Stanford CS224W: Finding Frequent Subgraphs

CS224W: Machine Learning with Graphs

Jure Leskovec, Stanford University

http://cs224w.stanford.edu

Till now we have seen the definition of subgraph and motifs. In previous section we saw NN method to check subgraph. Now problem is more complex of counting their frequency.

As we know by increasing nodes, all possible subgraph (connected) explodes super-exponentially so tradition method can handle 5-7 nodes and on top of this for each subgraph need to find frequency. So this makes problem double complex. In these slide we will address this problem.

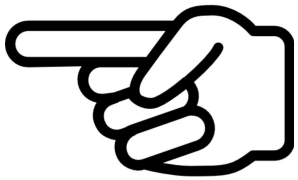
Plan for Today

1) Subgraphs and Motifs

- Defining Subgraphs and Motifs
- Determining Motif Significance

2) Neural Subgraph Representations

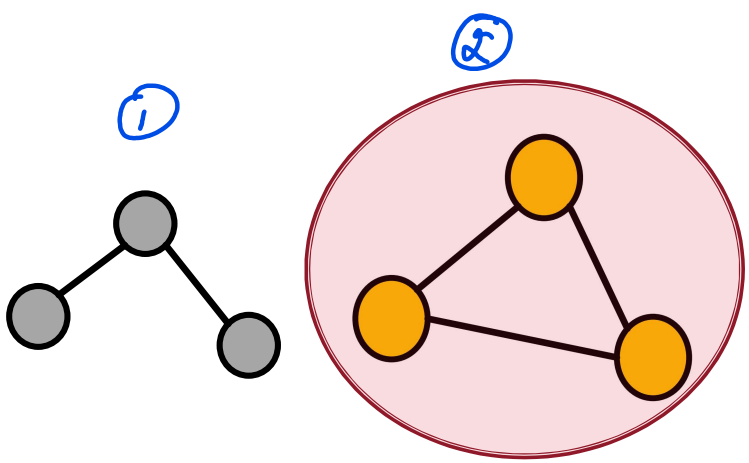
3) Mining Frequent Subgraphs



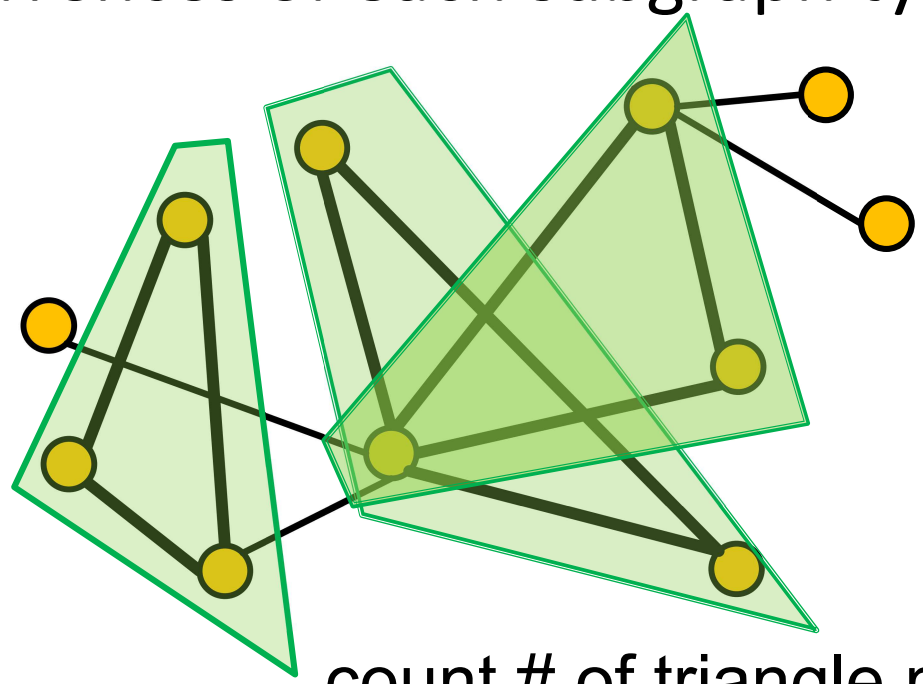
Finding Frequent Subgraphs

Combinatorial approach:

- Finding the most frequent size-k motifs requires solving two challenges:
  - **1) Enumerating** all size-k connected subgraphs
  - **2) Counting** #(occurrences of each subgraph type)



Possible size-3 motifs



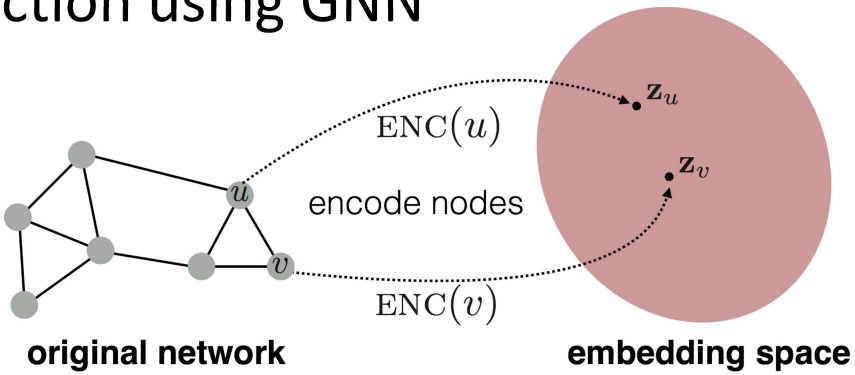
count # of triangle motifs

Why is it Hard?

- **Just knowing if a certain subgraph exists in a graph is a hard computational problem!**
  - Subgraph isomorphism is NP-complete (GNN)
    (for single subgraph)
- **Computation time grows exponentially as the size of the subgraphs increases** (Multiple subgraph)
  - Feasible motif size for traditional methods is relatively small (3 to 7)

Solution with Representation Learning

- Finding frequent subgraph patterns is **computationally hard** → Motif
  - **Combinatorial explosion** of number of possible patterns for given no. of node
  - Counting **subgraph frequency** is NP-hard
- **Representation learning** can tackle these challenges: with 2 step:
  - ① **Combinatorial explosion** → Find all possible subgraph from given # node. → organize the search space
  - ② **Subgraph isomorphism** → prediction using GNN



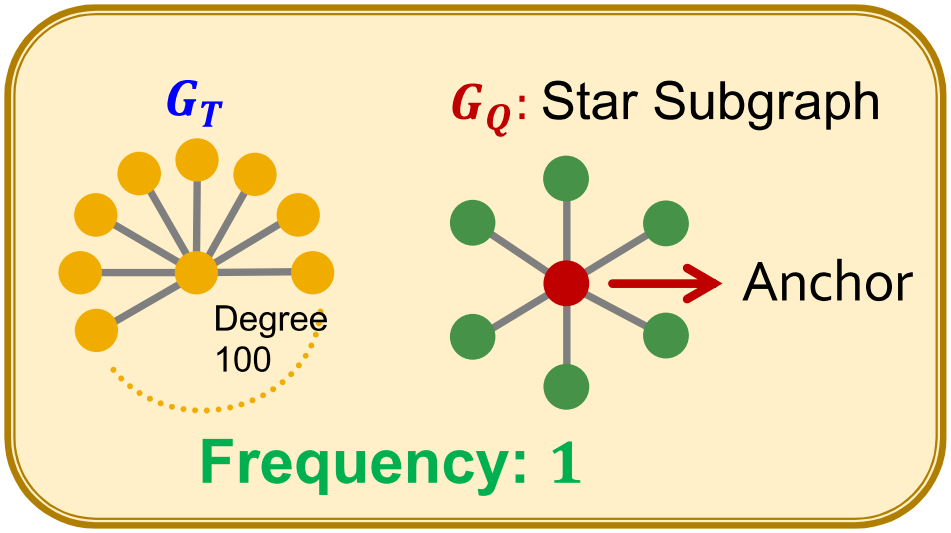
We need to keep in mind that problem is frequent subgraph mining. Here we do not need to find frequency of all possible subgraph rather which patter is most frequent. This challenge can be solved using representation learning with following steps:

1. Counting of given subgraph can be done using GNN as prediction problem. Rather counting we will predict.

2. Enumerating all size-k subgraph can be translated into search problem where we start with small size and grow step-by-step such that frequency count maximizes. With this we will be able to find most frequent motifs.

