How to Evaluate Clustering Techniques*

Daniel Delling[1], Marco Gaertler[1], Robert Görke[1], Zoran Nikoloski[2], and Dorothea Wagner[1]

[1] Faculty of Informatics, Universität Karlsruhe (TH),
{delling,gaertler,rgoerke,wagner}@informatik.uni-karlsruhe.de
[2] Department of Applied Mathematics, Charles University, Prague nikoloski@kam.mff.cuni.cz

Abstract. The quality of clustering algorithms is often based on their performance according to a specific quality index, in an experimental evaluation. Experiments either use a limited number of real-world instances or synthetic data. While real-world data is crucial for testing such algorithms, it is scarcely available and thus insufficient. Therefore, synthetic pre-clustered data has to be assembled as a test bed by a generator. Evaluating clustering techniques on the basis of synthetic data is highly non trivial. Even worse, we reveal several hidden dependencies between algorithms, indices, and generators that potentially lead to counterintuitive results. In order to cope with these dependencies, we present a framework for testing based on the concept of unit-tests. Moreover, we show the feasibility and the advantages of our approach in an experimental evaluation.

1 Introduction

Clustering is a crucial graph-theoretic problem in data and, more specifically, network analysis. It has been studied for decades and applied in many settings such as data mining [1], network analysis [2], biochemistry [3] and social studies [4]. Due to the large variety of these applications, the optimization criteria differ significantly. However, most applications share aspects of the underlying clustering model, i. e., the required intrinsic properties of clusters or the type of connection between clusters. We focus on the widely used paradigm of intra-cluster density versus inter-cluster sparsity [5,6,7].

All clustering techniques that have been introduced and studied in the literature suffer from drawbacks or artifical behavior; nevertheless, they are often applied. For example, quality indices are used as optimization criteria for finding good clusterings [8] or for quality assessment in general. A frequently given justification is that only very specific circumstances cause these drawbacks and artifacts and that such cases seldom occur in real-world applications. Furthermore, their applicability has been evaluated both theoretically [5,6,9,10] and experimentally [11,12,13,14] in the past. Although, theoretical examinations provide general characterizations, they are often hard to perform or tend to be highly specialized. On the other hand, the feasibility of experimental evaluations heavily depends on efficient implementations. However, to our knowledge, no conclusive framework for experimental evaluation has been investigated.

Our goal is to provide an experimental setup for benchmarking and evaluating clustering algorithms, quality indices, generators, and other clustering-related concepts. The main objective is to provide a *unit-test*. The concept of unit-tests was originally introduced in the field of software engineering and programming as an independent code module that ensures the correct functionality of a component. Such a test ensures that the associated methods of a data structure operate properly. They are frequently used when the implementation of a component is changed due to optimization, yet the functionality should remain. In our case, the provided experiments indicate the usability of a clustering technique. Similar to the tests in software engineering, our tests are only indicators, i. e., a meaningless technique can still successfully pass all tests, while a failed test reveals its impracticality. In addition, the results of the tests themselves can be used to compare techniques and deepen the understanding.

*This work was partially supported by the DFG under grant WA 654/14-3 and EU under grant DELIS (contract no. 001907) and CREEN (contract no. 012864) and grant MSM0021620838.

An essential part of the experimental setup is the large availability of pre-clustered graphs, i. e., graphs with a significant clustering. Since testing algorithms is one of our major interests, quality indices are required to measure the achieved quality of their output. The concepts in consideration, which are quality indices, clustering algorithms, and generators, are widely used in practice. We selected them based on our experience in this field which we gained by previous studies (see for example [14]).

This paper is organized as follows. Section 2 shortly introduces clustering algorithms, quality indices, and generators for pre-clustered graphs. Mutual dependencies and hidden pitfalls are revealed in Section 3. Our testing framework is introduced in Section 4, while Section 5 shows the feasibility of our approach. Section 6 concludes our work with a small discussion.

2 Preliminaries

We assume that G is an undirected and unweighted graph with n nodes and m edges. A partitioning of the nodes into several *clusters* C is called a *clustering* $\mathcal{C}$ of a graph. If each cluster is represented by only one node we speak of *singletons*; conversely, a clustering with only one cluster is called the *1-clustering*. The edges between nodes of the same cluster are called *intra-cluster edges* and those between nodes of different clusters *inter-cluster edges*.

In the following, we shortly introduce the three main clustering techniques. More precisely, we consider *indices* for measuring the quality of a given clustering, *generators* for creating graphs with a given clustering, and *algorithms* for finding clusterings in given graphs. For more details on the introduced techniques see [6,7,8,15].

2.1 Indices

Two basic indices *coverage* and *performance*, are based on counting edges. While coverage is the ratio of intra-cluster edges to m, performance normalizes the number of correctly classified pairs of nodes. Correct pairs are connected of pairs of nodes that are in the same cluster and unconnected pairs of nodes which are in different clusters. *Intra-* and *inter-cluster conductance* are two indices founded on the concept of bottlenecks, i. e., a cluster should not have a sparse cut separating non-trivial parts and in contrast, the connection of a cluster to the remaining graph should be sparse. To measure sparse cuts, we use *conductance* from random walk theory. Both intra- and inter-cluster conductance aggregate the values of the cut-measure applied to the individual clusters. In this case, the aggregation is just the maximum, therefore they are called *worst-case indices*. Similarly, we can define an average- or best-case index. An index that also incorporates statistical properties is *modularity* [8]. It measures the trade-off between the coverage of the clustering and the expected coverage when edges are rewired randomly and only the expected degree of a node remains fixed. Formal definitions can be found in [7].

