Generating Significant Graph Clusterings*

Daniel Delling, Marco Gaertler, and Dorothea Wagner

Faculty of Informatics, Universität Karlsruhe (TH),
{delling,gaertler,wagner}@informatik.uni-karlsruhe.de

Abstract. Many applications such as experimental evaluations of clustering algorithms require the existence of a significant reference clustering. This task is dual to finding significant clusterings of a given graph. We present several generators for pre–clustered graphs based on perturbation and geometry. In an experimental evaluation we confirm the applicability of our generators. Furthermore, the presented results lead to a better understanding of the correlation between the degree of perturbation and significance.

1 Introduction

Clustering, which consists of finding meaningful groups of elements in datasets, is an important technique in the exploration and analysis of data. It arises naturally in the fields of data mining [1], network analysis [2], biochemistry [3] and social studies [4] etc. Due to the large variety of these applications, the optimization criteria differ a lot. However, most applications share aspects of the founding clustering model, i. e., the required intrinsic properties of clusters or the type of connection between clusters. The general situation can be handled by blockmodels [5], while we restrict ourselves to the widely used paradigm of intra–cluster density versus inter–cluster sparsity [6,7,8].

Most research has been focused on finding the decomposition of a given graph into natural clusters. We consider the dual problem, namely generating a graph such that a given partition is significantly represented by the graph structure. The experimental comparison of clusterings is one scenario where this problem arises naturally. In the case of validating clustering algorithms, it is beneficial to know a significant clustering as a reference point. Measuring the significance is essentially founded on an implementation of an associated quality paradigm. These paradigms favor different ideal situations and generally agree only on the collection of disjoint cliques.

As mentioned previously, benchmark applications for clustering algorithms adopt the concept of pre–clustered graphs [7,9]. For example, the significance of a calculated clustering can be related to the reference. Since benchmarks comprise only a restricted scope, their testbed consists only of a few generators, thus making it hard to judge if the obtained results hold in general.

We present several generators and an experimental evaluation of them. The generators are based on the idea of perturbed disjoint cliques. In general, the collection of input parameters represent the amount of perturbation from the ideal case. The experimental evaluation confirms the intuitive connection between the parameters and the achieved

* 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).

quality. Furthermore, this results in a better understanding of the correlation between the amount of perturbation and the implementation of the quality paradigm.

This paper is organized as follows: We introduce the necessary preliminaries about graph clustering in Section 2. In Section 3, we define the generators used in the evaluation. The description and results of the performed evaluation are given in Section 4. We conclude the paper with a summary and a short outlook in Section 5.

2 Preliminaries

We assume that $G = (V, E)$ is an undirected and unweighted graph. Let $n := |V|, m := |E|$, and $\mathcal{C} := \{C_1, \dots, C_p\}$ be a partitioning of V. We call $\mathcal{C}$ a clustering and the C_i clusters of the graph. Let $E(\mathcal{C}) := \{\{u, v\} \in E \mid u, v \in C_i\}$ be the set of *intra-cluster edges* of $\mathcal{C}$ and $\overline{E}(\mathcal{C}) := \{\{u, v\} \in E \mid u \in C_i, v \in C_j, i \neq j\}$ the set of *inter-cluster edges* of $\mathcal{C}$. The cardinalities are indicated by $m(\mathcal{C}) := |E(\mathcal{C})|$ and $\overline{m}(\mathcal{C}) := |\overline{E}(\mathcal{C})|$. The graph $G[C_i] := (C_i, E(C_i))$ denotes the node-induced subgraph of the cluster C_i. We call G a *clustergraph* (with respect to $\mathcal{C}$), if every cluster induces a complete graph and no inter-cluster edges are present. For an arbitrary clustering $\mathcal{C}$, the *cluster editing set* $F_{\mathcal{C}}$ contains all edges that need to be added or deleted in order to transform G into a clustergraph with respect to $\mathcal{C}$.