Problem Setup: Frequent Motif Mining

- Target graph (dataset) G_T, size parameter k
- Desired number of results r
- **Goal**: Identify, among *all* possible graphs of k nodes, the r graphs with the highest frequency in G_T.
- **We use the node-level definition:**
  The number of nodes u in G_T for which some subgraph of G_T is isomorphic to G_Q and the isomorphism maps u to v.



SPMiner Overview

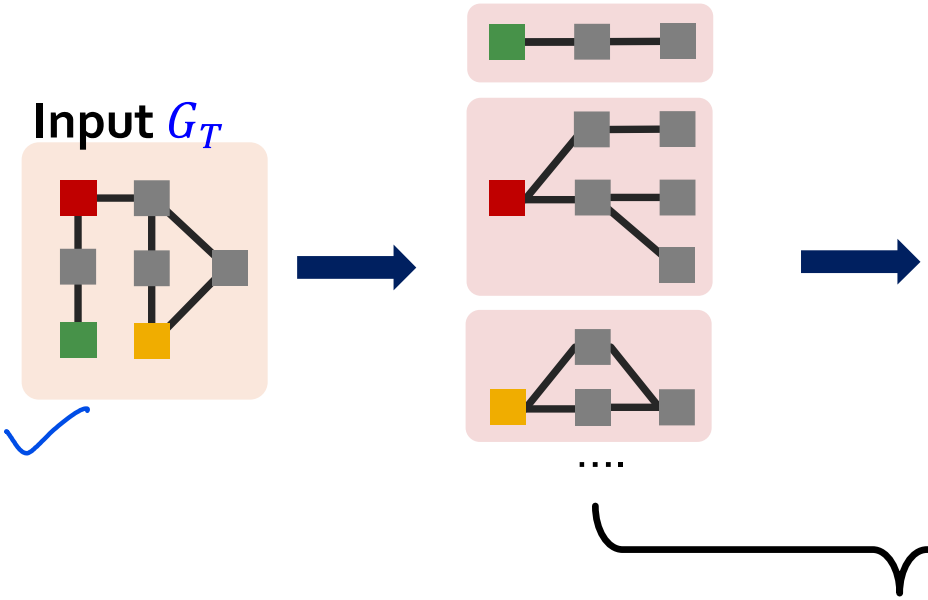
SPMiner: a neural model to identify frequent motifs

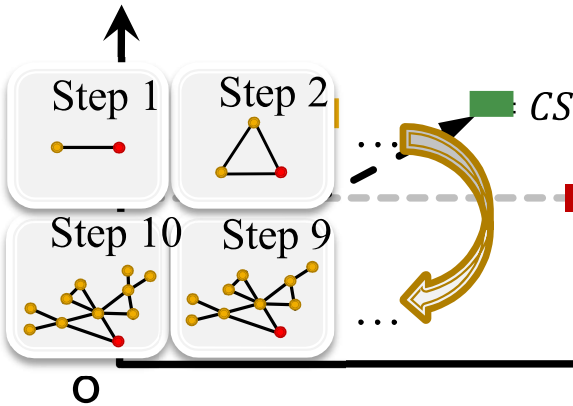
Decompose: overlapping *node-anchored* neighborhoods