As a new quality index for clusterings we introduce *density*. The intention of density is to directly formalize the paradigm of intra-cluster density versus inter-cluster sparsity.

$$\text{density}(\mathcal{C}) := \underbrace{\frac{1}{2}\left(\frac{1}{|\mathcal{C}|}\sum_{C\in\mathcal{C}}\frac{\#\text{ intra-cluster edges of } C}{\binom{|C|}{2}}\right)}_{\text{intra-cluster density}} + \underbrace{\frac{1}{2}\left(1-\frac{\#\text{ inter-cluster edges}}{\binom{n}{2}-\sum_{C\in\mathcal{C}}\binom{|C|}{2}}\right)}_{\text{inter-cluster sparsity}}$$

We use the convention, that the fraction in the term intra-cluster density for a singleton is defined to be 1, and that the inter-cluster sparsity of the 1-clustering is also 1. Note that all presented indices favor different ideal situations and generally agree only when the graph consists of disjoint and complete subgraphs.

2.2 Pre-Clustered Graph Generators

The *random pre-clustered graph generator* uses an integer array representing the cluster sizes and two probabilities p_{in} and p_{out} for the existence of intra-cluster edges and inter-cluster edges as the input. The graph is created by first assigning nodes randomly into clusters with respect to the given clusters' sizes and then inserting an edge between pairs of nodes in the same cluster with probability p_{in} and between other pairs with probability p_{out}. In other words, the generator extends the random graph model $\mathcal{G}(n, p)$ by a group structure.

The *Gaussian generator* is a restriction of the random pre-clustered graph generator using a Gaussian distribution for the cluster sizes. Due to the random selection of cluster sizes, the number of nodes is only approximated. One can insert additional nodes or delete nodes in order to obtain exactly n nodes, however this can introduce artifacts which affect quality indices. This phenomenon was observed by us for the index inter-cluster conductance in [14], which reacted heavily to the frequent emergence of very small clusters.

The *significant Gaussian generator* is a refinement of the Gaussian generator. For increasing n and a fixed pair $(p_{\text{in}}, p_{\text{out}})$ the growth of inter-cluster edges exceeds the growth of intra-cluster edges. Thus, the parameter p_{out} is substituted by the parameter ρ defined as the ratio of expected inter-cluster edges and expected intra-cluster edges and given in Equation (1), where k is the number of clusters.

$$\rho = \frac{p_{\text{out}}(n - n/k)}{p_{\text{in}}(n/k - 1)} \tag{1}$$

An experimental evaluation of the differences between the Gaussian generator and the significant Gaussian generator can be found in [15].

We have also introduced the *attractor generator* which uses geometric properties based on Voronoi diagrams to generate significant clusterings [15]. The generator first calculates a Voronoi diagram using few randomly placed nodes that act as Voronoi centers. The remaining nodes are randomly inserted in the plane and are connected to their Voronoi center node. Further edges are then inserted where the maximum Euclidean distance of two nodes being connected is determined by the perturbation parameter. We observed that the introduced perturbation parameter f is highly dependent on the number of nodes. Therefore, we modify this parameter in order to take the average node distance into account. The new perturbation parameter ρ is defined as follows. Instead of connecting all nodes with a distance less than $f \cdot \sqrt{2}/100$ we connect all nodes with a distance less than $\rho \cdot \sqrt{(\pi k)/2}$. The parameter k is the number of clusters, which is picked uniformly at random from the interval $[\log_{10}(n), \sqrt{n}]$ during the generation of the Voronoi diagram.

2.3 Clustering Algorithms

In the following, we briefly sketch two established algorithms that we used for our unit-test, namely Markov Clustering and a greedy algorithm for modularity.

The key intuition behind *Markov Clustering* (MCL) [6] is that a "random walk that visits a dense cluster will not likely leave the cluster until many of its vertices have been visited." Rather than actually simulating random walks, the MCL algorithm iteratively modifies a matrix of transition probabilities.

The modularity greedy algorithm [8] starts from singletons merging those clusters which leads to the highest increase in modularity. The algorithm stops if no further improvement of modularity is possible.

3 Dependencies

In the following, we reveal dependencies between the techniques introduced above. On the one hand, we deal with the dual nature of algorithms and generators. On the other hand, we point out circular dependencies between all three concepts.

