Experiments on Graph Clustering Algorithms *

Ulrik Brandes[1], Marco Gaertler[2], and Dorothea Wagner[2]

[1] University of Passau, Department of Mathematics & Computer Science, 94030 Passau, Germany. brandes@algo.fmi.uni-passau.de
[2] University of Karlsruhe, Faculty of Informatics, 76128 Karlsruhe, Germany. {dwagner,gaertler}@ira.uka.de

Abstract. A promising approach to graph clustering is based on the intuitive notion of intra-cluster density vs. inter-cluster sparsity. While both formalizations and algorithms focusing on particular aspects of this rather vague concept have been proposed no conclusive argument on their appropriateness has been given.
As a first step towards understanding the consequences of particular conceptions, we conducted an experimental evaluation of graph clustering approaches. By combining proven techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably.

1 Introduction

Clustering is an important issue in the analysis and exploration of data. There is a wide area of applications as e.g. data mining, VLSI design, computer graphics and gene analysis. See also [1] and [2] for an overview. Roughly speaking, clustering consists in discovering natural groups of similar elements in data sets. An interesting and important variant of data clustering is graph clustering. On one hand, similarity is often expressed by a graph. On the other hand, there is a growing interest in network analysis in general.

A natural notion of graph clustering is the separation of sparsely connected dense subgraphs from each other. Several formalizations have been proposed. However, the understanding of current algorithms and indices is still rather intuitive. As a first step towards understanding the consequences of particular conceptions, we concentrate on indices and algorithms that focus on the relation between the number of intra-cluster and inter-cluster edges.

In [3] some indices measuring the quality of a graph clustering are discussed. Conductance, an index concentrating on the intra-cluster edges is introduced and a clustering algorithm that repeatedly separates the graph is presented. A graph clustering algorithm incorporating the idea of performing a random walk on the graph to identify the more densely connected subgraphs is presented in [4] and the index *performance* is considered to measure the quality of a graph

* This work was partially supported by the DFG under grant BR 2158/1-1 and WA 654/13-1 and EU under grant IST-2001-33555 COSIN.

clustering. The idea of random walks is also used in [5] but only for clustering geometric data. Obviously, there is a close connection between graph clustering and the classical graph problem minimum cut. A purely graph-theoretic approach using this connection more or less directly is the recursive minimum cut approach presented in [6]. Other more advanced partition techniques involve spectral information as in [3, 7–9].

It is not precisely known how well indices formalizing the relation between the number of intra-cluster and inter-cluster edges measure the quality of a graph clustering. Moreover, there exists no conclusive evaluation of algorithms that focus on such indices. In this paper, we give a summary of those indices and conduct an experimental evaluation of graph clustering approaches. The already known algorithms under comparison are the iterative conductance cut algorithm presented in [3] and the Markov clustering approach from [4]. By combining proven techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably with respect to flexibility and running time.

In Section 2 the notation used throughout the paper is introduced and clustering indices considered in the experimental study are presented. Section 3 gives a detailed description of the three algorithms considered. The graph generators used for the experimental evaluation are described in Section 4.1 and the results of the evaluation are summarized in Section 4.3.

2 Indices for Graph Clustering

Throughout this paper we assume that $G = (V, E)$ is a connected, undirected graph. Let $|V| =: n, |E| =: m$ and $\mathcal{C} = (C_1, \ldots, C_k)$ a partition of V. We call $\mathcal{C}$ a *clustering* of G and the C_i *clusters*; $\mathcal{C}$ is called *trivial* if either $k = 1$, or all clusters C_i contain only one element. In the following, we often identify a cluster C_i with the induced subgraph of G, i.e. the graph $G[C_i] := (C_i, E(C_i))$, where $E(C_i) := \{\{v, w\} \in E : v, w \in C_i\}$. Then $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intra-cluster edges* and $E \setminus E(\mathcal{C})$ the set of *inter-cluster edges*. The number of intra-cluster edges is denoted by $m(\mathcal{C})$ and the number of inter-cluster edges by $\overline{m}(\mathcal{C})$. A clustering $\mathcal{C} = (C, V \setminus C)$ is also called a *cut* of G and $\overline{m}(\mathcal{C})$ the *size* of the cut. A cut with minimum size is called a *mincut*.