Encoder: Embed subgraphs into *order embedding space*

Search Procedure: find frequent subgraphs by *growing* patterns

Input G_T





Same as neural subgraph matching

SPMiner: Key Idea

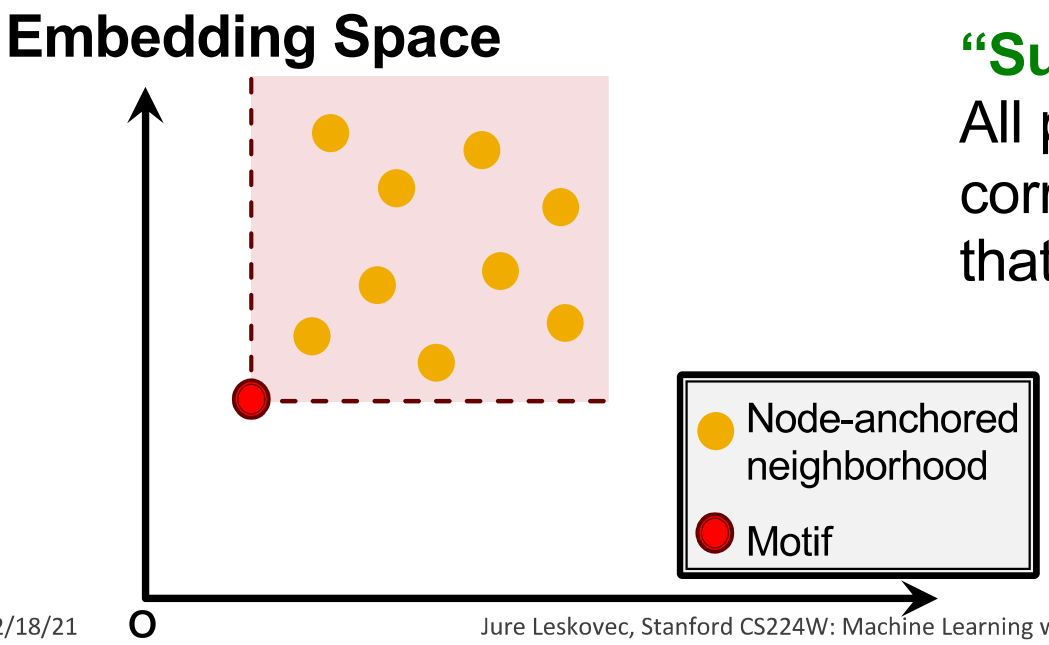
① - Decompose input graph G_T into neighborhoods

② - Embed neighborhoods into an order embedding space

③ - **Key benefit of order embedding:** → frequency prediction
We can quickly find out the frequency of a given subgraph G_Q → Predict the frequency based on order embedding

Motif Frequency Estimation

- **Given:** Set of subgraphs ("node-anchored neighborhoods") G_{N_i} of G_T (sampled randomly)
- **Key idea:** Estimate frequency of G_Q by counting the number of G_{N_i} such that their embeddings z_{N_i} satisfy $z_Q \leq z_{N_i}$
- Consequence of the **order embedding space property**