3.1 Duality

Consider the simplified view where clustering algorithms, quality indices, and pre-clustered graph generators are all interpreted as mappings: a clustering algorithm assigns to each graph G and significance threshold τ a clustering $\mathcal{C}$ which has a significance score larger than or equal to τ; a quality index maps a pair consisting of a graph G and a clustering $\mathcal{C}$ to a significance score τ; a pre-clustered graph generator assigns to each clustering $\mathcal{C}$ and a significance score τ a graph G such that $\mathcal{C}$ has at least significance τ with respect to G. This view is simplified, since algorithms and generators usually do not use an explicit significance threshold as input. However, the input parameter for algorithms or generators can usually be tuned in order to obtain various levels of significance. For example, one can restrict the number of clusters or the number of edges to generate. The duality is reflected by the fact, that a quality index or a clustering algorithm can be used to define a generator and vice versa. Let index be an arbitrary quality index. Then a corresponding generator maps a partition $\mathcal{C}$ of the (node-)set V and a quality threshold to a graph, where the edgeset is chosen from $\left\{E \subseteq \binom{V}{2} \mid \mathsf{index}((V,E),\mathcal{C}) \geq \tau\right\}$. Note that a necessary condition is that for every partition of V there is an edgeset that has significance 1, since otherwise the set of possible edgesets can be empty for some τ's. Similarly, a generator $\mathsf{generator}$ defines an index by mapping the pair $(G,\mathcal{C})$ to that τ such that G has the maximum probability to be generated by $\mathsf{generator}(\mathcal{C},\tau)$. A necessary condition is that every graph is generated with positive probability for every clustering. Similar correspondings can be made for clustering techniques; however, we omit them, since this is only an illustrating example and other suitable realizations may exist.

3.2 Circular Dependencies

The generators presented in Section 2.2 are defined to be mostly independent from the quality indices, since they are solely based on the perturbation of disjoint cliques. Thus they can be used to evaluate quality indices. However, hidden dependencies may exist and, even more severely, their perturbation parameters may be counterintuitive. For example, for increasing values of n and a fixed pair of parameters $(p_{\text{in}}, p_{\text{out}})$ the growth of the share of the inter-cluster edges exceeds that of the intra-cluster edges when using the Gaussian generator. Thus, larger instances require smaller values of p_{out} in order to obtain graphs with similar degrees of perturbation. Some of these drawbacks can be fixed by intelligent adjustments of parameters, e. g., Gaussian generator versus significant Gaussian generator. On the other hand, one can exploit them to reveal drawbacks in other concepts such as quality indices or clustering algorithms.

Generators, quality indices, and clustering algorithms are different aspects of one and the same problem, i. e., formalizing the term *natural groups*. Thus, there is a threefold interdependence between these concepts that implies certain pitfalls for meaningful benchmarking. As a simple example, consider an algorithm based on minimum cuts. Its evaluation may not only rely on cut size in order to maintain comparability with other techniques. More general, a too strongly correlated selection of generator, algorithms, and quality indices will only imply the high usability of the techniques, while in the opposite case of un- or anti-correlated concepts, the obtained results are more or less random. Thus the design of meaningful benchmarks does not only require a thorough understanding of each individual component such as generators, quality indices, and clustering algorithms, but also of their mutual interdependencies. In principle,

there is a theoretical and an experimental approach to improve this understanding. Although theoretical examinations provide general characterizations, they are often hard to perform or tend to be highly specialized. On the other hand, the feasibility of experimental evaluations mainly depends on efficient implementations. Moreover, they often reveal additional intrinsic patterns and provide guidance for theoretical analyses. In the following, we provide a framework of unit-tests for evaluating clustering techniques as well experimental results.

4 Engineering Experiments

In general, our evaluation framework is based on the repeated execution of experiments with fixed parameters. Each experiment consists of the following three stages: (1) generation of preclustered graphs, (2) execution of clustering algorithms, and, finally, (3) evaluation of obtained clusterings using quality indices. Due to the randomness inherent in the generators for preclustered graphs, each experiment has to be executed until (statistical) significance has been achieved.

Regardless of the concept to test, a general strategy would be to start with very basic and intuitive instances and then gradually increase the complexity of the test methods. In the following, we briefly sketch such a strategy for generators, algorithms, and quality indices. Although we formulate these strategies as unit-tests, no formal notion of passing such a test is given due to fact that there is no discrete transition between passing and failing. In fact, using specific thresholds implies further intrinsic dependencies.

4.1 Simple Strategies

The most basic quality index is *coverage*. Although it has drawbacks due to its simplicity, we can derive an initial unit-test for generators and clustering algorithms based on it.

Unit-Test 1 (Simple Generator/Algorithm) *For a fixed generator and a fixed number of nodes, an increase (decrease) in the perturbation must not cause an increase (decrease) in coverage of the clustering used by the generator or obtained with an algorithm.*

Suitable generators have to fulfill Unit-Test 1 as a necessary condition. However, for algorithms the situation is more complex. First, note that an algorithm may produce clusterings with a significantly different number of clusters as a result of varying the perturbation of the generator. This potentially requires a different interpretation of the results. On the other hand, coverage highly depends on the number of clusters, for example the 1–clustering always has maximum coverage. Thus a failed Unit-Test 1 does not necessarily imply a defect in the algorithm. We illustrate such a case in Section 4 in Figures 1(c) and 2(a) and discuss it in greater depth in Section 5.

Unit-Test 2 (Index) *For a fixed quality index and fixed cluster sizes, an increase (decrease) in the perturbation of the random pre-clustered graph generator must not cause an increase (decrease) in quality of the reference clustering.*

Unlike the previous unit-test, we have not experienced a violation of Unit-Test 2 for any indices considered here. However, special attention has to paid to worst-case quality indices that can have big jump discontinuities. For example, consider intra-cluster conductance, which rates star graphs with a maximum score of 1, but inserting an arbitrary edge in a star graph drastically decreases the scores.

