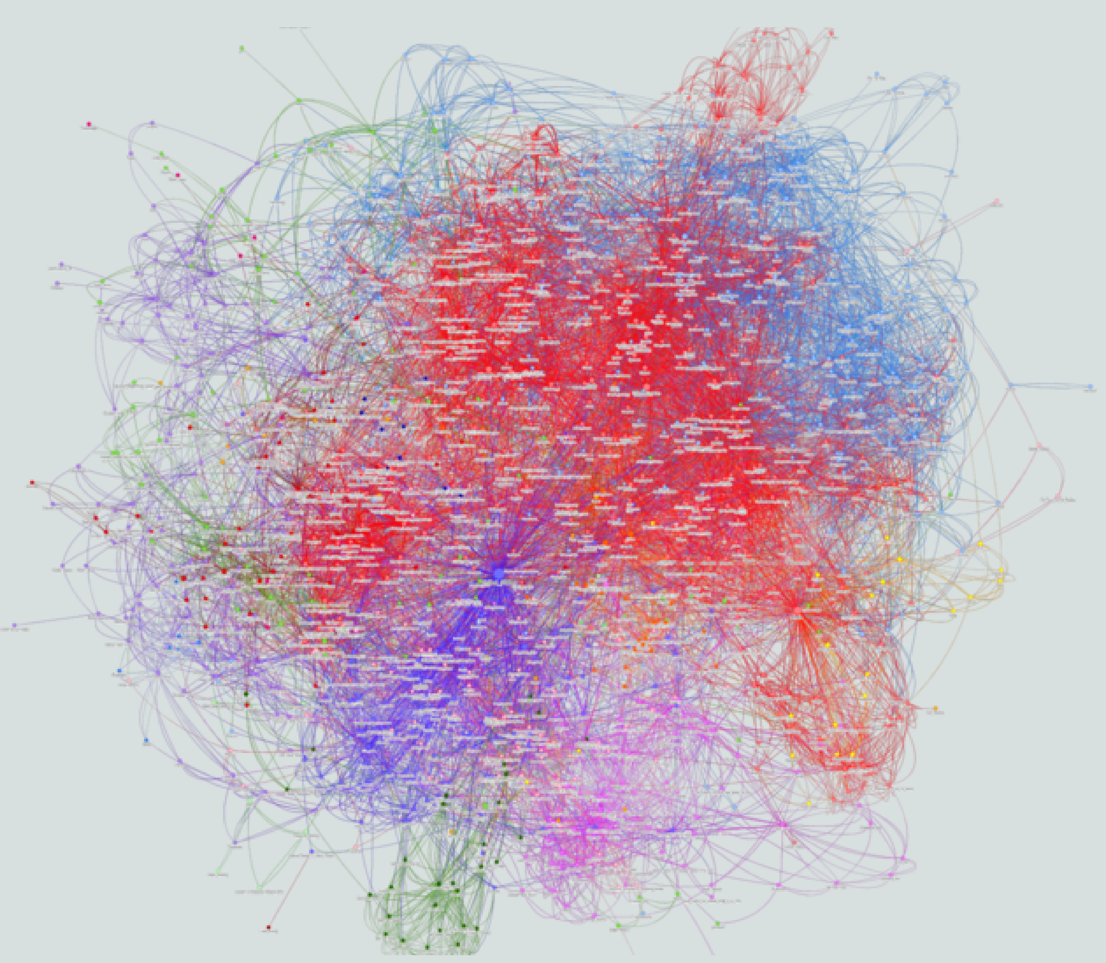
CLIQUE PERCOLATION METHOD : MEMORY EFFICIENT ALMOST EXACT COMMUNITIES

Automatic detection of relevant groups of nodes in large real-world graphs, i.e. community detection, has applications in many fields and has received a lot of attention in the last twenty years. The most popular method designed to find overlapping communities (where a node can belong to several communities) is perhaps the Clique Percolation Method (CPM).



AUTHORS

Alexis Baudin*, Maximilien Danisch*, Sergey Kirgizov**, Clémence Magnien*, and Marwan Ghanem*

* Sorbonne Université, CNRS, 75005 Paris, France

** LIB, Université de Bourgogne Franche-Comté, 21078 Dijon, France

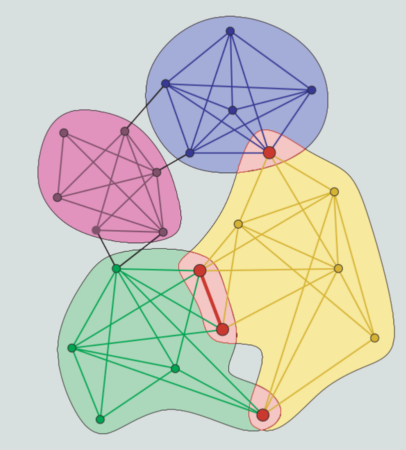
SORBONNE UNIVERSITÉ · cnrs · LIB · Complex Networks fr · ADMA'21

CONTEXT

COMMUNITIES IN A GRAPH

<u>Set of nodes:</u>

- Densely connected *inside*;
- Sparsly connected *outside*.