In the following, we introduce some indices measuring the quality of clusterings. More precisely, we consider five indices, namely, coverage, performance, intra- and inter-cluster conductance, and modularity. All have been used for various graph clustering applications. For more details on these measures see [8]. Two basic indices *coverage*, $\operatorname{cov}(\mathcal{C}) := m(\mathcal{C})/m$, and *performance*, $\operatorname{perf}(\mathcal{C}) := 1 - 2 \cdot |F_{\mathcal{C}}|/(n(n-1))$, are based on counting edges. *Intra-* and *inter-cluster conductance* are two indices founded on the concepts 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; the formula is given in Equation (1) with $\mathcal{C} = \{C, V \setminus C\}$.

$$\varphi_G(\mathcal{C}) = \frac{\overline{m}(\mathcal{C})}{\min\left(\sum_{v \in C} \deg v, \sum_{v \notin C} \deg v\right)} \tag{1}$$

This can be extended to general clusterings yielding intra- and inter-cluster conductance defined by $\alpha(\mathcal{C}) := \min_{C \in \mathcal{C}} \min_{C' \subseteq C} \varphi_{G[C]}(C')$ and $\delta(\mathcal{C}) := 1 - \max_{C \in \mathcal{C}} \varphi_G(C)$.

An index that also incorporates statistical properties is *modularity* [10], defined by $\operatorname{mod}(\mathcal{C}) := \operatorname{cov}(\mathcal{C}) - 1/(4m^2) \sum_{C \in \mathcal{C}} (\deg v)^2$. 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.

3 Generators

Here, we present our generators for creating pre–clustered graphs. The generators in Section 3.1 are based on initially creating a partition and adding edges afterwards, while in Section 3.2 we provide a generator using geometric properties to assign nodes to clusters.

3.1 Random Partition Generators

A *random partition generator* uses an integer array P and two probabilities $p_{\text{in}}, p_{\text{out}}$ as parameters. The integer array indicates the partition of the nodes, where $|P|$ is the number of clusters and each entry of the array indicates the size of the corresponding cluster. The generator connects each pair in the same cluster with probability p_{in} and a pair in different clusters with probability p_{out}. Note, that $p_{\text{in}} = 1.0$ and $p_{\text{out}} = 0.0$ leads to a clustergraph, i. e., an unperturbed graph, while an uniform random graph is obtained for $p_{\text{in}} = p_{\text{out}}$.

Gaussian Generators The *gaussian generator* uses the approximate number of nodes n instead of a partition P as a parameter. However, the generator initially creates a partition P and calls the above described random partition generator with parameters P, p_{in}, and p_{out}. For generating a gaussian partition, we pick $|P|$ uniformly at random between $\log_{10}(n)$ and $\sqrt{n}$. The mean of the entries of P is $a = \lfloor n/|P| \rfloor$ and the standard deviation d is $d = \lfloor a/4 \rfloor$. Note, that the sum of all entries of P approximately equals $|P| \cdot \lfloor n/|P| \rfloor$, which approximates n quite well.

Significant Gaussian Generators 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, we introduce the *significant gaussian generator* that substitutes the parameter p_{out} by an *edge–ratio* $\rho := E(\overline{m}(\mathcal{C}))/E(m(\mathcal{C}))$. Note, that ρ is highly dependent on the number of clusters. The generator initially creates a gaussian partition as described above, dynamically calculates p_{out} according to Equation (2) and calls the same procedure as the gaussian generator for building the edgeset.

$$\rho = \frac{\overline{m}(\mathcal{C})}{m(\mathcal{C})} = \frac{p_{\text{out}}\left(\binom{n}{2} - |P|\binom{n/|P|}{2}\right)}{p_{\text{in}}|P|\binom{n/|P|}{2}} = \frac{p_{\text{out}}(n - n/|P|)}{p_{\text{in}}(n/|P| - 1)} \tag{2}$$

3.2 Attractors

The *attractor generator* uses geometric properties to generate a significant clustering based on the following idea: k cluster centers, so called attractor nodes, are placed uniformly at random with a certain minimum distance t in the plane. Then, $n - k$ satellite nodes are assigned to the attractors and their corresponding clusters using the following policy. At a random position a satellite node u is inserted with probability $d(u,a)/t$, where $d(u,a)$ is the euclidean distance from u to the nearest attractor node a. If u is inserted, the edge $\{u,a\}$ is inserted. The parameter f sets the threshold for connecting nodes with a certain distance.