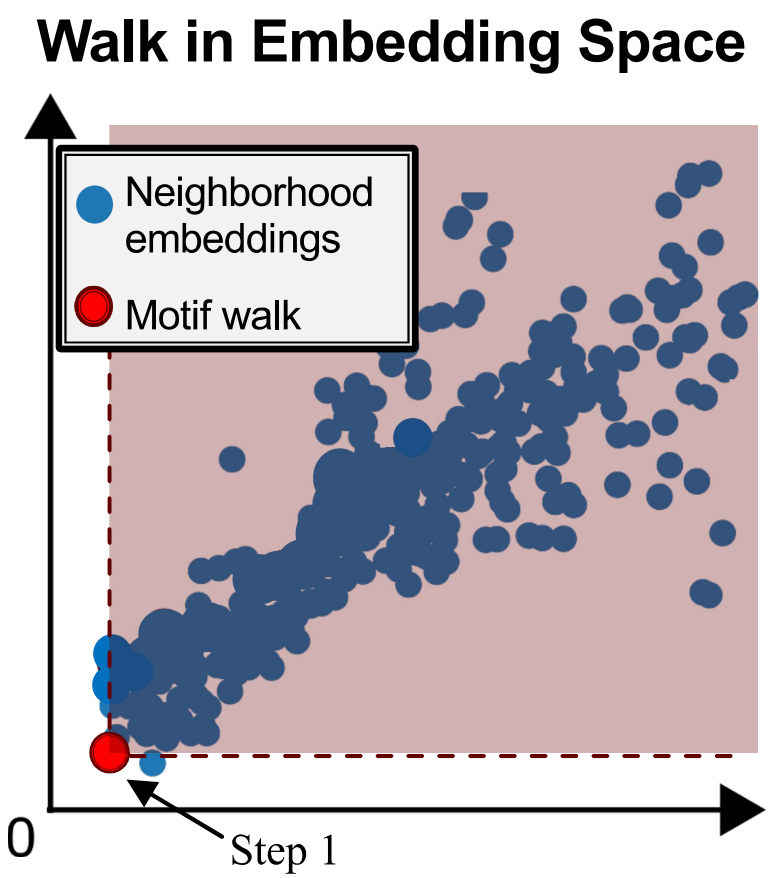


"Super-graph" region:
All points in the red shaded region correspond to neighborhoods in G_T that contain G_Q

Benefit: Super fast subgraph frequency counting!

SPMiner Search Procedure (1)

Initial step: Start by randomly picking a starting node u in the target graph. Set $S = \{u\}$



Each point in the shaded region represents a neighborhood in target graph that contains the motif pattern

Step 1

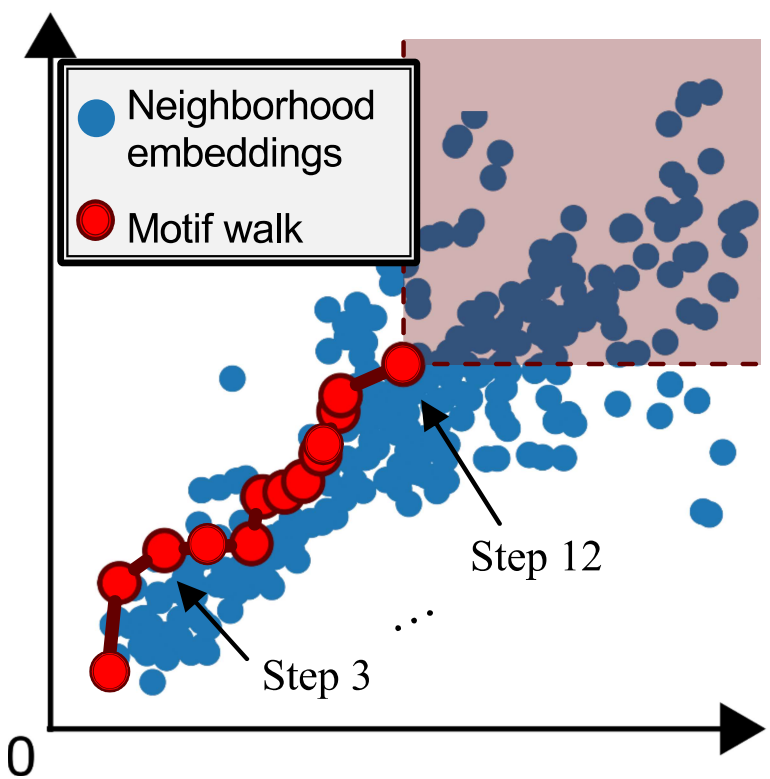
u

Initially, all neighborhoods contain the trivial subgraph

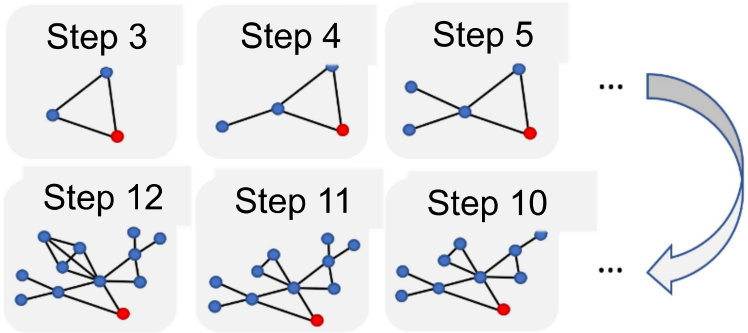
SPMiner Search Procedure (2)

Iteratively: Grow a motif by iteratively choosing a neighbor of a node in S, and adding that node to S
We want to grow motifs to find **larger** motifs!

