First publ. in: ACM Journal of Experimental Algorithmics 12 (2007), Article 1.1

Engineering Graph Clustering: Models and Experimental Evaluation

ULRIK BRANDES
University of Konstanz
and
MARCO GAERTLER and DOROTHEA WAGNER
Universität Karlsruhe

A promising approach to graph clustering is based on the intuitive notion of intracluster density versus intercluster sparsity. As for the weighted case, clusters should accumulate lots of weight, in contrast to their connection to the remaining graph, which should be light. 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. In order to deepen the understanding of particular concepts, including both quality assessment as well as designing new algorithms, we conducted an experimental evaluation of graph-clustering approaches. By combining proved techniques from graph partitioning and geometric clustering, we also introduce a new approach that compares favorably.

Categories and Subject Descriptors: G.2.3 [**Discrete Mathematics**]: Applications; H.3.3 [**Information Search and Retrieval**]: Clustering

General Terms: Algorithm, Design

Additional Key Words and Phrases: Graph clustering, experimental evaluation, quality measures, clustering algorithms

1. INTRODUCTION

Clustering is an important issue in the analysis and exploration of data. There is a wide area of applications, e.g., data mining, VLSI design, computer graphics, and gene analysis. (See also Jain and Dubes [1988] and Jain et al. [1999] for an overview.) Roughly speaking, clustering means discovering natural groups of similar elements in data sets. An interesting and important variant of data clustering is graph-clustering. On the one hand, similarity is often expressed by a graph. On the other hand, there is, in general, a growing interest in network analysis.

A natural notion of graph clustering is the separation of sparsely connected dense subgraphs from each other. Several formalizations have been proposed. However, the comprehension of current algorithms and indices is still rather intuitive. As a first step toward a deeper understanding, we performed a preliminary study in Brandes et al. [2003] using unweighted graphs. The experiments verified that algorithms, as well as quality measures, behave very well in the case of almost disjoint cliques. These cases have an incontrovertible clustering structure. However, the results became ambiguous as soon as intracluster density decreased or intercluster sparsity increased. Algorithms as well as quality measurements reacted quite differently. In the case of weighted graphs, the simple paradigm gains additional ambiguities, namely, the interpretation of sparse, yet heavy, or dense, yet light, subgraphs. These potential groups fulfill the density or weight criterion, while failing the other. Thus their relevance as clusters is questionable or at least depends on the application. Along the lines of Brandes et al. [2003], we concentrate on indices and algorithms that focus on the relation between the number of intra- and intercluster edges.

In Vempala et al. [2000], some indices measuring the quality of graph clustering are discussed. Conductance, an index concentrating on the intracluster 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 van Dongen [2000] and the index *performance* is considered to measure the quality of a graph clustering. The idea of random walks is also used in Harel and Koren [2001], 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 Hartuv and Shamir [2000]. Other more advanced partition techniques involve spectral information as in Vempala et al. [2000], Spielman and Teng [1996], and Chung and Yau [1994, 1997]. Very recently, the physics community presented techniques based on centralities and statistical properties. For example, an algorithm that iteratively prunes edges based on betweenness centrality was introduced as a clustering technique in Newman and Girvan [2004]. A related quality measure named *modularity* was presented in Clauset et al. [2004]. It evaluates the significance of clustering with respect to the graph structure by considering a random rewiring of the edge set.

It is not precisely known how well indices that formalize the relation between the number of intra- and intercluster edges measure the quality of a graph clustering. Moreover, there exists no conclusive evaluation of algorithms that focus on such indices. Therefore, our main goal is to perform an experimental evaluation that deepens the understanding of clustering techniques and quality assessment. As a partial result, we confirm the claim that all indices show certain artifical behavior (as presented for the unweighted case in Brandes et al. [2003]). Thus, it is natural to parameterize algorithms in order to incorporate different quality aspects. As a consequence, we engineered such an approach. In this paper, we give a summary of quality indices, including a comparison of unweighted versus weighted version, and conduct an experimental evaluation of graph-clustering approaches. The algorithms under comparison are the iterative conductance cut algorithm presented in Vempala et al. [2000], the Markov clustering approach from van Dongen [2000], and our method Brandes et al. [2003], which combines spectral embeddings and decomposition-based minimum spanning trees (MST). The idea of using a MST that way has been considered before [Zahn 1971]. However, to our knowledge, the MST decomposition was only used for geometric data or data embedded in metrical spaces Ho et al. [2003] and not for graphs. Since we consider general graphs with no additional geometric information and also unweighted ones, the initial spectral embedding is a neccessary and vital part in the algorithm. In order to keep the benchmarks and obtained results conclusive, we restricted ourselves to algorithms and quality measures that were used for similar purposes. In particular, we excluded the approaches of the physics community because of unresolved issues, such as the lack of description as an optimization problem, unknown complexity issues, tremendous increase of parameters for weighted versions, and artifical behavior. Since many of these approaches cover novel ideas, they might be subject of future research.