2.1 Coverage

The *coverage*($\mathcal{C}$) of a graph clustering $\mathcal{C}$ is the fraction of intra-cluster edges within the complete set of edges, i.e.

$$\textit{coverage}(\mathcal{C}) := \frac{m(\mathcal{C})}{m} = \frac{m(\mathcal{C})}{m(\mathcal{C}) + \overline{m}(\mathcal{C})}.$$

Intuitively, the larger the value of *coverage*($\mathcal{C}$) the better the quality of a clustering $\mathcal{C}$. Notice that a mincut has maximum coverage and in this sense would be an "optimal" clustering. However, in general a mincut is not considered

to be a good clustering of a graph. Therefore, additional constraints on the number of clusters or the size of the clusters seem to be reasonable. While a mincut can be computed in polynomial time, constructing a clustering with a fixed number k, $k \geq 3$ of clusters is NP-hard [10], as well as finding a mincut satisfying certain size constraints on the clusters [11].

2.2 Performance

The *performance*($\mathcal{C}$) of a clustering $\mathcal{C}$ counts the number of "correctly interpreted pairs of nodes" in a graph. More precisely, it is the fraction of intra-cluster edges together with non-adjacent pairs of nodes in different clusters within the set of all pairs of nodes, i.e.

$$\textit{performance}(\mathcal{C}) := \frac{m(\mathcal{C}) + \sum_{\{v,w\} \notin E, v \in C_i, w \in C_j, i \neq j} 1}{\frac{1}{2} n(n-1)}.$$

Calculating the performance of a clustering according to this formula would be quadratic in the number of nodes. Especially, if the performance has to be computed for a sequence of clusterings of the same graph, it might be more efficient to count the number of "errors" instead (Equation (1)). Maximizing the performance is reducible to graph partitioning which is NP-hard [12].

$$1 - \textit{performance}(\mathcal{C}) = \frac{2m\left(1 - 2\textit{coverage}(\mathcal{C})\right) + \sum_{i=1}^{k} |C_i| \left(|C_i| - 1\right)}{n(n-1)} \tag{1}$$

2.3 Intra- and Inter-Cluster Conductance

The *conductance of a cut* compares the size of the cut and the number of edges in either of the two induced subgraphs. Then the *conductance* $\phi(G)$ *of a graph* G is the minimum conductance value over all cuts of G. For a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of a graph G, the *intra-cluster conductance* $\alpha(\mathcal{C})$ is the minimum conductance value over all induced subgraphs $G[C_i]$, while the *inter-cluster conductance* $\delta(\mathcal{C})$ is the maximum conductance value over all induced cuts $(C_i, V \setminus C_i)$. For a formal definition of the different notions of conductance, let us first consider a cut $\mathcal{C} = (C, V \setminus C)$ of G and define *conductance* $\phi(C)$ and $\phi(G)$ as follows.

$$\phi(C) := \begin{cases} 1, & C \in \{\emptyset, V\} \\ 0, & C \notin \{\emptyset, V\} \text{ and } \overline{m}(\mathcal{C}) = 0 \\ \frac{\overline{m}(\mathcal{C})}{\min\left(\sum_{v \in C} \deg v, \sum_{v \in V \setminus C} \deg v\right)}, & \text{otherwise} \end{cases}$$

$$\phi(G) := \min_{C \subseteq V} \phi(C)$$

Then a cut has small conductance if its size is small relative to the density of either side of the cut. Such a cut can be considered as a bottleneck. Minimizing the conductance over all cuts of a graph and finding the according cut is

NP-hard [10], but can be approximated with poly-logarithmic approximation guarantee in general, and constant approximation guarantee for special cases, [9] and [8]. Based on the notion of conductance, we can now define intra-cluster conductance $\alpha(\mathcal{C})$ and inter-cluster conductance $\delta(\mathcal{C})$.

$$\alpha(\mathcal{C}) := \min_{i \in \{1,\ldots,k\}} \phi\left(G[C_i]\right) \quad \text{and} \quad \delta(\mathcal{C}) := 1 - \max_{i \in \{1,\ldots,k\}} \phi\left(C_i\right)$$

