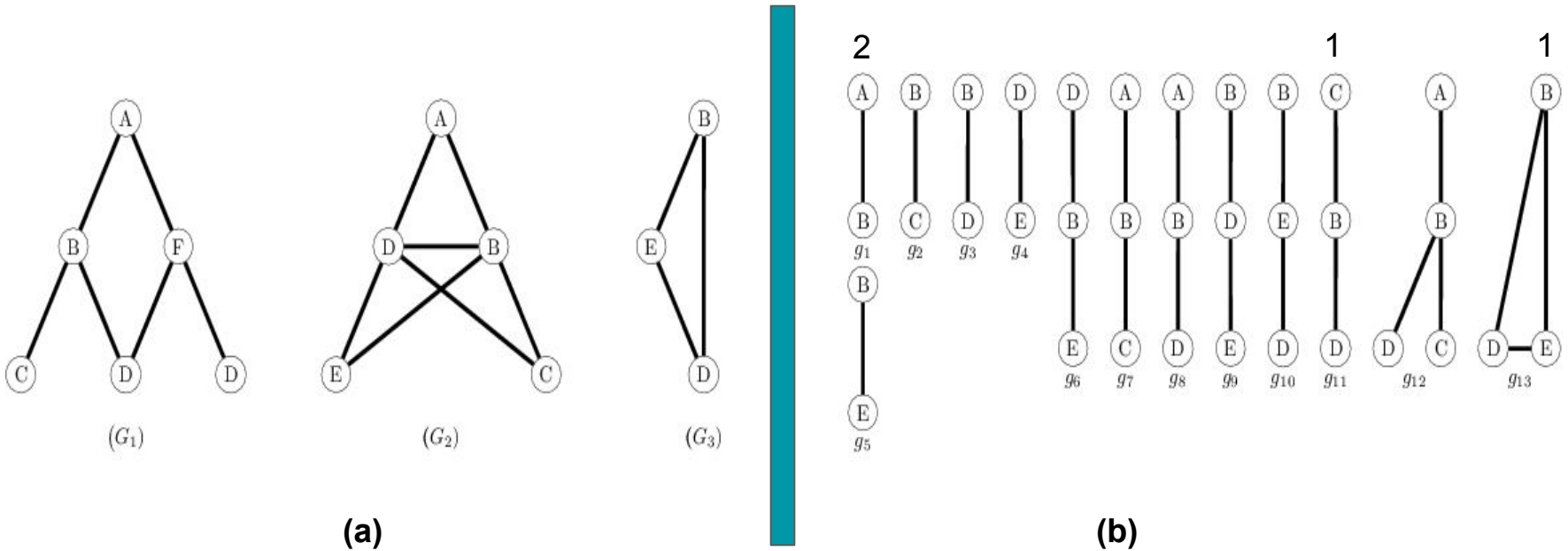
- It solves the lack of scalability problem by sampling fixed-size pattern instead of complete enumeration
- Instead of using support as a input, the method takes size as input,
  - For motif detection, typical size of functional motifs is usually known
- We superimpose each of the top-k frequent patterns to find functional motif in the conformation database and merge patches to obtain the complete functional motifs

FS^3 Algorithm

- A fixed size subgraph sampler
- Performs sampling in two stages. At the first stage, it choose one graph in the graph database. In the second stage, it samples a size-*l* subgraph from the chosen graph.
- The sampling distribution in second stage is biased such that it over-samples the graphs that are likely to be frequent over the entire database. The sampling is done via Markov chain Monte carlo (MCMC) sampling
- FS^3 algorithm repeat the sampling process for many times, and uses an innovative priority queue to hold a small set of most frequent subgraphs

Tanay Kumar Saha and Mohammad Hasan, "FS^3: A sampling based method for top-k frequent subgraph mining", Journal of statistical analysis and data mining, 2015

Mining Frequent Subgraphs (Sampling)

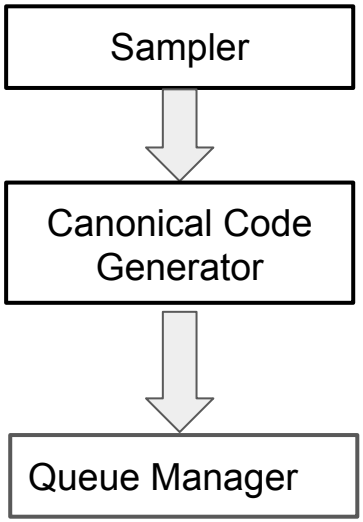


(a) **(b)**

(a) A graph database with 3 graphs (b) Sampled subgraphs show a number associated with those, which is their observed frequency through sampling

What sampling distribution should we use?