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 experiments are described in Section 4, which contains the generator model, implementational aspects, and the results. A summary and an outlook in Section 5 concludes the article.

2. INDICES FOR GRAPH CLUSTERING

Throughout this paper, we assume that $G = (V, E, \omega)$ is a simple, connected, and undirected graph with a positive edge weighting $\omega: E \to \mathbb{R}^+$. 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. The set of edges, which have one endnode in C_i and the other endnode in C_j, is denoted by $E(C_i, C_j) := \{\{v, w\} \in E : v \in C_i, w \in C_j\}$. 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 $E(C_i) := E(C_i, C_i)$. Then, $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intracluster edges* and $\overline{E(\mathcal{C})} := E \setminus E(\mathcal{C})$ the set of *intercluster edges*. The number of intracluster edges is denoted by $m(\mathcal{C})$ and the number of intercluster edges by $\overline{m}(\mathcal{C})$. In analogy, the weight of all intracluster edges is

denoted by $\omega(\mathcal{C})$ and the weight of all intercluster edges by $\overline{\omega}(\mathcal{C})$. For an edge subset $E' \subseteq E$, the symbol $\omega(E')$ is a short-cut for $\sum_{e \in E'} \omega(e)$. 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*.

The used indices exhibit a general structure that emphasizes the paradigm of intracluster density versus intercluster sparsity. This structure can be described by two independent, nonnegative functions f and g that measure the density and the sparsity, respectively, and that depend "only" on the clustering. In order to normalize the range of the index, a third function N, that depends only on the input graph, is used. An index $\mathsf{index}(\mathcal{C})$ is composed as shown in Eq. (1).

$$\mathsf{index}(\mathcal{C}) := \frac{f(\mathcal{C}) + g(\mathcal{C})}{N(G)} \tag{1}$$

The normalization function $N(G)$ should be set to the maximum of $f + g$ over all clusterings.

2.1 Coverage

The *coverage*$(\mathcal{C})$ of a graph clustering $\mathcal{C}$ is the fraction of the weight of intracluster edges with respect to the total weight of all edges, i. e., $f(\mathcal{C}) = \omega(E(\mathcal{C}))$, $g \equiv 0$ and $N(G) = \omega(E)$ or short:

$$\textit{coverage}(\mathcal{C}) := \frac{\omega(\mathcal{C})}{\omega(E)} = \frac{\omega(\mathcal{C})}{\omega(\mathcal{C}) + \overline{\omega}(\mathcal{C})}$$

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 and optimal coverage value, as well as finding a mincut satisfying certain size constraints on the clusters, is $\mathcal{NP}$-hard [Ausiello et al. 2002; Wagner and Wagner 1993].

2.2 Intra- and Intercluster Conductance

The *conductance of a cut* compares the size of the cut and the weight of edges in either of the two induced subgraphs. The *conductance* $\varphi(G)$ *of a graph* G is then the minimum conductance value over all cuts of G. For a clustering $\mathcal{C} = (C_1, \ldots, C_k)$ of a graph G, the *intracluster conductance* $\alpha(\mathcal{C})$ is the minimum conductance value over all induced subgraphs $G[C_i]$, while the *intercluster 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* $\varphi(C)$ and $\varphi(G)$ as follows:

$$a(C) := \sum_{v \in C} \sum_{\substack{cw \in V, \\ \{v,w\} \in E}} \omega(\{v,w\}) = 2 \sum_{e \in E(C)} \omega(e) + \sum_{f \in E(C, V \setminus C)} \omega(f)$$

$$\varphi(C) := \begin{cases} 1, & C \in \{\emptyset, V\} \\ 0, & C \notin \{\emptyset, V\} \text{ and } \overline{\omega}(\mathcal{C}) = 0 \\ \frac{\overline{\omega}(\mathcal{C})}{\min(a(C), a(V \setminus C))}, & \text{otherwise} \end{cases}$$
$$\varphi(G) := \min_{C \subseteq V} \varphi(C)$$

