PROJECT PROPOSAL

Group:
Deepak Kukkapalli [vk23p]
Karthik reddy Vemireddy [kv23b]
Muqueet Mohsin Shaik [ms23ch]

Project Type: Implementation-flavour Project.
`
Title: Efficient Progressive Minimum k-Core Search

Name of the Conference: Association for Computing Machinery

- Proceedings of the VLDB Endowment, 2019-11, Vol.13 (3), p.361-374

Year it was published: 2019

The problem this paper is addressing: The authors are trying to find the smallest k-core subgraph containing query vertices. This is very helpful for applications like targeted recommendations(advertisements, social media suggestions), biological analysis etc. The issue with the current solutions like the S-Greedy algorithm is that they are NP-hard problems. So they lack quality guarantees and often give outputs with oversized subgraphs, increasing costs and reducing its relevance.

Project goal and motivation: This project's main objective is to develop a Progressive Search Algorithm (PSA) to find near-optimal k-cores efficiently and provide theoretical lower and upper bounds on subgraph size for maximum quality assurance and validating them by running the algorithm on real-world datasets like Email, Yelp, Youtube etc.

The (rough) methodology and plan:
Lower Bound Computation: Mapping the problem to set multi-cover and use LP relaxation/greedy heuristics to estimate the minimal required vertices.
Upper Bound Computation: Using onion-layer decomposition to refine candidate subgraphs iteratively.
Progressive Refinement: Tightening bounds iteratively, enabling early termination when a user-defined approximation ratio is achieved.

Timeline and schedule:
Around 8 weeks where we design four sudo algorithms which fits C++ environment:

- PSA algorithm: 2 weeks, Lower Bound computation algorithm (consists two variations) 4 weeks (2 weeks for each variation),Upper Bound computation algorithm 2 weeks.

Implementing the algorithm in C++: 5 weeks:

- This includes implementing all the four algorithms which we designed, PSA algorithm and then integrating it with 2 lower bound variations and an upper bound variation and output the k-core subgraph by tightening the lower and upper bounds.

Testing/evaluation: 3 weeks:

- We test our implemented C++ code with the real world datasets which were taken from SNAP and reiterate the design and implementation steps for any optimisations and potential mistakes.

Resources needed: Knowledge of algorithms (for assessing time and space complexity),C++,Data Structures (vectors,unordered maps,sets,priority queues),algorithms (Adjacency lists,DFS,top elements using heaps,set theory) real-world datasets (SNAP, KONECT).

The Workload Distribution and Implementation Plan:
Deepak: Majorly responsible for sudo algorithms designs, going throughout the paper for the necessary theories and concepts with the help of other two members .
Karthik: Majorly responsible for Code implementation in C++ using the resources mentioned above and optimizing the code according to testing done with the help of other two members .
Muqeet: Majorly responsible for testing and evaluating the implemented code and algorithms against the real world datasets like email,yelp,youtube taken from SNAP and propose the necessary optimizations and changes that are to be done with the help of other two team members.