When Engagement Meets Similarity: Efficient (k, r)-Core Computation on Social Networks

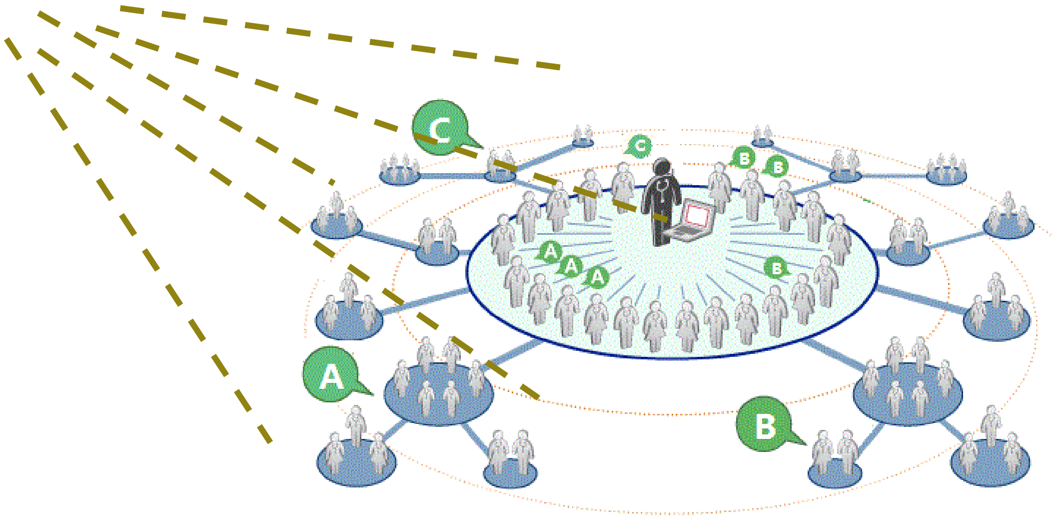
Fan Zhang[0,1], Ying Zhang[1], Lu Qin[1], Wenjie Zhang[2], Xuemin Lin[0,2]

0 *East China Normal University,* 1 *University of Technology Sydney,* 2 *University of New South Wales*

Social Network - Attributed Graph

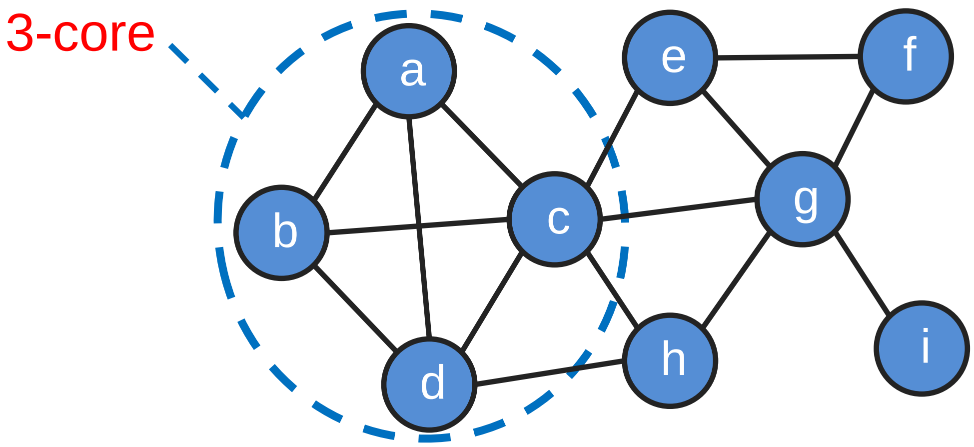
- Data becomes diverse and complex in real-life social networks, which not only consist of users and friendship but also have various attribute values on each user.

Attributes: location, keyword, age, interest, major, ...



k-Core

- Given a graph *G*, the *k*-core of *G* is a maximal subgraph where each node has at least *k* neighbors (i.e., *k* adjacent nodes, or a degree of *k*).

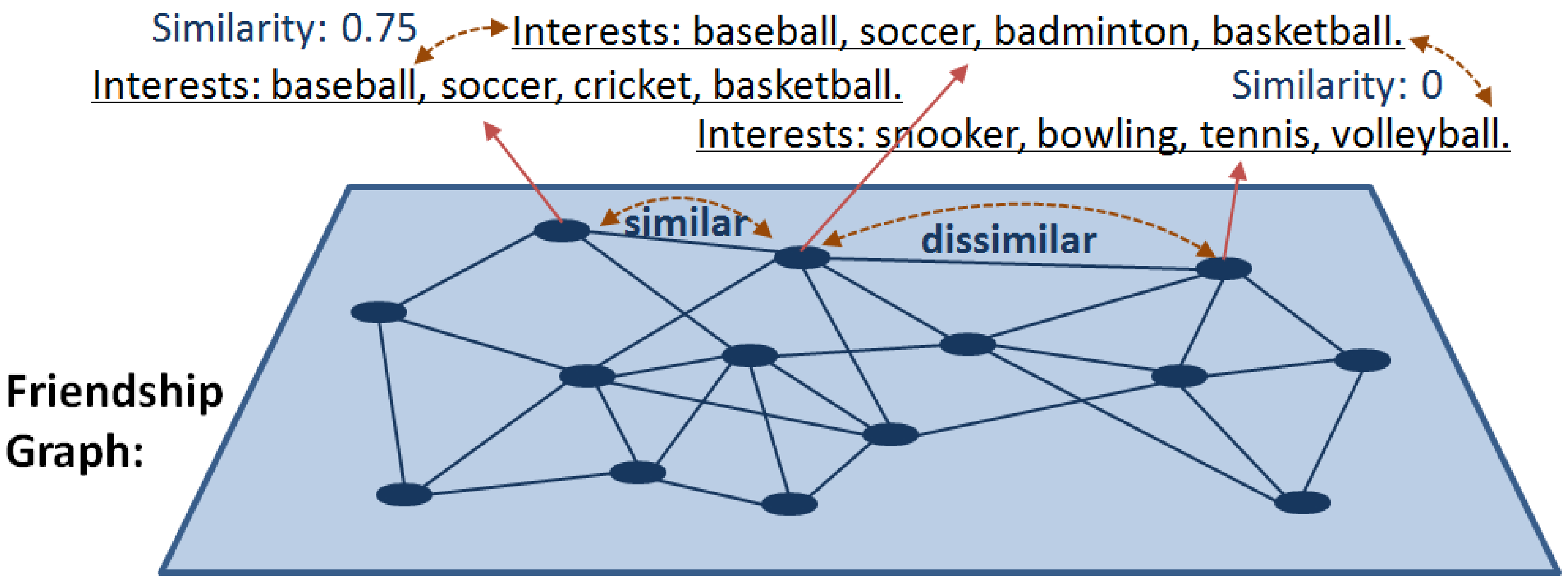


Applications: community detection, social contagion, user engagement, event detection, ……

S. B. Seidman. Network structure and minimum degree. Social networks, 5(3):269–287, 1983.

k-Core on Attributed Graph

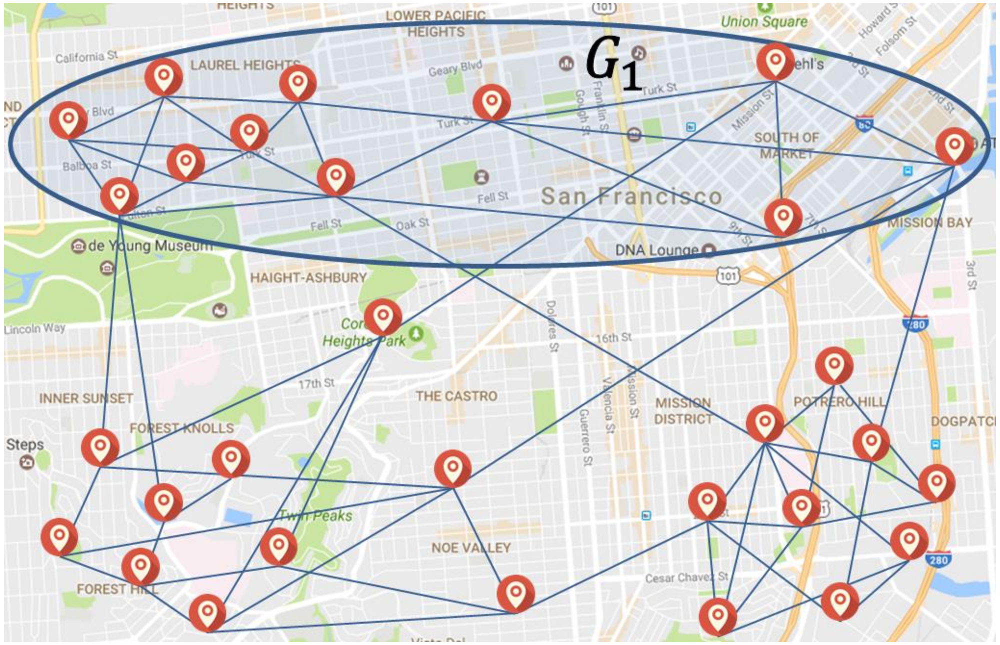
- Does not consider various kinds of attribute information on users.