In a clustering with small intra-cluster conductance there is supposed to be at least one cluster containing a bottleneck, i.e. the clustering is possibly too coarse in this case. On the other hand, a clustering with small inter-cluster conductance is supposed to contain at least one cluster that has relatively strong connections outside, i.e. the clustering is possibly too fine. To see that a clustering with maximum intra-cluster conductance can be found in polynomial time, consider first $m = 0$. Then $\alpha(\mathcal{C}) = 0$ for every non-trivial clustering $\mathcal{C}$, since it contains at least one cluster C_j with $\phi\left(G[C_j]\right) = 0$. If $m \neq 0$, consider an edge $\{u, v\} \in E$ and the clustering $\mathcal{C}$ with $C_1 = \{u, v\}$, and $|C_i| = 1$ for $i \geq 2$. Then $\alpha(\mathcal{C}) = 1$, which is maximum.

So, intra-cluster conductance has some artifical behavior for clusterings with many small clusters. This justifies the restriction to clusterings satisfying certain additional constraints on the size or number of clusters. However, under these constraints maximizing intra-cluster conductance becomes an NP-hard problem. Finding a clustering with maximum inter-cluster conductance is NP-hard as well, because it is at least as hard as finding a cut with minimum conductance.

3 Graph Clustering Algorithms

Two graph clustering algorithms that are assumed to perform well with respect to the indices described in the previous section are outlined. The first one iteratively emphazises intra-cluster over inter-cluster connectivity and the second one repeatedly refines an initial partition based on intra-cluster conductance. While both essentially operate locally, we also propose another, more global method. In all three cases, the asymptotic worst-case running time of the algorithms depend on certain parameters given as input. However, notice that for meaningful choices of these parameters, the time complexity of the new algorithm GMC is better than for the other two.

All three algorithms employ the *normalized adjacency matrix* of G, i.e., $M(G) = D(G)^{-1}A(G)$ where $A(G)$ is the adjacency matrix and $D(G)$ the diagonal matrix of vertex degrees.

3.1 Markov Clustering (MCL)

The key intuition behind *Markov Clustering* (MCL) [4, p. 6] is that a "random walk that visits a dense cluster will likely not leave the cluster until many of its vertices have been visited." Rather than actually simulating random walks, MCL iteratively modifies a matrix of transition probabilities. Starting from $M =$

$M(G)$ (which corresponds to random walks of length at most one), the following two operations are iteratively applied:

- *expansion*, in which M is taken to the power $e \in \mathbb{N}_{>1}$ thus simulating e steps of a random walk with the current transition matrix (Algorithm 1, Step 1)
- *inflation*, in which M is re-normalized after taking every entry to its rth power, $r \in \mathbb{R}^+$. (Algoritm 1, Steps 2–4)

Note that for $r > 1$, inflation emphasizes the heterogeneity of probabilities within a row, while for $r < 1$, homogeneity is emphasized. The iteration is halted upon reaching a recurrent state or a fixpoint. A recurrent state of period $k \in \mathbb{N}$ is a matrix that is invariant under k expansions and inflations, and a fixpoint is a recurrent state of period 1. It is argued that MCL is most likely to end up in a fixpoint [4]. The clustering is induced by connected components of the graph underlying the final matrix. Pseudo-code for MCL is given in Algorithm 1. Except for the stop criterion, MCL is deterministic, and its complexity is dominated by the expansion operation which essentially consists of matrix multiplication.

Algorithm 1: Markov Clustering (MCL)

Input: $G = (V, E)$, expansion parameter e, inflation parameter r

$M \leftarrow M(G)$
while M *is not fixpoint* **do**
1 $\quad M \leftarrow M^e$
2 $\quad$ **forall** $u \in V$ **do**
3 $\quad\quad$ **forall** $v \in V$ **do** $M_{uv} \leftarrow M_{uv}^r$
4 $\quad\quad$ **forall** $v \in V$ **do** $M_{uv} \leftarrow \frac{M_{uv}}{\sum_{w \in V} M_{uw}}$

$H \leftarrow$ graph induced by non-zero entries of M
$\mathcal{C} \leftarrow$ clustering induced by connected components of H

