Karlsruhe Institute of Technology

Parallel Dynamic and Selective Community Detection in Massive Streaming Graphs

European Conference on Data Analysis 2013, Luxembourg · July 11, 2013

Christian L. Staudt, Yassine Marrakchi, Aleksejs Sazonovs and Henning Meyerhenke

INSTITUTE OF THEORETICAL INFORMATICS · PARALLEL COMPUTING GROUP

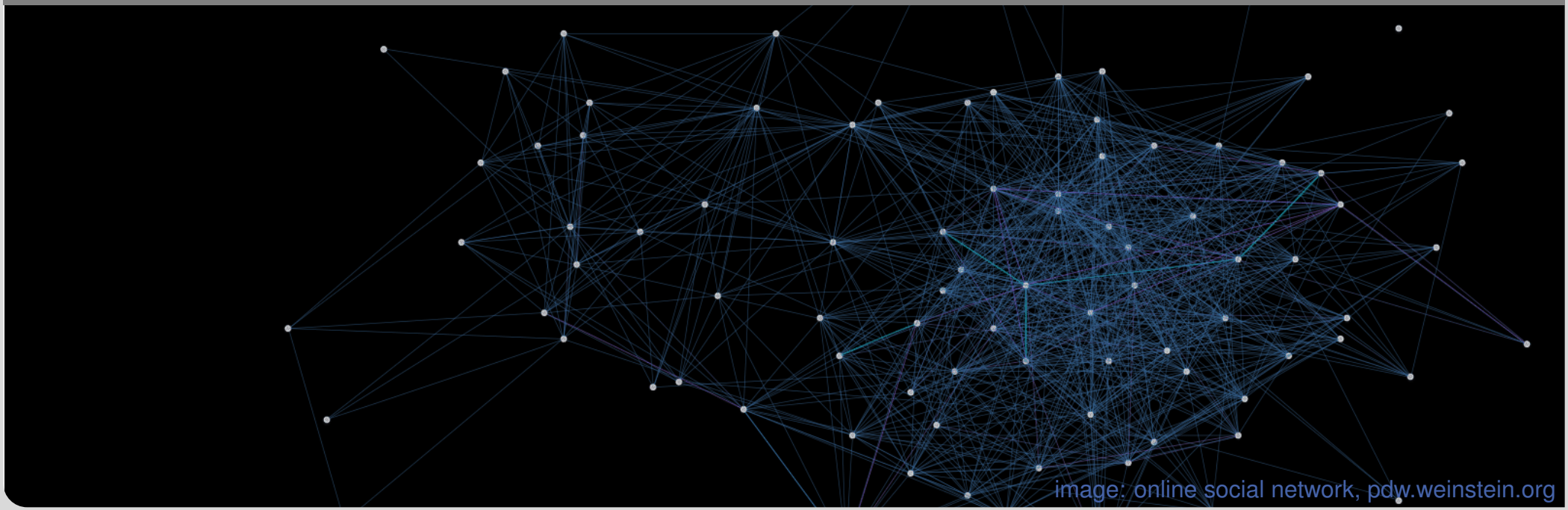


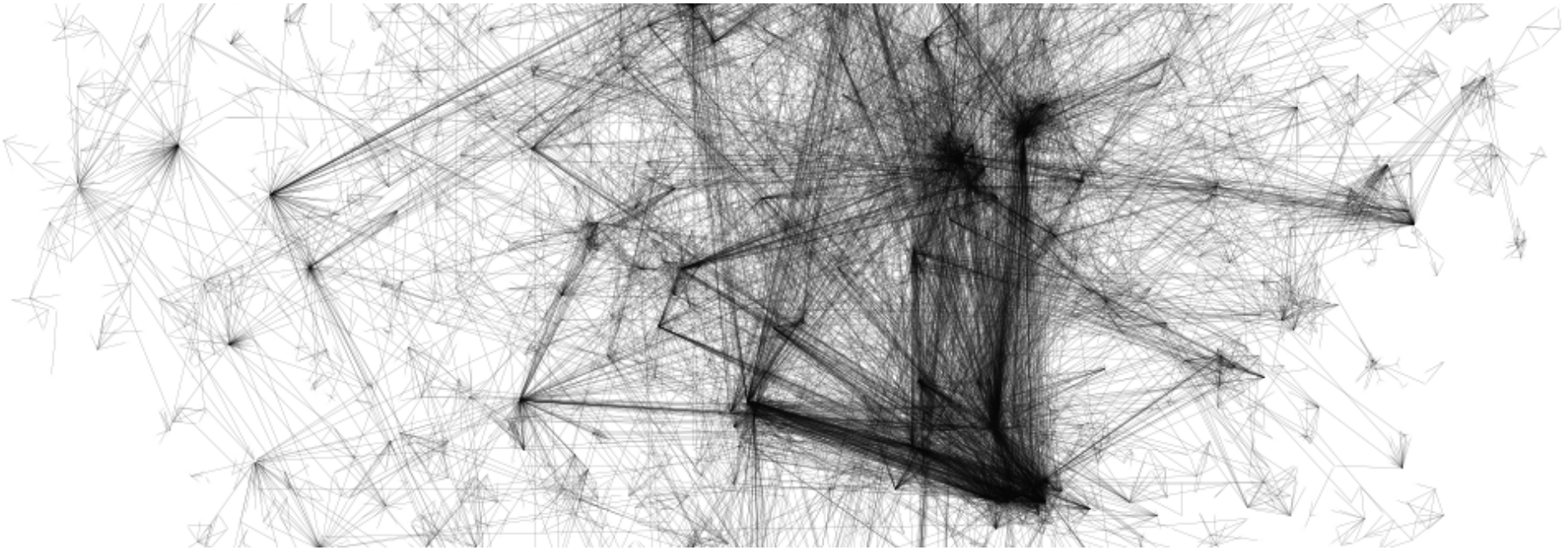
image: online social network, pdw.weinstein.org

KIT – University of the State of Baden-Wuerttemberg and
National Laboratory of the Helmholtz Association

www.kit.edu

Complex Networks

- **scale-free**: skewed degree distribution
- **community structure**: densely connected node groups



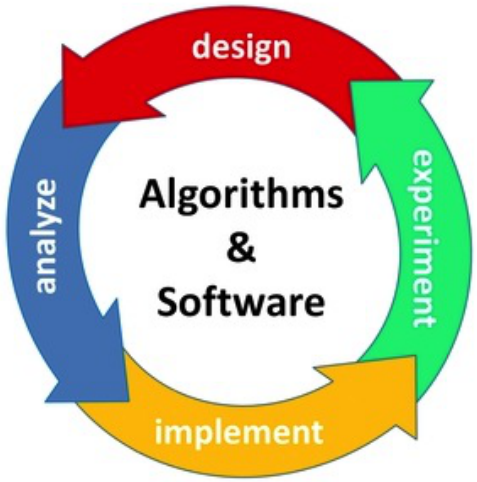
PGP giant component n=10680 m=24316 http://www.cc.gatech.edu/dimacs10/archive/clustering.shtml

- **structural features** $\rightarrow$ **computational challenges**, e.g.
  - degree distribution $\rightarrow$ load-balancing issues

Introduction

- fast heuristics for **community detection** in **large complex networks**
- **applications** e.g.
  - social communities in social networks
  - topical communities on the web
  - functional communities in metabolic networks
- **scenarios**

1. **dynamic community detection**
2. **selective community detection**



Community Detection

Problem Definition

- **given**: graph $G = (V, E)$, representation of a complex network
- **wanted**: partition $\zeta = \{C_1, \ldots, C_k\}$ of the node set into disjoint communities

- notion of a community is not precise
- various definitions and quality measures [Schaeffer 2007]
  - **Modularity** [Newman, Girvan 2004]: **coverage** minus expected coverage

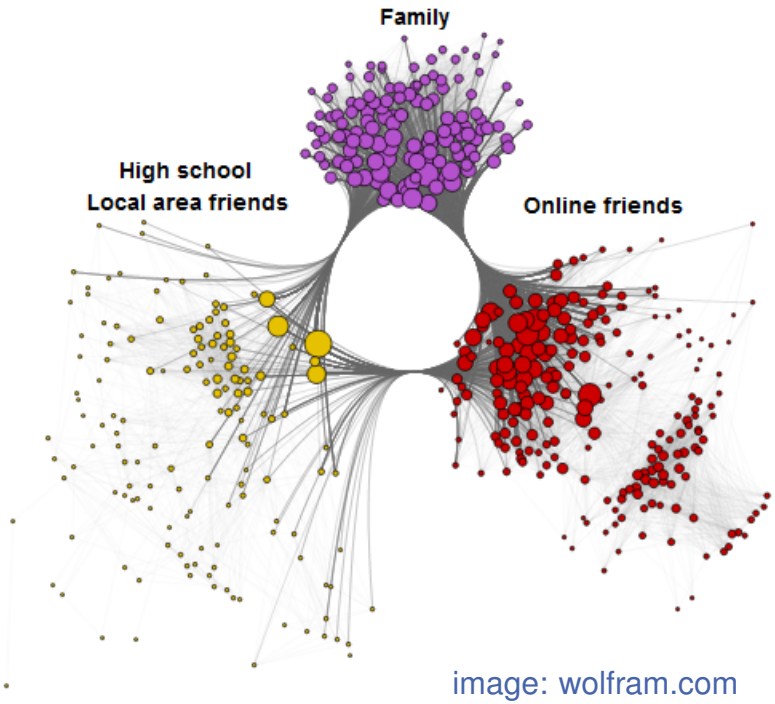


image: wolfram.com

Community Detection - Static and Global

- a quick recap of our **previous work** on scalable community detection
- **parallel algorithms**
  - **PLP**: label propagation
  - **PLM**: Louvain method
  - **EPP**: ensemble approach