This network is a 3-core while contains dissimilar nodes.

k-Core on Attributed Graph

- When the similarity of two users is measured by their distance.



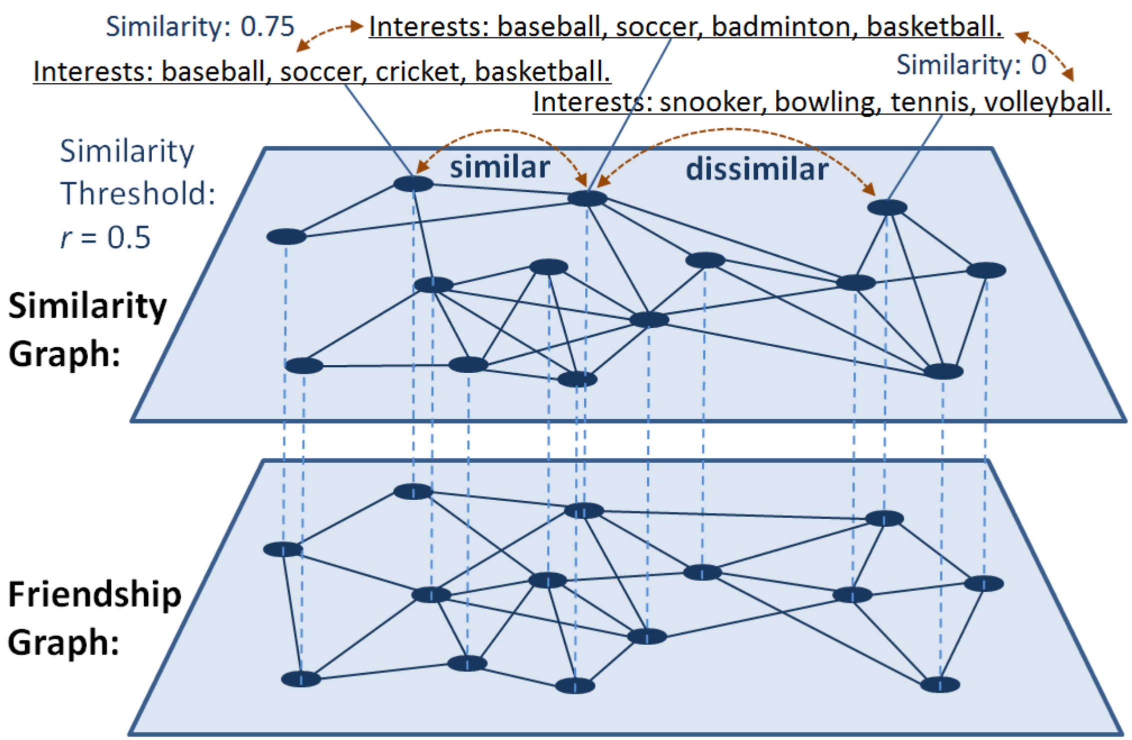
The group G_1 is a connected 3-core while contains users who are far away from others (dissimilar).

Similarity Graph

- The nodes in similarity graph and friendship graph are same.
- In similarity graph, there is an edge between two nodes if and only if they are similar.

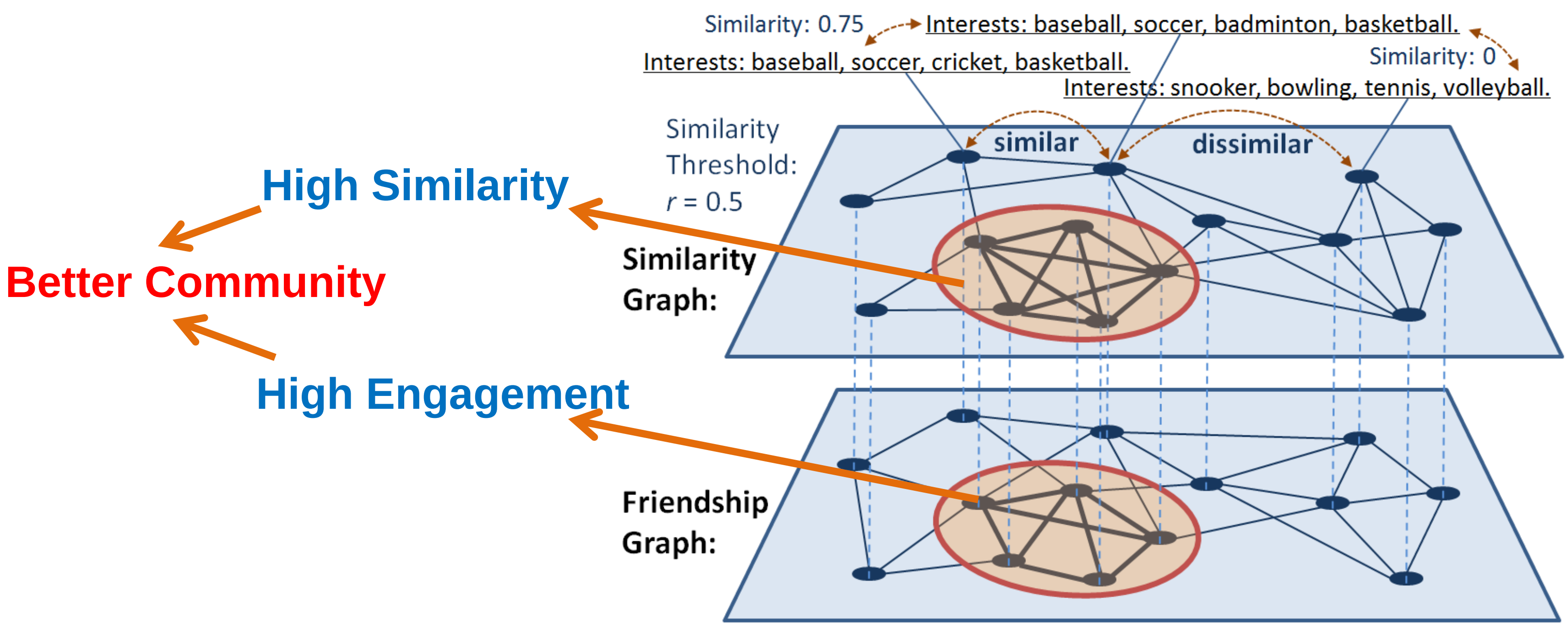
User Similarity ←

User Engagement ←



(*k*,*r*)-Core on Attributed Graph

- **(*k*,*r*)-Core:** a subgraph where each node has at least *k* neighbors and is similar to every other node in the subgraph.



The (k,r)-Core Problems

Problem Statement.

Given an attributed graph G, an integer k and a similarity threshold r, we aim to develop efficient algorithms for the following two fundamental problems:

(i) enumerate all maximal (k,r)-cores in G;

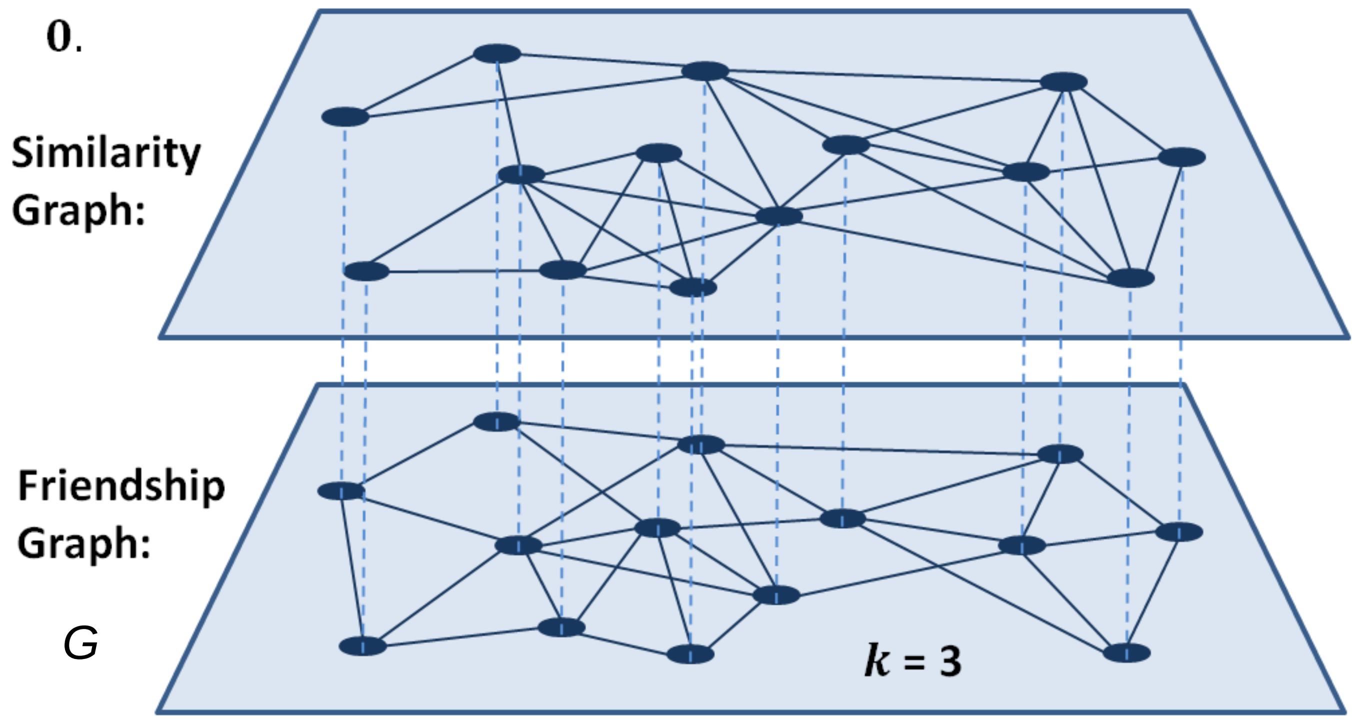
(ii) find the maximum (k,r)-core in G.