3.2 Iterative Conductance Cutting (ICC)

The basis of *Iterative Conductance Cutting* (ICC) [3] is to iteratively split clusters using minimum conductance cuts. Finding a cut with minimum conductance is NP–hard, therefore the following poly-logarithmic approximation algorithm is used. Consider the vertex ordering implied by an eigenvector to the second largest eigenvalue of $M(G)$. Among all cuts that split this ordering into two parts, one of minimum conductance is chosen. Splitting of a cluster ends when the approximation value of the conductance exceeds an input threshold α^* first. Pseudo-code for ICC is given in Algorithm 2. Except for the eigenvector computations, ICC is deterministic. While the overall running time depends on the number of iterations, the running time of the conductance cut approximation is dominated by the eigenvector computation which needs to be performed in each iteration.

Algorithm 2: Iterative Conductance Cutting (ICC)

Input: $G = (V, E)$, conductance threshold $0 < \alpha^* < 1$
$\mathcal{C} \leftarrow \{V\}$
while *there is a* $C \in \mathcal{C}$ *with* $\phi(G[C]) < \alpha^*$ **do**
  $x \leftarrow$ eigenvector of $M(G[C])$ associated with second largest eigenvalue
  $\mathcal{S} \leftarrow \left\{ S \subset C \colon \max_{v \in S}\{x_v\} < \min_{w \in C \setminus S}\{x_w\} \right\}$
  $C' \leftarrow \arg\min_{S \in \mathcal{S}}\{\phi(S)\}$
  $\mathcal{C} \leftarrow (\mathcal{C} \setminus \{C\}) \cup \{C', C \setminus C'\}$

3.3 Geometric MST Clustering (GMC)

Geometric MST Clustering (GMC), is a new graph clustering algorithm combining spectral partitioning with a geometric clustering technique. A geometric embedding of G is constructed from d distinct eigenvectors $x_1, \ldots, x_d$ of $M(G)$ associated with the largest eigenvalues less than 1. The edges of G are then weighted by a distance function induced by the embedding, and a minimum spanning tree (MST) of the weighted graph is determined. A MST T implies a sequence of clusterings as follows: For a threshold value τ let $F(T, \tau)$ be the forest induced by all edges of T with weight at most τ. For each threshold τ, the connected components of $F(T, \tau)$ induce a clustering. Note that there are at most $n - 1$ thresholds resulting in different forests. Because of the following nice property of the resulting clustering, we denote it with $\mathcal{C}(\tau)$. The proof of Lemma 1 is omitted. See [13].

Lemma 1. *The clustering induced by the connected components of* $F(T, \tau)$ *is independent of the particular MST* T*.*

Among the $\mathcal{C}(\tau)$ we choose one optimizing some measure of quality. Potential measures of quality are, e.g., the indices defined in Section 2, or combinations thereof. This genericity allows to target different properties of a clustering. Pseudo-code for GMC is given in Algorithm 3. Except for the eigenvector computations, GMC is deterministic. Note that, different from ICC, they form a preprocessing step, with their number bounded by a (typically small) input parameter. Assuming that the quality measure can be computed fast, the asymptotic time and space complexity of the main algorithm is dominated by the MST computation. GMC combines two proven concepts from geometric clustering and graph partitioning. The idea of using a MST that way has been considered before [14]. However, to our knowledge the MST decomposition was only used for geometric data before, not for graphs. In our case, general graphs without additional geometric information are considered. Instead, spectral graph theory is used [15] to obtain a geometric embedding that already incorporates insight about dense subgraphs. This induces a canonical distance on the edges which is taken for the MST computation.

Algorithm 3: Geometric MST Clustering (GMC)

Input: $G = (V, E)$, embedding dimension d, clustering valuation *quality*

$(1, \lambda_1, \ldots, \lambda_d) \leftarrow d+1$ largest eigenvalues of $M(G)$
$d' \leftarrow \max\{i \colon 1 \leq i \leq d, \lambda_i > 0\}$
$x^{(1)}, \ldots, x^{(d')} \leftarrow$ eigenvectors of $M(G)$ associated with $\lambda_1, \ldots, \lambda_{d'}$
forall $e = (u, v) \in E$ **do** $w(e) \leftarrow \sum_{i=1}^{d'} \left| x_u^{(i)} - x_v^{(i)} \right|$
$T \leftarrow$ MST of G with respect to w
$\mathcal{C} \leftarrow \mathcal{C}(\tau)$ for which *quality*$(\mathcal{C}(\tau))$ is maximum over all $\tau \in \{w(e) \colon e \in T\}$

4 Experimental Evaluation

First we describe the general model used to generate appropriate instances for the experimental evaluation. Then we present the experiments and discuss the results of the evaluation.

