Community Detection in Large Graphs

Davide Rucci, PhD Student in Computer Science

May 7, 2021

University of Pisa
Mauriana Pesaresi Seminar Series

Community Detection

Given a graph $G = (V, E)$ we say that

A community is a subset of nodes sharing "significantly many" connections with respect to the rest of the graph.

Community Detection

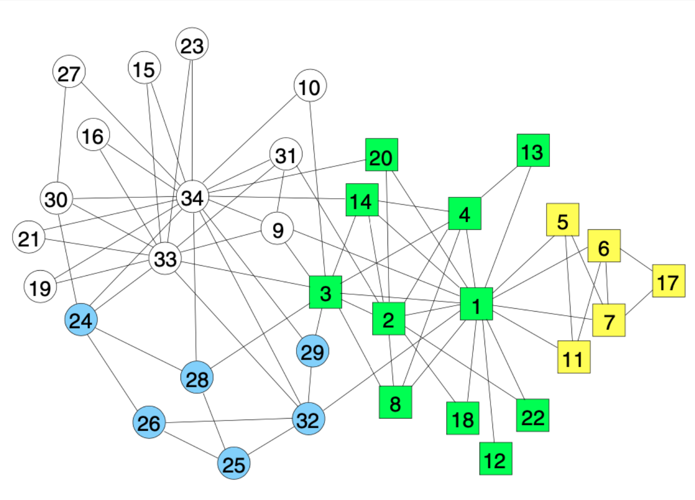


Figure 1: Real-life environment

Community Detection

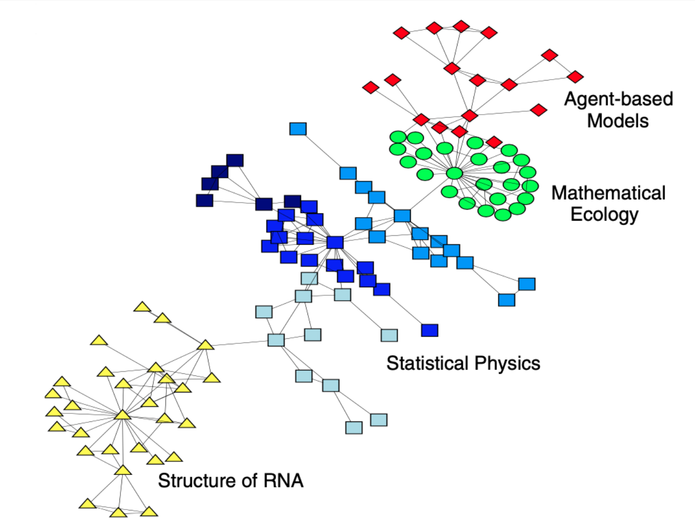


Figure 2: Co-authorships

Community Detection

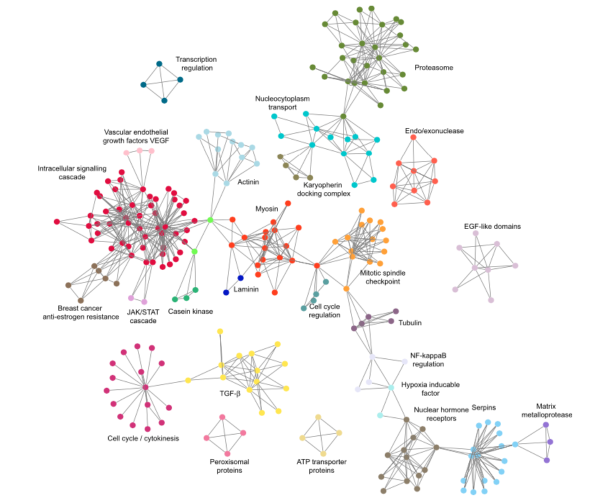
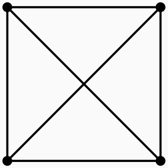


Figure 3: Protein-protein interactions

Community Formalizations

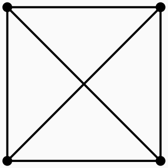
- Adjacency-based
  - Maximal Clique
  - Plexes
  - Graphlets
  - ... many others



