Modularity-Driven Clustering of Dynamic Graphs

SEA 2010 - 9th International Symposium on Experimental Algorithms
Robert Görke, Pascal Maillard, Christian Staudt, Dorothea Wagner | May 22, 2010
Structure

1. Basics
2. Static Algorithms
3. Dynamic Algorithms
4. Results
Overview

Modularity-Driven Clustering of Dynamic Graphs

In this work we

- pioneer **online dynamic modularity maximization**
- develop dynamizations of
  - the currently best heuristic algorithms
  - an optimal algorithm
- evaluate them on
  - dynamic clustered random graphs
  - dynamic real-world networks
- give sound recommendations for the choice of an algorithm
Overview

Modularity-Driven Clustering of Dynamic Graphs

In this work we

- pioneer **online dynamic modularity maximization**
- develop dynamizations of
  - the currently best heuristic algorithms
  - an optimal algorithm
- evaluate them on
  - dynamic clustered random graphs
  - dynamic real-world networks
- give sound recommendations for the choice of an algorithm
Overview

Modularity-Driven Clustering of Dynamic Graphs

In this work we

- pioneer **online dynamic modularity maximization**
- develop dynamizations of
  - the currently best heuristic algorithms
  - an optimal algorithm
- evaluate them on
  - dynamic clustered random graphs
  - dynamic real-world networks
- give sound recommendations for the choice of an algorithm
Modularity-Driven Clustering of Dynamic Graphs

In this work we

- pioneer **online dynamic modularity maximization**
- develop dynamizations of
  - the currently best heuristic algorithms
  - an optimal algorithm
- evaluate them on
  - dynamic clustered random graphs
  - dynamic real-world networks
- give sound recommendations for the choice of an algorithm
Clustering: Intuition and Formalization

Task
partition a graph into natural groups

Paradigm
intra-cluster density vs. inter-cluster sparsity

Formalization
a variety of clustering quality indices
Quality: Coverage & Modularity

- **Coverage**: fraction of intra-cluster edges
  - $\text{cov}(G, C) \in [0, 1]$

**Definition (Coverage)**

$$\text{cov}(G, C) := \sum_{C \in \mathcal{C}} \frac{|E(C)|}{|E|}$$

- **Modularity**: Coverage minus expected coverage
  - $\text{mod}(G, C) \in [-1, 1]$ [GIRVAN, NEWMAN 2004]

**Definition (Modularity)**

$$\text{mod}(G, C) := \text{cov}(G, C(G)) - \mathbb{E}[\text{cov}(G, C(G))]$$
Dynamic Graph Clustering

Dynamic Instances
changing networks with evolving group structure

⇓

Dynamic Approach
update previous clustering reacting to changes in the graph

\[ G \xrightarrow{\Delta} G' \]
\[ T \downarrow \quad \mathcal{A} \quad \downarrow T \]
\[ \mathcal{C}(G) \xrightarrow{\mathcal{A}} \mathcal{C}'(G') \]

Clustering update problem

Criteria
- speed
- quality
- smooth transitions
Dynamic Graph Clustering

Dynamic Instances
changing networks with evolving group structure

⇓

Dynamic Approach
update previous clustering reacting to changes in the graph

\[ G \xrightarrow{\Delta} G' \]
\[ \mathcal{T} \downarrow \quad \mathcal{T} \downarrow \]
\[ \mathcal{C}(G) \xrightarrow{\mathcal{A}} \mathcal{C'}(G') \]

Criteria
- speed
- quality
- smooth transitions
Theorem

\textbf{ModOpt} is \( \mathcal{NP} \)-hard \cite{BRANDES et al. 2008}

\[ \downarrow \]

Corollary

\textbf{DynModOpt} is \( \mathcal{NP} \)-hard
Heuristics: Global

[NEWMAN et al. 2004]

- globally greedy
- cluster agglomeration

Algorithm 1: Global

1. repeat
2. compute all $\Delta_{mod}(C_i, C_j)$
3. merge best cluster pair
4. until $\max. \Delta_{mod} \leq 0$
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering
Illustration: Global