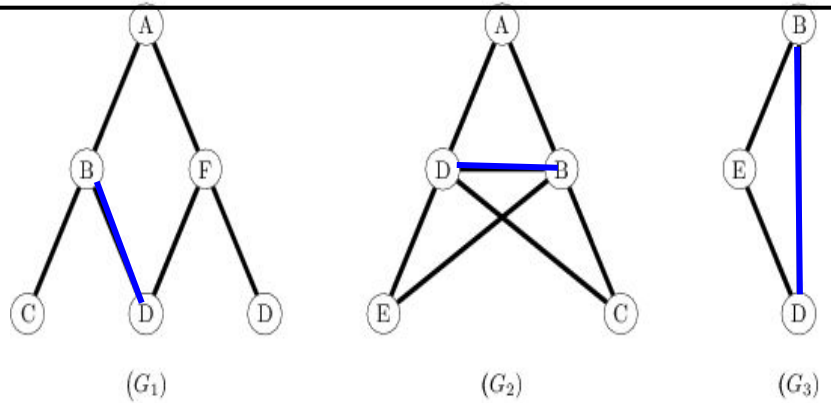
FS^3 Algorithm



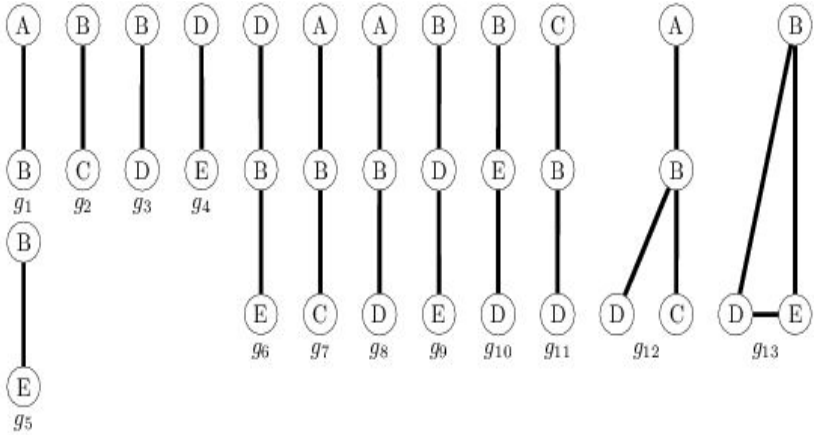
- Sampler samples a *l*-size subgraph in proportion to the set-intersection support of a subgraph
- Set intersection support is an upper bound of actual support of that pattern
- Queue is sorted based on canonical code, their max support and arrival time in the queue (3-criterion)
- Queue manager dynamically maintains the top-k frequent patterns
- Generation of canonical code is the most time consuming step.

FS^3 (Target Distribution of sampling)

Support (BD) = 3 {1,2,3}, Support (BE) = 2 {2,3},
Support (ED) =2 {2,3}



(a)



(b)

(a) A graph database with 3 graphs (b) All frequent subgraph of the graph database in (a) using minimum support value of 2

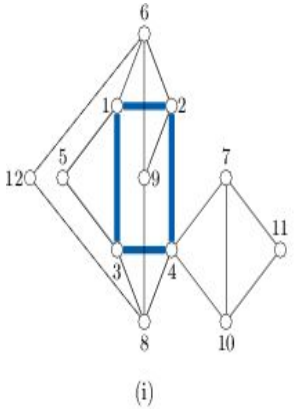
Sampling Induced Subgraph using Metropolis-Hastings Algorithms

```
SAMPLEINDSUBGRAPH(G_i, p)
 1  x = State saved at G_i
 2  d_x = Neighbor-count of x
 3  a_sup_x = score of graph x
 4  while (a neighbor state y is not found)
 5        y = a random neighbor of x
 6        d_y = Possible neighbor of y
 7        a_sup_y = score of graph y
 8        accp_val = (d_x * a_sup_y)/(d_y * a_sup_x)
 9        accp_probablility = min(1, accp_val)
10        if uniform(0, 1) ≤ accp_probability
11              return y
```