Design Goal

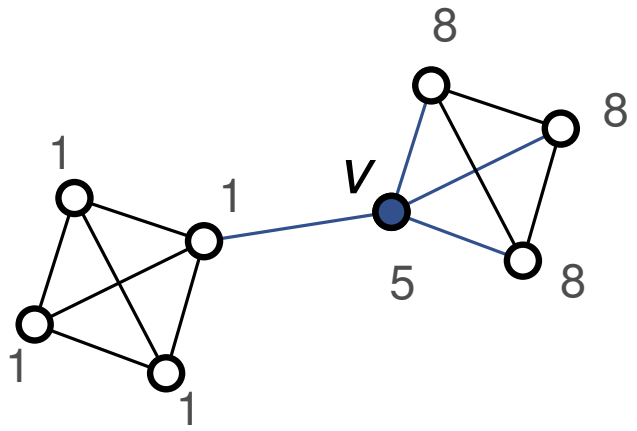
- scale to networks with billions of edges
  - **PLP** on 32 cores processes 3 billion edge web graph in 2 minutes

[Staudt, Meyerhenke *ICPP* 2013: *Engineering High-Performance Community Detection Heuristics for Massive Graphs* - http://arxiv.org/abs/1304.4453]

Parallel Label Propagation **PLP**

- local coverage maximizer
- purely local update rule
  $\rightarrow$ embarassingly parallel

initialize nodes with unique labels
while *labels not stable* **do**
  parallel for $v \in V$
    adopt dominant label in $N(v)$
  endfor
end
return *communities from labels*

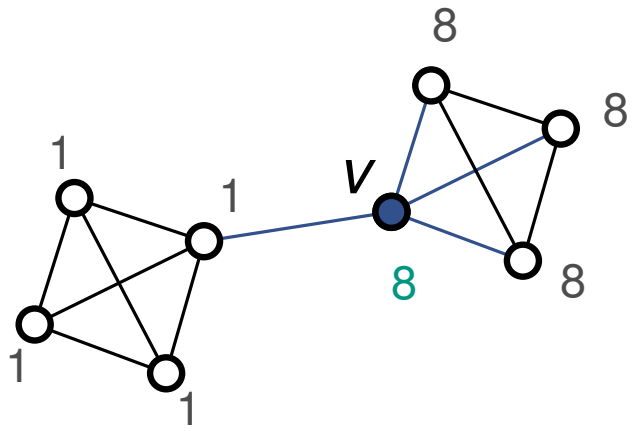


[sequential: Raghavan et al. 2007] [parallel: Staudt, Meyerhenke 2013]

Parallel Label Propagation **PLP**

- local coverage maximizer
- purely local update rule
  $\rightarrow$ embarassingly parallel

initialize nodes with unique labels
while *labels not stable* **do**
  parallel for $v \in V$
    adopt dominant label in $N(v)$
  endfor
end
return *communities from labels*

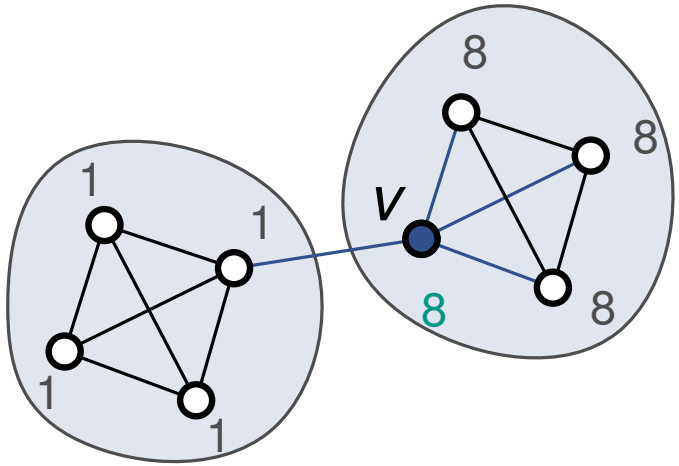


[sequential: Raghavan et al. 2007] [parallel: Staudt, Meyerhenke 2013]

Parallel Label Propagation **PLP**

- local coverage maximizer
- purely local update rule
  $\rightarrow$ embarassingly parallel

initialize nodes with unique labels
while *labels not stable* **do**
  parallel for $v \in V$
    adopt dominant label in $N(v)$
  endfor
end
return *communities from labels*



[sequential: Raghavan et al. 2007] [parallel: Staudt, Meyerhenke 2013]

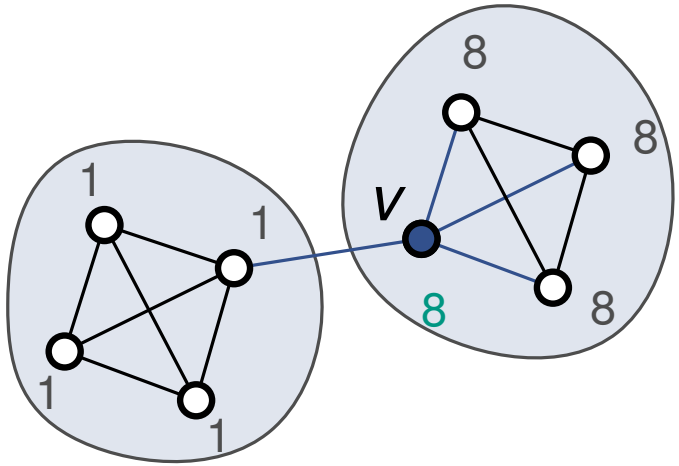
Parallel Label Propagation **PLP**

- local coverage maximizer
- purely local update rule
  $\rightarrow$ embarassingly parallel

initialize nodes with unique labels
while *labels not stable* **do**
  parallel for $v \in V$
    adopt dominant label in $N(v)$
  endfor
end
return *communities from labels*

Strengths / Weaknesses

- \+ very fast and highly scalable
- \- moderate community quality



[sequential: Raghavan et al. 2007] [parallel: Staudt, Meyerhenke 2013]

Ensemble Preprocessing **EPP**

Parallel Louvain Method PLM

- parallel implementation
  [sequential: Blondel 2008,
  parallel: Staudt, Meyerhenke 2013]
- higher quality, less scalable

Ensemble Preprocessing EPP

Parallel Louvain Method PLM

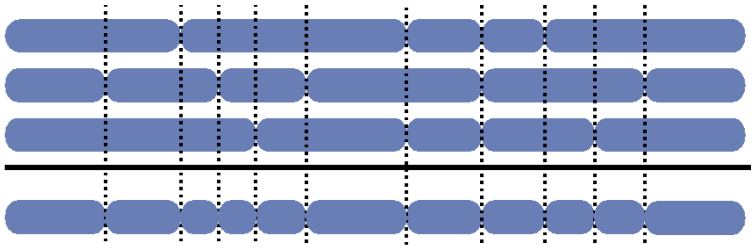
- parallel implementation
  [sequential: Blondel 2008,
  parallel: Staudt, Meyerhenke 2013]
- higher quality, less scalable

Ensemble Preprocessing EPP

- combines the strengths of **PLP** and **PLM** $\rightarrow$ good **quality-speed tradeoff**
  - run multiple **PLP** in parallel
  - consensus of results
    $\rightarrow$ finds contested nodes
  - large graphs easily handled by **PLM** after coarsening

[ensemble approach: Ovelgönne, Geyer-Schulz 2012]
[specific algorithm: Staudt, Meyerhenke 2013]

parallel for Base *in ensemble*
  $\zeta_i \leftarrow \textbf{Base}_i(G)$
endfor
$\bar{\zeta} \leftarrow \textbf{consensus}(\zeta_1, \ldots, \zeta_b)$
$G^1 \leftarrow \textbf{coarsen}(G, \bar{\zeta})$
$\zeta^1 \leftarrow \textbf{Final}(G^1)$
$\zeta \leftarrow \textbf{prolong}(\zeta^1, G)$
return ζ

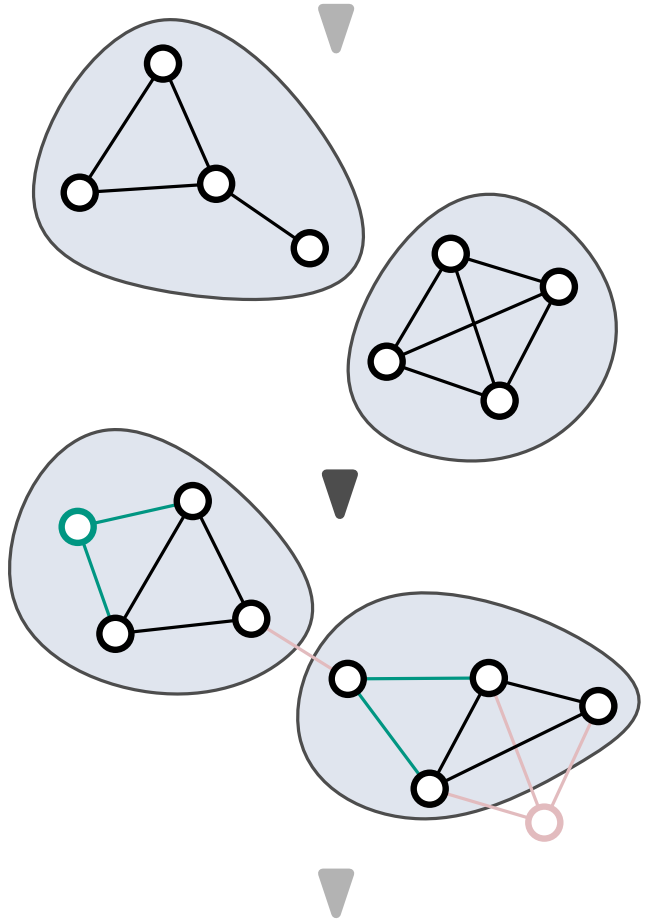


consensus communities

Dynamic Community Detection

Problem Definition

- **given**: dynamic graph $(\ldots, G_t, G_{t+1}, \ldots)$ being modified through a stream of events $\Delta(G_t, G_{t+1})$
- **wanted**: communities at requested points in time