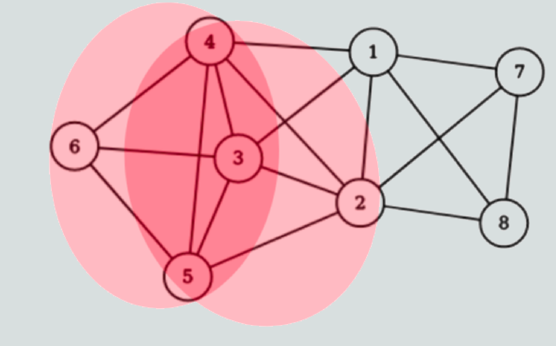
Palla *et al.*, 2005

INTEREST

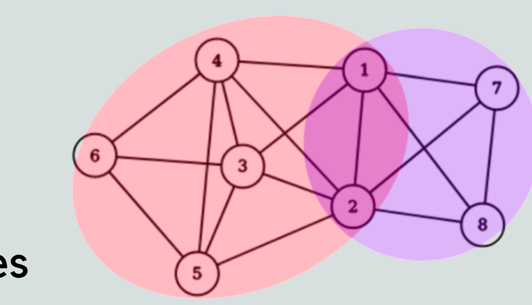
- Massive graphs : zoom in and out;
- Biological interactions;
- Content recommendation;
- ...

K-CLIQUE COMMUNITY

- **k-clique:** set of k not fully connected to each other.
- Two k-clique are **adjacent** if they share k-1 nodes.
- A **k-clique community** (CPM community), is the set of nodes of a <u>maximal set of adjacent k-cliques.</u>



<u>Two adjacent k-cliques</u>



<u>k-clique communities</u>

RELATED WORK

- **Palla *et al.*, 2005:** first definition of CPM communities;
- **Kumpula *et al*, 2008:** solution based on k-clique enumeration;
- **Reid *et al*, 2012:** solution based on maximal clique enumeration;
- **Gregori *et al*, 2013:** parallel computation, based on maximal clique enumeration.

ALGORITHMS

METHOD

- Compute a stream of k-cliques in parallel;
- <u>For each k-clique:</u>
  - **Find** communities of each of its (k-1)-clique;
  - **Merge** them;
  - **Add** all new (k-1)-clique.

