Scalable Interactive Dynamic Graph Clustering on Multicore CPUs

SON et. al., IEEE Trans.KDE 2018

Wissem Inoubli

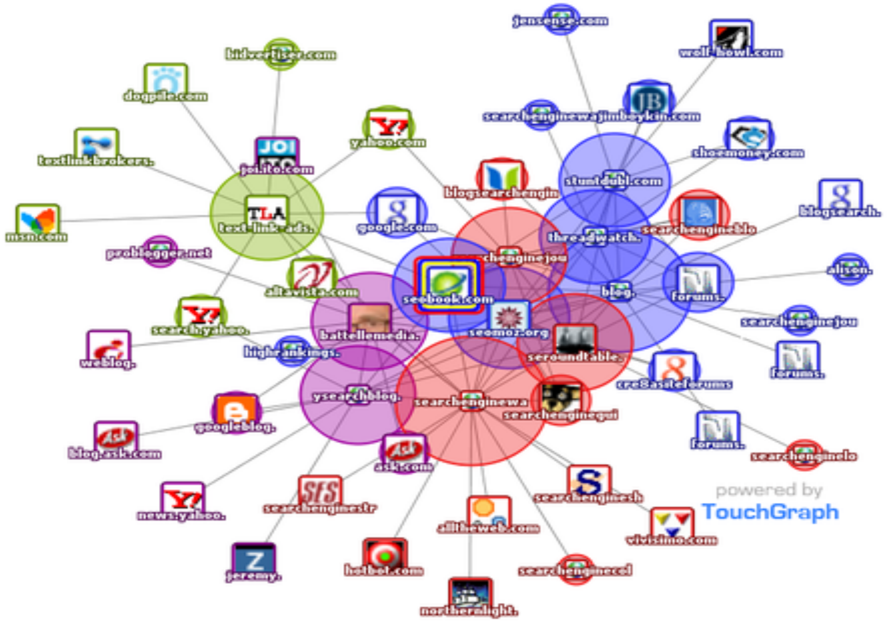
FST, LIPAH

Context

Graphs

Graphs are ubiquitous and can model complex relationships

Social network



Web

Context

Graph clustering

- Group vertices into clusters;
  - e.g: detect the strong connected sub-graphs;
- Obtimization problem:
  - **Maximze** the distance between **sub-graphs**;
  - **Minimize** the distance between the vertices in the same **sub-graphe.**

Applications

- **Digital marketing**: differ intelligenlty the messages or offers;
- **Bio-informatic**: identify the target proteins or the functions of the protein groups;
- **Social network**: community detection, user profiling;
- **Scientific Research** : identify the scientific community .

challenges

- **High volume of the graph**: Graph storage;
- **Scalability**: Graph processing;
- **Velocity**: Fast updating of graph.

1.49 trillion user/ month

500 Million Tweets/ dy
320 Million user/month

AnyTimeSCAN

Scalable Interactive Dynamic Graph Clustering on Multicore CPUs

SON et. al., IEEE Trans. KDE 2018

Contributions

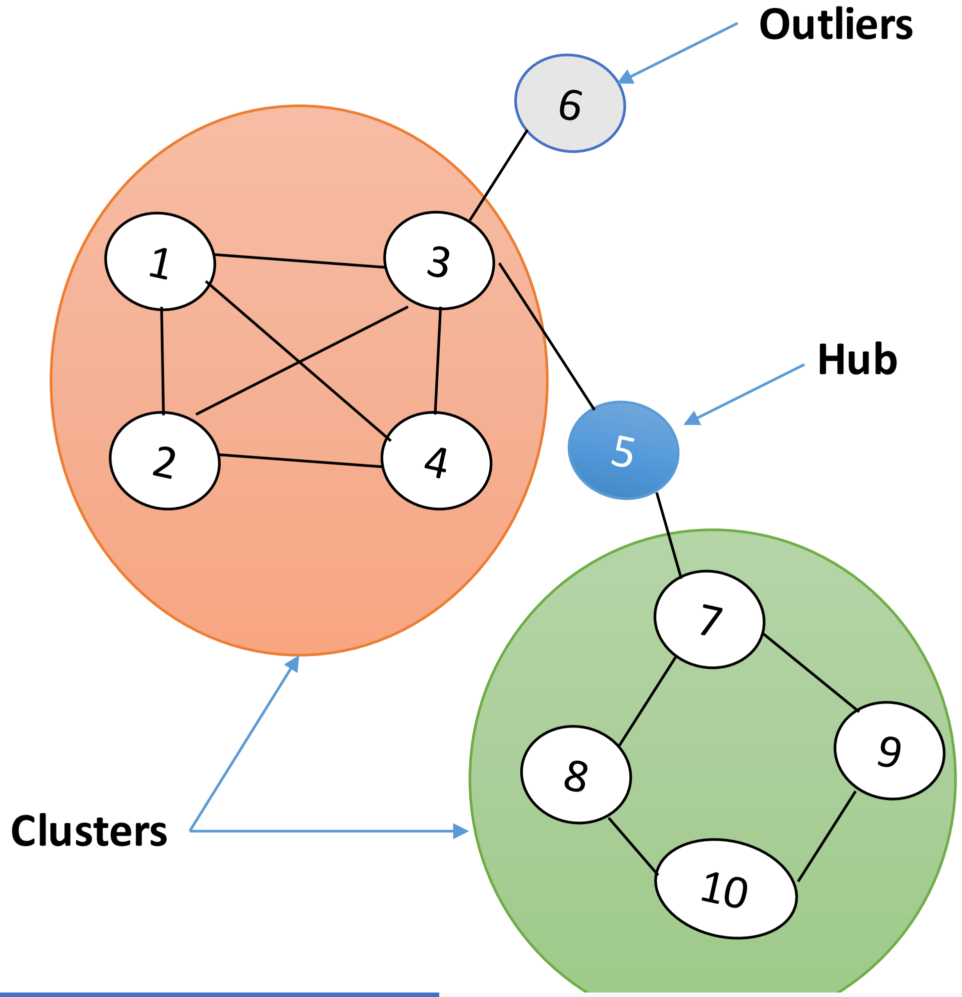
1. Structural clustering: SCAN;
2. Interactive clustering in real time: intermediate results;
3. Parallel processing on multi-core and shared memory;
4. Dynamic graph clustering.