Challenge.

Both problems are NP-hard.

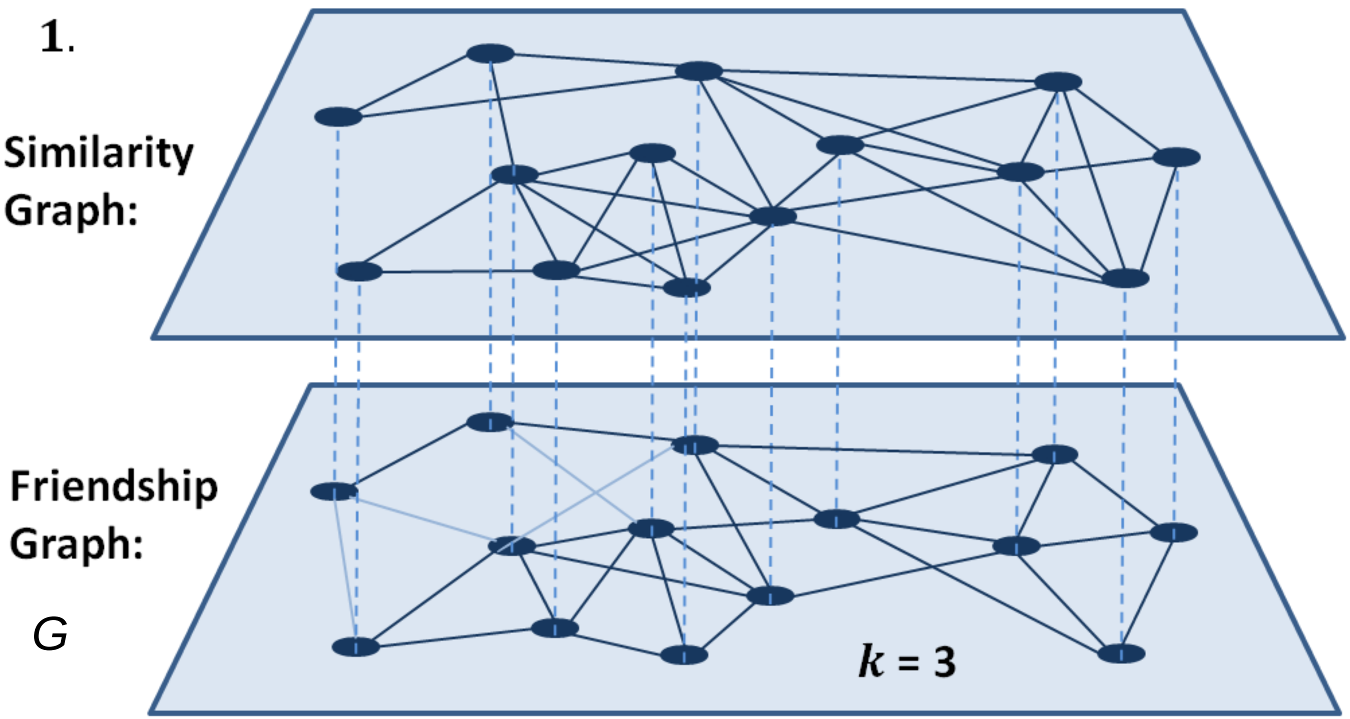
The Clique-based Approach

1. Delete every edge in *G* if its two endpoints are dissimilar.
2. Compute *k*-core (*S*) on *G*.
3. Enumerate maximal cliques in the similarity graph of *S*.
4. Compute *k*-core on the induced subgraph in *S* for each maximal clique.



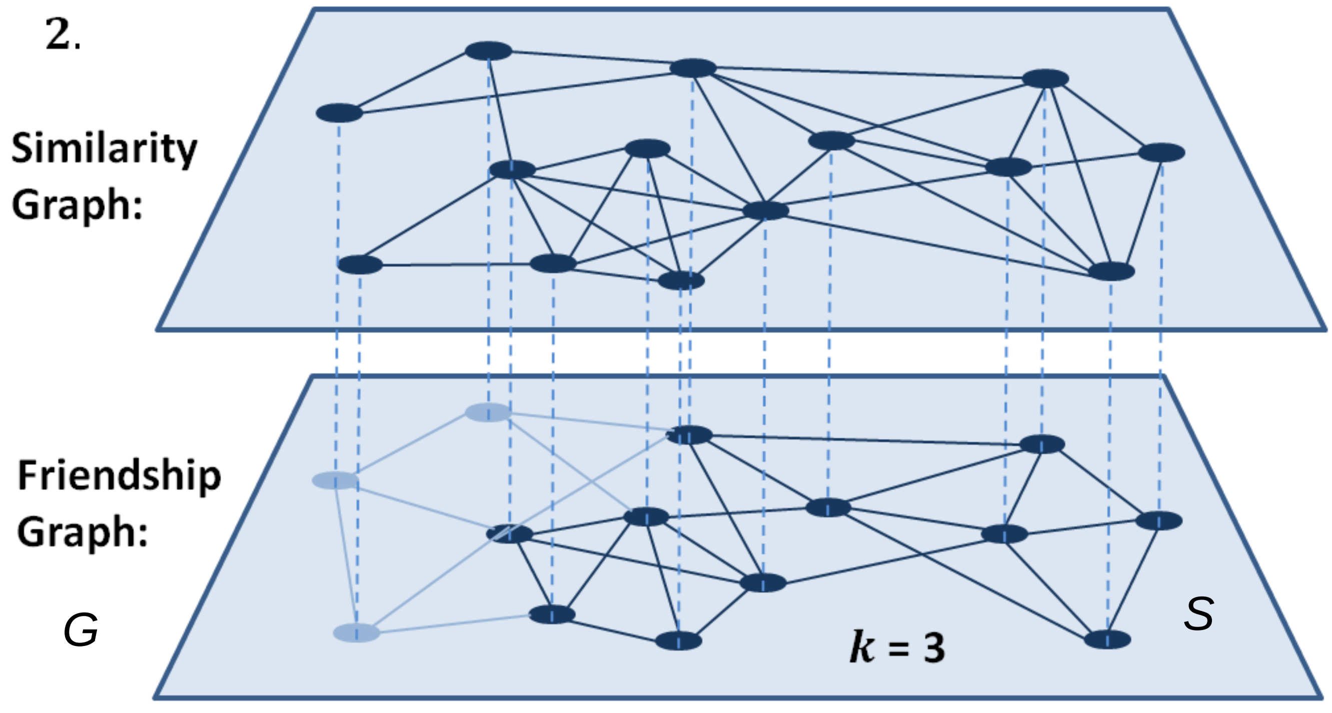
The Clique-based Approach

1. **Delete every edge in *G* if its two endpoints are dissimilar.**
2. Compute *k*-core (*S*) on *G*.
3. Enumerate maximal cliques in the similarity graph of *S*.
4. Compute *k*-core on the induced subgraph in *S* for each maximal clique.