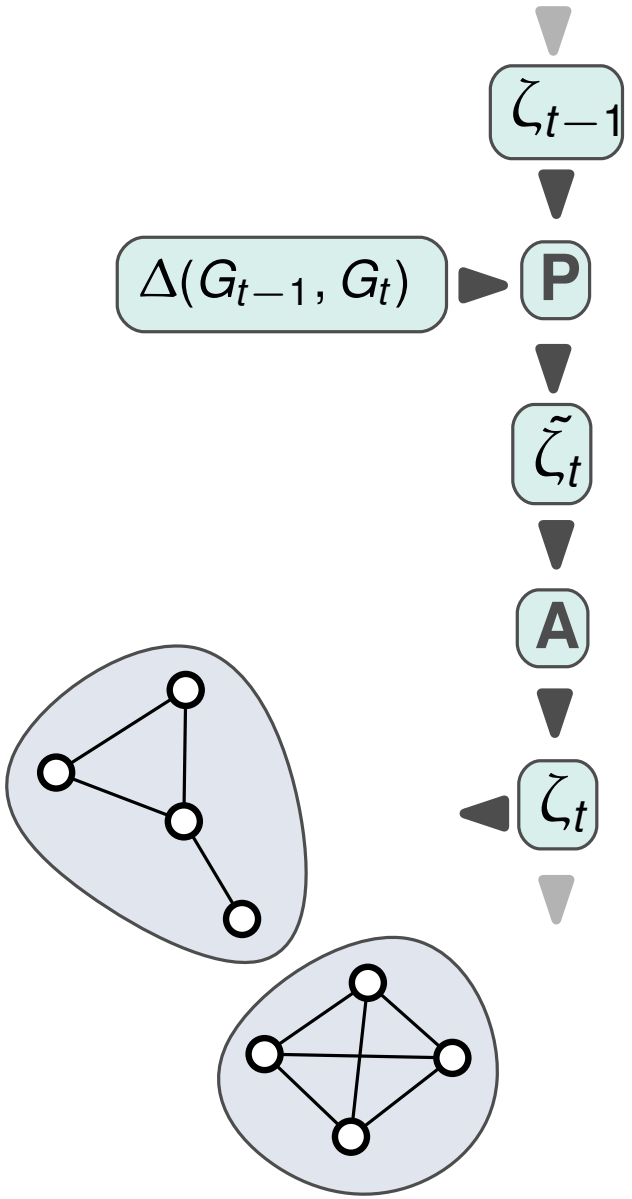
Goals

- **speedup** by adapting previous solution
- **continuity**: consistency of communities over time
- **community quality** in view of changing network structure

[Görke et al. 2010: *Modularity-driven clustering of dynamic graphs*]

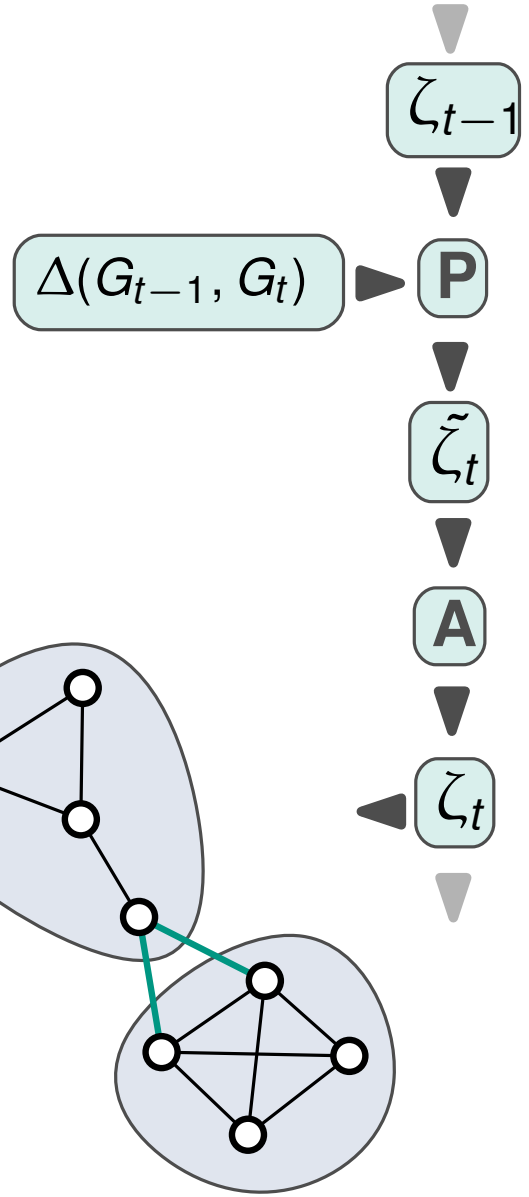
Dynamic Parallel Label Propagation **dynPLP**

1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time



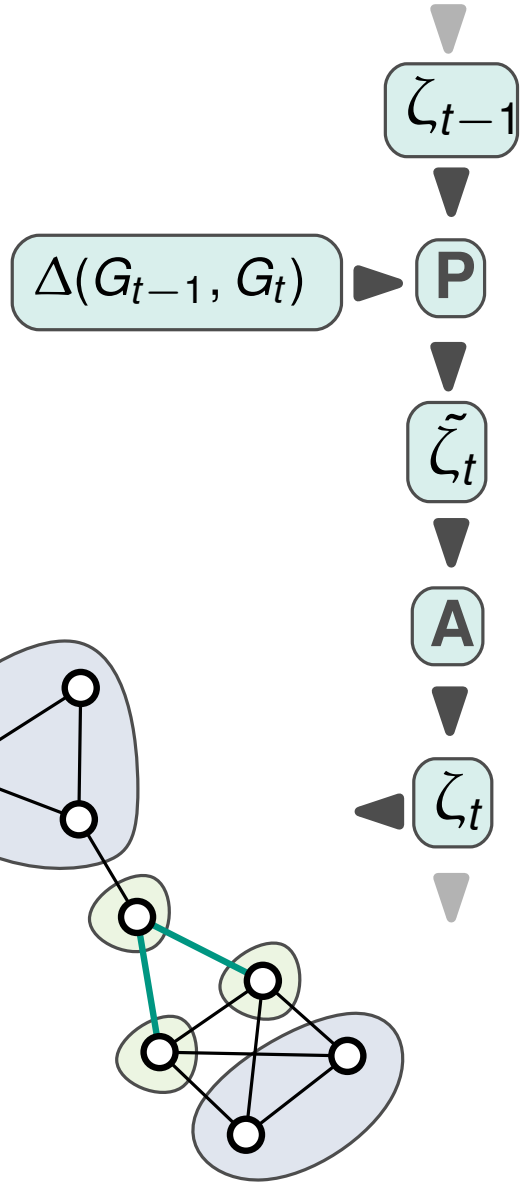
Dynamic Parallel Label Propagation **dynPLP**

1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time



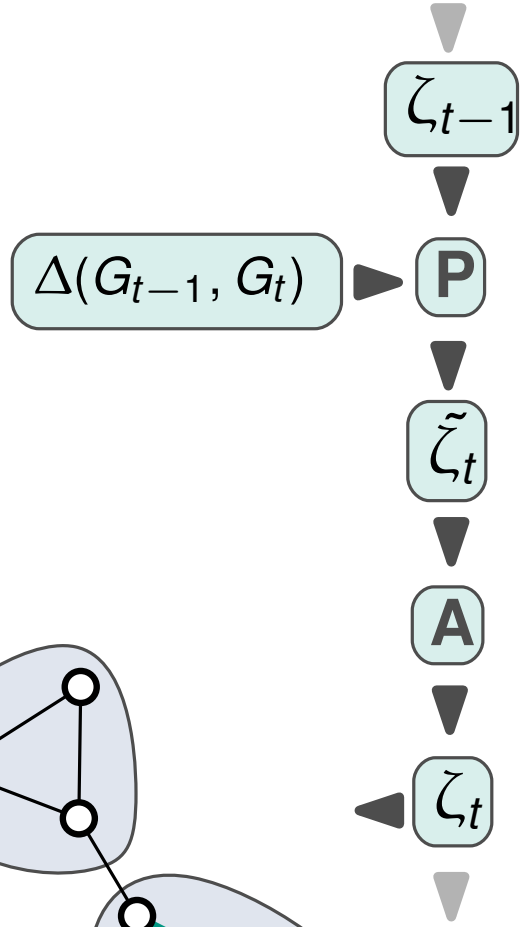
Dynamic Parallel Label Propagation **dynPLP**