The quality index coverage and the random pre-clustered graph generator are reliably simple and thus suffice as a first indicator. In order to reveal additional drawbacks of the concepts to be

tested and in order to reduce any potential correlations between the considered concepts in the experiments, each can be replaced by a related or more sophisticated one, such as modularity or geometric generators. On the one hand, this provides more meaningful insights, on the other hand, more complex artifacts can arise.

Besides replacing fundamental concepts in the basic unit-tests, each can be refined in order to focus on individual properties of generators, algorithms, or indices.

4.2 Advanced Strategies

Recalling our clustering paradigm of intra-cluster density and inter-cluster sparsity, an extension of the Unit-Tests 1 and 2 distinguishes between the two aspects of the paradigm. More precisely, quality indices should exhibit appropriate behavior with respect to both aspects.

Unit-Test 3 (Advanced Paradigm Incorporation) *Keeping inter-cluster sparsity roughly constant, for a fixed quality index and fixed cluster sizes, an increase (decrease) in the intra-cluster density of the random pre-clustered graph generator must not cause a decrease (increase) in quality of the reference clustering. Analogously, keeping intra-cluster density roughly constant, an increase (decrease) in the inter-cluster sparsity must not cause a decrease (increase) in quality of the reference clustering.*

The founding motivation is that quality indices should react accordingly to each parameter of the perturbation, ruling out one-sided dependencies. On the other hand, a failure of this unit-test can either imply such a defect or suggest a reduction of the parameter set. The evaluation of quality indices is a necessary foundation for benchmarking algorithms. In Unit-Test 4, we give the founding unit-test for algorithms.

Unit-Test 4 (Algorithm) *Let $\mathcal{G}$ be a generator passing Unit-Test 1 and* index *be an index passing Unit-Test 2. For small perturbation, the quality of the clustering calculated by an algorithm has to be close to the quality of the clustering used by the generator. Analogously, large perturbation must result in greater quality than the reference clustering of the generator.*

For small perturbation, the initial clustering used by the generator should be highly significant and thus it is very unlikely that a much better clustering exists. Therefore, a decent clustering algorithm has to find a clustering with similar quality. For large perturbation, the algorithm should find a better clustering due to the fact that the generated graph structure does not significantly represent the initial clustering. A failure of Unit-Test 4 for small perturbations indicates a potential defect in the algorithm. However, detecting the reason for a failed test for large perturbations is quite challenging. Potentially, each of the three components or yet unknown interdependencies may cause the failure. For example, if the perturbation exceeds a certain level, the generated graph likely contains no significant clustering and thus the algorithm cannot extract one. We detail a more specific example in Section 4. Similar to the duality of Unit-Test 1 and 2, Unit-Test 4 can be turned around in order to obtain a stronger test for generators.

Unit-Test 5 (Generator) *Let $\mathcal{A}$ be an algorithm passing Unit-Test 4 and* index *be the corresponding index used in that unit-test. The expected behavior of generators should be the following: The initial clustering used by a generator has to be at least as significant as the clustering calculated by $\mathcal{A}$ with respect to* index.