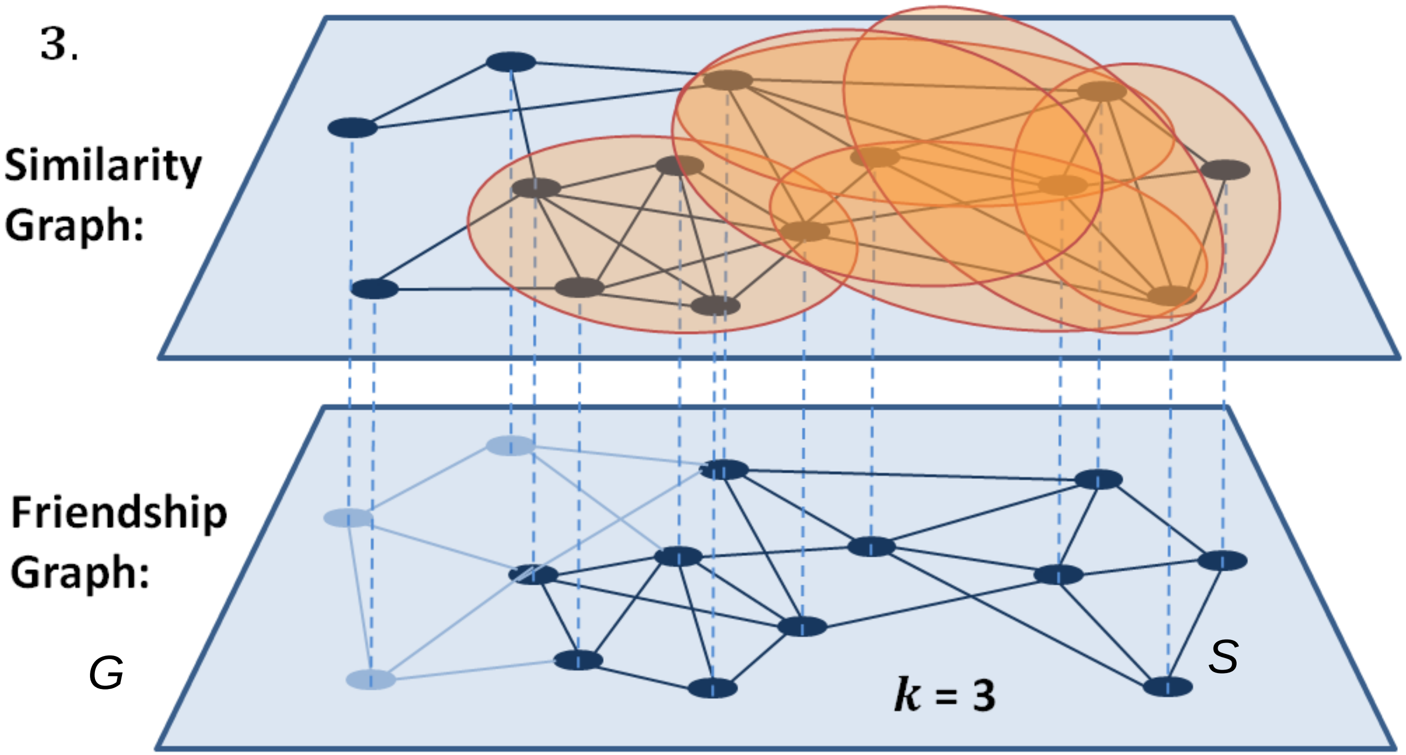
The Clique-based Approach

1. Delete every edge in *G* if its two endpoints are dissimilar.
2. **Compute *k*-core (*S*) on *G*.**
3. Enumerate maximal cliques in the similarity graph of *S*.
4. Compute *k*-core on the induced subgraph in *S* for each maximal clique.



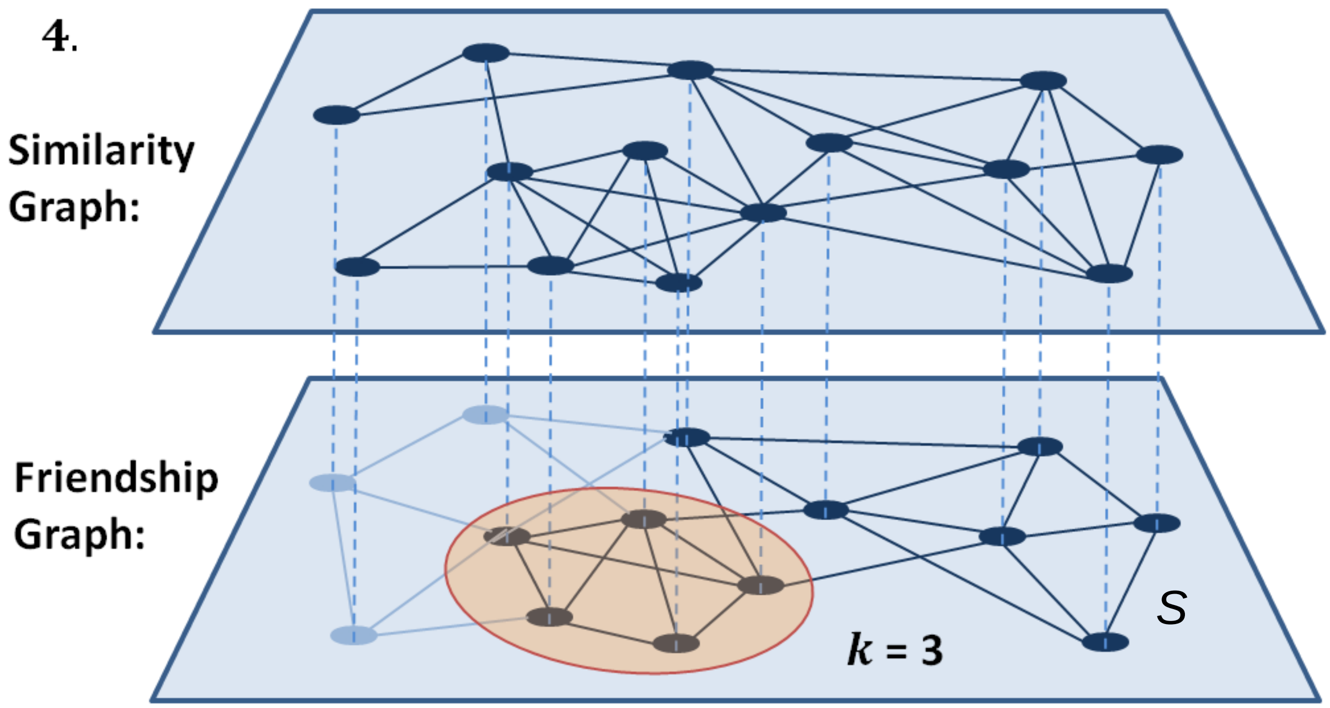
The Clique-based Approach

1. Delete every edge in *G* if its two endpoints are dissimilar.
2. Compute *k*-core (*S*) on *G*.
3. **Enumerate maximal cliques in the similarity graph of *S*.**
4. Compute *k*-core on the induced subgraph in *S* for each maximal clique.



The Clique-based Approach

1. Delete every edge in *G* if its two endpoints are dissimilar.
2. Compute *k*-core (*S*) on *G*.
3. Enumerate maximal cliques in the similarity graph of *S*.
4. **Compute *k*-core on the induced subgraph in *S* for each maximal clique.**



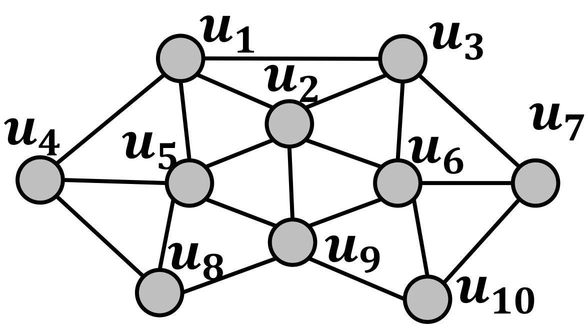
The Clique-based Approach

1. Delete every edge in G if its two endpoints are dissimilar.
2. Compute *k*-core (S) on G.
3. Enumerate maximal cliques in the similarity graph of S.
4. Compute *k*-core on the induced subgraph in S for each maximal clique.