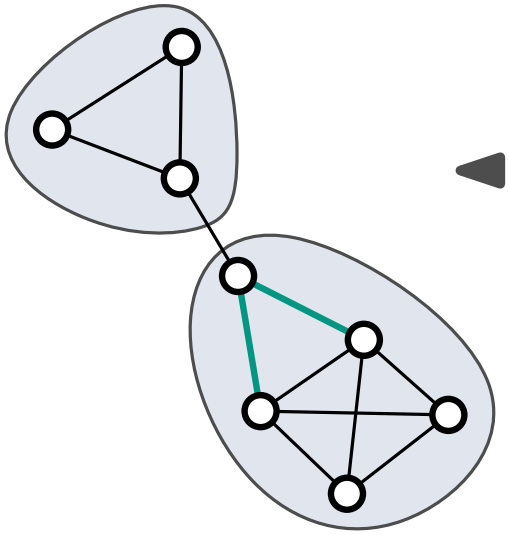
1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time



Dynamic Parallel Label Propagation **dynPLP**

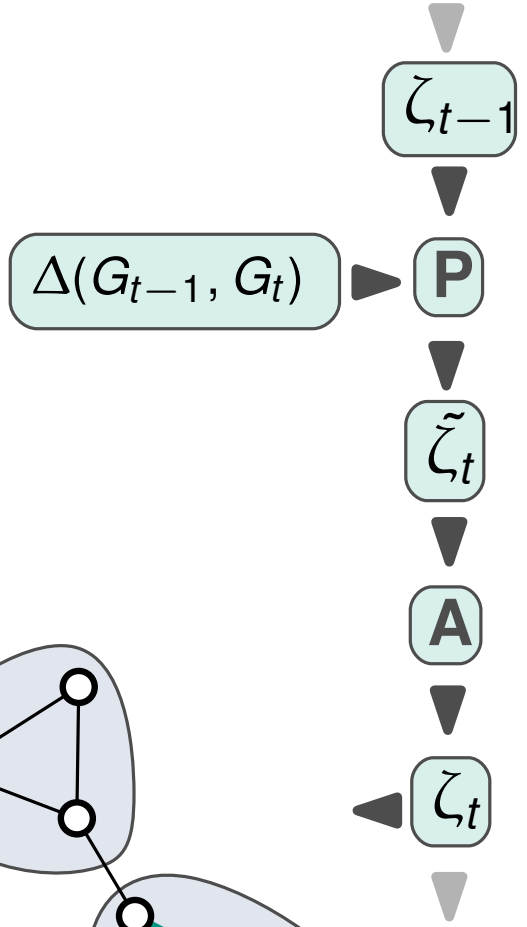
1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time



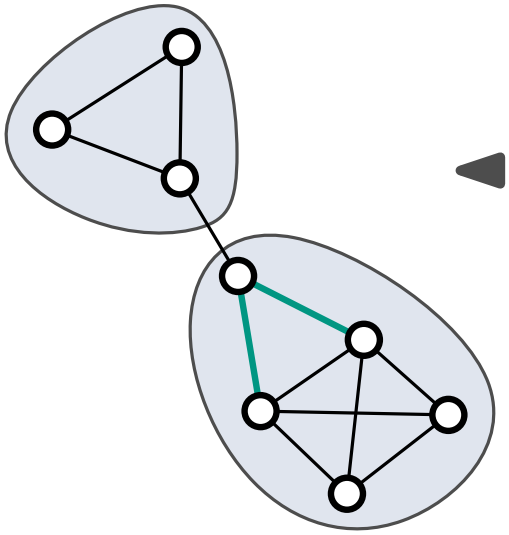


Dynamic Parallel Label Propagation **dynPLP**

1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time

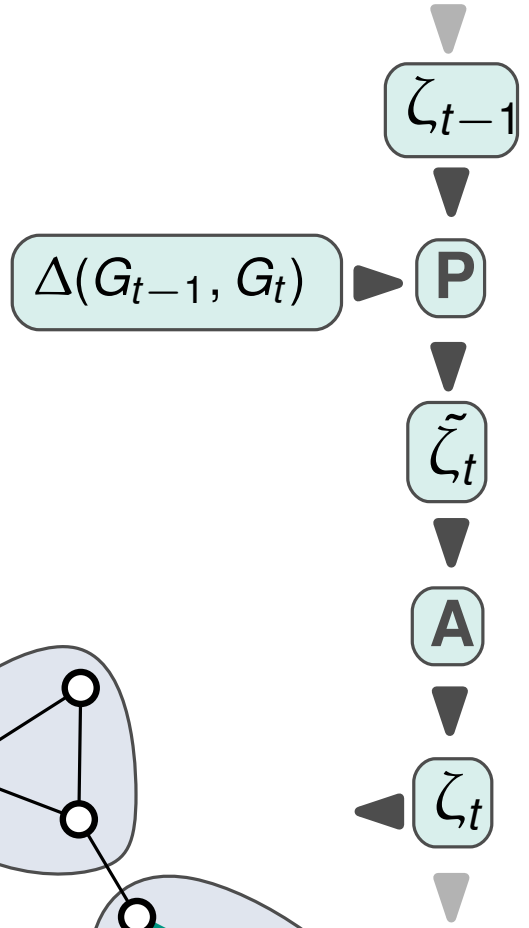


- applicable as base algorithm in ensemble (**EPP**)



Dynamic Parallel Label Propagation **dynPLP**

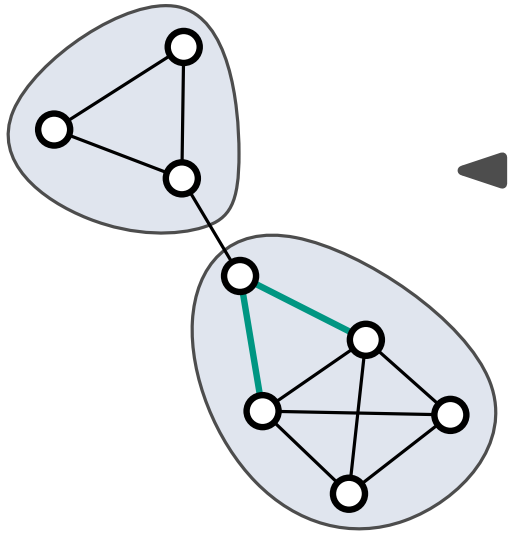
1. **prep strategy** reacts to event stream
   $\rightarrow$ temporary communities
   (a) **strategy I**: affected nodes to singletons
   (b) **strategy IN$_1$**: affected nodes and 1-neighborhood to singletons
2. parallel label propagation **PLP** applied to return communities at requested time



- applicable as base algorithm in ensemble (**EPP**)

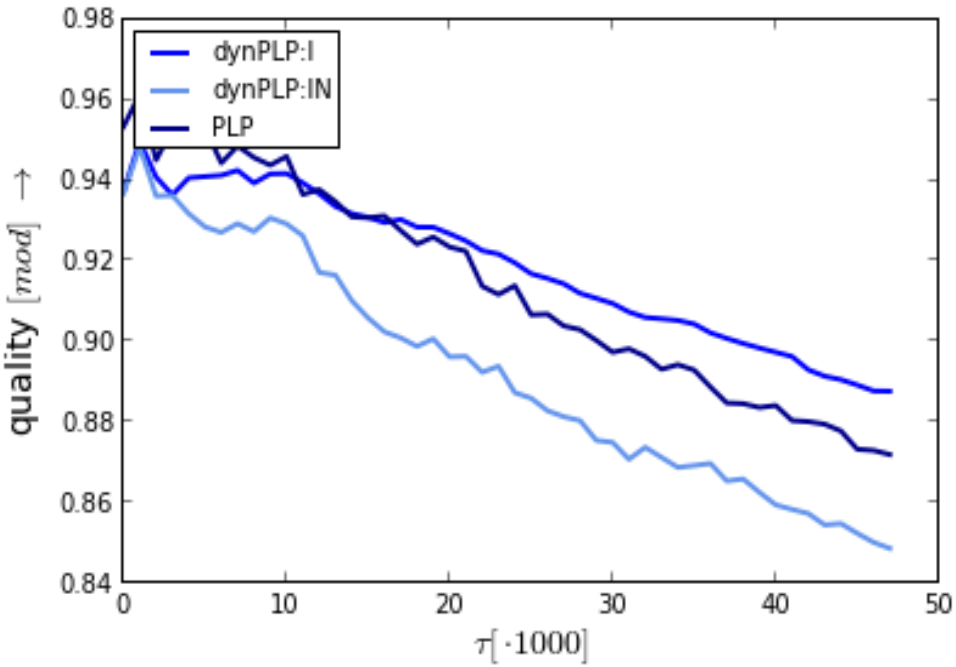
Research Questions

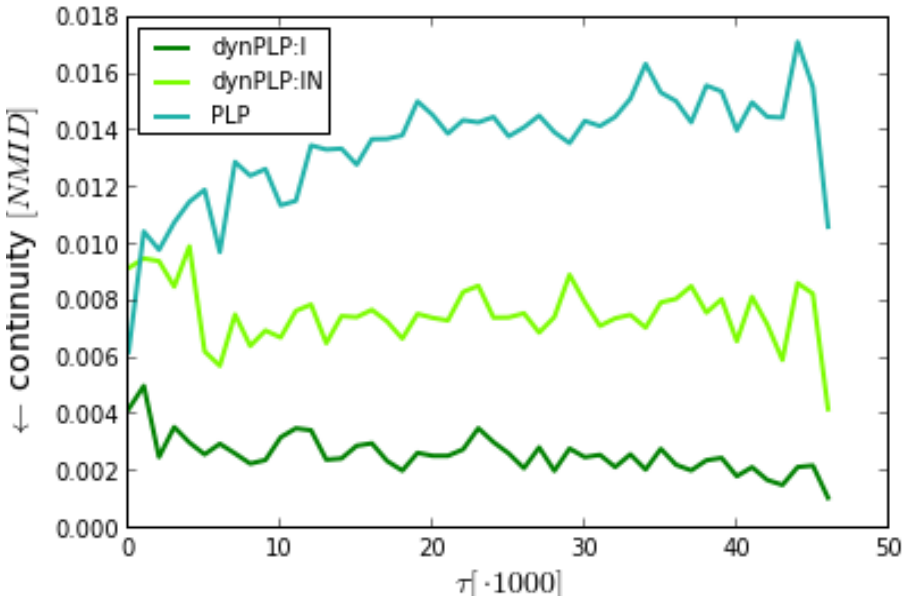
- quality as good as static recomputation?
  - are small local revisions sufficient?