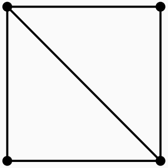
(a) 4-clique

Community Formalizations

- Adjacency-based
  - Maximal Cliques
  - Plexes
  - Graphlets
  - . . . many others



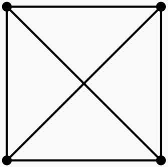
(a) 4-clique



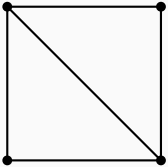
(b) 2-plex, 4 nodes

Community Formalizations

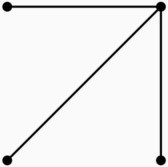
- Adjacency-based
  - Maximal Cliques
  - Plexes
  - Graphlets
  - . . . many others



(a) 4-clique



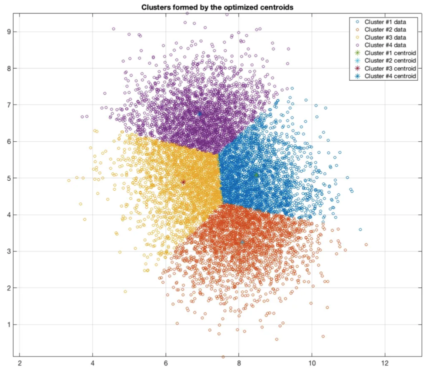
(b) 2-plex, 4 nodes



(c) 4-node graphlet

Community Formalizations

- Adjacency-based
  - Maximal Cliques
  - Plexes
  - Graphlets
  - ... many others
- Metric-based
  - Clusters



Problem Statement

- **Input**: A graph $G = (V, E)$, an integer k
- **Output**: A list of all the communities contained in G made up by (at least or exactly) k nodes

Enumerating Cliques

Listing all (maximal) cliques contained in a graph is a well-known NP-hard problem. Known algorithms include:

- Binary Partition Scheme
- Reverse Search Scheme

Enumerating Cliques

Listing all (maximal) cliques contained in a graph is a well-known NP-hard problem. Known algorithms include:

- Binary Partition Scheme
  - Bron-Kerbosch $\rightarrow O(4^{|V|/3})$ time
  - Tomita et al. $\rightarrow O(3^{|V|/3})$ time
  - Eppstein et al. $\rightarrow O(d|V|3^{d/3})$ time

  $\left.\begin{array}{l} \\ \\ \\ \end{array}\right\} O(|V| + q\Delta)$ space
- Reverse Search Scheme

Enumerating Cliques

Listing all (maximal) cliques contained in a graph is a well-known NP-hard problem. Known algorithms include:

- Binary Partition Scheme
  - Bron-Kerbosch $\rightarrow O(4^{|V|/3})$ time
  - Tomita et al. $\rightarrow O(3^{|V|/3})$ time
  - Eppstein et al. $\rightarrow O(d|V|3^{d/3})$ time

  $\left.\right\}$ $O(|V| + q\Delta)$ space
- Reverse Search Scheme
  - Conte et al. $\rightarrow \alpha\tilde{O}(min\{|E|d, qd\Delta\})$ time, $O\left(\sqrt{|E|}\right)$ space

Enumerating Cliques

- Binary Partition Scheme
  - Bron-Kerbosch $\rightarrow O(4^{|V|/3})$ time
  - Tomita et al. $\rightarrow O(3^{|V|/3})$ time
  - Eppstein et al. $\rightarrow O(d|V|3^{d/3})$ time

  $\left.\begin{array}{l} \\ \\ \\ \end{array}\right\} O(|V| + q\Delta)$ space
- Reverse Search Scheme
  - Conte et al. $\rightarrow \alpha\tilde{O}(min\{|E|d, qd\Delta\})$ time, $O\left(\sqrt{|E|}\right)$ space
- $O(3^{|V|/3})$ is worst-case optimal (Moon-Moser graphs).

Binary Partition

Binary Partition is a traditional technique largely adopted for enumeration.