4.1 Random Uniform Clustered Graphs

We use a random partition generator $\mathcal{P}(n, s, v)$ that determines a partition $(P_1, \ldots, P_k)$ of $\{1, \ldots, n\}$ with $|P_i|$ being a normal random variable with expected value s and standard deviation $\frac{s}{v}$. Note that k depends on the choice of n, s and v, and that the last element $|P_k|$ of $\mathcal{P}(n, s, v)$ is possibly significantly smaller than the others. Given a partition $\mathcal{P}(n, s, v)$ and probabilities p_{in} and p_{out}, a uniformly random clustered graph $(G, \mathcal{C})$ is generated by inserting intra-cluster edges with probability p_{in} and inter-cluster edges with probability p_{out} [3]. For a clustered graph $(G, \mathcal{C})$ generated that way, the expected values of m, $m(\mathcal{C})$ and $\overline{m}(\mathcal{C})$ can be determined. We obtain

$$\mathbb{E}\left[\overline{m}(\mathcal{C})\right] = \frac{p_{\text{out}}}{2}\left(n(n-s)\right) \qquad \text{and} \qquad \mathbb{E}\left[m(\mathcal{C})\right] = \frac{p_{\text{in}}}{2}\left(n(s-1)\right),$$

and accordingly for coverage and performance

$$\mathbb{E}\left[\textit{coverage}(\mathcal{C})\right] = \frac{(s-1)p_{\text{in}}}{(s-1)p_{\text{in}} + (n-s)p_{\text{out}}}$$
$$1 - \mathbb{E}\left[\textit{performance}(\mathcal{C})\right] = \frac{(n-s)p_{\text{out}} + (1-p_{\text{in}})(s-1)}{n-1}.$$

In the following, we can assume that for our randomly generated instances the initial clustering has the expected behavior with respect to the indices considered.

[3] In case a graph generated that way is not connected, additional edges combining the components are added.

4.2 Technical Details of the Experiments and Implementation

For our experiments, randomly generated instances with the following values of (n, s, v) respectively $p_{\text{in}}, p_{\text{out}}$ are considered. We set $v = 4$ and choose s uniformly at random from $\left\{\frac{n}{\ell} : 2 \leq \ell \leq \sqrt{n}\right\}$. Experiments are performed for $n = 100$ and $n = 1000$. On one hand, all combinations of probabilities p_{in} and p_{out} at a distance of 0.05 are considered. On the other hand, for two different values $p_{\text{in}} = 0.4$ and $p_{\text{in}} = 0.75$, p_{out} is chosen such that the ratio of $\overline{m}(\mathcal{C})$ and $m(\mathcal{C})$ for the initial clustering $\mathcal{C}$ is at most 0.5, 0.75 respectively 0.95.

The free parameters of the algorithms are set to $e = 2$ and $r = 2$ in MCL, $\alpha^* = 0.475$ and $\alpha^* = 0.25$ in ICC, and dimension $d = 2$ in GMC. As objective function *quality* in GMC, *coverage*, *performance*, intra-cluster conductance α, inter-cluster conductance δ, as well as the geometric mean of *coverage*, *performance* and δ is considered [4].

All experiments are repeated at least 30 times and until the maximal length of the confidence intervals is not larger than 0.1 with high probability. The implementation is written in C++ using the GNU compiler g++(2.95.3). We used LEDA 4.3[5] and LAPACK++[6]. The experiments were performed on an Intel Xeon with 1.2 ($n = 100$) and 2.4 ($n = 1000$) GHz on the Linux 2.4 platform.