Fig: Sample Induced Subgraph

Proposal distribution: Uniform
Target distribution: Proportional to Upper bound of support

Neighborhood Generation in FS^3



(i)

1: $\langle 1,2,3,5\rangle, \langle 1,2,4,5\rangle, \langle 1,3,4,5\rangle, \langle 1,2,3,6\rangle, \langle 1,2,4,6\rangle, \langle 1,3,4,6\rangle$

2: $\langle 1,2,3,6\rangle, \langle 1,2,4,6\rangle, \langle 2,3,4,6\rangle, \langle 1,2,3,9\rangle, \langle 1,2,4,9\rangle, \langle 2,3,4,9\rangle$

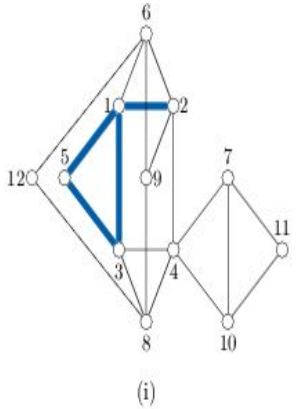
3: $\langle 1,2,3,5\rangle, \langle 1,3,4,5\rangle, \langle 2,3,4,5\rangle, \langle 1,2,3,8\rangle, \langle 1,3,4,8\rangle, \langle 2,3,4,8\rangle$

4: $\langle 1,2,4,7\rangle, \langle 1,3,4,7\rangle, \langle 2,3,4,7\rangle, \langle 1,2,4,8\rangle, \langle 1,3,4,8\rangle, \langle 2,3,4,8\rangle$
$\langle 1,2,4,10\rangle, \langle 1,3,4,10\rangle, \langle 2,3,4,10\rangle$

(ii)

(a)

(i) A database graph G_i with the current state of FS^3's random walk (ii) Neighborhood information of the current state <1,2,3,4>



(i)

1: $\langle 1,2,3,6\rangle, \langle 1,2,5,6\rangle, \langle 1,3,5,6\rangle$

2: $\langle 1,2,3,4\rangle, \langle 1,2,3,6\rangle, \langle 1,2,3,9\rangle, \langle 1,2,4,5\rangle, \langle 1,2,5,6\rangle, \langle 1,2,5,9\rangle$
$\langle 2,3,4,5\rangle, \langle 2,3,5,6\rangle, \langle 2,3,5,9\rangle$

3: $\langle 1,2,3,4\rangle, \langle 1,2,3,8\rangle, \langle 1,3,4,5\rangle, \langle 1,3,5,8\rangle, \langle 2,3,4,5\rangle, \langle 2,3,5,8\rangle$

5: $\langle\rangle$

(ii)

(b)

(i) The state of random walk on G_i (Figure (a)) after one transition (ii) Updated Neighborhood information

Finding sub-network embedding in the interface graph

- Most of the top-frequent subgraphs are almost identical
- After embedding, they map to a patch of the functional motif in such a way that superimposition of the embedded patches of multiple top-frequent patterns cover the entire motif
- For HIV-1 Protease, we consider 10 of the frequent subgraphs, and the embedding with superimposition covers the entire 16-residue dimeric lock motif in 323 out of 329 patterns
- Similar treatment for the TIM protein using 20 most frequent subgraphs finds the dimeric lock in 50 out of 86 structures.

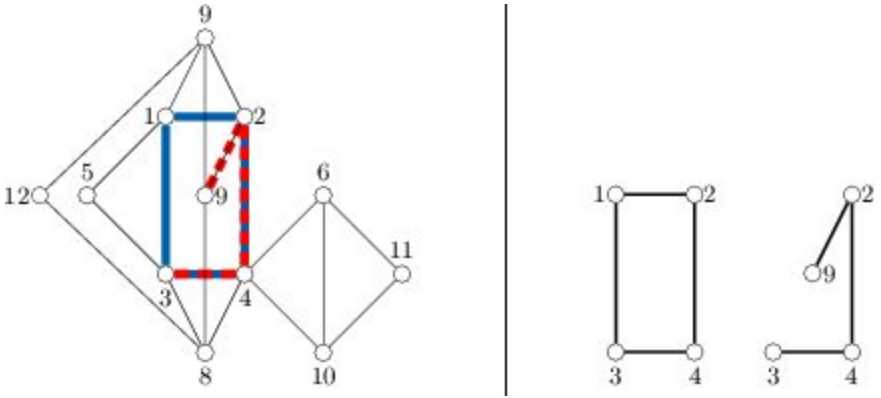


Fig: Subnetwork patches embedded in an interface graph.

