Algorithm Engineering Applied To Graph Clustering
Insights and Open Questions in Designing Experimental Evaluations

Marco Gaertler

Faculty of Informatics
Universität Karlsruhe (TH)

Workshop on Communities in Networks
14. March, 2008 – Louvain-la-Neuve
1 Motivation

2 Framework?

3 Demonstration

4 Conclusion
## What is Graph Clustering?

Jain et al. – Data Clustering: A Review

Clustering is the unsupervised classification of patterns into groups.

---

Clustering is the mathematical study of methods for recognizing natural groups within a class of entities.
What is Graph Clustering?

**Jain et al. – Data Clustering: A Review**
Clustering is the unsupervised classification of patterns into groups.

**van Dongen – Graph Clustering by Flow Simulation**
Cluster Analysis is the mathematical study of methods for recognizing natural groups within a class of entities.
What is Graph Clustering?

Jain et al. – Data Clustering: A Review
Cluster Analysis is the unsupervised classification of patterns into groups.

van Dongen – Graph Clustering by Flow Simulation
Cluster Analysis is the mathematical study of methods for recognizing natural groups within a class of entities.

What are interesting patterns or natural groups?
classification
classification

partitioning

Universität Karlsruhe (TH)
Faculty of Informatics

Gaertler

AE in Clustering

http://i11www.iti.uka.de
Questions:

- What are suitable models / paradigms for clusterings?
  → formalization of clustering / quantification of quality

- How can we objectively evaluate clustering algorithms?
  → theoretical guarantees versus experimental validation
Questions:

- What are suitable models / paradigms for clusterings?
  \[\rightarrow\] formalization of clustering / quantification of quality

- How can we objectively evaluate clustering algorithms?
  \[\rightarrow\] theoretical guarantees versus experimental validation
1. Motivation

2. Framework?

3. Demonstration

4. Conclusion
Task

experimental evaluation of clustering algorithms

testbed:

application-oriented:
- large relevance
- not always available

generated data:
- easy to produce
- need not be realistic
Setup for Statistical Evaluation

- generate data
- algorithms
- quality assessment
- statistical summary
Setup for Statistical Evaluation

- generate data
- algorithms
- quality assessment
- statistical summary

**generator:** random graph model with clustering information
Setup for Statistical Evaluation

- **generator**: random graph model with clustering information
- **algorithms**: sets of technique to test
Setup for Statistical Evaluation

**generator**: random graph model with clustering information

**algorithms**: sets of technique to test

**quality**: quantification of achieved quality

- generate data
- algorithms
- quality assessment
- statistical summary
Setup for Statistical Evaluation

- **generator:** random graph model with clustering information
- **algorithms:** sets of technique to test
- **quality:** quantification of achieved quality
- **summary:** statements of average behavior

Diagram:
- **generate data**
  - **algorithms**
  - **quality assessment**
  - **statistical summary**
Advantages & Disadvantages

advantages:

- easy to setup and perform
- “average”-case analysis
- benchmark-like behavior
  - reproducible (without having the implementation)
  - comparable with former/future evaluations

disadvantages:

- worst cases can be arbitrary bad
- hidden dependencies between generator, algorithms and quality measures can lead to wrong conclusions
Motivation

Framework?

Demonstration

Conclusion

Detail Setup

- generator
  - parameters

- algorithms
  - parameters
  - model

- evaluation

- clustering model implicitly affects
  - generator
  - algorithms
  - quality evaluation

Universität Karlsruhe (TH)
Faculty of Informatics
Gaertler
AE in Clustering
Algorithmics
http://i11www.iti.uka.de
Questions:

1. How severe are the (hidden) dependencies between generator, algorithms and quality measures?

2. What kind of graph generator do we need?
Hidden Dependencies

setting:

- (uniform) random graph
- two arbitrary clustering algorithms
setting:
- (uniform) random graph with random equi-partition
- two arbitrary clustering algorithms

observations:
- both algorithms perform fairly good
setting:
- (uniform) random graph with random equi-partition
- two arbitrary clustering algorithms

observations:
- both algorithms perform fairly good
- or not?!
\[ G(n, p) \] does not generally have a clustering structure (for large \( p \)).

- **Coverage** of \( \approx 0.5 \) means half of the edges are inside clusters → not very good for dense graphs.
- **Modularity** of \( \approx 0 \) means the clustering structure is not significant (compared to random rewiring).