4.3 Computational Results

We concentrate on the behavior of the algorithms with respect to running time, the values for the initial clustering in contrast to the values obtained by the algorithms for the indices under consideration, and the general behavior of the algorithms with respect to the variants of random instances. In addition, we also performed some experiments with grid-like graphs.

Running Time The experimental study confirms the theoretical statements in Section 3 about the asymptotic worst-case complexity of the algorithms. MCL is significantly slower than ICC and GMC. Not surprisingly as the running time of ICC depends on the number of splittings, ICC is faster for $\alpha^* = 0.25$ than for $\alpha^* = 0.475$. Note that the coarseness of the clustering computed by ICC results from the value of α^*.

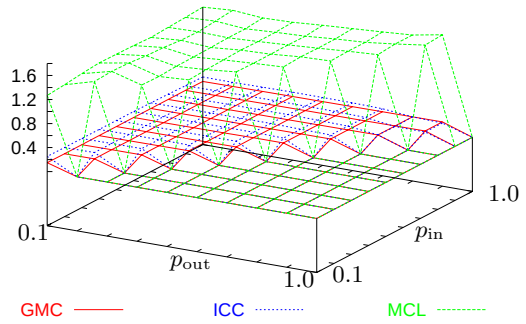
For all choices of *quality* except intra-cluster conductance, GMC is the most efficient algorithm. Note that the higher running time of GMC with *quality* set to intra-cluster conductance is only due to the elaborate approximation algorithm for the computation of the intra-cluster conductance value. In summary, GMC with *quality* being the geometric mean of *coverage*, *performance* and inter-cluster conductance, respectively *quality* being an appropriate combination of those indices is the most efficient algorithm under comparison. See Figure 1.

[4] Experiments considering the geometric mean of all four indices showed that incorporation of intra-cluster conductance did not yield significantly different results. We therefore omit intra-cluster conductance because of efficiency reasons.

[5] `http://www.algorithmic-solutions.com`

[6] `http://www.netlib.org/lapack/`

a)



b)

$(p_{\text{in}}, p_{\text{out}})$	GMC	ICC
$(0.25, 0.25)$	71	102
$(0.50, 0.25)$	72	103
$(0.50, 0.50)$	72	73
$(0.75, 0.25)$	74	101
$(0.75, 0.50)$	74	78
$(0.75, 0.75)$	74	73

Fig. 1. Running-time in seconds for $n = 100$ (a) and $n = 1000$ (b).

Indices for the Initial Clustering Studying *coverage*, *performance*, intra- and inter-cluster conductance of the initial clustering gives some useful insights about these indices. Of course, for *coverage* and *performance* the highest values are achieved for the combination of very high p_{in} and very low p_{out}. The *performance* value is greater than the *coverage* value, and the slope of the *performance* level curves remains constant while the slope of the *coverage* level curves decreases with increasing p_{in}. This is because *performance* considers both, edges inside and non-edges between clusters, while *coverage* measures only the fraction of intra-cluster edges within all edges.

The fluctuations of the inter-cluster conductance values for higher values of p_{out} can be explained by the dependency of inter-cluster conductance $\delta(\mathcal{C})$ from the cluster $C_i \in \mathcal{C}$ maximizing ϕ. This shows that inter-cluster conductance is very sensitive to the size of the cut induced by a single small cluster. Due to the procedure how instances are generated for a fixed choice of n, the initial clustering often contains one significantly smaller cluster. For higher values of p_{out}, this cluster has a relatively dense connection to the rest of the graph. So, in many cases it is just this cluster that induces the inter-cluster conductance value.

In contrast to the other three indices, intra-cluster conductance shows a completely different behavior with respect to the choices of p_{in} and p_{out}. Actually, intra-cluster conductance does not depend on p_{out}.

Comparing the Algorithms A significant observation when comparing the three algorithms with respect to the four indices regards their behavior for dense graphs. All algorithms have a tendency to return a trivial clustering containing only one cluster, even for combinations of p_{in} and p_{out} where p_{in} is significantly higher than p_{out}. This suggests a modification of the algorithms to avoid trivial clusterings. However, for ICC such a modification would be a significant deviation from its intended procedure. The consequences of forcing ICC to split even if

the condition for splitting is violated are not clear at all. On the other hand, the approximation guarantee for intra-cluster conductance is no longer maintained if ICC is prevented from splitting even if the condition for splitting is satisfied. For MCL it is not even clear how to incorporate the restriction to non-trivial clusterings. In contrast, it is easy to modify GMC such that only non-trivial clusterings are computed. Just the maximum and the minimum threshold values τ are ignored.