Experimental result

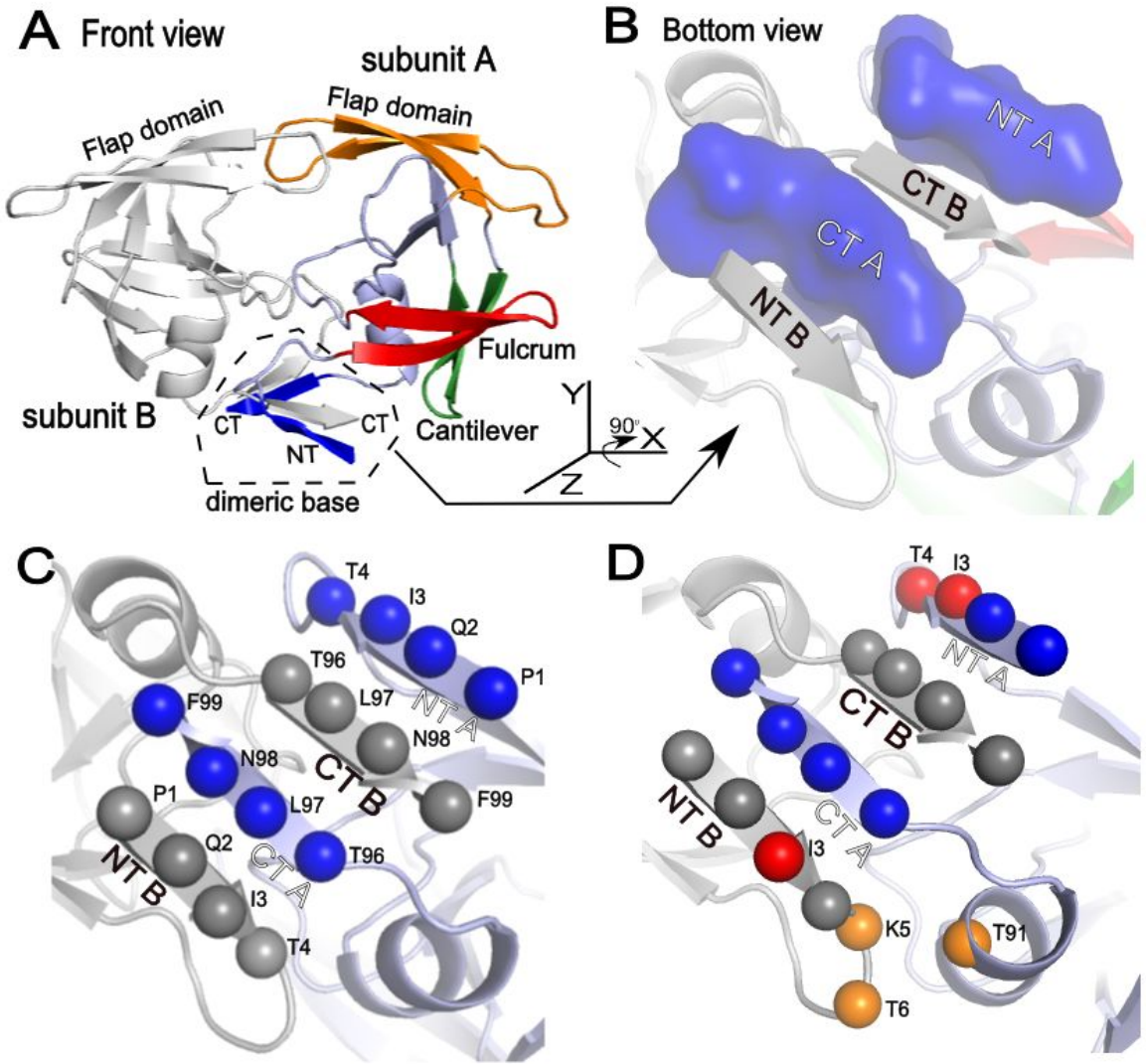


Fig: HIV-1 Protease (Lock structure)