SCAN

Scan: a structural clustering algorithm for networks

Xu et. al.,KDD'07.

- Based on common neighbors between the vertices;
- Uses the structural similarity;
- Identifies Clusters, Outliers and Hubs;



SCAN

The main steps of the basis SCAN algorithm

1. Define the neighbors of each vertex;
2. Calculate the *structural similarity* in the graph **G;**
3. Define the **core** vertices**;**
4. Build the **clusters;**
5. Define the **Hub** and **outliers** vertices**.**

SCAN

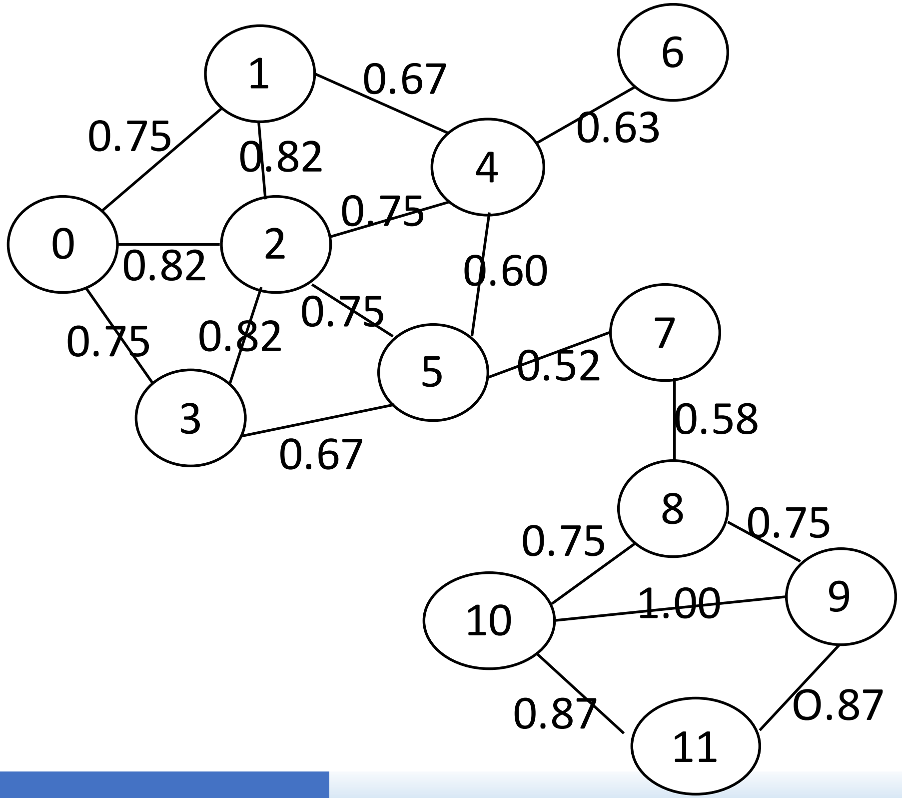
1. Define the neighbors of each vertex;

List of neighbors for each vertex

$$\Gamma(v) = \{w \in V \mid (v,w) \in E\} \cup \{v\}$$

2. Structural similarity of the graph **G**

$$\sigma(v,w) = \frac{|\Gamma(v) \cap \Gamma(w)|}{\sqrt{|\Gamma(v)||\Gamma(w)|}}$$



SCAN

3. Cores detection

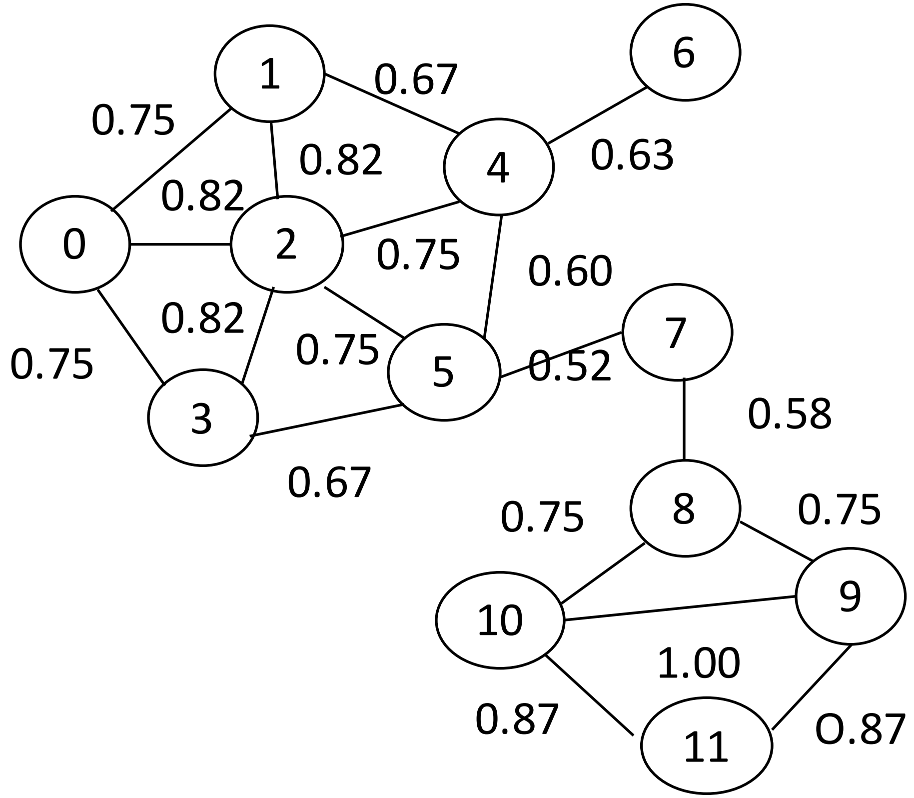
ε =0.7
μ=3

Uses a threshold **ε** to determine a dense connections:

$$N_\varepsilon(v) = \{w \in \Gamma(v) \mid \sigma(v,w) \geq \varepsilon\}$$

A **core** vertex shares structural similiarity of at least **ε** with at least **μ** neighbors:

$$CORE_{\varepsilon,\mu}(v) \Leftrightarrow |N_\varepsilon(v)| \geq \mu$$



SCAN

$\varepsilon = 0.7$
$\mu = 3$

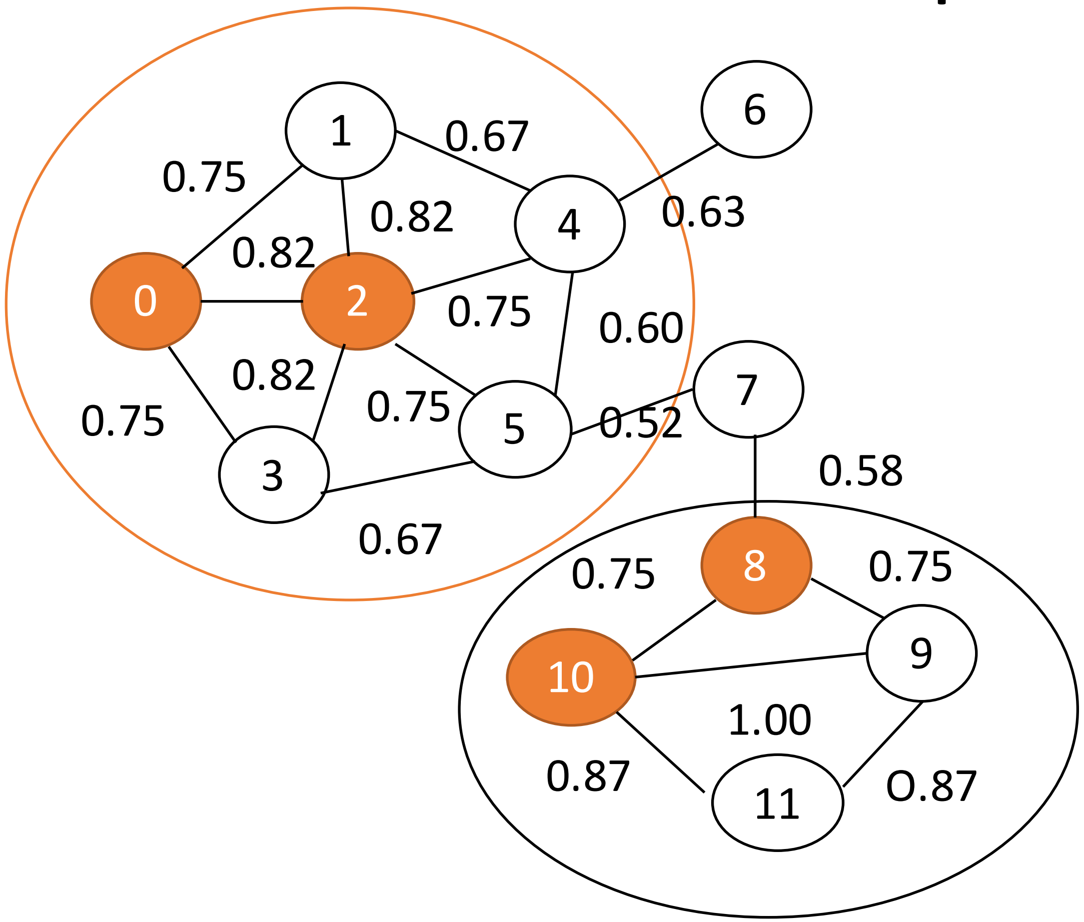
4. Build the **Clusters**

Direct structure reachability:

IF vertex is in **ε–Neighborhood** of a **core** vertex,

they should be **IN** the **same cluster**

$$DirREACH_{\varepsilon,\mu}(v,w) \Leftrightarrow CORE_{\varepsilon,\mu}(v) \wedge w \in N_{\varepsilon}(v)$$