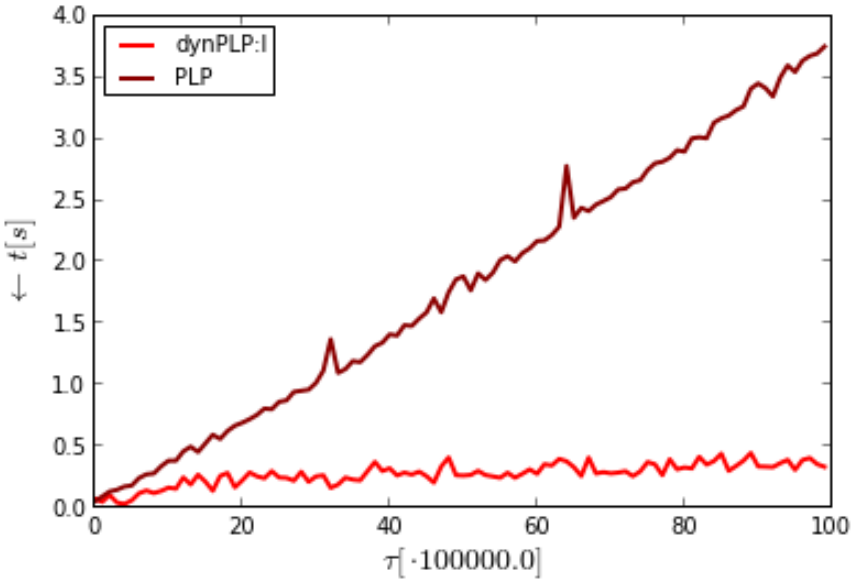


Evaluation dynPLP

- **dynPLP quality** matches **PLP**
- isolating affected nodes only works best
- **continuity** significantly improved
- **running time** stays proportional to network change, not network size



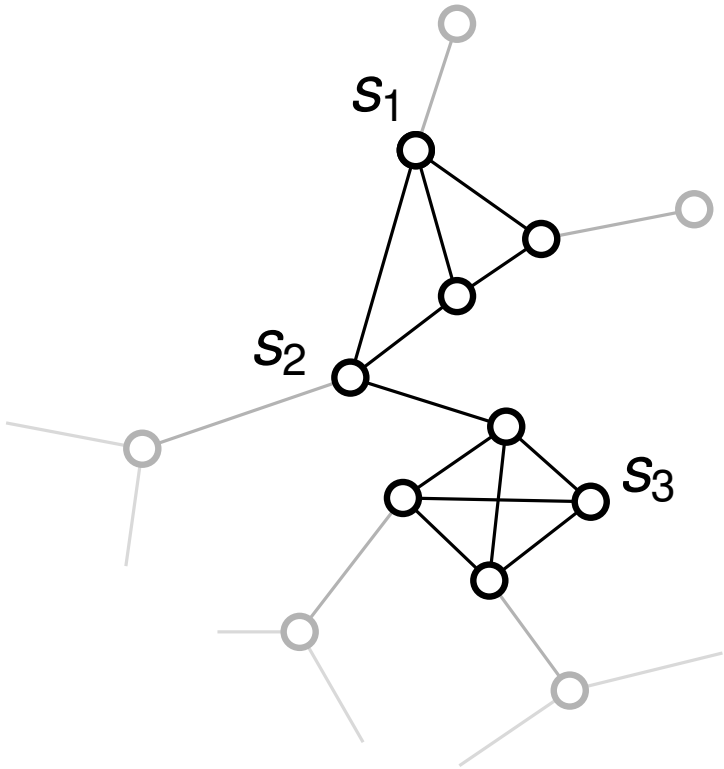




Selective Community Detection

Problem Definition

- **given**: graph $G = (V, E)$ and set of seed nodes S
- **wanted**: assignment of seed node s to community C_s



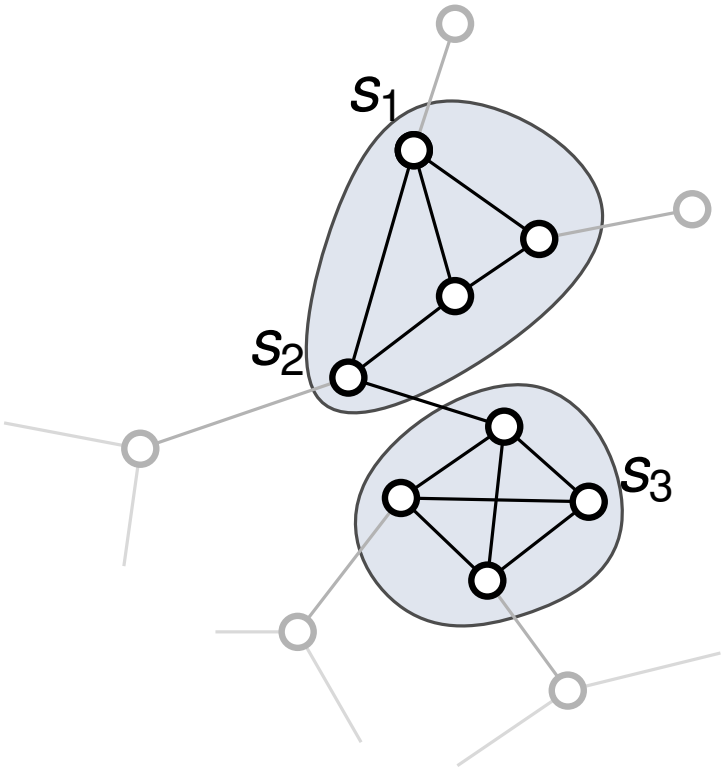
Goals

- finding high-quality communities fast
  - when **global solution not needed / feasible**
  - in networks which are **not globally known**

Selective Community Detection

Problem Definition

- **given**: graph $G = (V, E)$ and set of seed nodes S
- **wanted**: assignment of seed node s to community C_s



Goals

- finding high-quality communities fast
  - when **global solution not needed / feasible**
  - in networks which are **not globally known**

Selective Community Detection

a different game...

- **modularity** is a global measure
  - $\rightarrow$ **Louvain method** not suitable
- **label propagation** not suitable
  - communities need to restrict each other's expansion

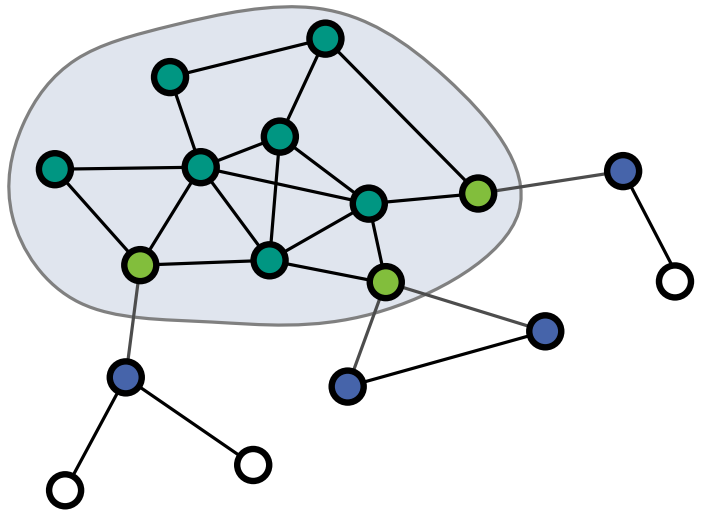
Related Work

- greedy community quality maximization (node-by-node)
  - objective functions [Clauset 2005] [Luo 2008] [Chen 2009] [. . .]
  - prioritizing nodes [Chen 2009] [Xu 2012]