Then a cut has small conductance if its size is small in relation 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 $\mathcal{NP}$-hard [Ausiello et al. 2002], but can be approximated with polylogarithmic approximation guarantee, in general, and constant approximation guarantee for special cases [Chung and Yau 1994, 1997]. Based on the notion of conductance, we can now define intra- $\alpha(\mathcal{C})$ and intercluster conductance $\delta(\mathcal{C})$.

$$\alpha(\mathcal{C}) := \min_{i \in \{1,\ldots,k\}} \varphi(G[C_i]) \quad \text{and}$$
$$\delta(\mathcal{C}) := \begin{cases} 1, & \text{if } \mathcal{C} = \{V\} \\ 1 - \max_{i \in \{1,\ldots,k\}} \varphi(C_i), & \text{otherwise} \end{cases}$$

Expressing both indices in the general framework, we obtain $g \equiv 0$ for intracluster conductance and $f \equiv 0$ for intercluster conductance, while, in both cases, $N \equiv 1$, which is also the maximum of $f + g$. In a clustering with small intracluster 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 intercluster 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 intracluster conductance can be found in polynomial time, first consider $m = 0$. Then, $\alpha(\mathcal{C}) = 0$ for every nontrivial clustering $\mathcal{C}$, since it contains at least one cluster C_j with $\varphi(G[C_j]) = 0$. 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 the maximum.

Thus, intracluster 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 intracluster conductance becomes an $\mathcal{NP}$-hard problem. Finding a clustering with maximum intercluster conductance is $\mathcal{NP}$-hard as well, because it is at least as hard as finding a cut with minimum conductance.

2.3 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 intracluster edges together with nonadjacent pairs of nodes in different clusters within the set of all pairs of nodes. The function f counts the number of edges within all clusters while the function g counts the number of nonadjacent pairs belonging

to different clusters (Eq. 2).

$$\begin{aligned} f(\mathcal{C}) &= \sum_{i=1}^{k} |E(C_i)| \\ g(\mathcal{C}) &= \sum_{i=1}^{k} \sum_{j>i} |\{\{u,v\} \notin E \mid u \in C_i, v \in C_j\}| \\ \mathit{performance}(\mathcal{C}) &:= \frac{f(\mathcal{C}) + g(\mathcal{C})}{\frac{1}{2}n(n-1)} \end{aligned} \tag{2}$$

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 (Eq. 3).

$$1 - \mathit{performance}(\mathcal{C}) = \frac{2m(1 - 2\,\mathit{coverage}(\mathcal{C})) + \sum_{i=1}^{k} |C_i|(|C_i| - 1)}{n(n-1)} \tag{3}$$

Maximizing the performance is $\mathcal{NP}$-hard [Shamir et al. 2002]. There are several ways to extend the definition of performance for weighted graphs. For example, one can use more complex models for classifications, however, such models highly depend on the underlying application. Thus, we engineered two alternatives that integrate the weights in their counting schema. Since, a weighted analogon needs to assign a weight to node pairs that are not connected with an edge, an estimate or a corresponding interpretation is required. Therefore, let M be a meaningful upper bound on the values of ω (see Gaertler [2005] for a detail discussion of the meaning of M). The first version uses:

$$\begin{aligned} f(\mathcal{C}) &:= \sum_{i=1}^{k} \omega(E(C_i)) \\ g(\mathcal{C}) &:= \sum_{i=1}^{k} \sum_{j>i} M\,|\{\{u,v\} \notin E \mid u \in C_i, v \in C_j\}| \end{aligned}$$