Dendrogram

Current Clustering

Robert Görke, Pascal Maillard, Christian Staudt, Dorothea Wagner – MDCoDG
May 22, 2010
Illustration: Global

Dendrogram

Current Clustering
Heuristics: Global

[Newman et al. 2004]

- globally greedy
- cluster agglomeration

Algorithm 1: Global

1. repeat
2. compute all $\Delta_{mod}(C_i, C_j)$
3. merge best cluster pair
4. until max. $\Delta_{mod} \leq 0$

Pseudo-Dynamic Algorithm

sGlobal: start Global from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change

[BLONDEL et al. 2008]
Heuristics: Local

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]
- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

\texttt{sLocal}: start \texttt{Local} from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDE et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDELL et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]
- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change
Heuristics: Local

[Blondel et al. 2008]

- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change
Heuristics: Local

[BLOMDEL et al. 2008]
- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm
sLocal: start Local from scratch after each change
Heuristics: Local

[BLONDEL et al. 2008]
- locally greedy
- node shifts
- hierarchical contractions

Pseudo-Dynamic Algorithm

sLocal: start Local from scratch after each change
Optimization: ILP approach

1. introduce decision variables

∀\{u, v\} ∈ \binom{V}{2}: X_{uv} = \begin{cases} 0 & \text{if } C(u) = C(v) \\ 1 & \text{otherwise} \end{cases}

2. ensure valid clustering with constraints (transitivity):

∀\{u, v, w\} ∈ \binom{V}{3}: \begin{cases} X_{uv} + X_{vw} - X_{uw} ≥ 0 \\ X_{uv} + X_{uw} - X_{vw} ≥ 0 \\ X_{uw} + X_{vw} - X_{uv} ≥ 0 \end{cases}
Optimization: ILP approach

3 optimize target function:

\[
\text{mod}_{\text{ILP}}(G, C_G) = \sum_{\{u,v\} \in \binom{V}{2}} \left( \omega(u, v) - \frac{\omega(u) \cdot \omega(v)}{2 \cdot \omega(E)} \right) \cdot X_{uv}
\]

[BRANDES et. al 2008]

Pseudo-Dynamic Algorithm

sILP: start ILP from scratch after each change. Infeasible!
Optimization: ILP approach

3. Optimize target function:

$$\text{mod}_{\text{ILP}}(G, C_G) = \sum_{\{u, v\} \in \binom{V}{2}} \left( \omega(u, v) - \frac{\omega(u) \cdot \omega(v)}{2 \cdot \omega(E)} \right) \cdot X_{uv}$$

[BRANDES et. al 2008]

Pseudo-Dynamic Algorithm

sILP: start ILP from scratch after each change. Infeasible!
locality assumption
Prep Strategies

locality assumption
Prep Strategies
locality assumption
Prep Strategies: Concept

prep strategy $S$
- reacts to changes
- prepares half-finished preclustering $\tilde{C}$
- passes $\tilde{C}$ on to algorithm
Prep Strategy

BU: Break up the affected clusters entirely
Prep Strategy

**BU: Break up the affected clusters entirely**
Prep Strategy

N: free a neighborhood up to a depth $d$ with BFS

Experiments: What is the best choice for $d / k$?
Prep Strategy

N: free a *neighborhood* up to a depth $d$ with BFS

Experiments: What is the best choice for $d / k$?
Prep Strategy

**N**: free a neighborhood up to a depth $d$ with **BFS**

Experiments: What is the best choice for $d / k$?
Prep Strategy

**BN**: free a *bounded neighborhood* of up to $k$ of nodes with **BFS**

Experiments: What is the best choice for $d/k$?
Prep Strategies: N/BN

**Prep Strategy**

**BN**: free a **bounded neighborhood** of up to $k$ of nodes with **BFS**

**Experiments**: What is the best choice for $d / k$?
Prep Strategy BT: Illustration

Prep Strategy

BT: Backtrack Global’s mergers according to heuristic rules
Prep Strategy BT: Illustration