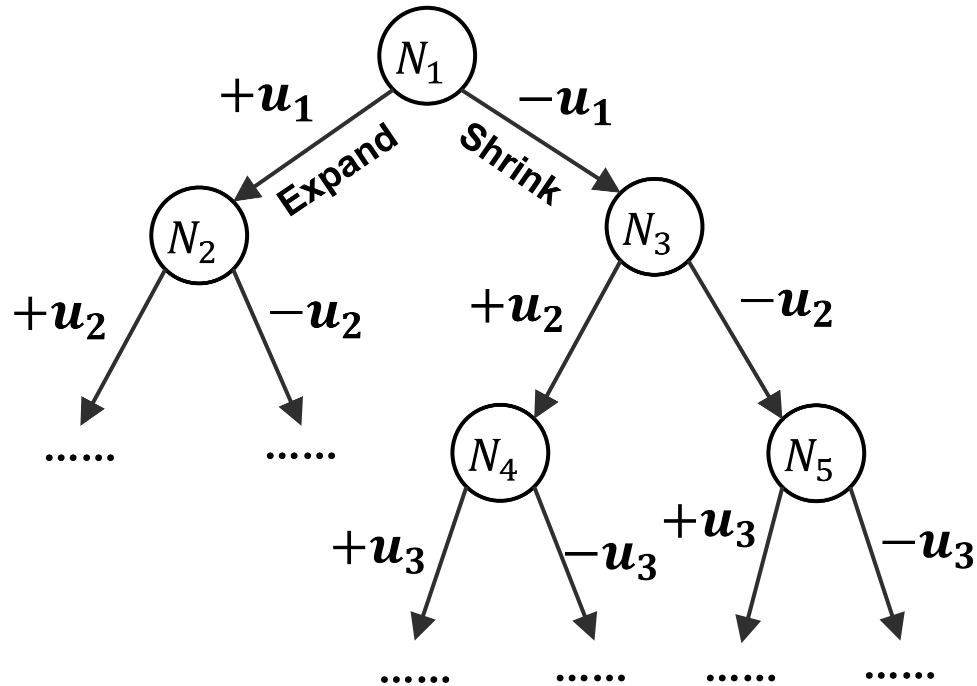
Time-consuming for two reasons:

1. Still too many maximal cliques.
2. Isolated processing of k-core and clique computations.

Enumerate Maximal (*k*,*r*)-Cores



A graph



Search tree

$\boldsymbol{M}$: Must in (*k*,*r*)-core

$\boldsymbol{C}$: Candidate node

$\boldsymbol{E}$: Excluded node

N_1 : $\boldsymbol{M} = \emptyset, \boldsymbol{C} = \{\boldsymbol{u_1}, \boldsymbol{u_2}, \ldots \boldsymbol{u_9}, \boldsymbol{u_{10}}\}, \boldsymbol{E} = \emptyset$

N_4 : $\boldsymbol{M} = \{\boldsymbol{u_2}\}, \boldsymbol{C} = \{\boldsymbol{u_3}, \ldots \boldsymbol{u_9}, \boldsymbol{u_{10}}\}, \boldsymbol{E} = \{\boldsymbol{u_1}\}$

Enumerate Maximal (k,r)-Cores

Pruning Rules.

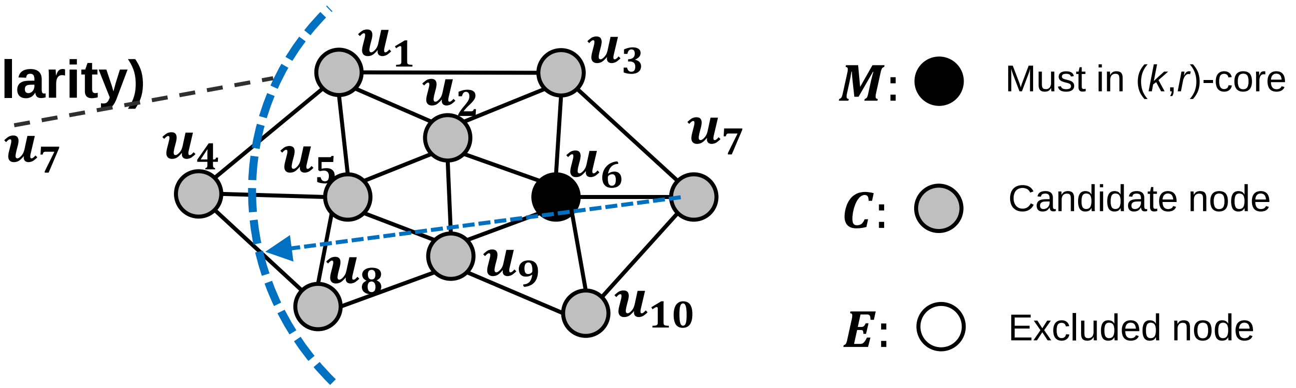
(1) Eliminate Candidates

Structural based pruning.
We can discard a node u in C if $deg(u, M \cup C) < k$.

Similarity based pruning.
We can discard a node u in C if $sim(u, v) < r$ for any v in M.

distance (similarity) constraint for u_7

$k = 3$



Enumerate Maximal (*k*,*r*)-Cores

Pruning Rules.

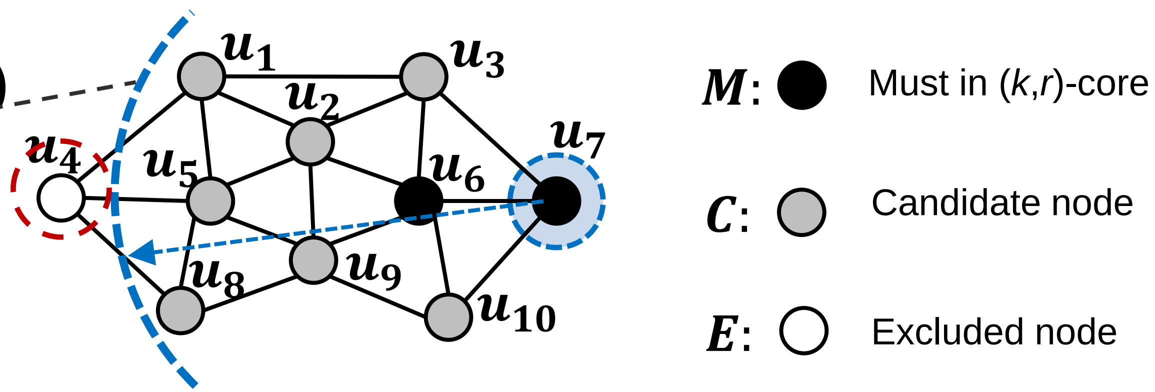
(1) Eliminate Candidates

Structural based pruning.
We can discard a node *u* in *C* if *deg(u, M* ∪ *C)* < *k*.

Similarity based pruning.
We can discard a node *u* in *C* if *sim(u, v)* < *r* for any *v* in *M*.

distance (similarity) constraint for u_7

***k* = 3**



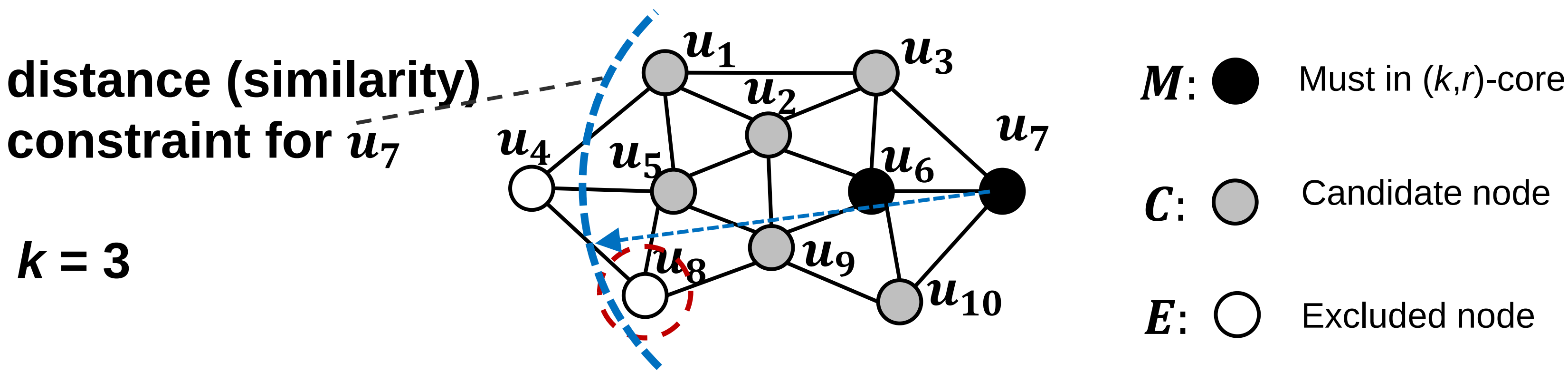
Enumerate Maximal (*k*,*r*)-Cores

Pruning Rules.

(1) Eliminate Candidates

Structural based pruning.
We can discard a node *u* in *C* if *deg(u, M ∪ C) < k*.

Similarity based pruning.
We can discard a node *u* in *C* if *sim(u, v) < r* for any *v* in *M*.