The normalization factor is $1/2n(n-1)M$. The idea is to count the weight of the edges and assuming that not-existing edges "have" maximum weight. However, the weight of the intercluster edges is neglected. This can be integrated by modifying g:

$$g'(\mathcal{C}) := g(\mathcal{C}) + M\,|\overline{E(\mathcal{C})}| - \omega(\overline{E(\mathcal{C})})$$

By scaling this addition term accordingly, we can define the influence of the intercluster edges. Let $\vartheta \in [0,1]$ be a scaling parameter, then the complete formula is given by:

$$\mathit{performance}_w(\mathcal{C}) := \frac{f(\mathcal{C}) + g(\mathcal{C}) + \vartheta \cdot (M\,|\overline{E(\mathcal{C})}| - \omega(\overline{E(\mathcal{C})}))}{\frac{1}{2}n(n-1)M}$$

This process can also be applied to Eq. (3) that results in:

$$\tilde{f}(\mathcal{C}) := \sum_{i=1}^{k} \left(M \frac{1}{2} |C_i|(|C_i| - 1) - \theta\omega(E(C_i)) \right)$$
$$\tilde{g}(\mathcal{C}) := \omega(\overline{E(\mathcal{C})})$$

where $\theta \in [0, 1]$ is a scaling parameter controlling the influence of the intracluster edges. We used different symbols $\tilde{f}$ and $\tilde{g}$ to clarify that these functions count errors. The complete formula is:

$$performance_m(\mathcal{C}) = 1 - \frac{\tilde{f}(\mathcal{C}) + \tilde{g}(\mathcal{C})}{\frac{1}{2}n(n-1)M}$$

Note that both versions are the same for $\vartheta = \theta = 1$. In general, this is not true for other choices of ϑ and θ. In the following, we will only use $performance_w$ with scaling parameter $\vartheta = 1$.

More information about quality indices can be found in Gaertler [2005].

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- over intercluster connectivity and the second one repeatedly refines an initial partition based on intracluster 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 weighted adjacency matrix and $D(G)$ the diagonal matrix of the weighted node degrees. In order to define $D(G)^{-1}$, we require that G contains no isolated nodes.

3.1 Markov Clustering (MCL)

The key intuition behind *Markov clustering* (MCL) [van Dongen 2000, 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 a length of 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 renormalized after taking every entry to its rth power, $r \in \mathbb{R}^+$. (Algorithm 1, Steps 2–4)

Algorithm 1. Markov Clustering (MCL)

Input: $G = (V, E, \omega)$, expansion parameter e, inflation parameter r
$M \leftarrow M(G)$
while *M is not fixed point* **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

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 fixed point. A recurrent state of period $k \in \mathbb{N}$ is a matrix that is invariant under k expansions and inflations, and a fixed point is a recurrent state of period 1. It is argued that MCL is most likely to end up in a fixed point [van Dongen 2000]. The clustering is induced by connected components of the graph underlying the final matrix. Pseudocode for MCL is given in Algorithm 1. Except for the stop criteria, MCL is deterministic, and its complexity is dominated by the expansion operation, which essentially consists of matrix multiplication.

3.2 Iterative Conductance Cutting (ICC)

The basis of *iterative conductance cutting* (ICC) [Vempala et al. 2000] is to iteratively split clusters using minimum conductance cuts. Finding a cut with minimum conductance is $\mathcal{NP}$–hard, therefore, the following polylogarithmic approximation algorithm is used. Consider the node 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. Pseudocode 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.

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,

Algorithm 2. Iterative Conductance Cutting (ICC)

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

the connected components of $F(T, \tau)$ induce a clustering. Note that there are, at most, $n-1$ thresholds resulting in different forests. The resulting clustering of $F(T, \tau)$ does not depend on the actual MST T (see Lemma 3.4), therefore, we denote it with $\mathcal{C}(\tau)$. In order to verify this statement, we prove the following three lemmas, which handle locality in the connected components (Lemma 3.1), very similar MSTs (Lemma 3.2), and sequences of MSTs (Lemma 3.3).