Walk in Embedding Space



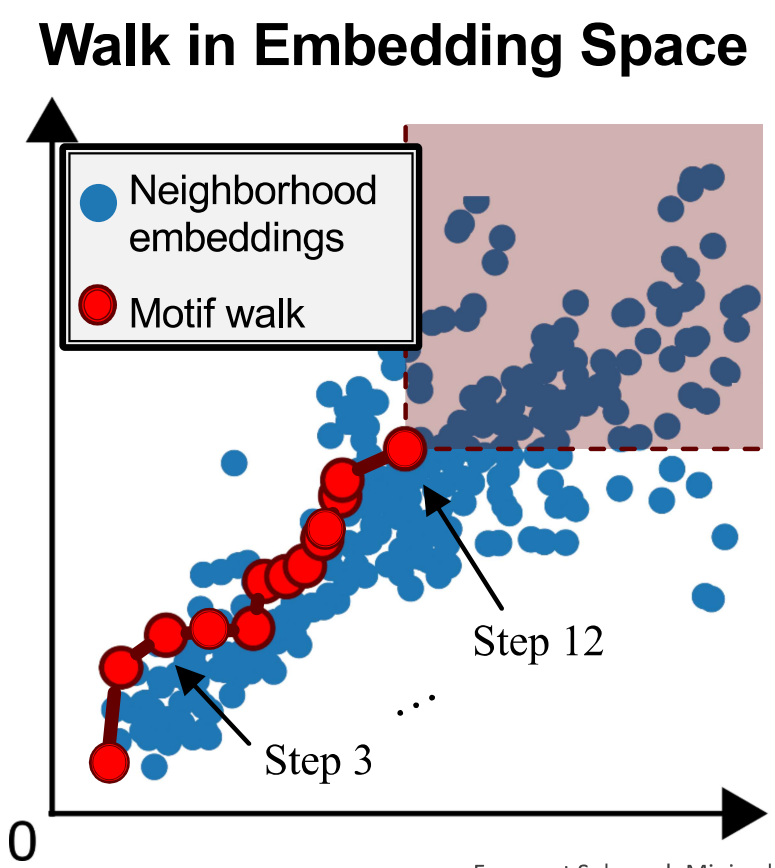
- Small motifs grow by adding neighbors
- Their embeddings correspond to red points on the left

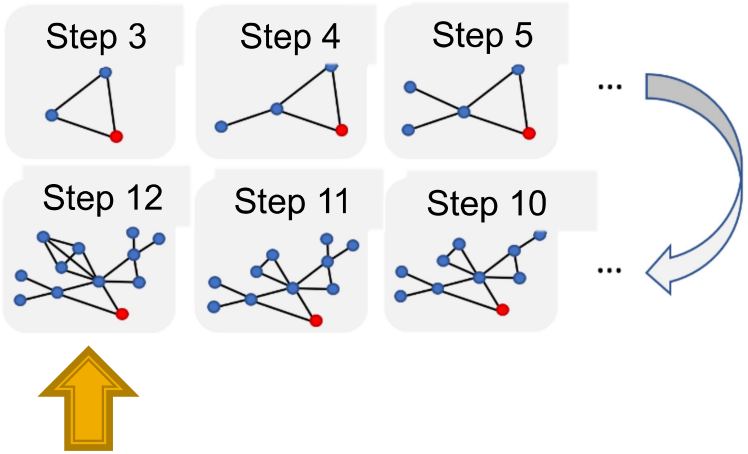


Goal: maximize number of neighborhoods in red shaded area after k step!

SPMiner Search Procedure (3)

Termination: Upon reaching a desired motif size, take the subgraph of the target graph induced by S





Identified frequent motif of size 12: It has the largest number of blue points in super-graph region, among all embeddings of possible subgraphs of size 12