SCAN

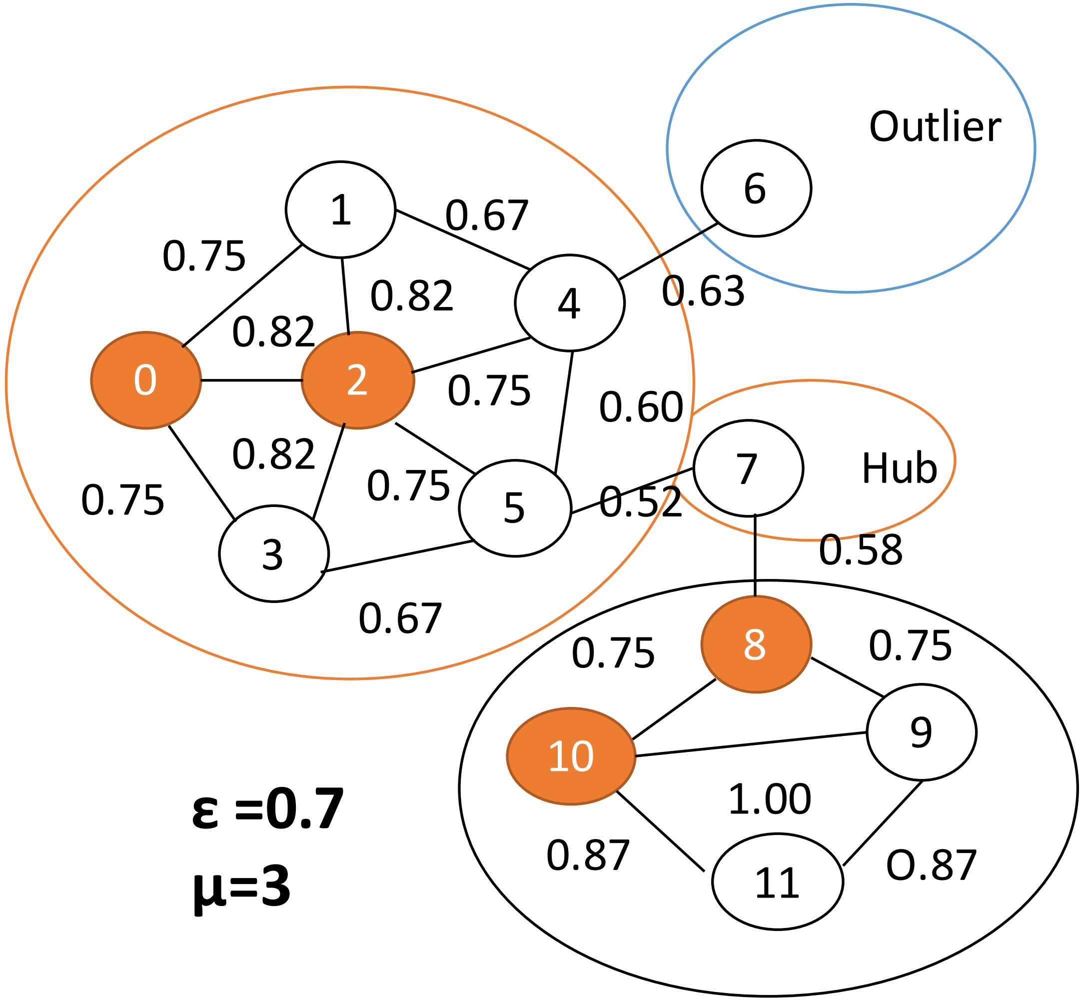
5. Define the **Hub** and **outliers** vertices

Hub:

Is an ***isolated vertex*** that's neighbors ***belonging*** of two or more different clusters.

Outlier:

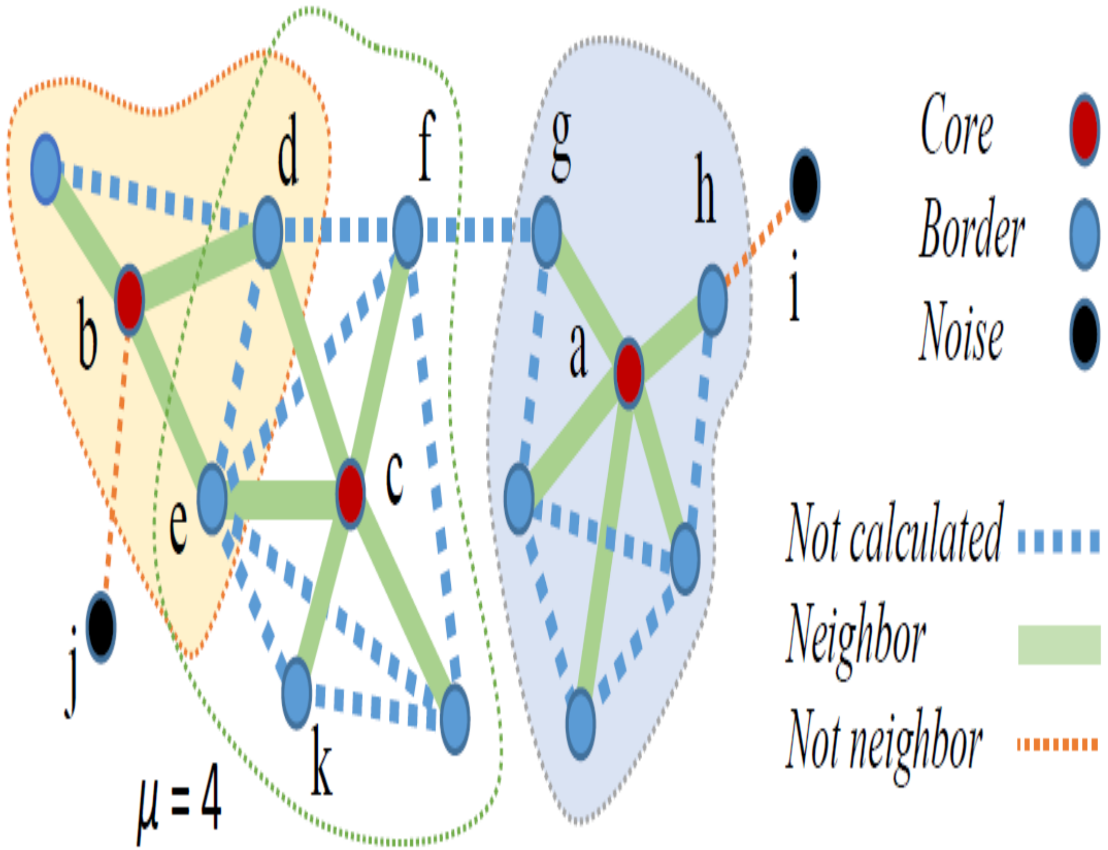
Is an ***isolated vertex*** that ***do not belong*** to any cluster.