EXACT ALGORITHM

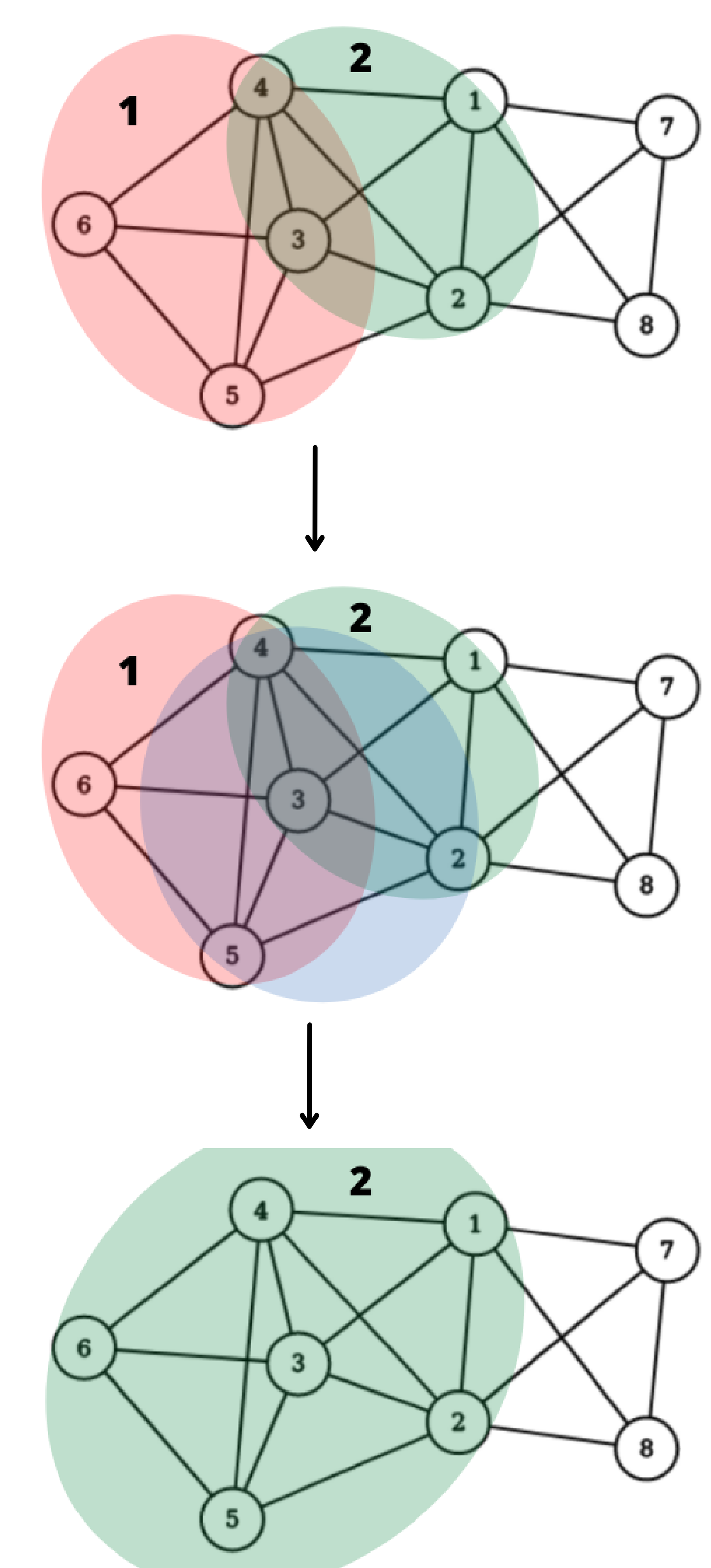
- <u>**Store a dictionnary on (k-1)-cliques**</u>
- <u>On the example:</u>
  - New k-clique of the stream : (2,3,4,5)
    - **Find:**
      - (2,3,4) → 2
      - (2,3,5) → X
      - (2,4,5) → X
      - (3,4,5) → 1
    - **Merge (Union-Find):**
      - 2 ⇆ 1
    - **Add:**
      - (2,3,5) → 2
      - (2,4,5) → 2

COMPLEXITY

Operations per k-cliques : $\mathcal{C}_k \approx \mathcal{O}(k)$

(Find each (k-1)-clique)

EXAMPLE WITH K=4



MEMORY ISSUE OF EXACT ALGORITHM

Massive graphs : the larger k is, the more k-cliques there are.

HOW TO REDUCE MEMORY

- Exact algorithm : store all (k-1)-cliques;
- Approximate algorithm : store all z-cliques, for z < k-1.

APPROXIMATE ALGORITHM

- <u>**Store a dictionnary on z-cliques with z < k-1**</u>
- Difference with exact algorithm = **Find**
- <u>On the example, with z=2 (store edges):</u>
  - New k-clique of the stream : (2,3,4,5)
    - **Find:**
      - (2,3,4): <u>2</u>
        - (2,3) → {<u>2</u>}
        - (2,4) → {<u>2</u>}
        - (3,4) → {1,<u>2</u>}
      - (3,4,5): <u>1</u>
        - (3,4) → {<u>1</u>,2}
        - (3,5) → {<u>1</u>}
        - (4,5) → {<u>1</u>}