Frequent Subgraph Mining by Walking in Order Embedding Space

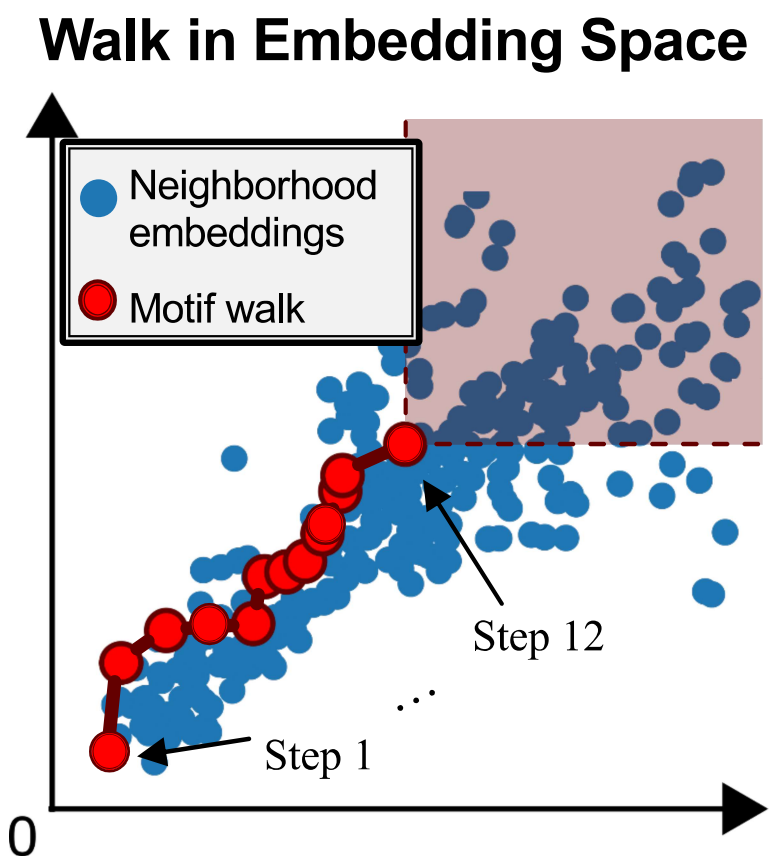
SPMiner Search Procedure (4)

How to pick which node to add at each step?

Total violation of a subgraph G: the number of neighborhoods that do not contain G.

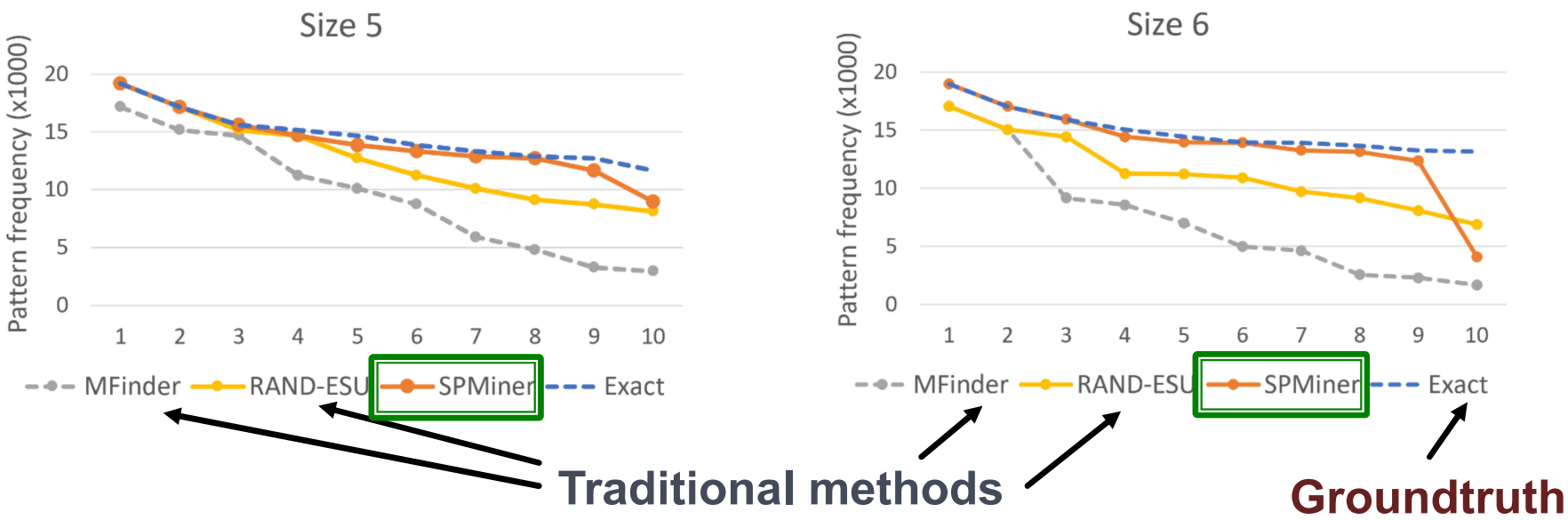
- The number of neighborhoods G_{N_i} that do **not** satisfy $z_Q \preccurlyeq z_{N_i}$
- Minimizing total violation = maximizing frequency

