Color-Coding

Tianyi Chen, Zixiong Cheng, Yuan Zhang
<ctony@bu.edu> <chouchou@bu.edu> <yuan0331@bu.edu>

Motivation - PPI networks

Huge amount of gene data ⟶ Highly time consuming

By using **a method** ↓

The motifs or signaling pathways can be found in polynomial time.

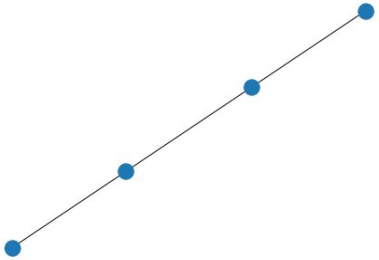
⟶ Enable us to explore more complex structures in PPI networks.

- The detection of signaling pathways in protein-protein interaction (PPI) networks.
- To discover and count the number of motifs in PPI networks.

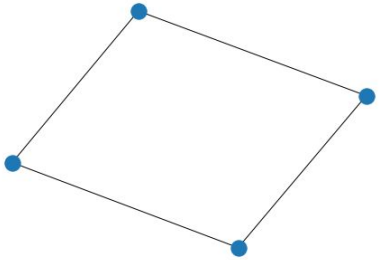
Problem definition

Given a graph G = (V,E), find small non-induced subgraphs (a.k.a. motifs) of a specific size k.

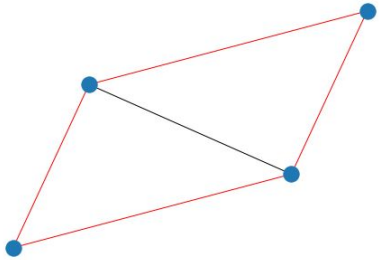
In this project, we focus mainly on sparse motifs including simple paths, cycles and forests of a specified length k.



4 - Path



4 - Cycle



Non-induced 4 - Cycle

Naive Method

We implemented a naive method to ensure finding **all the k-length paths** in the given graph **for all the nodes**.

- It can be used as the correct answer when we evaluate the random colorings results.

Implementation:

- Start from each nodes in the given graph
- Use backtrack strategy to find all possible k-length paths

Disadvantage: It's **slow**!

Color-Coding Method

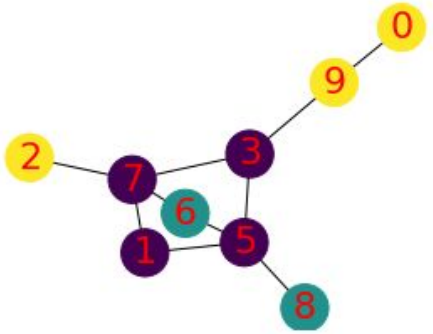
The color-coding method is a **randomized** method.

Color-Coding is useful in the **discovery of network motifs** and can solve **sparse motif counting problems** efficiently.

Time Complexity:

- $2^{O(k)} \cdot |E|$ in expectation to find paths/cycles of length k from one node.
- $2^{O(k)} \cdot |V||E|$ in expectation to find all paths/cycles of length k.

Color-Coding Method



Goal:

- Finding a subgraph $H = (V_H, E_H)$ in a given graph $G = (V, E)$, where H can be a path or a cycle and $k = |V_H| = O(\log V)$.
- Answer whether the given graph G contains a path or a cycle of length k?

Algorithm:

1. Randomly coloring each vertex of G with $k = |V_H|$ colors.
2. Try to find a colorful copy of H in colored G.

Probability of finding the colorful path

Colorful Path:

- A k-length colorful path means a path has a **k sequence of vertices** and every vertex in it is colored with a **distinct color**.

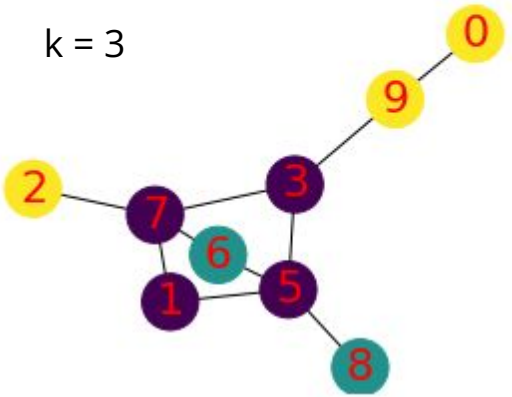
If we randomly color the whole given graph vertices with k different colors, then the probability P of finding a colorful path start from a random vertex is:

$$P = \frac{k!}{k^k} > e^{-k}$$

There are k^k ways of coloring the k vertices on the path, among which there are $k!$ colorful occurrences.