Measuring Community Quality

Structure of a Community

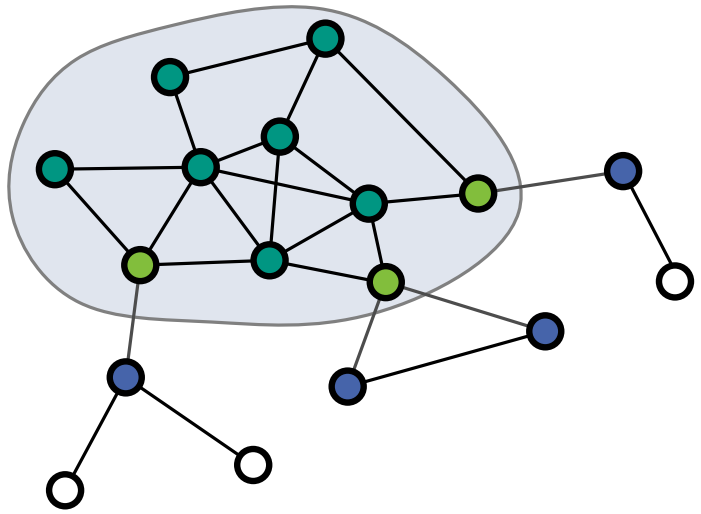
- **core** $K(C)$
- **boundary** $B(C)$
- **shell** $\Omega(C)$



Measuring Community Quality

Structure of a Community

- **core** $K(C)$
- **boundary** $B(C)$
- **shell** $\Omega(C)$



Quality Measures

community: high ratio of core edges to boundary-shell edges

- **Conductance** $[0, 1] \downarrow$ [Andersen, Lang 2006]
- **"Local Modularity" M** $[0, \infty) \uparrow$ [Luo et. al 2008]
- **"Local Modularity" L** $[0, \infty) \uparrow$ [Chen et al. 2009]

Greedy Community Expansion **GCE**

- generic greedy algorithm
- interchangeable components
  1. **quality objective** $Q(C)$
     - **conductance** $\downarrow$
     - **L** $\uparrow$
     - **M** $\uparrow$
  2. **node acceptability** $A(v, C)$
     - **neighborhood overlap** of C and v

$C_s \leftarrow \{s\}$
while *candidates in shell with* $\Delta Q > 0$ **do**
  (prioritize candidates by acceptability)
  include node with max. ΔQ
end
return C_s

Return Value

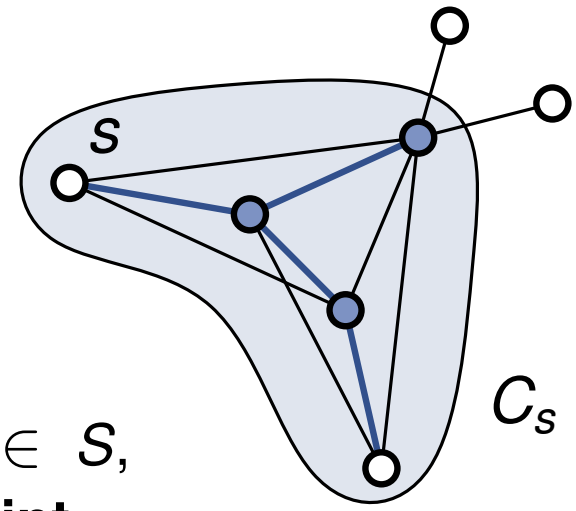
Assignment of seed s to community C_s.
For $s_i, s_j \in S$ communities C_{s_i} and C_{s_j} may **overlap**.

Selective SCAN **seISCAN**

- our adaption of global algorithm **SCAN** [Xu et al. 2007] for selective scenario
- community: **core nodes** and their **close neighbors** according to **distance measure** D and threshold ϵ

Concepts

- **core node** has at least μ close neighbors
- **reachability**: u and v are reachable from eachother $\iff$ there is a path $(u, c_1, \ldots, c_k, v)$ where inner nodes are cores
- **community** C_s is the set of nodes reachable from s



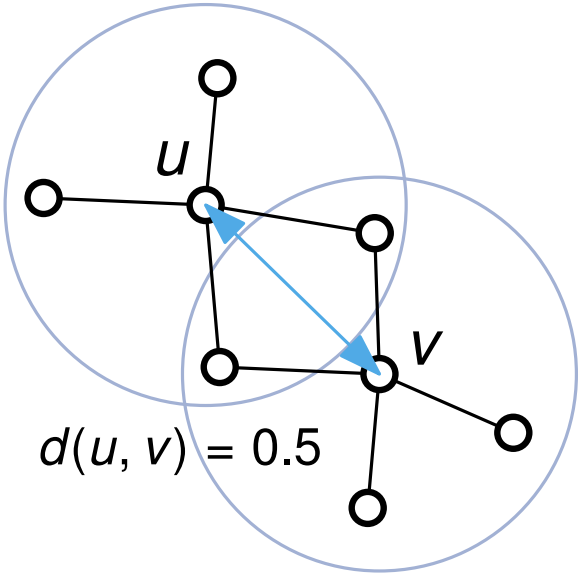
Return Value

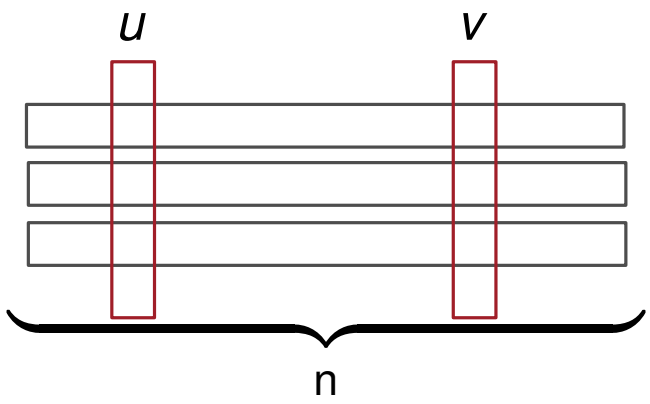
Assignment of seed s to community C_s. For $s_i, s_j \in S$, communities C_{s_i} and C_{s_j} are either **identical** or **disjoint** or **empty** ($\rightarrow$ node is an **outlier** or **hub**)

Selective SCAN **seISCAN**

Distance Measures

- **neighborhood distance (ND)**
  $d(u, v) := 1 - \frac{|N(u) \cap N(v)|}{\sqrt{|N(u)| \cdot |N(v)|}}$ [Xu 2007]
- **algebraic distance (AD)** [Chen, Safro 2011]
  - graph-structural distance between nodes
  - iterative preprocessing



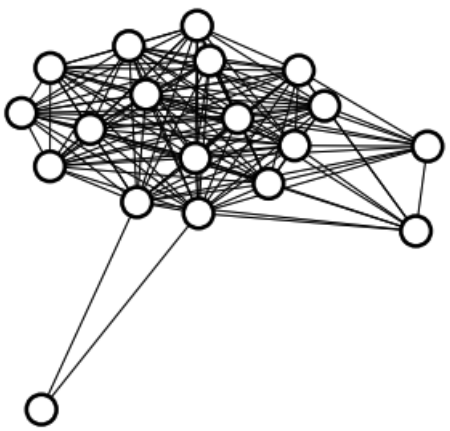


Evaluation GCE

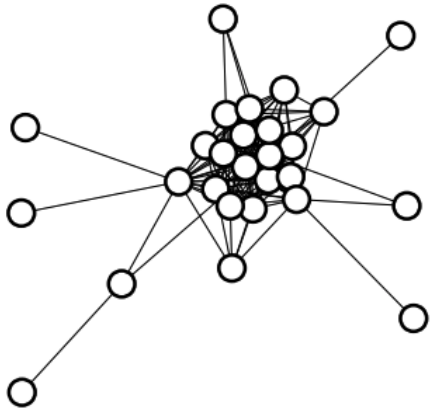
Quality Objectives

- **conductance** and **M** are equivalent
- **L** agrees more with intuition (exclusion of satellites)

L



conductance

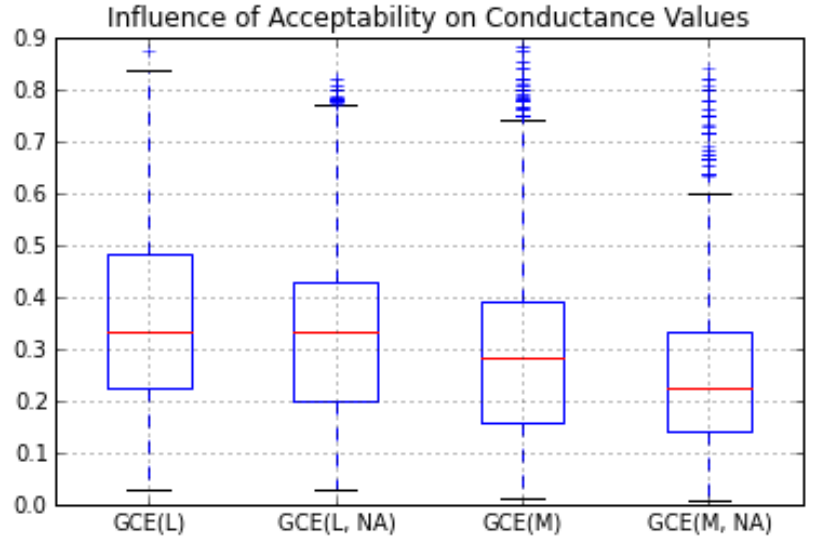


Node Acceptability

- quality improvement possible when prioritizing by **neighborhood overlap**
- factor 15 slower