LEMMA 3.1. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega: E \to \mathbb{R}^+$. Let $T = (V, E')$ be a spanning tree and V' the node set of a connected subtree T' in T. Then the following equation holds for every threshold τ:*

$$F(T, \tau) \restriction V' = F(T', \tau) \ . \tag{4}$$

PROOF. The clustering $F(T, \tau) \restriction V'$ of V' can be rewritten as

$$F(T, \tau) \restriction V' = \{C \cap V' | C \in F(T, \tau) \wedge C \cap V' \neq \emptyset\}$$

We prove Eq. (4) by using mutual inclusion. First, we show that the left side is included in the right one. Let $C' \in F(T, \tau) \restriction V'$ and $C \in F(T, \tau)$ such that $\emptyset \neq C' = C \cap V'$. Then, for every pair of nodes contained in C', there exists an unique path p in T such that each edge has a weight less than τ. Since T' is connected and spans V', every path in T connecting two nodes in V' is totally contained in T'. Thus, there exists a $C'' \in F(T', \tau)$ such that $C' \subseteq C''$. For every node pair in C'' there exists an unique path in T' such that each edge has a weight less than τ. This path is also a path in T with the same property; therefore $C' = C''$.

Second, we show that the right side is included in the left one. Let $C \in F(T', \tau)$, then there exists a unique path in T' between every pair in C such that each edge has weight less than τ. This path is also a path in T with the same property; thus, there exists a cluster $C' \in F(T, \tau)$ with $C \subseteq C'$. Moreover the following inclusion holds:

$$C = C \cap V' \subseteq C' \cap V'$$

Thus, it is sufficient to show that $C' \cap V' = C$. Suppose otherwise and let u be a node in C and v a node in $(C' \cap V') \setminus C$. Then there exists a unique path p connecting u and v in T such that every edge in p has weight less than τ. Since T' is connected and $u, v \in V'$, the path p has to be contained in T' as well. However, every path connecting u and v in T' contains an edge weight greater or equal to τ, otherwise v would be in C. This contradicts $C \subsetneq C' \cap V'$. □

LEMMA 3.2. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega\colon E \to \mathbb{R}^+$. Let $T = (V, E')$ and $T' = (V, E'')$ be two MSTs such that $E'' = E' \setminus \{e'\} \cup \{e''\}$. Then, the clusterings $F(T, \tau)$ and $F(T', \tau)$ are the same.*

PROOF. Since both trees T and T' are MST, both edges e' and e'' have the same weight. Furthermore, let $C = (V', E_C)$ denote the cycle formed by e'' and the path (in T) connecting its endnodes. This cycle also contains e'. The subgraph $(V', E_C \setminus \{e''\})$ is the unique path in T connecting the two endnodes of e''. Suppose this path does not contain e', then it is also a path in T'. Thus C is contained in T', which contracts T' being a tree.

Using Lemma 3.1, it is sufficient to show the following equality

$$F(T, \tau) \restriction C = F(T', \tau) \restriction C$$

In the case that $\omega(e') < \tau$, both clusterings equal $\{V'\}$ and are thus the same. Therefore, let us assume that $\omega(e') \geq \tau$. We divide the cycle into subpaths p_i such that each edge in the paths has weight less than τ. Since this division is independent of e' and e'', we obtain the following equation:

$$F(T, \tau) \restriction C = \{V_i | V_i \text{ is the node set of path } p_i\} = F(T', \tau) \restriction C$$

which concludes the lemma. □

LEMMA 3.3. *Let $G = (V, E, \omega)$ be an undirected weighted graph with $\omega\colon E \to \mathbb{R}^+$, $T = (V, E')$ and $T' = (V, E'')$ be two different MSTs. Then there exists an MST $\widetilde{T} = (V, \widetilde{E})$ such that*

$$\exists\, e'' \in E'' \setminus E', e' \in E'\colon \widetilde{E} = E' \setminus \{e'\} \cup \{e''\}$$