Note, that the attractor nodes represent the point sites of a Voronoi Diagram and the satellite nodes are assigned to the clusters according to the Voronoi cells.

Our implementation uses the unit square for modeling the plane and $t = \sqrt{2/(\pi k)}$ as threshold for the minimum distance between attractor nodes. $\textsc{connectNodes}(t)$ connects the nodes with distance less than t according to the plane, while $\textsc{placeAttractor}(t)$

places a node in the plane with a distance greater than t to existing nodes according to the plane. Algorithm 1 provides the pseudo–code.

Algorithm 1: AttractorGenerator(n, f)

```
$V \leftarrow \emptyset$, $E \leftarrow \emptyset$, $P \leftarrow [0;1] \times [0;1]$ //P is the plane
$k \leftarrow$ integer uniformly at random $\in [\log_{10}(n), \sqrt{n}]$
$t \leftarrow \sqrt{2/(\pi k)}$ //threshold
for $i = 0; i < k; i{+}{+}$ do
    $v \leftarrow$ placeAttractor($t$)
    $V \leftarrow V \cup \{v\}$, $C_i \leftarrow \{v\}$
connectNodes($2.5 \cdot t$) // ensures global connectivity
$i \leftarrow 0$
while $i < n - k$ do
    $r \leftarrow$ randomCoord $\in [0;1] \times [0;1]$
    $a \leftarrow$ nearestAttractor($r$)
    $d \leftarrow$ distance($r$, $a$)
    $p \leftarrow$ double uniformly at random $\in [0;1]$
    if $d/t_1 < p$ then
        $C \leftarrow$ cluster($a$) //the cluster $a$ represents
        $V \leftarrow V \cup \{v\}$, $E \leftarrow E \cup \{v, a\}$, $C \leftarrow C \cup \{v\}$
        $i{+}{+}$
connectNodes($f \cdot \sqrt{2}/100$)
$G \leftarrow (V, E)$, $\mathcal{C} \leftarrow (C_1, \ldots, C_k)$
```

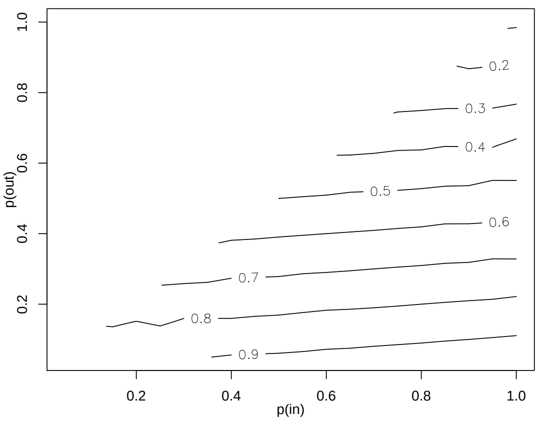
Note, that connectNodes() is called after placing the attractor nodes in the plane to ensure global connectivity of the graph. An input of $f = 0$ leads to a graph with star–like clusters and inter–cluster edges only between the attractor nodes, while $f = 100$ leads to the complete graph.