Example of finding the k-length colorful path

k = 3



Try: Start from 1

Cannot find a colorful path! (Since 5 and 7 have the same color as 1)

Try: Start from 2

2 → 7 → 6

Try: Start from 6

6 → 7 → 2

Derandomization

If we want to give every simple path of length k a chance of being discovered, what we need is a list of colorings of V such that for every subset $V' \subseteq V$ of size $|V'| = k$, there exists a coloring in the list that gives each vertex in V' a distinct color.

In other words, we need a k-perfect hash family mapping from $\{1,2,\ldots,|V|\}$ to $\{1,2,\ldots,k\}$.

Perfect hash family (1-probe hash family)

- Perfect hash function (1-probe function):
  - let the data set S comprise n elements belonging to the universe U: {1,2,3, … ,m}
  - A function h is called a 1-probe hash function if each element s ∈ S can be located in [1 … n] distinctly by applying h to s

- Perfect hash family:
  - A family H is a perfect hash family for U if every n-element subset S ⊂ U has some perfect hash function in H
  - In other words, what we need is a perfect hash family for vertex set V, so that each node in a random n-element subset S ⊂ V can be assigned to a distinct color
  - Proposed by Schimidt and Siegel, can be specified with O(k + loglogn) bits

Open Source

GitHub:

- https://github.com/c752334430/ColorCoding

PyPI install:

- pip install -i https://test.pypi.org/simple/ colorCoding

Implementation detail

- Python 3 + networkx
- Universal hash family $h(x) = (ax + b) \, mod \, p$
- Use integer+bitwise operation to store any possible color combination on one node
  - e.g. k=3, colors (1,0,1) can be encoded as 5
- Dynamic programming with n by 2^k bit matrix
  - Int list of size n In Python
- Retrive all pathes with backtracking

Experiment setup

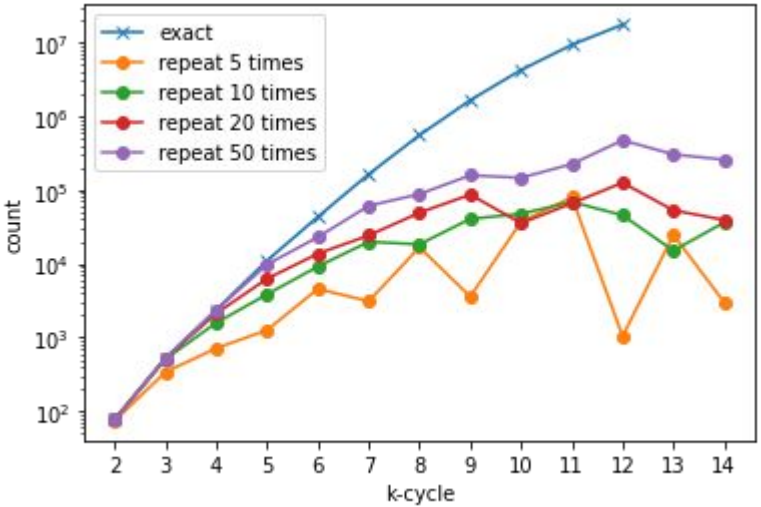
Dataset	Num of nodes	Num of edges
Karate club	34	78
Bitcoin-Alpha	3783	14124

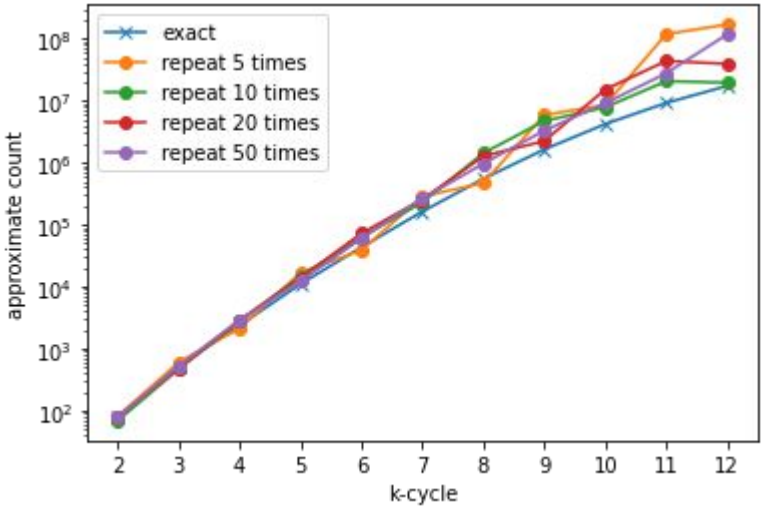
Naive: DFS like (but worse than DFS) algorithm.