ε =0.7
μ=3

AnyTimeSCAN method: an extension of SCAN

- AnyTimeSCAN: extension of the SCAN;
- Gives the same result as SCAN;
- (+) Reduce the calculation operation of the structural similarity



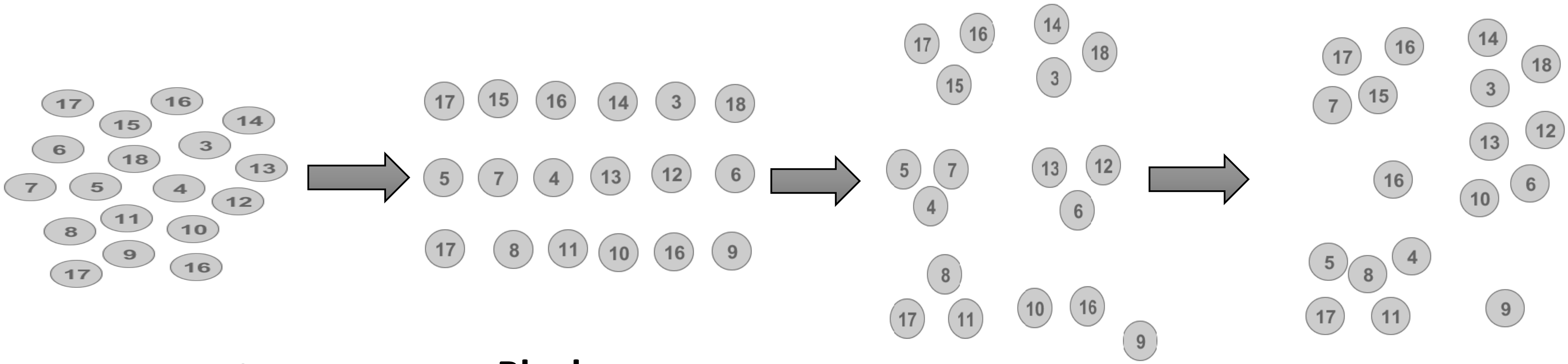
a, b and C: **Core**

i and J: **Noise**

AnyTimeSCAN

The steps of the AnyTime SCAN algorithm

1. Summarization: decomposes the set of vetrices into equals blocks
2. Combines the sub-clusters;
3. Determines the borders.



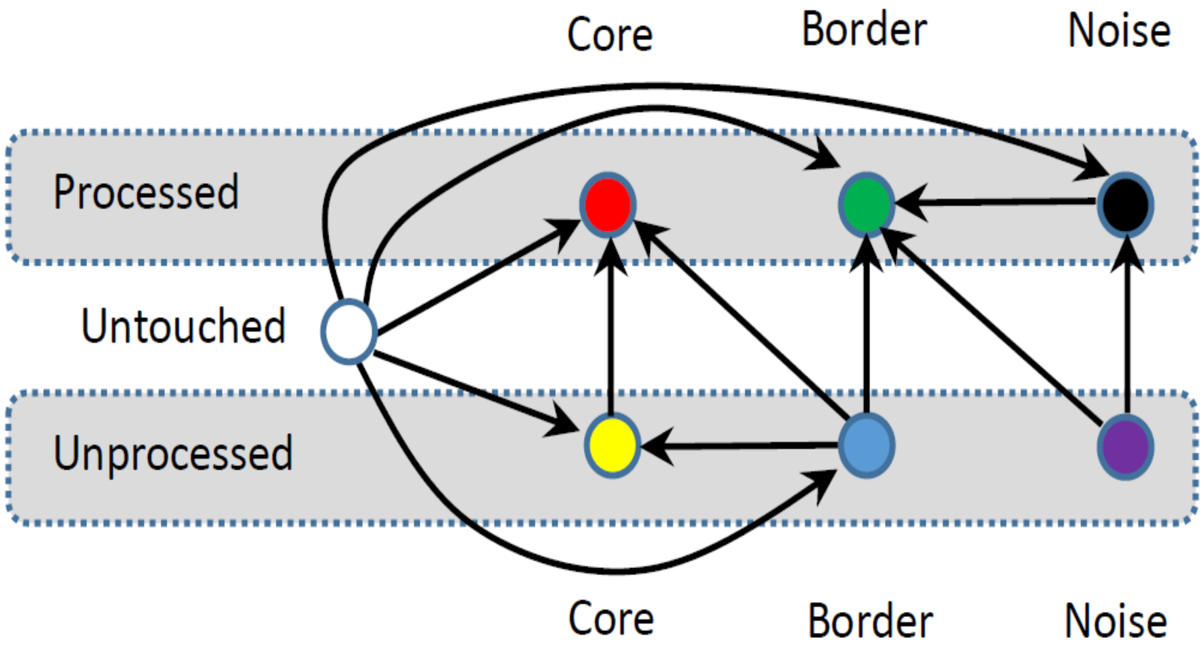
Input: Graph G **Blocks** **Local clustering** **Merge the sub-clusters**

AnyTimeSCAN

AnyTimeSCAN: Summarization step

- At t =0, all vertices **V** are marked as untouched vertices.

- For each vertex **v** in a bloc **b,** we determine its state according to its neighbors.