Scaling

- **GCE does not scale to massive graphs**
  - shell grows with graph size

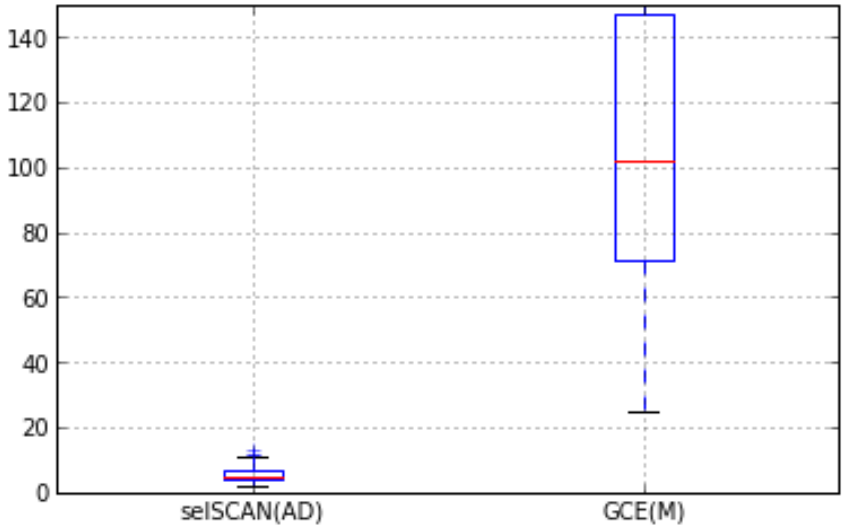


running time [ms] **selSCAN** vs **GCE**

Evaluation **selSCAN**

Efficiency

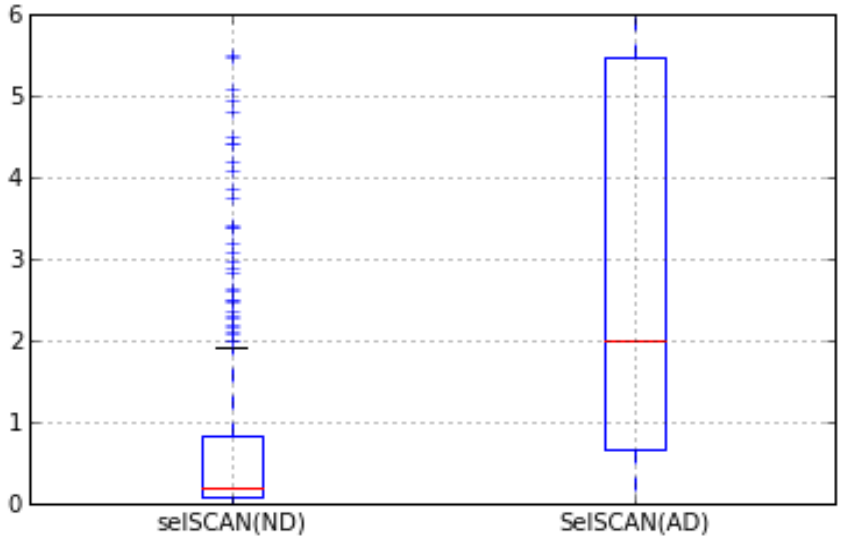
- **selSCAN factor 100 faster** than **GCE**
  - faster operations
  - avoids redundancy for close seeds
  - significant fraction of nodes are classified as outliers or hubs
- **quality**: better *L*, slightly worse conductance
- scales well to massive graphs



running time [ms] **selSCAN** vs **GCE**

Distance Measures

- **AD** can yield significantly improved quality
- **AD** preprocessing: minutes for large graphs
  - pays off for repeated requests on same graph
- best **selSCAN** and **AD** parameters depend on graph structure



quality L for **selSCAN** variants

Conclusion

Dynamic Community Detection

- dynamic approach: **speedup** and improved **continuity without loss of quality**
- **PLP** and **EPP** adapted to dynamic scenario

Selective Community Detection

- different objective functions and algorithmic ideas
- scaling to massive graphs requires new algorithm variants such as **selSCAN**
- new node distance measures like **AD** seem promising for speedup and quality improvement
  - choice of parameters $\rightarrow$ parameter studies needed

NetworKit

- a toolkit for **engineering high-performance network analysis algorithms**
  - C++11 and OpenMP
- **free software** (*MIT License*)
  - 1.0 release in spring 2013: static global community detection
  - 2.0 release coming in fall 2013

[http://parco.iti.kit.edu/software/networkit.shtml]

NetworKit

- a toolkit for **engineering high-performance network analysis algorithms**
  - C++11 and OpenMP
- **free software** (*MIT License*)
  - 1.0 release in spring 2013: static global community detection
  - 2.0 release coming in fall 2013

[http://parco.iti.kit.edu/software/networkit.shtml]

Thank you for your attention

Acknowledgements
This work was partially funded by *MWK Baden-Württemberg*

Appendix

Karlsruhe Institute of Technology

Parallel Louvain Method **PLM**

- greedy modularity maximization
  - local moves
  - multi-level coarsening
- parallelization requires locking of global data structures

```
initialize to singletons
repeat
    while communities not stable do
        parallel for v ∈ V
            move v to neighbor
            community for max.
            modularity improvement
        endfor
    end
    coarsen graph
until no change in communities
return communities induced by coarsest
graph
```

Strengths / Weaknesses

- \+ high community quality
- \- does not scale to billions of edges

[Blondel et al. 2008: *Fast unfolding of communities in large networks*]

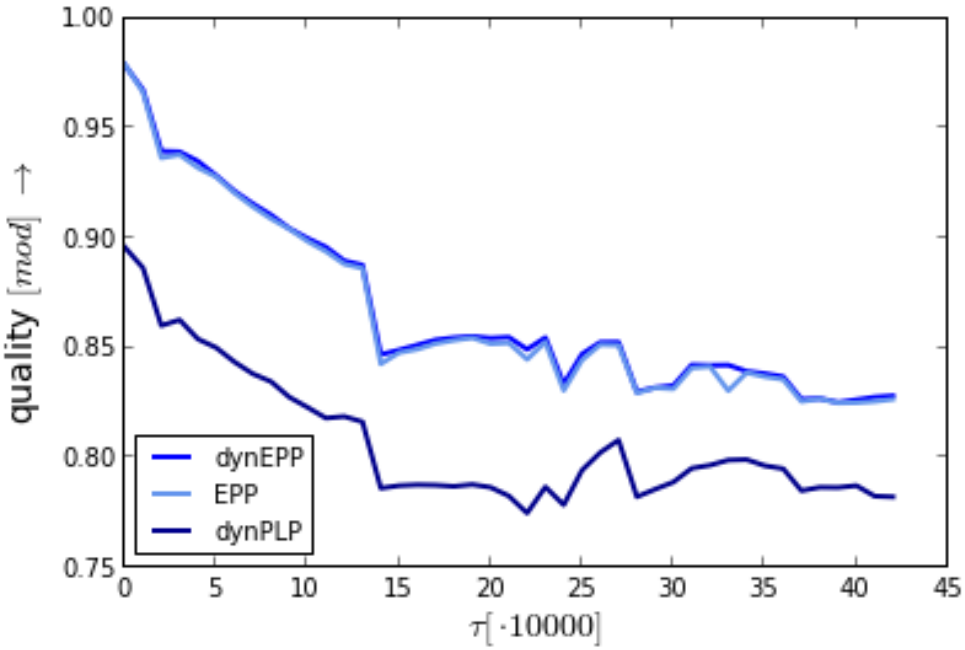
Data Sources

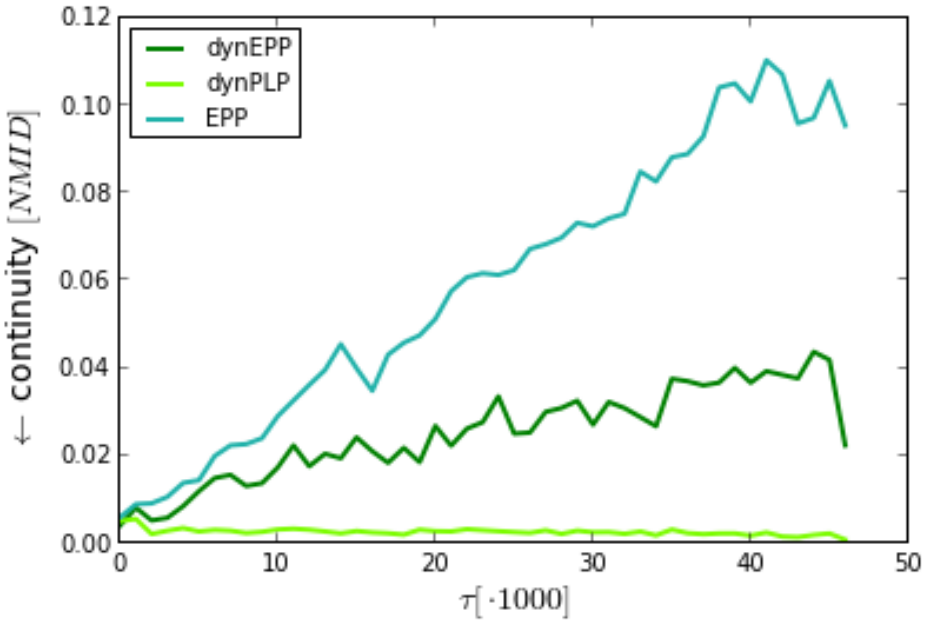
including *arxiv.org* dynamic collaboration network