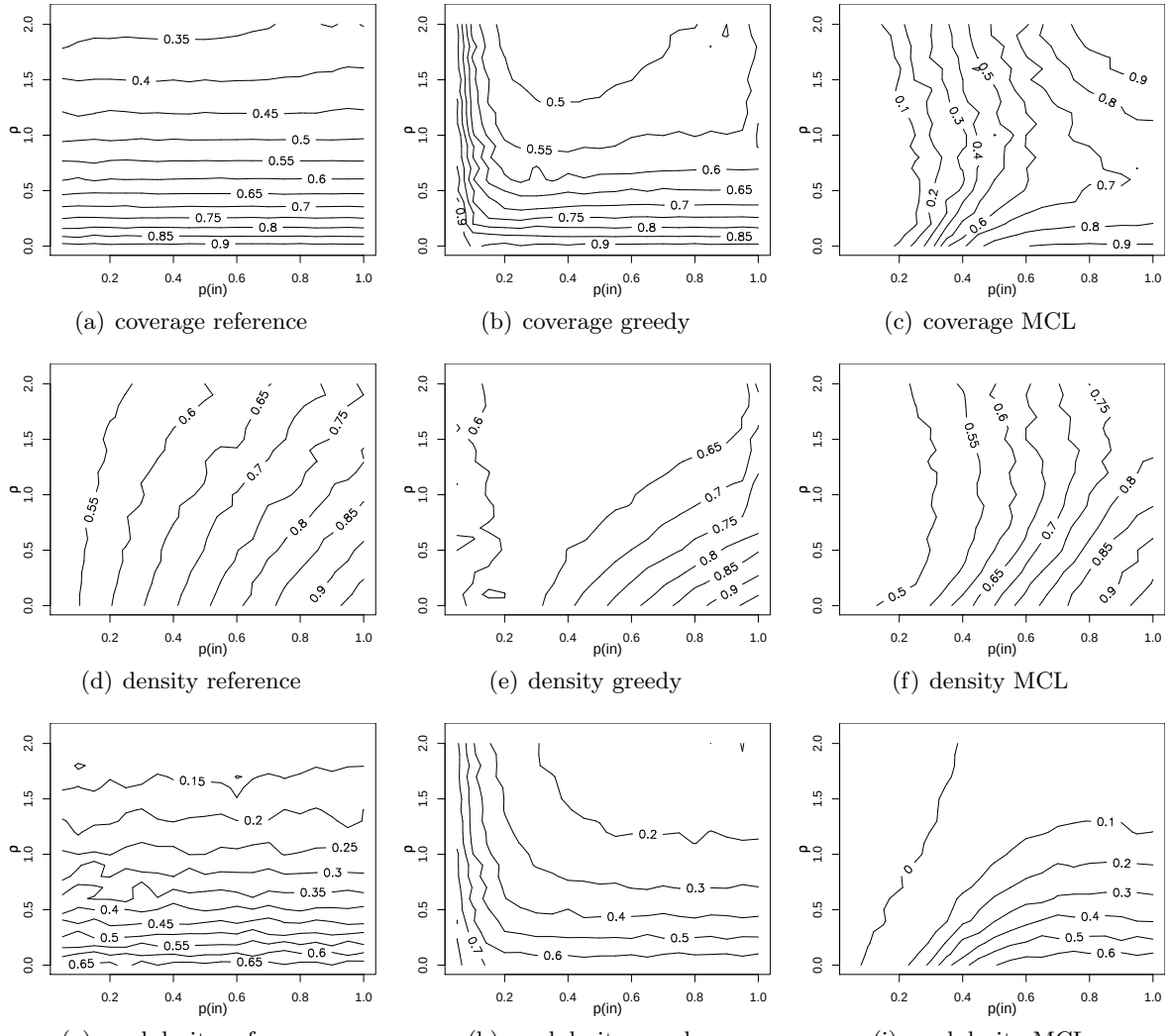
Note, that Unit-Test 4 and 5 can be alternately executed in order to find and evaluate improvements of algorithms and generators. More generally, all the above mentioned unit-tests constitute our benchmark foundation and can be combined in order to develop a sound test

suite. Theoretical and experimental insight should be incorporated in the unit-tests to further deepen the understanding. In the next section, we present a collection of performed unit-tests including their results and interpretations.

5 Results and Guidelines

In this section, we present several evaluations of algorithms, generators and quality indices according to our introduced framework. As we mentioned in Section 4, statistical significance is important for evaluating clustering techniques. In our case, we consider the average of the selected quality indices and repeated each experiment at least 50 times and at most 1000 times, but stopped when the length of the confidence interval fell below 0.1 with a probability of 0.95.

The two algorithms that serve as examples in our framework are those introduced in Section 2.3, namely the greedy algorithm and the MCL algorithm. The main unit-test for algorithm is Unit-Test 4 which requires a generator passing Unit-Test 1 and a quality index passing Unit-Test 2. As generator, we use the significant Gaussian generator fulfilling the corresponding unit-test, as can be seen in Figure 1(a).



(a) coverage reference (b) coverage greedy (c) coverage MCL

(d) density reference (e) density greedy (f) density MCL

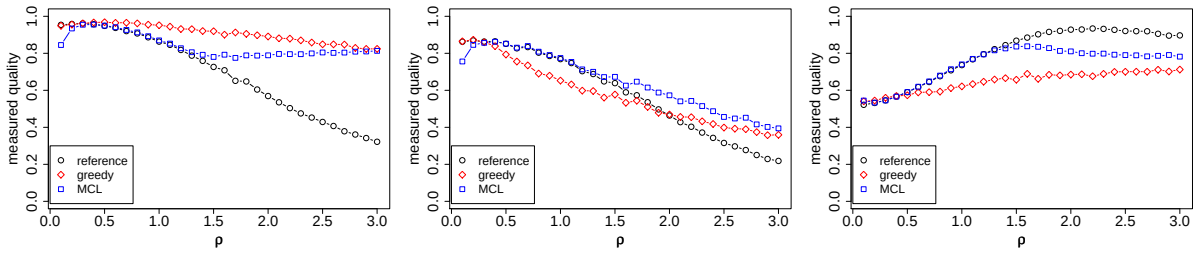
(g) modularity reference (h) modularity greedy (i) modularity MCL

Fig. 1. Quality of the reference clustering and the clusterings computed by the MCL and greedy algorithm, operating on instances of the significant Gaussian generator. Number of node is roughly 100.

Thus, we use it as the generator for Unit-Tests 2 on the indices modularity and density. Figures 1(d) and 1(g) indicate that both indices pass these tests. Since density is sensitive to both parameters of perturbation, it clearly passes Unit-Test 3. This confirms the sound definition of density. As coverage is independent of p_{in}, so is the first part of modularity, as described in Section 2.1. Moreover, since it has been shown that the expected value of coverage can be interpreted as a random rewiring of edges, the expected coverage for a fixed value of ρ will hardly change for varying p_{in}. Thus, modularity is independent of changes in p_{in}. Strictly speaking, coverage and modularity do not fail Unit-Test 3, but their independence of p_{in} constitutes a degeneracy one has to keep in mind when employing these indices.