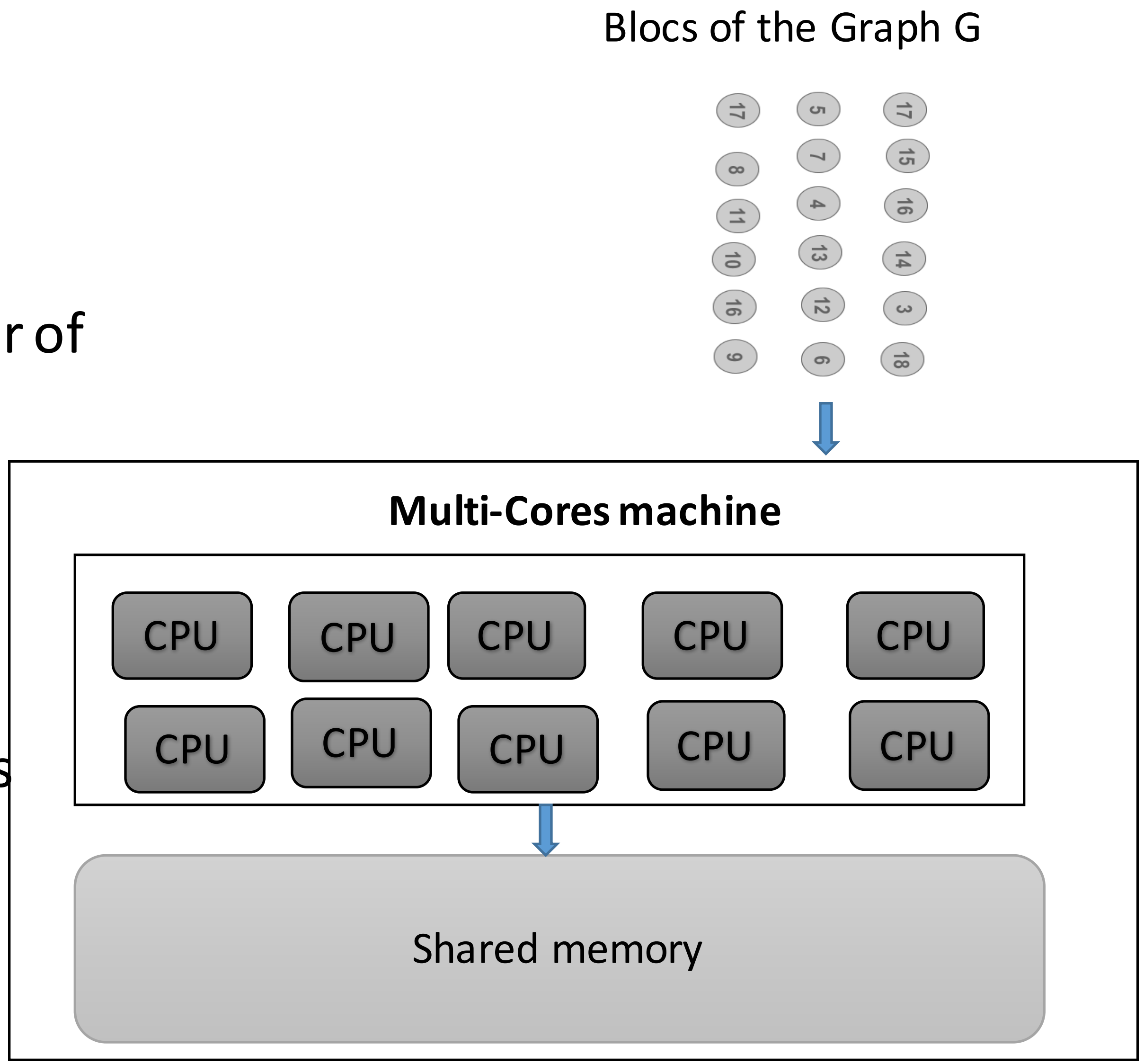


The state transition schema for vertices

AnyTimeSCAN

AnyTimeSCAN: Parallel processing

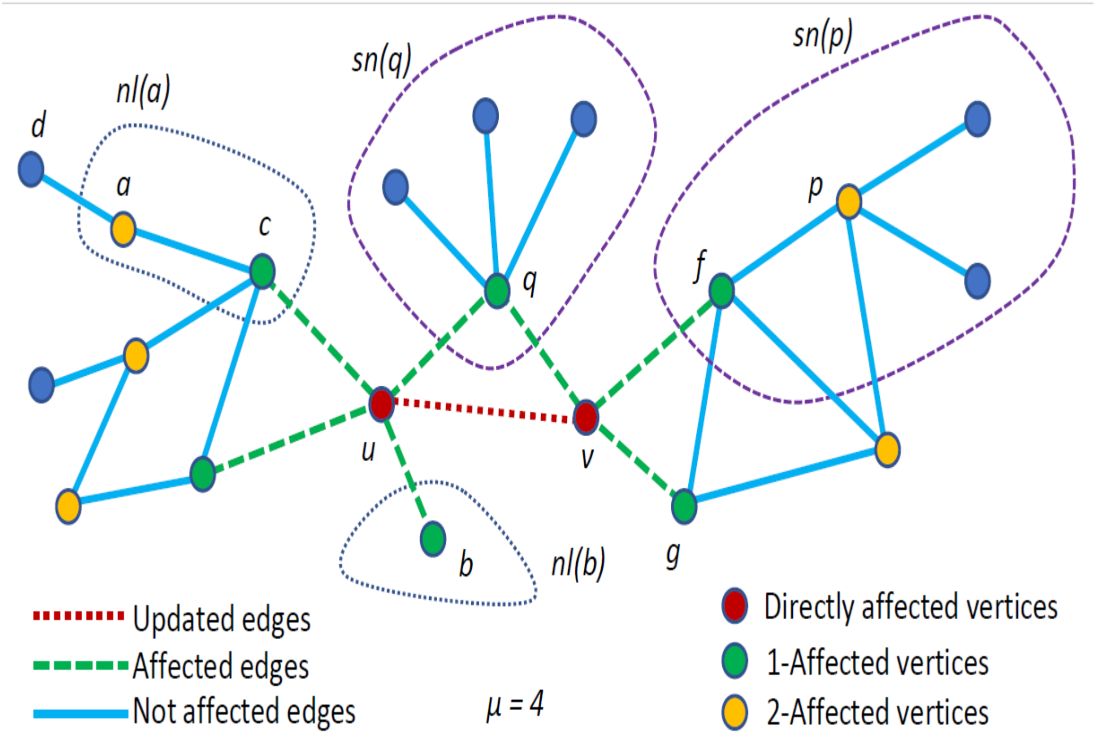
- Implementation based on OpenMP API
- Degree of parallelism according to the number of blocks or the parameter α >> 1
- Shared memory:
  - Group all the blocks;
  - Keep the links between the sub-graphs
  - Keep the shared data.