**dendrogram**

**current clustering**

**BT:** Backtrack Global’s mergers according to heuristic rules
Prep Strategy BT: Illustration

Prep Strategy

**BT: Backtrack Global**’s mergers according to heuristic rules
Dynamic Heuristics: dGlobal

1. take **preclustering** defined by **prep strategy**
2. run **Global** to complete

Dynamic Algorithm

dGlobal: Global with $\tilde{C}$ as search space
Dynamic Heuristics: dGlobal

1. take preclustering defined by prep strategy
2. run Global to complete

Dynamic Algorithm

dGlobal: Global with Ĉ as search space
Illustration: dLocal
Dynamic Heuristics: dLocal

1. extract some nodes defined by prep strategy from supernodes
2. run Local to complete

Dynamic Algorithm

dLocal: Local with ˜C as search space
Dynamic Heuristics: dLocal

1. extract some nodes defined by prep strategy from supernodes
2. run Local to complete

Dynamic Algorithm

dLocal: Local with $\tilde{\mathcal{C}}$ as search space
Dynamic Algorithms: dILP

1. construct smaller instance of the ILP using the preclustering
2. solve ILP

Dynamic Algorithm

dILP: ILP with $\tilde{C}$ as search space
Dynamic Algorithms: dILP

1. construct smaller instance of the ILP using the preclustering
2. solve ILP

Dynamic Algorithm

dILP: ILP with $\tilde{C}$ as search space
Dynamic Graph Instances

- generator for dynamic clustered random graphs
  [GÖRKE, STAUDT 2009]

- e-mail graph of KIT CompSci

- arXiv collaboration graph
Dynamics vs Statics: Speed

Small search spaces work best!

Result

Dynamics run faster
Dynamics vs Statics: Speed

Running time [ms] per time step

- **sLocal**
- **sGlobal**
- **dGlobal@BN_{16}**
- **dLocal@BN_{4}**
- **dGlobal@BT**

Small search spaces work best!

**Result**

Dynamics run faster
Dynamics vs Statics: Transitions

Result

Dynamics yield smoother clustering transitions
Dynamics vs Statics: Transitions

Result
Dynamics yield smoother clustering transitions
Dynamics often surpass static counterparts in terms of quality.
Dynamics vs Statics: Quality

Result

Dynamics often surpass static counterparts in terms of quality
Heuristics vs dILP: Quality

Result

Local optimality is worse than dynamic heuristics.
Heuristics vs dILP: Quality

quality [modularity] per time step; e-mail graph

Result
Local optimality is worse than dynamic heuristics.
Conclusion

- dynamic versions of state-of-the-art heuristics
  - evaluation: dynamics yield better
    - speed
    - quality
    - smooth transitions
  - local changes call for local updates
  - local optimality does not help
- recommendations for the choice of an algorithm

Thank you for your attention!
Conclusion

- dynamic versions of state-of-the-art heuristics

- evaluation: dynamics yield better
  - speed
  - quality
  - smooth transitions

- local changes call for local updates

- local optimality does not help

- recommendations for the choice of an algorithm

Thank you for your attention!
Conclusion

- dynamic versions of state-of-the-art heuristics
- evaluation: dynamics yield better
  - speed
  - quality
  - smooth transitions
- local changes call for local updates
- local optimality does not help
- recommendations for the choice of an algorithm

Thank you for your attention!
Conclusion

- dynamic versions of state-of-the-art heuristics

- evaluation: dynamics yield better
  - speed
  - quality
  - smooth transitions

- local changes call for local updates

- local optimality does not help

- recommendations for the choice of an algorithm

Thank you for your attention!
Conclusion

- dynamic versions of state-of-the-art heuristics
- evaluation: dynamics yield better
  - speed
  - quality
  - smooth transitions
- local changes call for local updates
- local optimality does not help
- recommendations for the choice of an algorithm

Thank you for your attention!
Conclusion

- dynamic versions of state-of-the-art heuristics
- evaluation: dynamics yield better
  - speed
  - quality
  - smooth transitions
- local changes call for local updates
- local optimality does not help
- recommendations for the choice of an algorithm

Thank you for your attention!