Figure 1 shows the quality, with respect to coverage, modularity and density, of the reference clustering and the clustering calculated by the MCL algorithm and by the greedy algorithm. In both cases the obtained clustering is similar to the reference clustering for small perturbations. Thus, both algorithms pass this part of Unit-Test 4. However, the situation is different for high perturbations. While the MCL algorithm yields worse clusterings than the reference clustering, the greedy algorithm produces clusterings with higher quality. Note the exceptional behavior for low p_{in} and high ρ.

Figure 1(c) reveals that the MCL algorithm in combination with the significant Gaussian generator fails Unit-Test 1, since for high values of p_{in} coverage grows for increasing values of ρ beyond 0.6. Clearly, the large values of coverage for high ρ stem from the fact that the algorithm has a tendency to yield the 1-clustering, since a random walk tends to visit all nodes of the graph, as soon as the majority of edges incident to a node lead out of its cluster.



(a) coverage (b) modularity (c) density

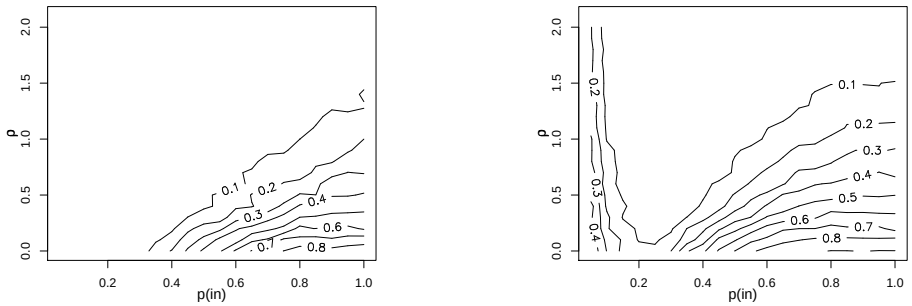
Fig. 2. Quality of the reference clustering and the clusterings computed by the MCL and greedy algorithm, operating on instances of the attractor generator. Number of node is roughly 1000.

Attractor Generator. The corresponding setup on the attractor generator is presented in Figure 2. We observe in Figure 2(a) that the generator passes Unit-Test 1 and both algorithms pass Unit-Test 4 for coverage, although for the MCL algorithm, we observe an artifact for very small values of ρ. Furthermore, the MCL algorithm again fails Unit-Test 1, since coverage grows for increasing values of ρ beyond 1.5. The reason for this again is the tendency of the algorithm to yield the 1-clustering for high values of ρ.

However, the situation is different for modularity and density, as can be seen in Figure 2(b) and 2(c). For modularity—which passes Unit-Test 2—the three clusterings yield the same value which may be interpreted as a failure of Unit-Test 4 for both algorithms. More important, the attractor generator in combination with modularity passes Unit-Test 5. Analyzing density we observe three facts: Both algorithms fail Unit-Test 4 and thus the generator itself cannot be considered to pass Unit-Test 5. But more important, we reveal a disadvantage of combining the attractor generator and density. Since an increase in perturbation leads to an increase in the measured quality, Unit-Test 2 fails. This originates from the fact that for very small perturbation the sparsity-part of density yields one, while the density-part yields zero. With increasing ρ,

edges are added, leading to an increase in the density-part and a decrease of the sparsity-part of the quality index density. However, the growth of intracluster-density exceeds the loss of the intercluster-sparsity.

Another interesting fact observable in Figure 3 is the difference between the average and the mininum inter-cluster conductance for the MCL algortihm for low p_{in} and high ρ. This is due to the fact that the MCL algorithm tends to produce high numbers of clusters, including clusters consisting of only one node, for these parameters. Thus, inter-cluster conductance, as a worst-case index, is dominated by the zero value for these clusters, while average inter-cluster conductance is not. We observed this artifical behavior of the quality index inter-cluster



(a) inter-cluster cond. MCL

(b) average inter-cluster cond. MCL

Fig. 3. Discrepancy between minimum and average inter-cluster conductance for the significant Gaussian generator. Number of node is roughly 100.

conductance in [14] under different circumstances. There, as a basic and intuitive generator, we introduced a variant of the Gaussian generator that generates graphs with the given fixed number of nodes. We observed, that Unit-Test 1 was obviously fulfilled. However, the index inter-cluster conductance exhibits notable fluctuations for large values of p_{out}. As pointed out in Section 4.1, this can either be an artifact of the generator or the index. In fact this behavior does not stem from only one component, but from the interplay of the generator and the index. Due to the procedure how instances are generated, the initial clustering often contains one significantly smaller cluster which is densely connected with the remaining graph. On the other hand, inter-cluster conductance is a worst-case index and as such is very sensitive to the size of the cut induced by a single small cluster. We introduced an enhanced version in [15], namely the Gaussian generator as defined in Section 2.2. This generator still fulfills Unit-Test 1, while avoiding the above mentioned drawback. This serves as an illustrating example of the hidden dependencies and their impact on designing and re-engineering clustering techniques.

A further selection of Figures is included in Appendix A.