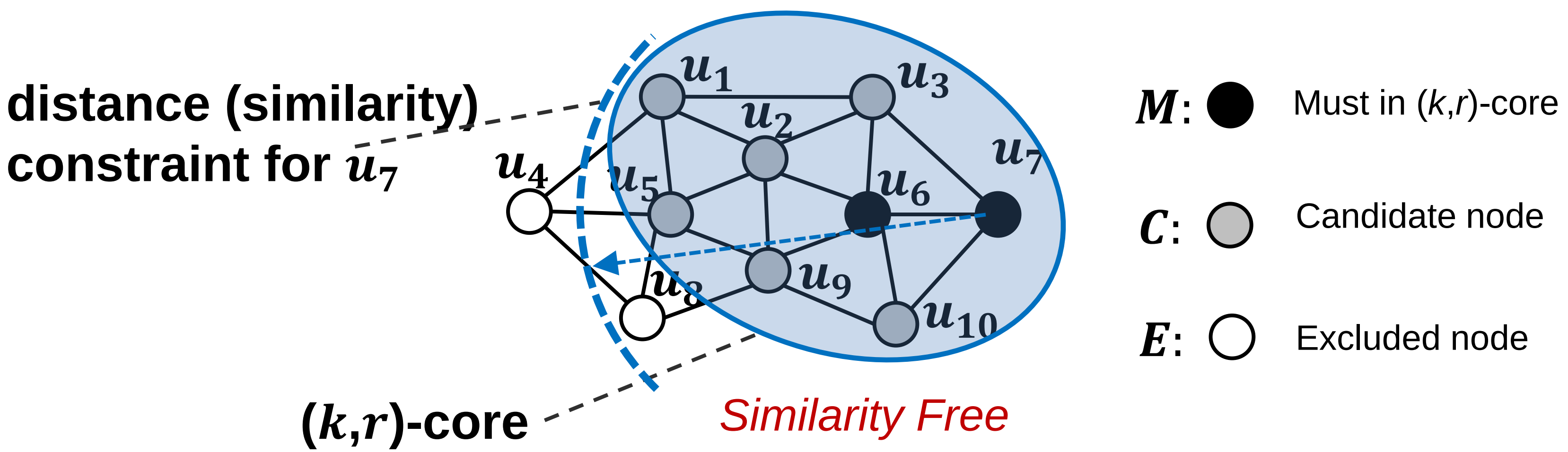
Enumerate Maximal (k,r)-Cores

Pruning Rules.

(2) Candidate Retaining

A node u is *similarity free* w.r.t C if u is similar to all nodes in C.

$M \cup C$ is a (k,r)-core if we have every node in C is similarity free *w.r.t. C.*



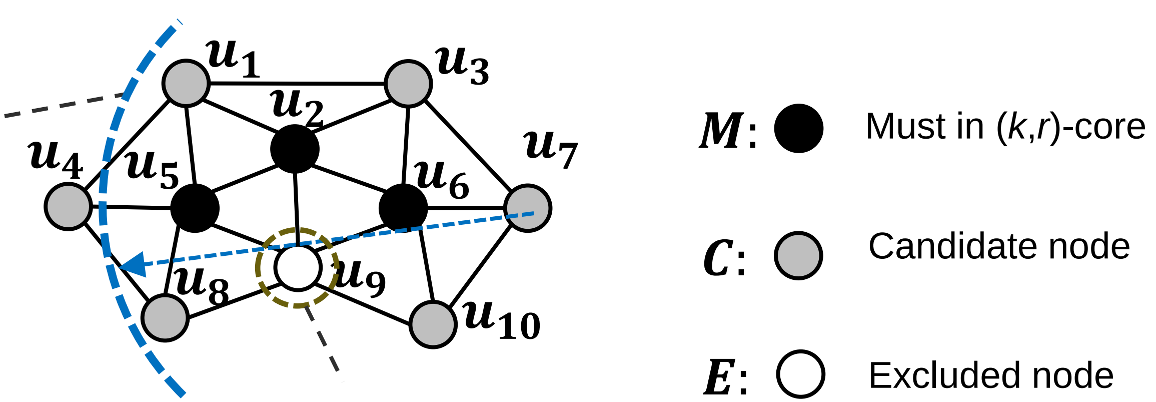
Enumerate Maximal (*k*,*r*)-Cores

Pruning Rules.

(3) Early Termination

Terminate the current search if there is a node $u \in E$ with $deg(u,M) \geq k$ and similarity free w.r.t. $M \cup C$;

distance (similarity) constraint for u_7



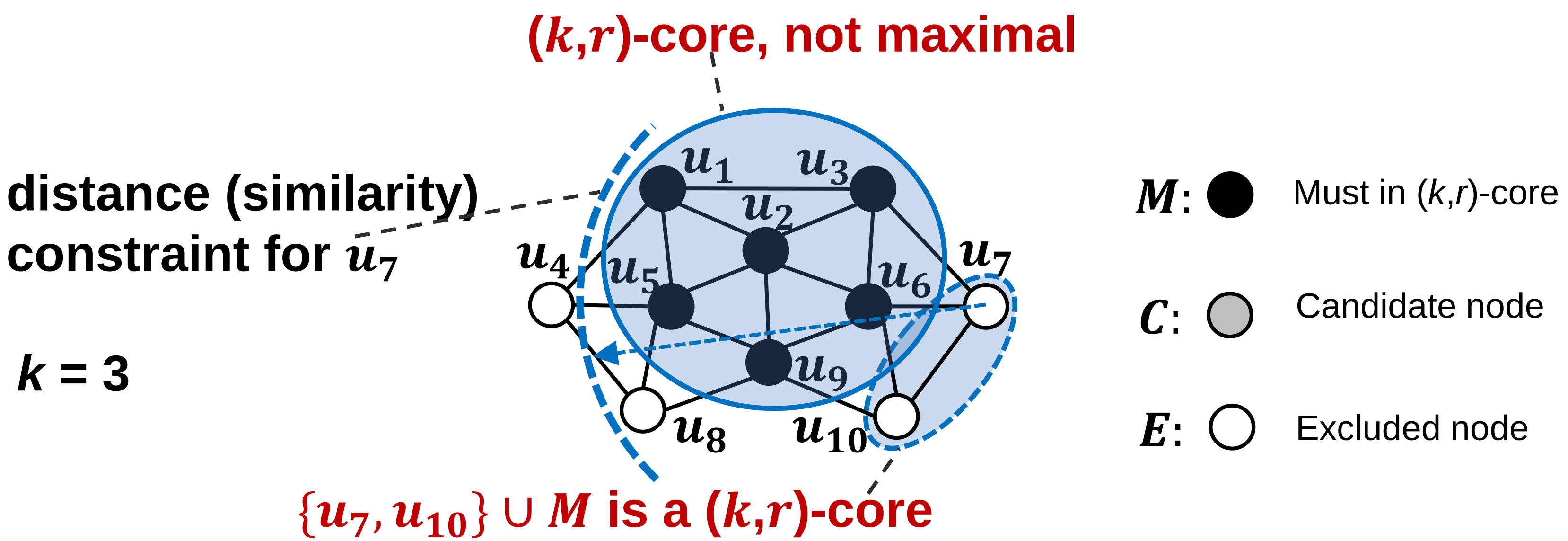
u_9 is similarity free and
***deg*(u_9, M) $\geq$ 3**

Enumerate Maximal (*k*,*r*)-Cores

Pruning Rules.

(4) Maximal Check

Given a (*k*,*r*)-core *R*, we claim that *R* is a maximal (*k*,*r*)-core if there doesn't exist a non-empty set $U \subseteq E$ such that $R \cup U$ is a (*k*,*r*)-core.

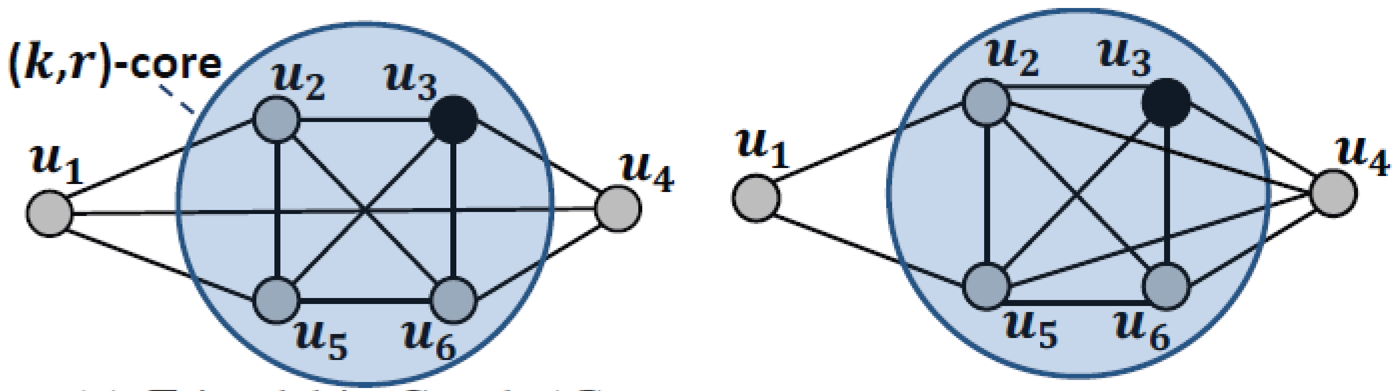


Finding the Maximum (*k*,*r*)-Core

Colour based Size Upper Bound of *(k,r)-core*   *s: (k,r)*-core size

Let c_{min} denote the minimum number of colors to *color* the nodes in the similarity graph *J'* such that every two adjacent nodes in *J'* have different colors.
Since a *k*-clique needs *k* number of colors to be *colored*, we have $s \le c_{min}$.