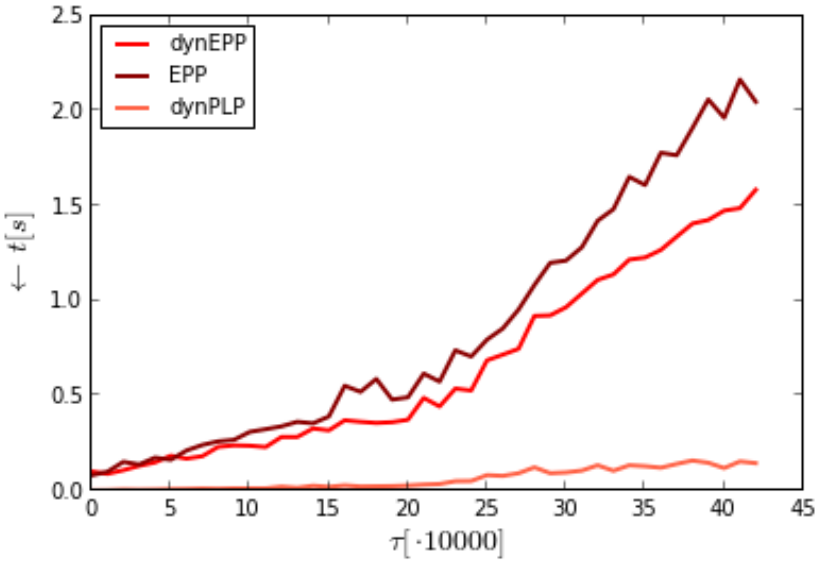
- extracted by crawler [available on parco.iti.kit.edu/software]
- coauthorship relations as
  - author-author graph
  - paper-paper graph

Evaluation dynEPP

- dynEPP **quality** matches EPP

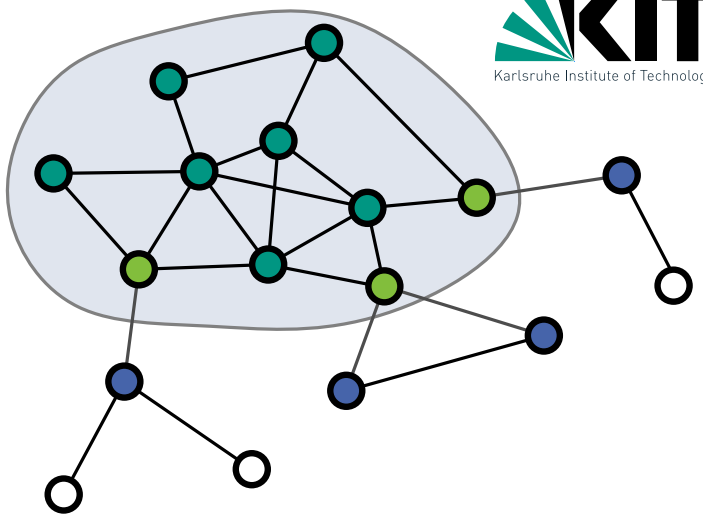






- **continuity**: middle ground between **dynPLP** and **EPP**
- **running time** dominated by **PLM**

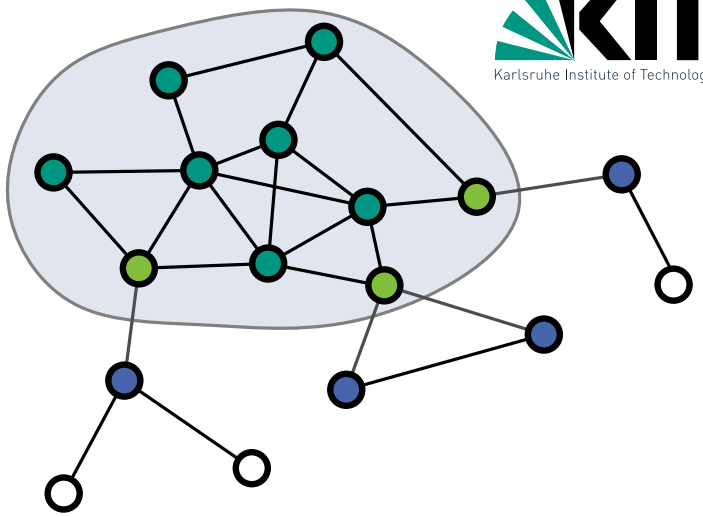
Measuring Community Quality



Structure of a Community

- **core** $K(C) := \{u \in C : \forall\{u, v\} \in E : v \in C\}$
- **boundary** $B(C) := \{u \in C : \exists\{u, v\} \in E : v \notin C\}$
- **shell** $\Omega(C) := \{u \notin C : \exists\{u, v\} \in E : v \in C\}$

Measuring Community Quality



Structure of a Community

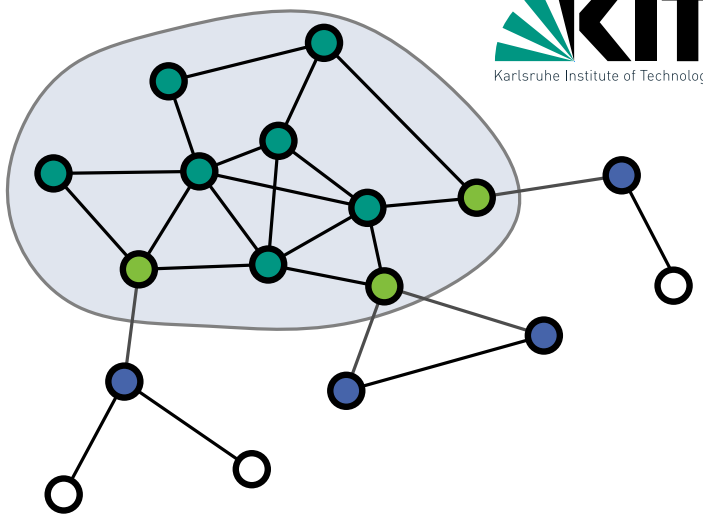
- **core** $K(C) := \{u \in C : \forall \{u, v\} \in E : v \in C\}$
- **boundary** $B(C) := \{u \in C : \exists \{u, v\} \in E : v \notin C\}$
- **shell** $\Omega(C) := \{u \notin C : \exists \{u, v\} \in E : v \in C\}$

Quality Measures

- **Conductance** [Andersen, Lang 2006]

$$\Phi(\zeta) = \underbrace{\frac{|E_{ext}(C)|}{vol(C)}}_{vol(C) \ll vol(V \setminus C)} \qquad [0, 1] \downarrow$$

Measuring Community Quality



Structure of a Community

- **core** $K(C) := \{u \in C : \forall \{u, v\} \in E : v \in C\}$
- **boundary** $B(C) := \{u \in C : \exists \{u, v\} \in E : v \notin C\}$
- **shell** $\Omega(C) := \{u \notin C : \exists \{u, v\} \in E : v \in C\}$

Quality Measures

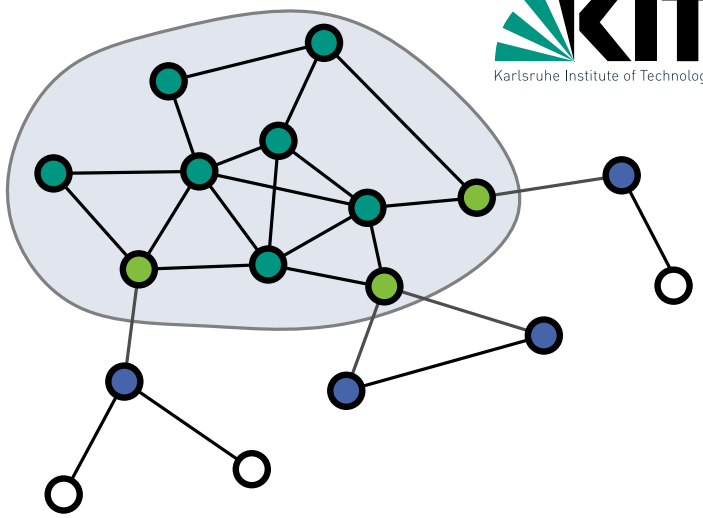
- **Conductance** [Andersen, Lang 2006]

$$\Phi(\zeta) = \underbrace{\frac{|E_{ext}(C)|}{vol(C)}}_{vol(C) \ll vol(V \setminus C)} \qquad [0, 1] \downarrow$$

- **"Local Modularity" M** [Luo et. al 2008]

$$M(C) := \frac{|E_{int}(C)|}{|E_{ext}(C)|} \qquad [0, \infty) \uparrow$$

Measuring Community Quality



Structure of a Community

- **core** $K(C) := \{u \in C : \forall \{u, v\} \in E : v \in C\}$
- **boundary** $B(C) := \{u \in C : \exists \{u, v\} \in E : v \notin C\}$
- **shell** $\Omega(C) := \{u \notin C : \exists \{u, v\} \in E : v \in C\}$

Quality Measures

- **Conductance** [Andersen, Lang 2006]

$$\Phi(\zeta) = \underbrace{\frac{|E_{ext}(C)|}{vol(C)}}_{vol(C) \ll vol(V \setminus C)} \qquad [0, 1] \downarrow$$

- **"Local Modularity" M** [Luo et. al 2008]

$$M(C) := \frac{|E_{int}(C)|}{|E_{ext}(C)|} \qquad [0, \infty) \uparrow$$

- **"Local Modularity" L** [Chen et al. 2009]

$$L_{int} := \frac{2 \cdot |E_{int}(C)|}{|C|} \quad L_{ext} := \frac{E_{ext}(B(C))}{|B(C)|} \quad L(C) := \frac{L_{int}}{L_{ext}} \qquad [0, \infty) \uparrow$$