6 Conclusion

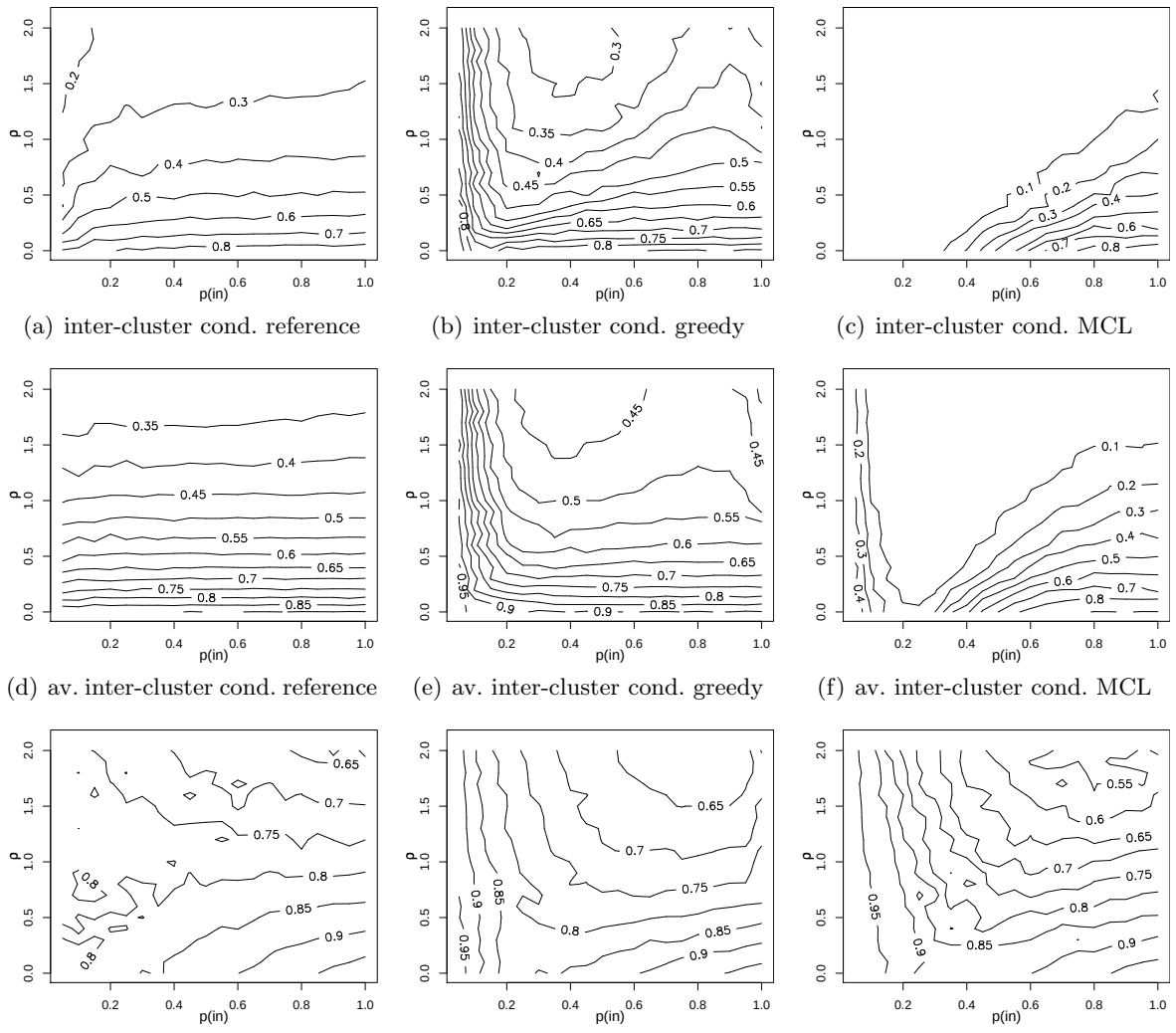
We presented a first design of a universal testing framework for evaluating clustering techniques. The prime difficulty in engineering such frameworks is incorporating the hidden dependencies between the different concepts, which are algorithms, indices, and generators. Due to our study of basic and intuitive setups, we were able to identify such dependencies and, furthermore, could extract them in form of unit-tests. Through alternately performing and redesigning tests, the quality and complexity of our framework increased, i. e., advanced unit-tests for algorithms, indices, and generators were obtained. In its current state, the framework is mature enough to investigate techniques with respect to basic properties of the paradigm intra-cluster density