***k* = 3**



(a) Friendship Graph (J)   (b) Similarity Graph (J')

We need at least 5 colors to color J', so the color based upper bound is 5.

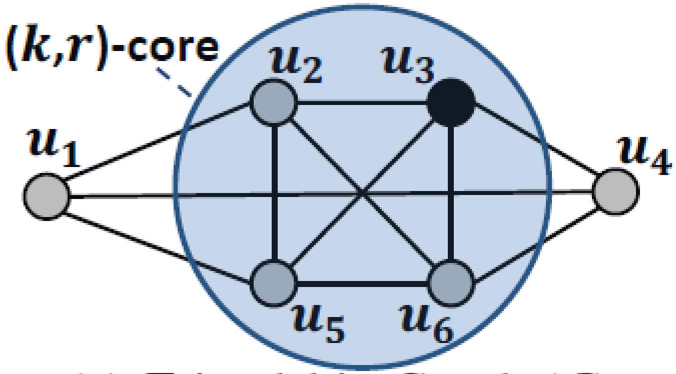
Finding the Maximum (*k*,*r*)-Core

***k*-core based Size Upper Bound** of *(k,r)-core*

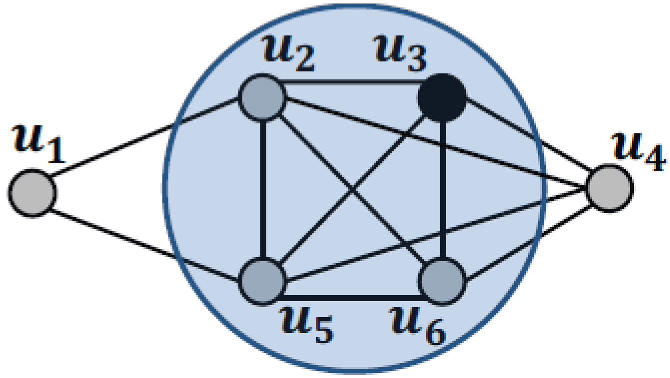
s: (k,r)-core size

Let k_{max} denote the maximum *k* value such that *k*-core of *J'* is not empty.
Since a *k*-clique is also a (*k*-1)-core, this implies that we have $s \le k_{max} + 1$

***k* = 3**



(a) Friendship Graph (J)



(b) Similarity Graph (J')

By core decomposition on similarity graph J', we get that the k-core based upper bound is 5 since k_{max} = 4 with 4-core {u_2, u_3, u_4, u_5, u_6}.

Finding the Maximum (*k*,*r*)-Core

***(k,k')*-core based Size Upper Bound** of *(k,r)-core* *s: (k,r)*-core size

By core decomposition on similarity graph J', we get that the k-core based upper bound is 5 *since* k_{max} = 4 *with 4-core* {u_2, u_3, u_4, u_5, u_6}.

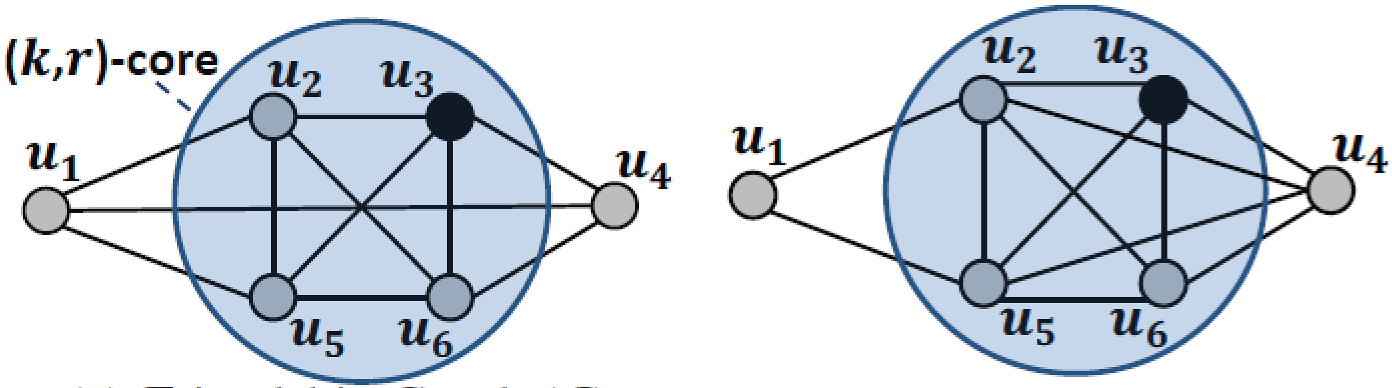
However, the induced subgraph of {u_2, u_3, u_4, u_5, u_6} *on friendship graph J is NOT a 3-core (degree of* u_4 < 3*).*

delete u_4

$k_{max} = k_{max} - 1 = 3$ *with 3-core* {u_2, u_3, u_5, u_6}*. The nodes also form a 3-core on J.*

(k,k')-core based Size Upper Bound is **4**

***k* = 3**



(a) Friendship Graph (*J*) (b) Similarity Graph (*J'*)

Search Orders

(1) Node visiting order: the order of which node is chosen from candidate set C.

(2) Branch visiting order: the order of which search branch (expand or shrink branch) goes first.

Measurements for a chosen node is extended to *M* or discarded:

- $\mathbf{\Delta_1}$: the change of the number of dissimilar pairs, where

$$\Delta_1 = \frac{DP(\mathbf{C}) - DP(\mathbf{C}')}{DP(\mathbf{C})}$$

M' and *C'* denote the updated *M* and *C*

- $\mathbf{\Delta_2}$: the change of the number of edges, where

$$\Delta_2 = \frac{|E(\mathbf{M} \cup \mathbf{C})| - |E(\mathbf{M}' \cup \mathbf{C}')|}{|E(\mathbf{M} \cup \mathbf{C})|}$$

Search Orders

(1) Find the maximum *(k,r)-core*

a *cautious greedy strategy:* $\boldsymbol{\lambda\Delta_1 - \Delta_2}$**.** where λ is to make a trade-off.

In this way, each candidate has two scores (for expand or shrink). Then the vertex with the highest score will be chosen and its branch with higher score will be explored first.

(2) Enumerate all maximal *(k,r)-cores*

we adopt the Δ_1**-then-**Δ_2 strategy; that is, we prefer the larger Δ_1, and the smaller Δ_2 is considered if there is a tie.

(3) Maximal Check

we adopt a *short-sighted greedy heuristic*. In particular, we choose the vertex with the **largest degree** and the expand branch is always preferred.

- $\boldsymbol{\Delta_1}$: the change of the number of dissimilar pairs, where

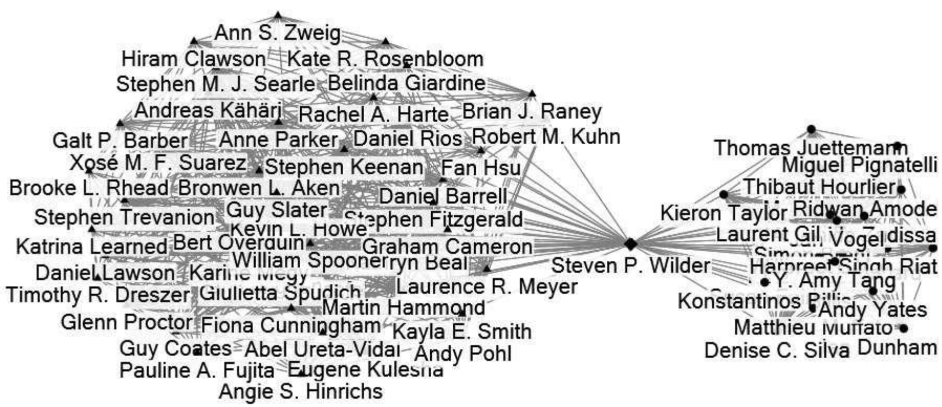
$$\Delta_1 = \frac{DP(\mathbf{C}) - DP(\mathbf{C'})}{DP(\mathbf{C})}$$

- $\boldsymbol{\Delta_2}$: the change of the number of edges, where

$$\Delta_2 = \frac{|E(\mathbf{M} \cup \mathbf{C})| - |E(\mathbf{M'} \cup \mathbf{C'})|}{|E(\mathbf{M} \cup \mathbf{C})|}$$

Case Study on DBLP

DBLP is a computer science bibliography website.