Chiba-Nishizaki [4] (C-N): Exact clique counting algorithm.

Motivo [2]: induced motif occurrence estimation.

Experiments - Karate - Counting

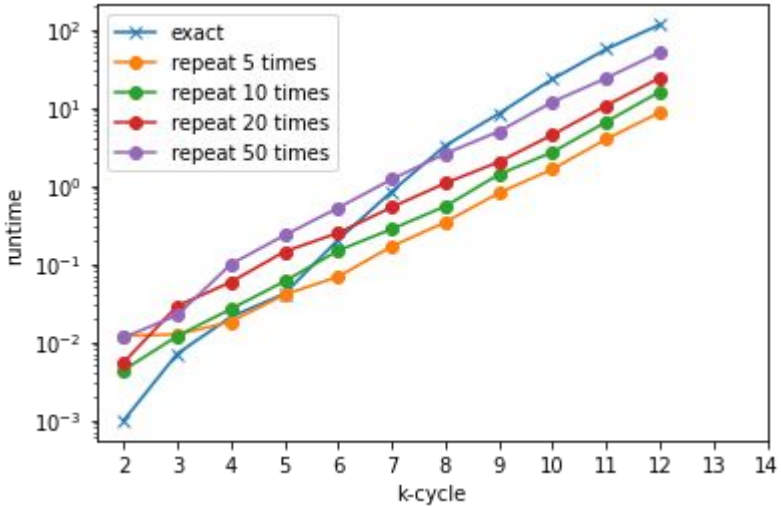




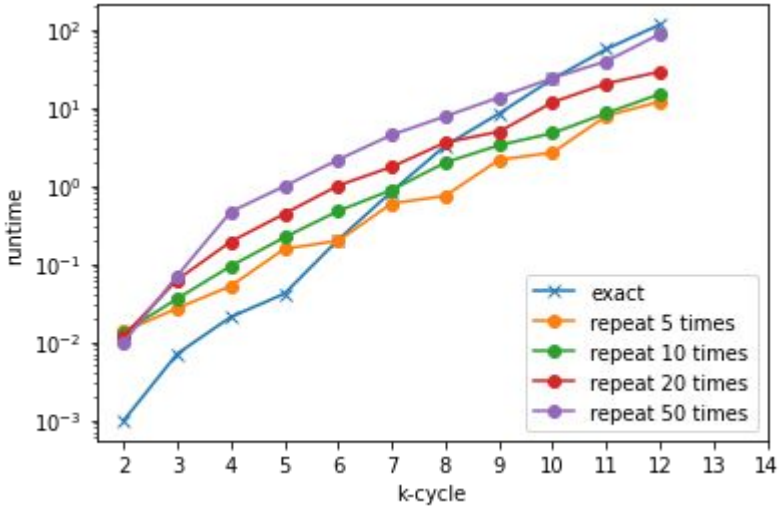
Exact count

Approximate by $\frac{k!}{k^k}$

Experiments - Karate - Running time



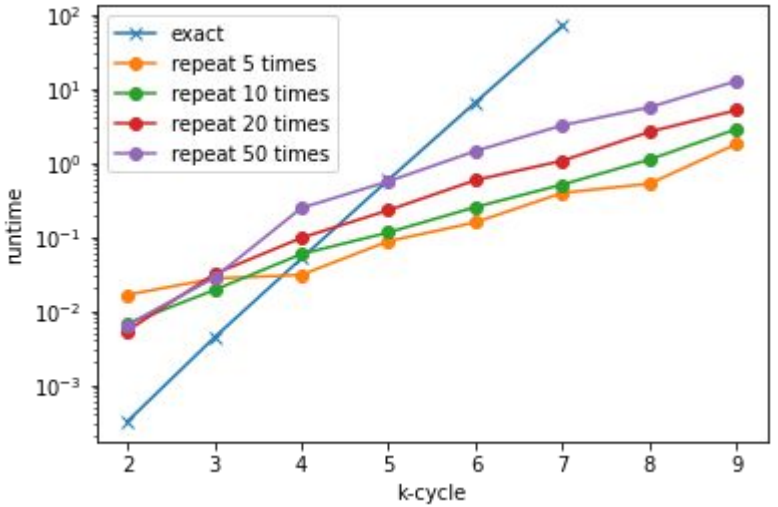
Without backtrack



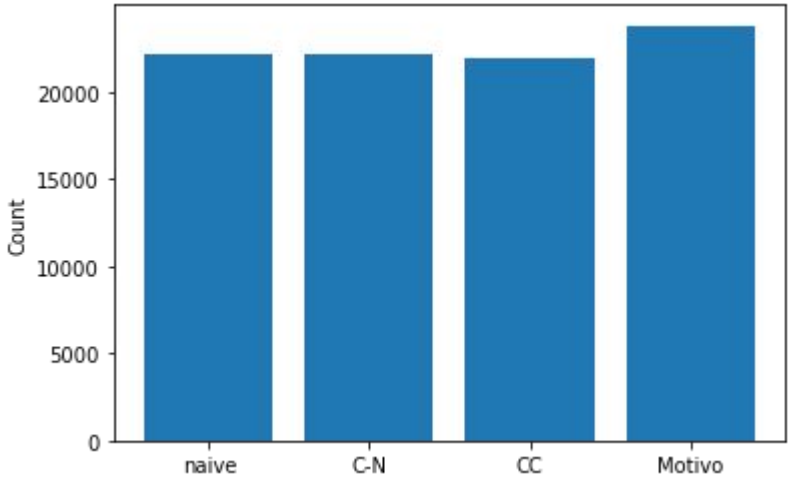
With backtrack

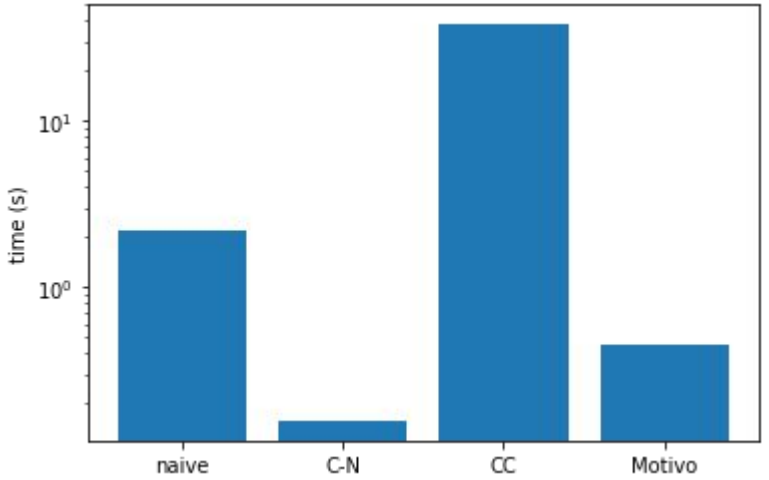
Experiment - synthetic graph

- Erdos renyi graph with n=25 and p=0.6.
- The brute force approach is way slower when the graph is dense.
    - The program for k=8 cannot finish within half an hour.



Triangle counting on Bitcoin-Alpha





Conclusions

In this project, we implement an open-sourced color-coding algorithm that is able to approximately count the non-induced sparse subgraphs.

The package is written in Python3 and its correctness is verified with both synthetic and real-life datasets.

While this is so far the first open-sourced Python color coding package, we observe the efficiency of it is not satisfying, possibly due to the limited bitwise operation support offered by Python. Therefore, we consider this package not efficient and comprehensive for practical usage, but rather an experimental tool.

References

[1]. Alon, N., Yuster, R., & Zwick, U. (1994, May). Color-coding: a new method for finding simple paths, cycles and other small subgraphs within large graphs. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing* (pp. 326-335).

[2]. Bressan, M., Leucci, S., & Panconesi, A. (2019). Motivo: fast motif counting via succinct color coding and adaptive sampling. *arXiv preprint arXiv:1906.01599*.

[3]. Schmidt, J. P., & Siegel, A. (1990). The spatial complexity of oblivious k-probe hash functions. *SIAM Journal on Computing*, *19*(5), 775-786.

[4]. Chiba, Norishige and Takao Nishizeki. “Arboricity and Subgraph Listing Algorithms.” *SIAM J. Comput.* 14 (1985): 210-223.

Q & A

Thanks for watching!