→ due to the 'structure' of the generator and the selected evaluation mechanism this outcome was to be expected.
Generators

What are suitable graph generators?

preferred properties:

- efficient computation
- direct correspondence to a clustering model/paradigm
- parameters control the significance of the clustering

Do we need an associated clustering?

Yes/No, but it serves as indicator for the clusterability of the generated graph
If we come up with a generator, how do we know that it is suitable for evaluation?
If we come up with a generator, how do we know that it is suitable for evaluation?

A generator is suitable, if the quality of the generated clustering is acceptable (and comparable to that of suitable algorithms).
If we come up with a generator, how do we know that it is suitable for evaluation?

A generator is suitable, if the quality of the generated clustering is acceptable (and comparable to that of suitable algorithms).

more generally:

How to design suitable components for (statistical) experimental evaluation?

Every combination of two suitable components can be used to evaluated the missing third one.
How to break this cyclic dependency?

bad news: no chance, all components formalize the model

good news: can break the dependencies into smaller/easier blocks
→ concept of unit tests
**general:**
a simple rule describing a behavioral pattern of a component

**example for generators:**
as the level of perturbation increases (modeled by parameters) the coverage of the clustering should not increase

\[
\text{coverage} = \frac{\# \text{ intra-cluster edges}}{\# \text{edges}}
\]
desired outcome:

- basic rules for general behavioral patterns
- advanced rules building on tested components
- application-specific requirements as constraints
Outline

1. Motivation
2. Framework?
3. Demonstration
4. Conclusion
Random Clustered Graph Generator

classification model

generator:
- create $n$ nodes
- partition nodes in $k$ clusters
- create edges inside of clusters with probability $p_{\text{in}}$
- create edges between clusters with probability $p_{\text{out}}$

overall:
- random graph with clustering structure
- significance of clustering depends on probabilities $p_{\text{in}}$ and $p_{\text{out}}$
Example: $\mathcal{G}((12, 18, 13, 18, 20, 13, 10), 0.85, 0.01)$

Node partitioning in 7 clusters
Example: $G((12, 18, 13, 18, 20, 13, 10), 0.85, 0.01)$

- node partitioning in 7 clusters
- intra-cluster edges
Example: $G((12, 18, 13, 18, 20, 13, 10), 0.85, 0.01)$

- node partitioning in 7 clusters
- intra-cluster edges
- inter-cluster edges
Validation via Basic Unit Test

- Increases in perturbation implies non-increase in coverage: ✓
- Increases in perturbation implies non-increase in modularity: ✓

Overall measuring perturbation vs. quality: ✓
Validation via Basic Unit Test

- increases in perturbation implies non-increase in coverage: ✓
- increases in perturbation implies non-increase in modularity: ✓

overall measuring perturbation vs. quality: ✓
ratio of intra- and inter-cluster edges depends on $p_{\text{in}}, p_{\text{out}}$ and $n$
Scalability?

questions:

- what is the dependency of number and size of clusters and $n$?
- should the ratio (intra- vs. inter-cluster edges) be independent of $n$?
- what about other properties? quality?
choosing $p_{out}$ individually according to $k$, $p_{in}$ and the ratio of (expected) intra- versus inter-cluster edges
choosing $p_{\text{out}}$ individually according to $k$, $p_{\text{in}}$ and the ratio of (expected) intra- versus inter-cluster edges

\[ n = 100 \]  
\[ n = 1000 \]
Comparing Algorithms

comparing coverage:

generator

greedy optimization

iterative pruning
Comparing Algorithms

comparing modularity:

generator

greedy optimization

iterative pruning
Observations

greedy optimization:
- general good performances (wrt. generator)
- minor artifacts for very very sparse graphs

iterative pruning:
- good performance for small-perturbed instances
- artifacts for sparse graphs ($p_{in} \leq 0.2$)
Outline

1. Motivation
2. Framework?
3. Demonstration
4. Conclusion
Summary

experimental evaluation:

- good and flexible mean for average-case analysis
- easy to reproduce and compare with each other
- implicit assumption of the model can have a large impact
- not all combination of generators, algorithms and quality measures makes sense
- designing and evaluating good components is non-trivial
concept of unit test:

- engineering approach to systematic evaluations
- formalization of rules of thumb
- easy integration of application specifics
- knowledge of basic building blocks required (e.g. formalization of model as quality index)
- results target only “average” cases
concept of unit test:
- engineering approach to systematic evaluations
- formalization of rules of thumb
- easy integration of application specifics
- knowledge of basic building blocks required (e.g. formalization of model as quality index)
- results target only “average” cases

Thank you!