COMPLEXITY

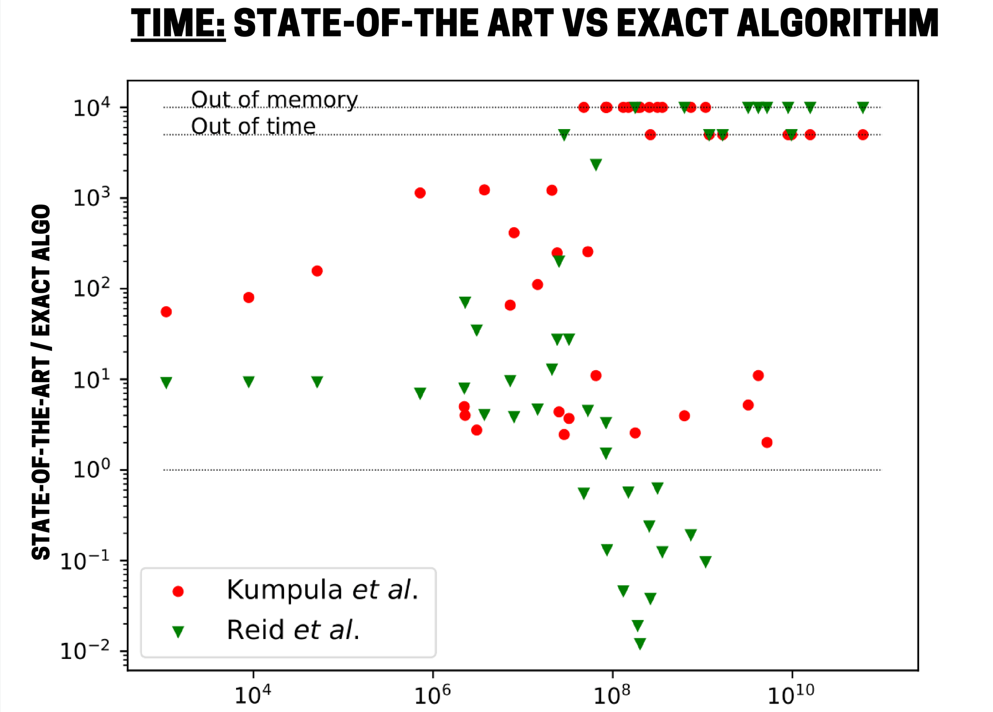
Operations per k-cliques: $\mathcal{C}_k \approx \mathcal{O}\left(k \cdot \binom{k-1}{z}\right)$

(Find each z-clique of each (k-1)-clique)

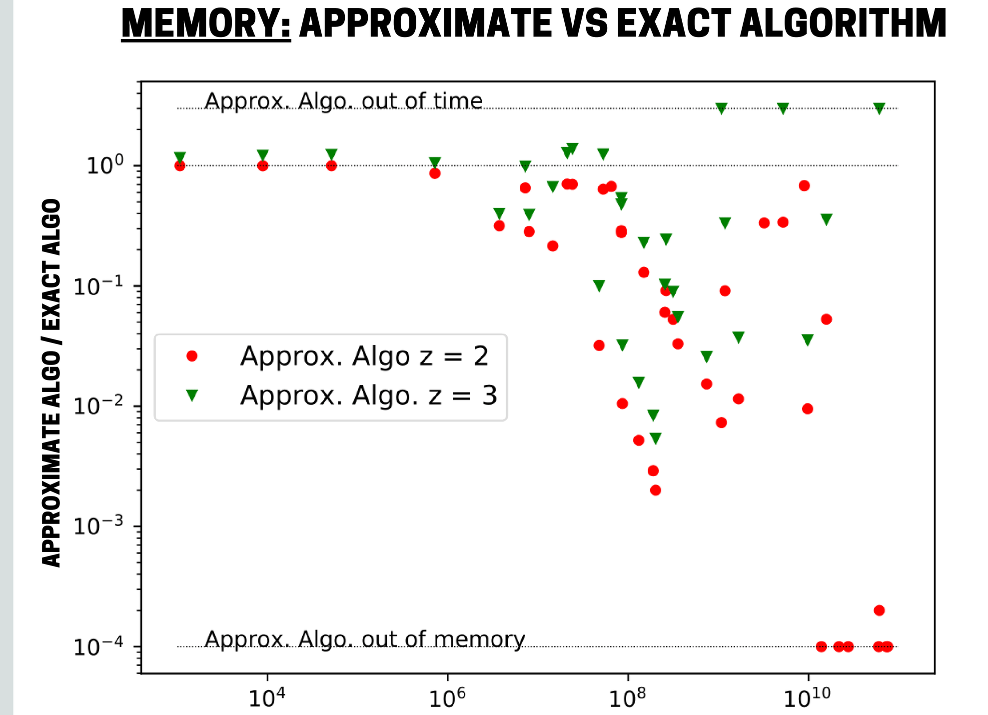
RESULTS

Communities of approximate algorithm are communities of exact algorithm, with some of them merged.

<u>**TIME:**</u> **STATE-OF-THE ART VS EXACT ALGORITHM**



<u>**MEMORY:**</u> **APPROXIMATE VS EXACT ALGORITHM**



GOOD APPROXIMATION

Comparison Approximate VS Exact communites:

<u>**On all experiments:**</u>

- z=2: >93.8% of similarity, with mean=98.6%
- z=3: >99.5% of similarity, with mean=99.95%

Community comparisons with ONMI (MacAid et al. 2013)

CONCLUSION

- **Exact algorithm:** gain of time;
- **Approximate algorithm:** gain of <u>memory</u>.
  - best scale;
  - accurate approximate communities.