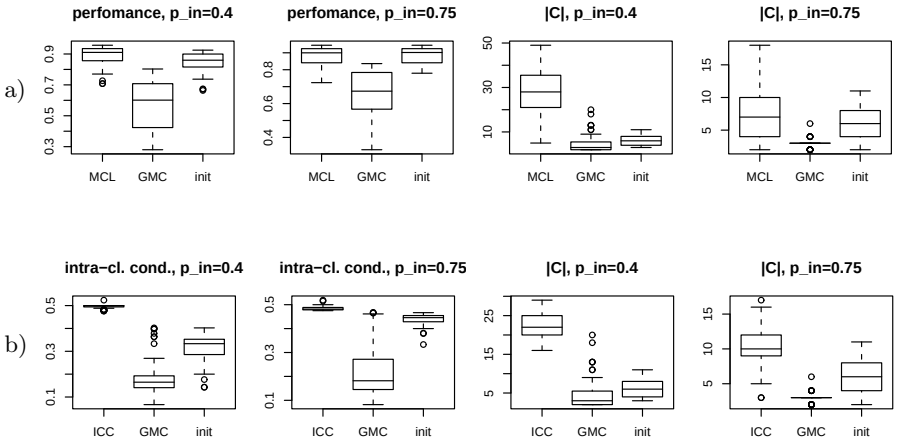
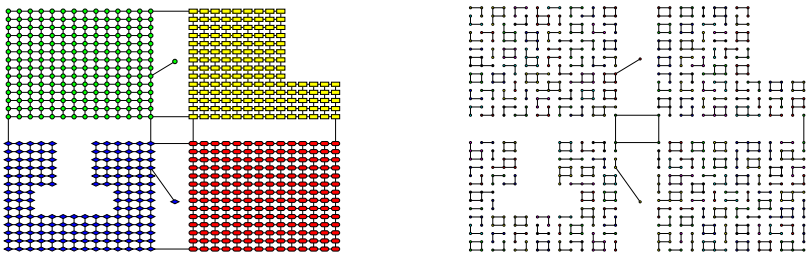


Fig. 2. The diagrams show the distribution of performance respectively intra-cluster conductance and the number of clusters for $p_{\text{in}} = 0.4$ respectively $p_{\text{in}} = 0.75$, and p_{out} such that at most one third of the edges are inter-cluster edges. The boxes are determined by the first and the third quantile and the internal line represents the median. The shakers extend to 1.5 of the boxes' length (interquartile distance) respectively the extrema. The first two diagrams in 2a) compare the performance values for MCL, GMC and the initial clustering, whereas the last two compare the number of clusters. The first two diagrams in 2b) compare the intra-cluster conductance for MCL, GMC and the initial clustering, whereas the last two compare the number of clusters.

Regarding the cluster indices, MCL does not explicitely target on any of those. However, MCL implicitly targets on identifying loosely connected dense subgraphs. It is argued in [4] that this is formalized by *performance* and that MCL actually yields good results for *performance*. In Figure 2a), the behavior of MCL and GMC are compared with respect to *performance*. The results suggest that MCL indeed performs somewhat better than GMC. The *performance* values for MCL are higher than for GMC and almost identical to the values of the initial clustering. However, MCL has a tendency to produce more clusters than GMC and actually also more than contained in the initial clustering. For instances with high p_{in}, the results for MCL almost coincide with the initial clustering