4 Experiments

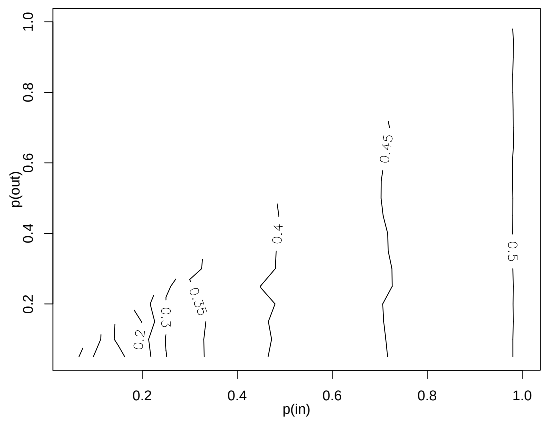
In this section, we summarize the results of the evaluation. All tests use $n = 1000$ nodes. For the gaussian generator we choose p_{in} between 0.05 and 1.0 using steps of 0.05 and p_{out} between 0.05 and p_{in} with steps of 0.05. For the significant gaussian generator we use the same values for p_{in} while we pick ρ between 0.1 and 3.0 with steps of 0.1. For the attractor generator we choose f between 0 and 50 with steps of 0.5. The tests are repeated at least 50 times until the maximal length of the 0.95–confidence intervals is not larger than 0.1.

For the gaussian generator, all five indices achieve their highest value for combinations of high p_{in} and low p_{out}. Performance (Figure 1(a)) and intra–cluster conductance (Figure 1(b)) exhibit an almost linear behavior. While performance strongly depends on the p_{out}, intra–cluster conductance only depends on p_{in}. In the case of modularity the dependency seems to be balanced.

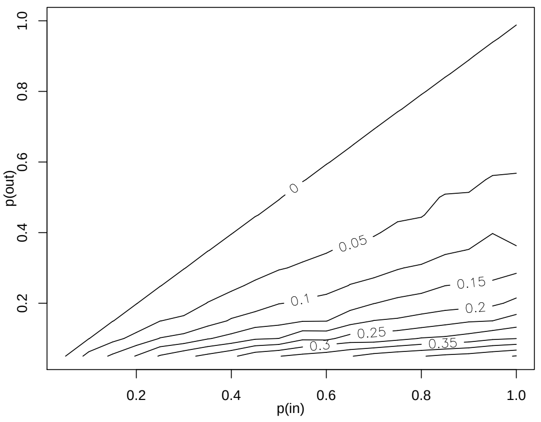
The situation is even more extreme for the significant gaussian generator, i. e., the five indices mainly depend on one parameter, namely on p_{in} or ρ. More precisely, coverage, modularity (Figure 1(d)) and inter–cluster conductance (Figure 1(f), interCC) are invariant with respect to variations in p_{in}, while intra–cluster conductance (Figure 1(e), intraCC) is invariant with respect to ρ, by virtue of construction. Performance has a very uniform behavior with high scores of approximately 0.9.



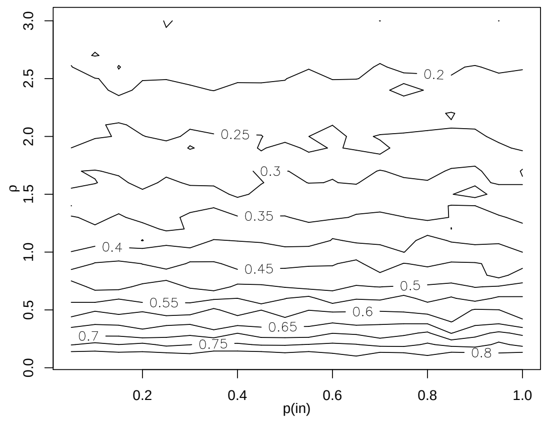
(a) gaussian generators: performance



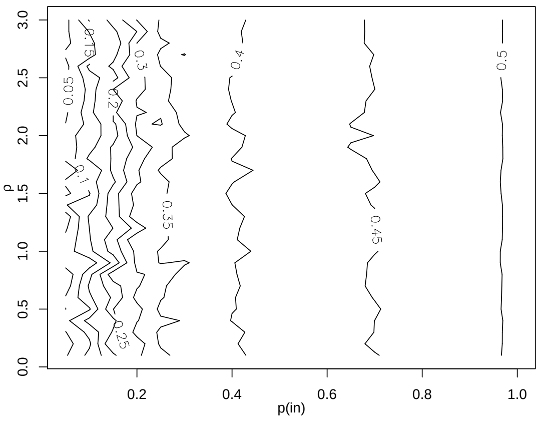
(b) gaussian generators: intra–cluster conductance