AnyTimeSCAN

DanySCAN: Dynamic Clustering

- Incremental approach;
- Each update in the graph, a set of vertices and edges will be affected by this change;



AnyTimeSCAN

Experimental study

Dataset:

Graph	Vertices	Edges	$\overline{d}$	c
Ego-Gplus (GR01)	107,614	13,673,453	127.06	0.4901
Soc-LiveJournal1 (GR02)	4,847,571	68,993,773	14.23	0.2742
Soc-Poket (GR03)	1,632,803	30,622,564	18.75	0.1094
Com-Orkut (GR04)	3,072,441	117,185,083	38.14	0.1666
Kron_g500-logn21 (GR05)	2,097,152	182,082,942	86.82	0.1649
Twitter (GR06)	41,652,230	1,369,000,750	32.8	0.0730

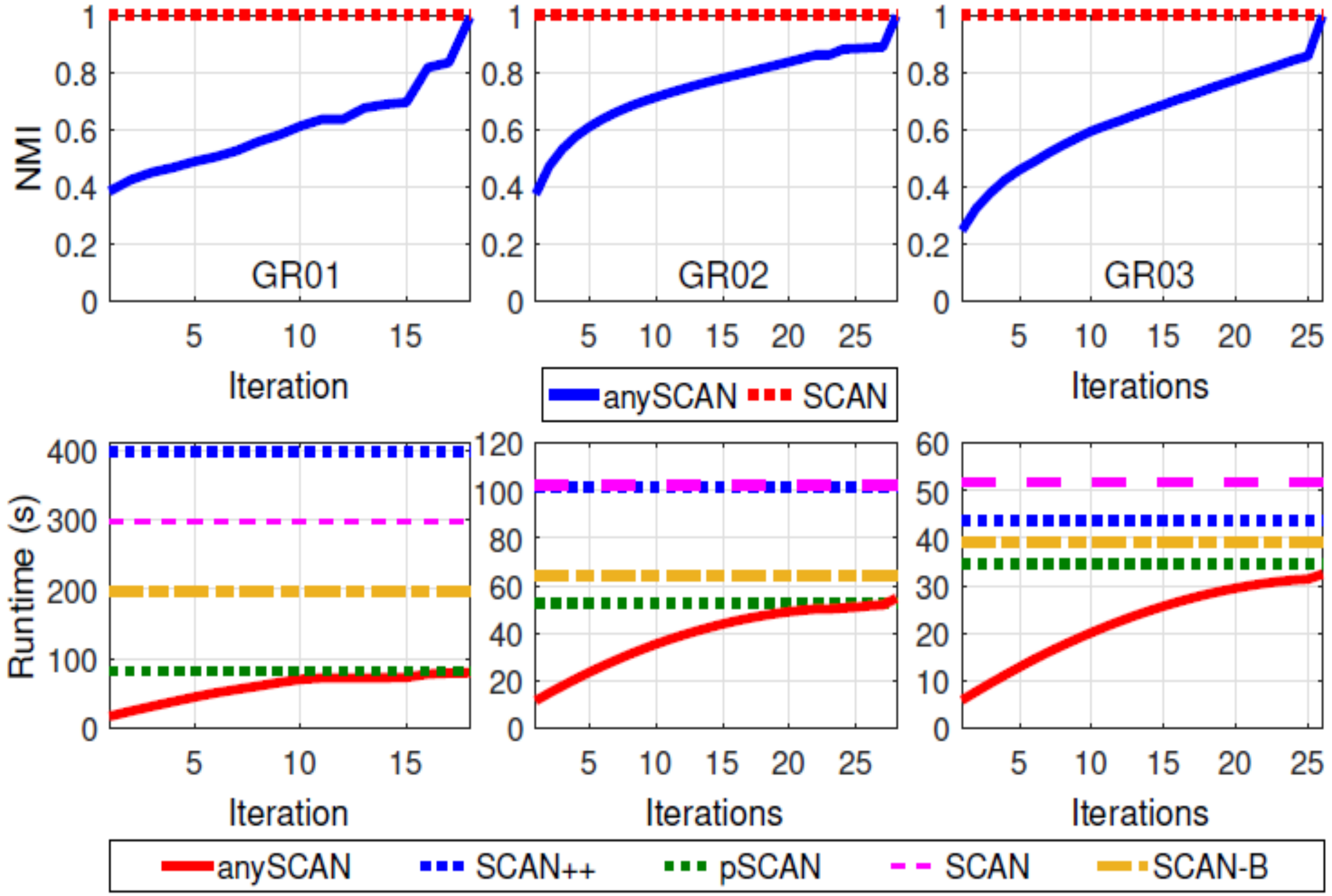
Environment:

- 2 * 3.1 GHz Intel Xeon CPUs with 64 GB local RAM
- OpenMP 3.2

AnyTimeSCAN

AnyTimeSCAN: experimental results

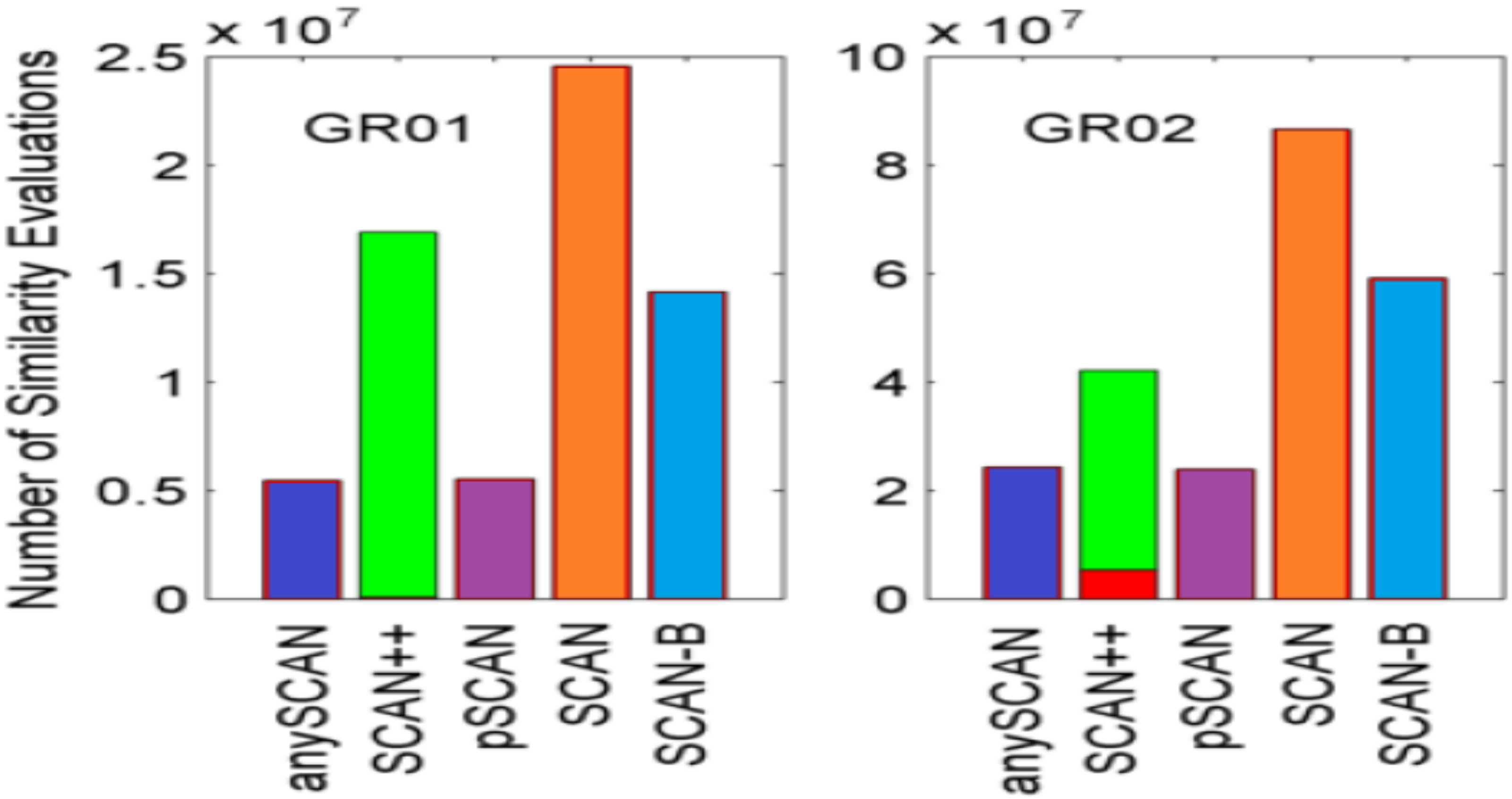
- Give an intermediate results, which has not been guaranteed by the others approaches;
- Runtimes comparison with others approaches.



AnyTimeSCAN

AnyTimeSCAN: experimental results

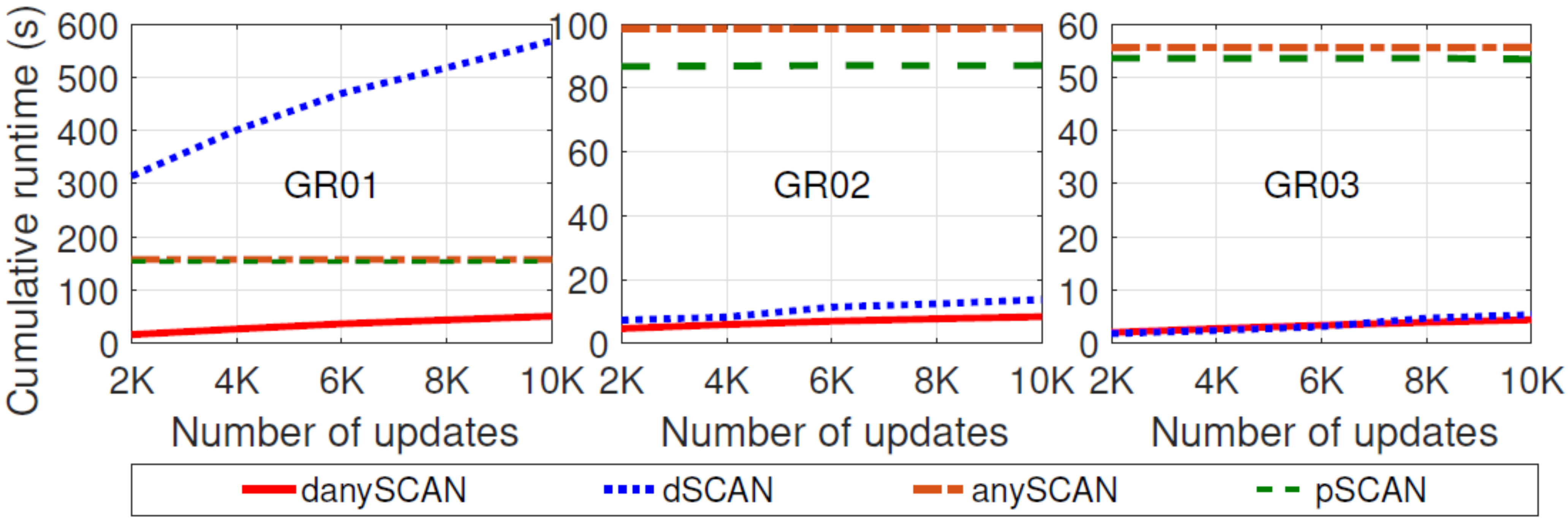
Reduce the structural similarity operation between a pairs of vertices in the graph.



Comparison in terms of the number of structural similarity operation

DanySCAN

DanySCAN: **experimental results**



Performance of danySCAN according the number of updates for different datasets

Conclusion

- A new paradigm to reduce the similarity calculation operation;
- An incremental approach for dynamic graphs clustering ;
- Interactive model to provide an intermediate results during the execution of the algorithm;
- A parallel approach based on shared memory architecture;
- A strong experimental study;