but the variance is greater. ICC targets explicitely at intra-cluster conductance and its behavior depends on the given α^*. Actually, ICC computes clusterings with intra-cluster conductance α close to α^*. For $\alpha^* = 0.475$, ICC continues the splitting quite long and computes a clustering with many small clusters. In [3] it is argued that *coverage* should be considered together with intra-cluster conductance. However, ICC compares unfavorable with respect to *coverage*. For both choices of α^*, the variation of the *performance* values obtained by ICC is comparable while the resulting values are better for $\alpha^* = 0.475$. This suggests that besides intra-cluster conductance, ICC implicitly targets at *performance* rather than at coverage. Comparing the performance of ICC (with $\alpha^* = 0.475$) and GMC with respect to intra-cluster conductance suggests that ICC is much superior to GMC. Actually, the values obtained by ICC are very similar to the intra-cluster conductance values of the initial clustering. However, studying the number of clusters generated shows that this is achived at the cost of generating many small clusters. The number of clusters is even significantly bigger than in the initial clustering. This suggests the conclusion that targeting at intra-cluster conductance might lead to unintentional effects. See Figure 2b). Finally, Figure 3 confirms that ICC tends to generate clusterings with many clusters. In contrast, GMC performs very well. It actually generates the ideal clustering.



(a) (b)

Fig. 3. In 3(a) the clustering determined by GMC for a grid-like graph is shown. The clusters are shown by the different shapes of vertices. In contrast, 3(b) shows the clustering determined by ICC. Inter-cluster edges are not omitted to visualize the clusters.

5 Conclusion

The experimental study confirms the promising expectations about MCL, i.e. in many cases MCL seems to perform well. However, MCL often generates a trivial clustering. Moreover, MCL is very slow. The theoretical result on ICC is reflected by the experimental study, i.e., ICC computes clusterings that are good with respect to intra-cluster conductance. On the other hand, there is the suspect that

the index intra-cluster conductance does not measure the quality of a clustering appropriately. Indeed, the experimental study shows that all four cluster indices have weaknesses. Optimizing only with respect to one of the indices often leads to unintended effects. Considering combinations of those indices is an obvious attempt for further investigations. Moreover, refinement of the embedding used by GMC offers additional potential. So far, only the embedding canonically induced by the eigenvectors is incorporated. By choosing different weightings for the distances in the different dimensions, the effect of the eigenvectors can be controlled.

Actually, because of its flexibility with respect to the usage of the geometric clustering and the objective function considered, GMC is superior to MCL and ICC. Finally, because of its small running time GMC is a promising approach for clustering large graphs.

References

1. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall (1988)
2. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Computing Surveys **31** (1999) 264–323
3. Kannan, R., Vampala, S., Vetta, A.: On Clustering — Good, Bad and Spectral. In: Foundations of Computer Science 2000. (2000) 367–378
4. van Dongen, S.M.: Graph Clustering by Flow Simulation. PhD thesis, University of Utrecht (2000)
5. Harel, D., Koren, Y.: On clustering using random walks. Foundations of Software Technology and Theoretical Computer Science **2245** (2001) 18–41
6. Hartuv, E., Shamir, R.: A clustering algorithm based on graph connectivity. Information Processing Letters **76** (2000) 175–181
7. Spielman, D.A., Teng, S.H.: Spectral partitioning works: Planar graphs and finite element meshes. In: IEEE Symposium on Foundations of Computer Science. (1996) 96–105
8. Chung, F., Yau, S.T.: Eigenvalues, flows and separators of graphs. In: Proceeding of the 29th Annual ACM Symposium on Theory of Computing. (1997) 749
9. Chung, F., Yau, S.T.: A near optimal algorithm for edge separators. In: Proceeding of the 26th Annual ACM Symposium on Theory of Computing. (1994) 1–8
10. Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: Complexity and Approximation – Combinatorial optimization problems and their approximability properties. Springer–Verlag (1999)
11. Wagner, D., Wagner, F.: Between Min Cut and Graph Bisection. In Borzyszkowski, A.M., Sokolowski, S., eds.: Lecture Notes in Computer Science, Springer–Verlag (1993) 744–750
12. Garey, M.R., Johnson, D.S., Stockmeyer, L.J.: Some simplified NP-complete graph problems. Theoretical Computer Science **1** (1976) 237–267
13. Gaertler, M.: Clustering with spectral methods. Master's thesis, Universität Konstanz (2002)
14. Zahn, C.: Graph-theoretical methods for detecting and describing gestalt clusters. IEEE Transactions on Computers **C-20** (1971) 68–86
15. Chung, F.R.K.: Spectral Graph Theory. Number 52 in Conference Board of the Mathematical Sciences. American Mathematical Society (1994)