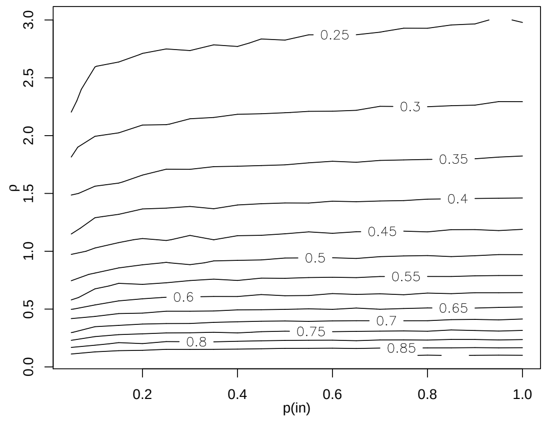
(c) gaussian generators: modularity



(d) significant gaussian generators: modularity



(e) significant gaussian generators: intraCC



(f) significant gaussian generators: interCC

Fig. 1. Results for gaussian and significant gaussian generators

The measured quality for attractors of all five indices are shown in Figure 2. Coverage, inter–cluster conductance and modularity indicate a decreasing significance with increasing density of the attractors. Intra–cluster conductance has an artificial behavior for star–like clusters which causes extreme jumps for very small densities. In the case of performance, the maximum score is achieved for $f \approx 13$. It increases from 0 to 13 and decreases with further increase in density.

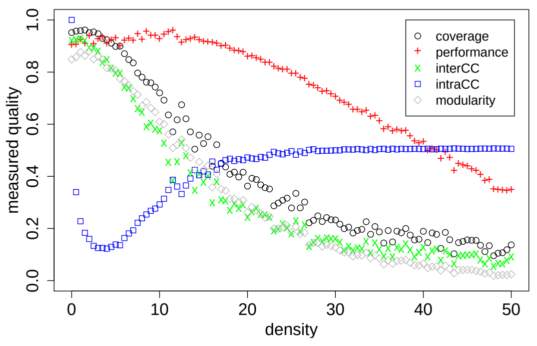


Fig. 2. Results for attractors

5 Conclusion

We presented several generators for creating graphs with a given partition such that the significance of the clustering is adjustable by the parameters. The experimental evaluation confirms that the generators respect the paradigm of intra–cluster density versus inter–cluster sparsity, since all indices attain their maximum score for instances with very small perturbation. Both gaussian generators have two degrees of freedom, one for intra–cluster density and one for inter–cluster sparsity. In the case of gaussian generators, most indices depend on p_{in} as well as p_{out}. By substitution of p_{out} by ρ, we observe that indices which formerly reacted to changes of p_{in} and p_{out} are only dependent on ρ. Thus, the usage of these indices, namely coverage, modularity and inter–cluster conductance, is easier because the choice of p_{in} is irrelevant. Still, intra–cluster conductance only depends on p_{in}, while performace hardly differs for the chosen parameters. The attractor generator offers a meaningful embedding additionally, which can be used to visually support the underlying clustering. The one parameter controlling the density simultaneously adjusts intra–cluster density and inter–cluster sparsity. For high values the impact on the inter–cluster sparsity is larger.

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. Nunkesser, M., Sawitzki, D.: Blockmodels. [2] 253–292
6. 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
7. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
8. Gaertler, M.: Clustering. [2] 178–215
9. 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
10. Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E **70** (2004)