1,566,919 nodes,
6,461,300 edges.

k=15, *r*=3‰

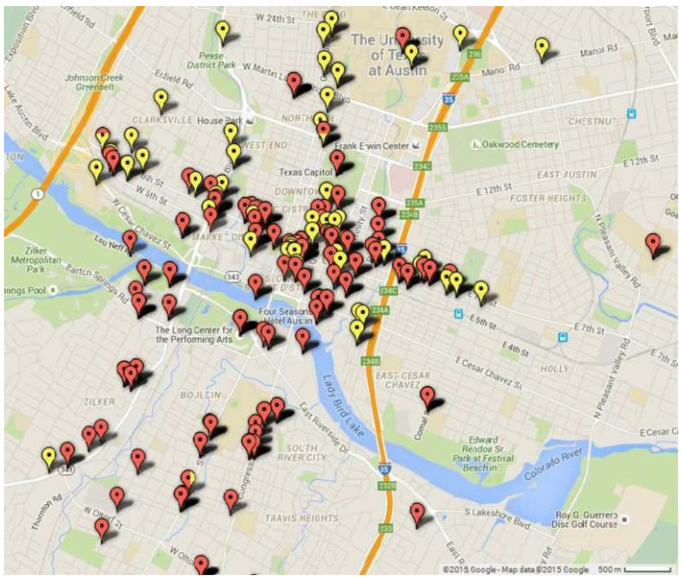
Each node is an author.

Each edge represents there are at least 3 co-authored papers for two authors.

For *r*, we used the thousandth of the pairwise similarity distribution in decreasing order which grows from top 1‰ to top 15‰ (i.e., the similarity threshold value drops).

Case Study on Gowalla

Gowalla is a location-based social network launched in 2007.

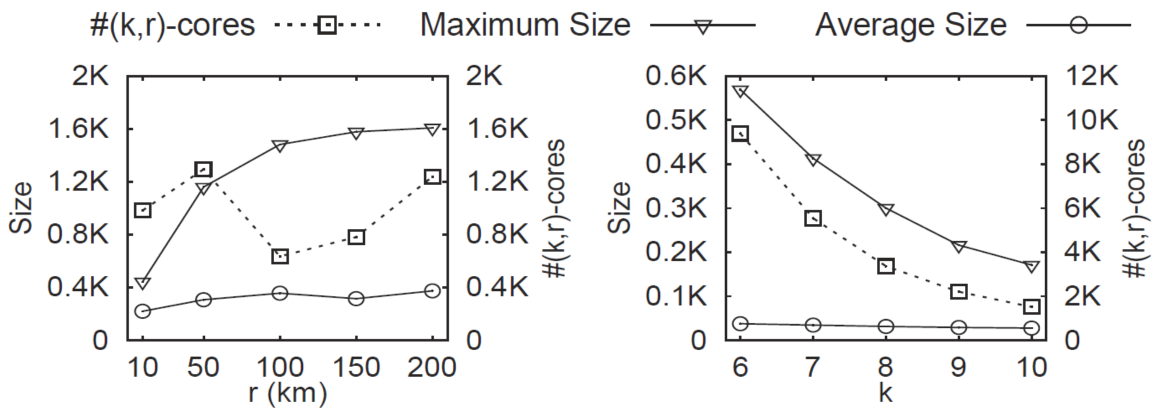


Two maximal (k,r)-cores
When k=10, r=10 km

196,591 nodes,
456,830 edges.

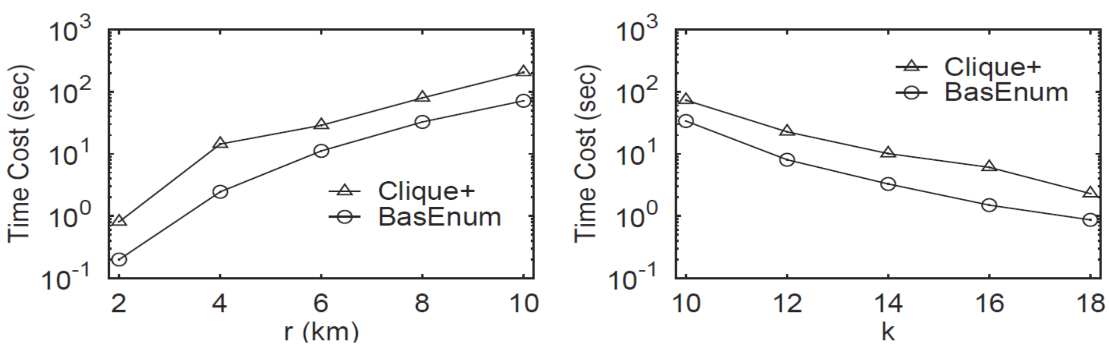
Maximal (k,r)-cores when k=20 and r=3 km

(*k*,*r*)-Core Statistics



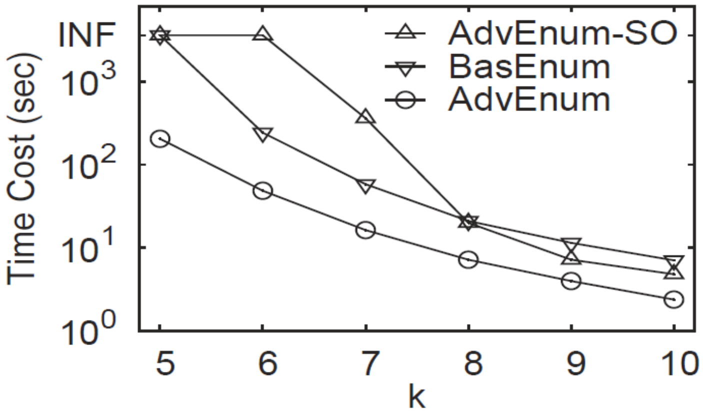
(a) Gowalla, k=5 (b) DBLP, r=top 3‰

Efficiency of Baseline

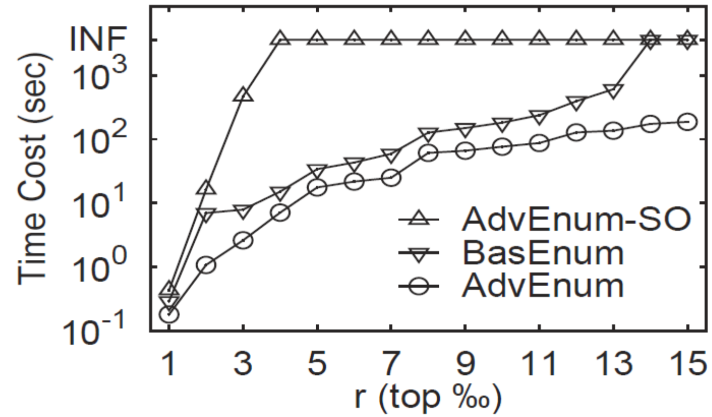


(a) Gowalla, k=5 (b) DBLP, r=top 3‰

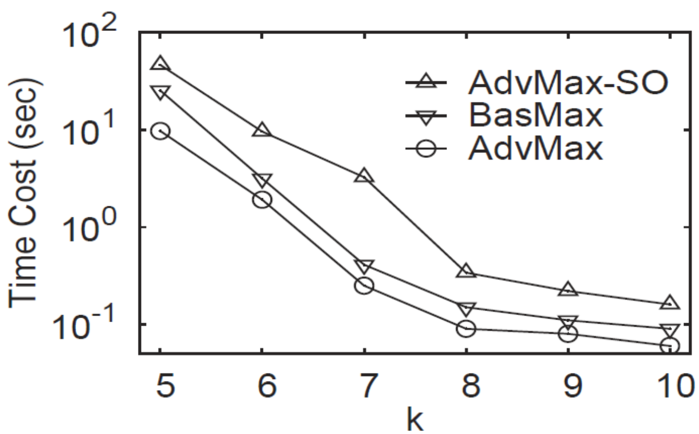
Efficiency



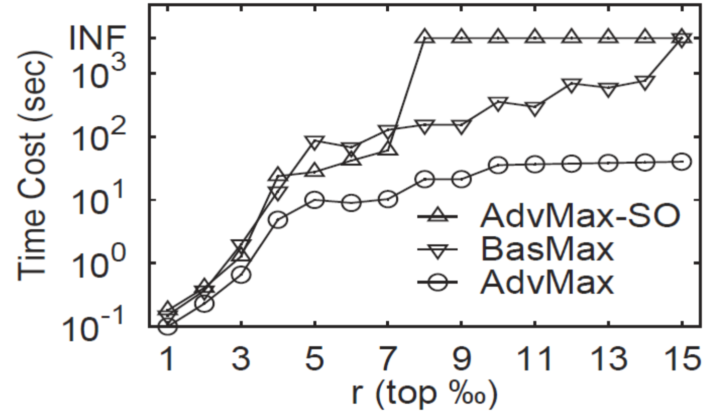
(a) Gowalla, r=100



(b) DBLP, k=15



(a) Gowalla, r=100



(b) DBLP, k=15

THANK YOU

Q&A