- Recursive approach
- Given an element v of the input and a partial solution S, recursively proceed with
  - $ENUM(S \cup \{x\})$;
  - $ENUM(S)$ removing x from the input.

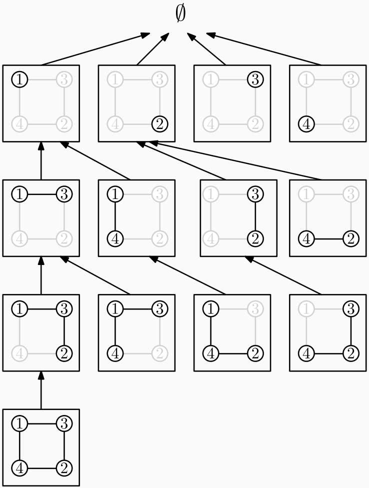
Binary Partition

The Binary Partition scheme on graphs corresponds to:

1. Pick $v \in V$
2. $ENUM(G, S \cup \{v\})$
3. $ENUM(G \setminus \{v\}, S)$

Reverse Search

Key idea: given a solution construct another solution using a *parent* rule. It explores the solution space.



Bron-Kerbosch Algorithm

Algorithm 1 Bron-Kerbosch Algorithm

function $\textsc{Bron-Kerbosch}(P, R, X)$
    if $P = \emptyset$ **and** $X = \emptyset$ **then**
        return ▷ R is a maximal clique
    end if
    for all $v \in P$ **do**
        $\textsc{Bron-Kerbosch}(P \cap N(v), R \cup \{v\}, X \cap N(v))$
        $P \leftarrow P \setminus \{v\}$
        $X \leftarrow X \cup \{v\}$
    end for
end function

Tomita and Eppstein

- Tomita et al. algorithm uses *pivoting* to achieve optimal worst-case running time

Tomita and Eppstein

- Tomita et al. algorithm uses *pivoting* to achieve optimal worst-case running time
  - Informally, this means choosing $v \in P \cup X$ such that $|P \cap N(v)|$ is maximized, then loop through $P \setminus N(v)$.

Tomita and Eppstein

- Tomita et al. algorithm uses *pivoting* to achieve optimal worst-case running time
    - Informally, this means choosing $v \in P \cup X$ such that $|P \cap N(v)|$ is maximized, then loop through $P \setminus N(v)$.
- Eppstein et al. proposed a variant built on top of Tomita et al.'s

Tomita and Eppstein

- Tomita et al. algorithm uses *pivoting* to achieve optimal worst-case running time
  - Informally, this means choosing $v \in P \cup X$ such that $|P \cap N(v)|$ is maximized, then loop through $P \setminus N(v)$.
- Eppstein et al. proposed a variant built on top of Tomita et al.'s
  - It exploits a *degeneracy ordering* of G.

Degeneracy Ordering

Definition: The *degeneracy* of a graph $G = (V, E)$ is the minimum $d \in \mathbb{N}$ for which there exists an ordering of V such that every vertex has at most d neighbors *later* in the ordering.

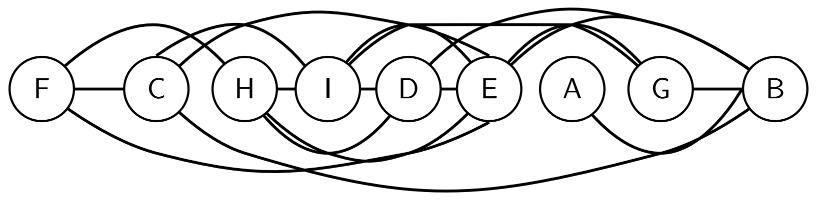


Figure 4: A graph with degeneracy $d = 3$

Output Sensitivity

- The algorithm by Eppstein et al. shows that clique enumeration is fixed parameter tractable $\rightarrow O(d|V|3^{d/3})$;

Output Sensitivity

- The algorithm by Eppstein et al. shows that clique enumeration is fixed parameter tractable $\rightarrow O(d|V|3^{d/3})$;
- *Output-sensitivity*: complexity expressed as a function of the output size;