versus inter-cluster sparsity. Application-specific details can easily be incorporated by adding more unit-tests. We will extend our framework by adapting it to further techniques and concepts such as the comparison of graph clusterings. Additionally, we plan to use the gained insights to continously enhance the considered concepts and moreover, we will theoretically analyze the phenomena observed here.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall (1988)
2. Brandes, U., Erlebach, T., eds.: Network Analysis: Methodological Foundations. Volume 3418 of Lecture Notes in Computer Science. Springer-Verlag (2005)
3. Vidal, M.: Interactome modeling. FEBS Lett. **579** (2005) 1834–1838
4. Wasserman, S., Faust, K.: Social Network Analysis: Methods and Applications. Cambridge University Press (1994)
5. Vempala, S., Kannan, R., Vetta, A.: On Clusterings - Good, Bad and Spectral. In: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00). (2000) 367–378
6. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
7. Gaertler, M.: Clustering. [2] 178–215
8. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E **70** (2004)
9. Doddi, S., Marathe, M.V., Ravi, S.S., Taylor, D.S., Widmayer, P.: Approximation Algorithms for Clustering to Minimize the Sum of Diameters. Nordic Journal of Computing **7** (2000) 185–203
10. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31** (1999) 264–323
11. Harel, D., Koren, Y.: On Clustering Using Random Walks. In: Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer Science. Volume 2245 of Lecture Notes in Computer Science., Springer-Verlag (2001) 18–41
12. Harel, D., Koren, Y.: Clustering spatial data using random walks. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data minin, ACM Press (2001) 281–286
13. Hartuv, E., Shamir, R.: A Clustering Algorithm based on Graph Connectivity. Information Processing Letters **76** (2000) 175–181
14. Brandes, U., Gaertler, M., Wagner, D.: Experiments on Graph Clustering Algorithms. In: Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03). Volume 2832 of Lecture Notes in Computer Science. (2003) 568–579
15. Delling, D., Gaertler, M., Wagner, D.: Generating Significant Graph Clusterings. In: Proceedings of the European Conference of Complex Systems ECCS. (2006)

A Figures

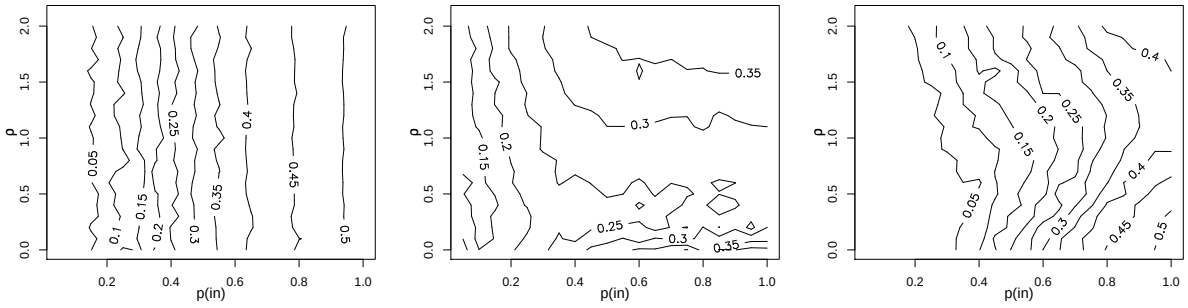


(a) inter-cluster cond. reference (b) inter-cluster cond. greedy (c) inter-cluster cond. MCL

(d) av. inter-cluster cond. reference (e) av. inter-cluster cond. greedy (f) av. inter-cluster cond. MCL

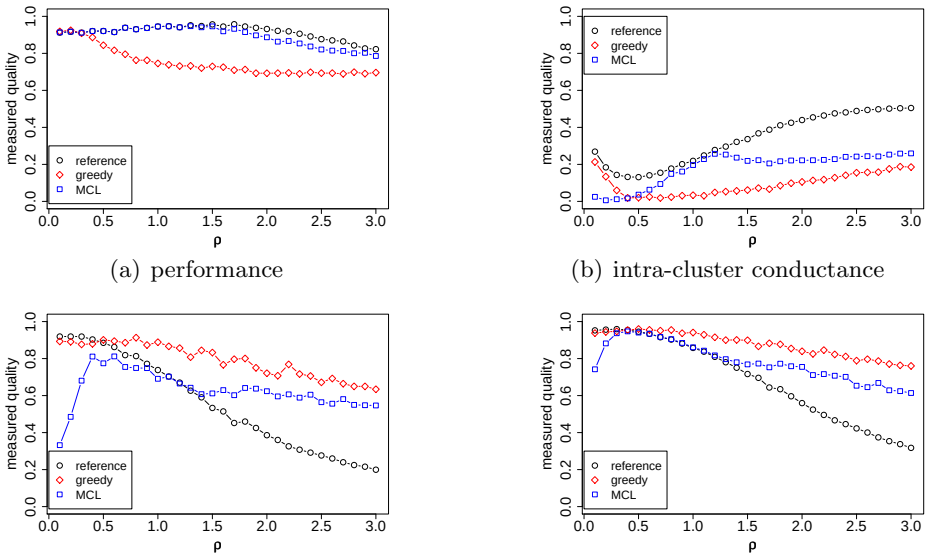
(g) performance reference (h) performance greedy (i) performance MCL

Fig. 4. Quality of the reference clustering and the clusterings computed by the MCL and greedy algorithm, operating on instances of the significant Gaussian generator. Number of node is roughly 100.



(a) intra-cluster cond. reference (b) intra-cluster cond. greedy (c) intra-cluster cond. MCL

Fig. 5. Quality of the reference clustering and the clusterings computed by the MCL and greedy algorithm, operating on instances of the significant Gaussian generator. Number of node is roughly 100.



(a) performance (b) intra-cluster conductance

(c) inter-cluster conductance (d) average inter-cluster conductance

Fig. 6. Quality of the reference clustering and the clusterings computed by the MCL and greedy algorithm, operating on instances of the attractor generator. Number of node is roughly 1000.