Greedy strategy (heuristic): At every step, add the node that results in the **smallest total violation**



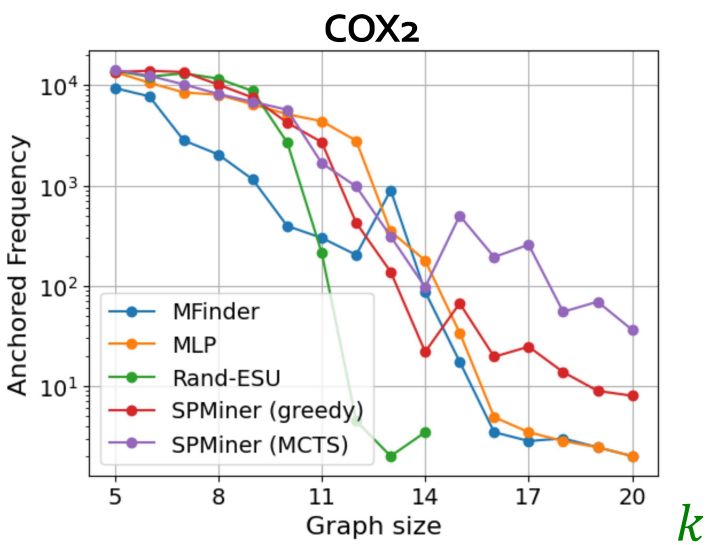
Results: Small Motifs

- **Ground truth**: find most frequent 10 motifs in dataset by brute-force exact enumeration (expensive)
- **Question**: Can the model identify frequent motifs?
- **Result**: The model identifies 9 and 8 of the top 10 motifs, respectively.

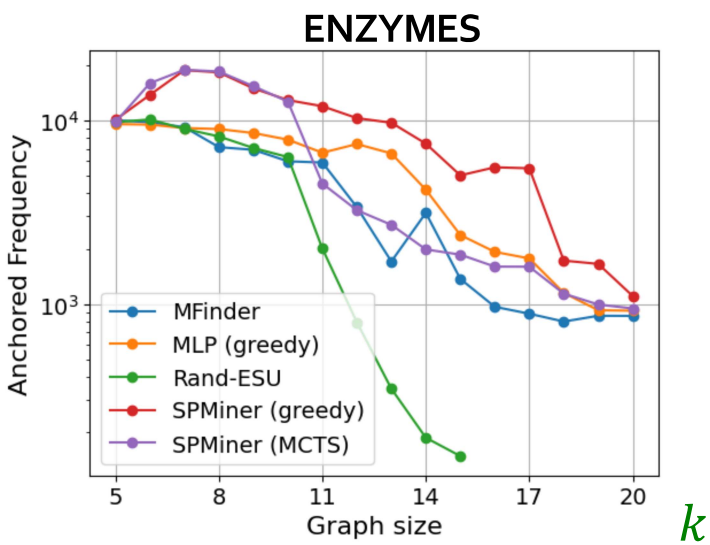


Experiments: Large motifs

- **Question**: how do the frequencies of the identified motif compare?
- **Result**: SPMiner identifies motifs that appear 10-100x more frequently than the baselines



Molecule dataset



Protein dataset

Summary

- **Subgraphs** and **motifs** are important concepts that provide insights into the structure of graphs. Their counts can be used as features for nodes and graphs.
- We covered **neural approaches** to prediction subgraph isomorphism relationship.
- **Order embeddings** have desirable properties and can be used to encode subgraph relations
- **Neural embedding-guided search** in order embedding space can enable ML model to identify motifs much more frequent than existing methods