Output Sensitivity

- The algorithm by Eppstein et al. shows that clique enumeration is fixed parameter tractable $\rightarrow O(d|V|3^{d/3})$;
- *Output-sensitivity*: complexity expressed as a function of the output size;
- Delay: Time spent between the output of two consecutive solutions;

Output Sensitivity

- The algorithm by Eppstein et al. shows that clique enumeration is fixed parameter tractable $\rightarrow O(d|V|3^{d/3})$;
- *Output-sensitivity*: complexity expressed as a function of the output size;
- Delay: Time spent between the output of two consecutive solutions;
- Eppstein et al. has exponential delay;

Output Sensitivity

- The algorithm by Eppstein et al. shows that clique enumeration is fixed parameter tractable $\rightarrow O(d|V|3^{d/3})$;
- *Output-sensitivity*: complexity expressed as a function of the output size;
- Delay: Time spent between the output of two consecutive solutions;
- Eppstein et al. has exponential delay;
- Tomita et al. also has exponential delay unless $\mathrm{P} = \mathrm{NP}$.

- There exist many algorithms for $k-$graphlet enumeration.

Graphlets

- There exist many algorithms for $k-$graphlet enumeration.
- The best state-of-the-art algorithm has a delay of $O(k^2\Delta)$.
    - k: size of the graphlets desired
    - Δ: maximum degree of the graph

Graphlets

- There exist many algorithms for $k-$graphlet enumeration.
- The best state-of-the-art algorithm has a delay of $O(k^2\Delta)$.
  - k: size of the graphlets desired
  - Δ: maximum degree of the graph
- Currently working on an improvement of this bound, using the so called Push-Out Amortization

Push-Out Amortization

Push-Out Amortization is a novel technique for output sensitive, bounded-delay enumeration.

Push-Out Amortization

Push-Out Amortization is a novel technique for output sensitive, bounded-delay enumeration. It exploits the structure of recursive (binary partition) trees.

Push-Out Amortization

Push-Out Amortization is a novel technique for output sensitive, bounded-delay enumeration. It exploits the structure of recursive (binary partition) trees.

Theorem: Let T^* be the time taken by a leaf node of the recursion tree. If all non-leaf nodes have

$$\sum_{Y \in C(X)} T(Y) \geq \alpha T(X) - \beta(|C(X)| + 1) T^* \qquad \alpha > 1, \beta \geq 0$$

then the delay of the algorithm is bounded by $O(T^*)$.

Push-Out Amortization in Short

- Child nodes should pay more than their parent;
- A non-leaf node must have at least two children;
- $|\text{leaves}| \geq |\text{internal nodes}|$
- The overall cost is dominated by the cost of leaves;

Graphlets

Our current findings are:

- A $O(k^2\Delta)$ delay practical algorithm;
- A $O(k^2)$ delay algorithm using push-out amortization;
- A $O(1)$ delay algorithm using push-out amortization (currently in development).

Conclusions

- Enumeration is easy to think, yet extremely difficult to optimize;
- There exists a whole hierarchy of complexity classes for enumeration;
- Push-out amortization is a very powerful technique and can be applied to a huge variety of enumeration problems;
  - k-edge subgraphs
  - All graphlets (no bounds on the size)
  - Matchings
  - Elimination Orderings
  - ...

Bibliography

- BK: https://doi.org/10.1145%2F362342.362367
- Tomita: https://doi.org/10.1016/j.tcs.2006.06.015
- Eppstein: https://doi.org/10.1145/2543629
- Enum. Complexity: http://bulletin.eatcs.org/index.php/beatcs/article/view/596/605
- Reverse Search Cliques: https://doi.org/10.4230/LIPIcs.ICALP.2016.148
- Push-Out Amortization: https://link.springer.com/content/pdf/10.1007/978-3-319-21840-3_49.pdf

Thank You!

Thank you for your attention!

My contacts:
email: davide.rucci@phd.unipi.it
office: Room 300, Computer Science Dept. (Polo Fibonacci, ed. C, 2nd floor)