Experimental Result

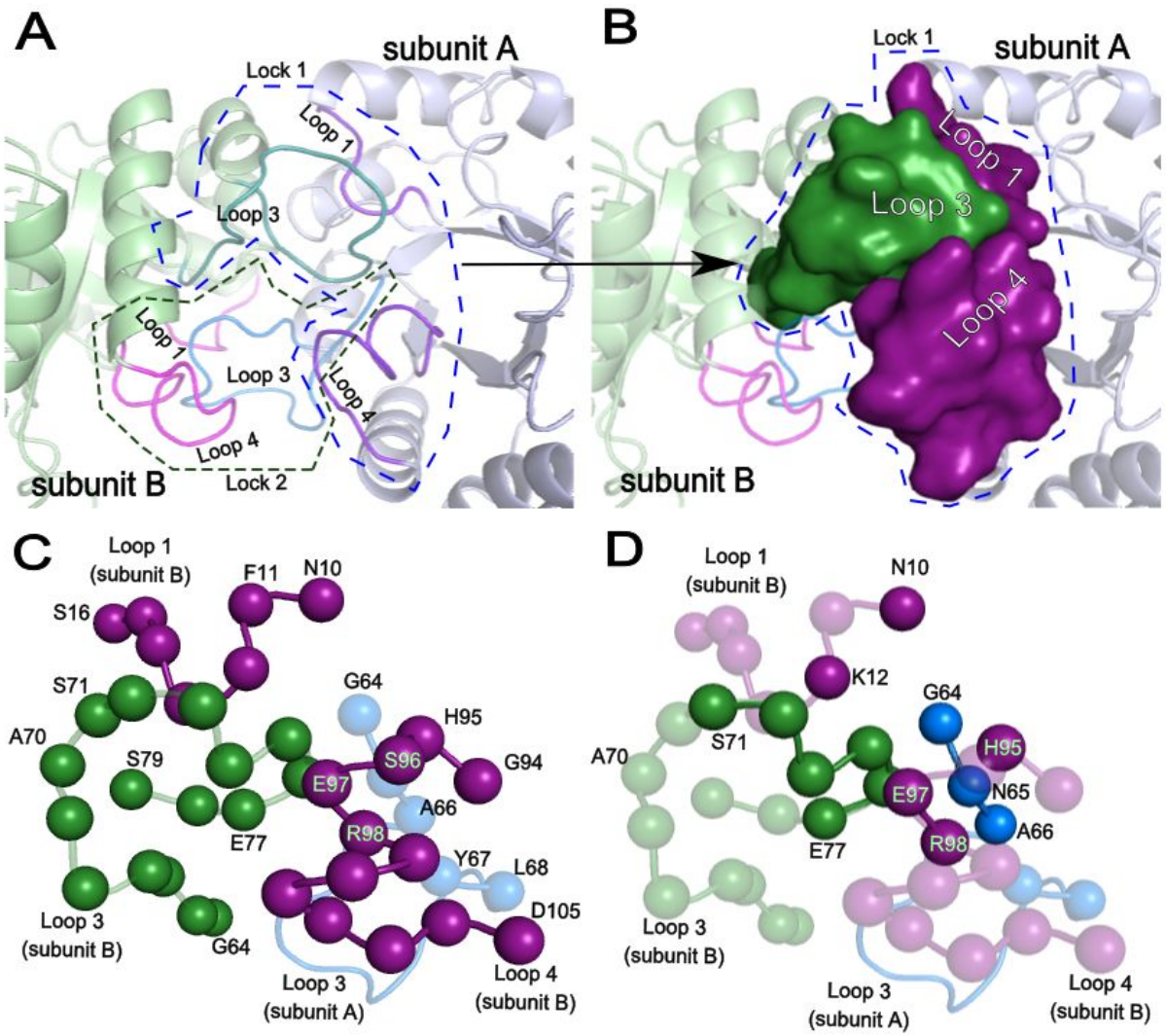


Fig: TIM
(hugging point)

Future work

- Using more datasets to see the hypothesis holds for multiple datasets
- Using clustering techniques for cluster the mined frequent subgraphs to overlap the patches to obtain the complete functional motif.

Conclusion

- Propose a method for the discovery of functional motifs from the interface region of dimeric protein structures
- The method uses the graphical representation of the interface region of these structures and mine fixed size highly frequent subgraphs
- The method captures the locking mechanism at the dimeric interface by taking only into account the spatial positioning of the interfacial residues through graphs