PROOF. Let $\Delta E' := E'' \setminus E'$ be the set of tree edges (in T') that are not contained in T. If $|\Delta E'| = 1$ then $\widetilde{T} = T'$ suffices. Otherwise, let $e' \in \Delta E'$. This edge splits T' and thus partitions V into two nonempty parts V_1 and V_2. Since T is a spanning tree that does not contain e', there exists an edge e that connects V_1 and V_2. Both edges e' and e have the same weight, otherwise not both trees T and T' could have minimum weight. We define $\widetilde{T} := (V, E' \setminus \{e\} \cup \{e'\})$, it is still spanning, and has the same weight as T; therefore, it is an MST. □

LEMMA 3.4. *The clustering induced by the connected components of $F(T, \tau)$ is independent of the particular MST T.*

PROOF. Let T and T' be two different MSTs. By Lemma 3.3, we can construct a sequence of MSTs such that every two consecutive MSTs differ in exactly one edge. Using Lemma 3.2, the clusterings induced by such a pair are the same, therefore, the clustering of T and T' are the same. □

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 universality allows targeting of different properties of a clustering. Pseudocode for GMC is given in Algorithm 3. Except for the eigenvector computations, GMC is deterministic. Note that, opposite to 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

Algorithm 3. Geometric MST Clustering (GMC)

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

and space complexity of the main algorithm is dominated by the MST computation. GMC combines two proved concepts from geometric clustering and graph partitioning.

4. EXPERIMENTAL EVALUATION

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

4.1 Random Uniform Clustered Graphs

It is possible to obtain a random clustered graph with n nodes with (almost) uniform cluster size by the following process Brandes et al. [2003]: First, a random partition generator $\mathcal{P}(n, s, v)$ 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 $\frac{s}{v}$. The parameter k depends on the choice of n, s, and v. 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 intracluster edges with probability p_{in} and intercluster edges with probability p_{out}. In case a generated graph is not connected, additional edges combining the components are added.

A disadvantage of this process is that the "last" cluster P_k is possibly significantly smaller than the others in order to achieve a graph with exactly n nodes. Correspondingly, indices that depend on cluster size, such as intercluster conductance can produce artefacts. In order to obtain an undisturbed behavior, we relaxed the size constraint i.e., if the last cluster size variable $|P_k|$ is too small or too large but the number of unassigned or additional nodes is less than one-third of the expected cluster size, we add or delete the corresponding nodes. However, if the gap exceeds one-third, we reject the partition and generate a new one. This may bias the generation process, yet we observed only few rejections during our experiments.

In order to judge both weighted and unweighted versions of the indices, we extended the above generation process to produce random weights as well. Since the weight should reflect the given partitioning, a weight from $[0, p_{\text{out}}]$ for each intercluster edge and from $[p_{\text{in}}, 1]$ each for intracluster edge is uniformly at random selected and assigned. In Addition, small disturbances or shuffles

could yield more realistic weightings. However, we chose not to perform such postprocessings in order to be independent of the used model and to keep the parameter set small.

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_{in}, p_{out} are considered. We set $v = 4$ and choose s uniformly at random from $\{\frac{n}{\ell} | \log n \leq \ell \leq \sqrt{n}\}$. Experiments are performed for $n = 1000$. On the one hand, all combinations of probabilities p_{in} and p_{out} at a distance of 0.05 are considered. On the other hand, a second group of experiments used a dynamic adaptation of p_{out}. Partial results of the tests with the above given parameters is that the ratio of p_{in} and p_{out} hardly reflect the ratio of potential intra- and intercluster edges. Therefore, a scaling parameter f is introduced to replace p_{out}, which estimates a suitable p_{out} value to bound the number of expected intercluster edges in terms of expected intracluster edges. The experiments are performed with $n = 1000$, inner probability p_{in} between 0.7 and 0.95, with a step size of 0.05, and the scaling parameter f between 0.25 and 2.25 with a step size of 0.25.

The free parameters of the algorithms are set to $e = 2$ and $r = 2$ in MCL, $\alpha^* = 0.4$ and $\alpha^* = 0.2$ in ICC, and dimension $d = 2$ in GMC. As objective function *quality* in GMC, *coverage*, *performance*, intercluster conductance δ, as well as the geometric mean of *coverage*, *performance* and δ is considered.[1]

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 Java (1.4.2). In addition, we used yFiles.[2] and colt.[3] The experiments were performed on an AMD Opteron 248 with 2.2 GHz on a Linux 2.6 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.

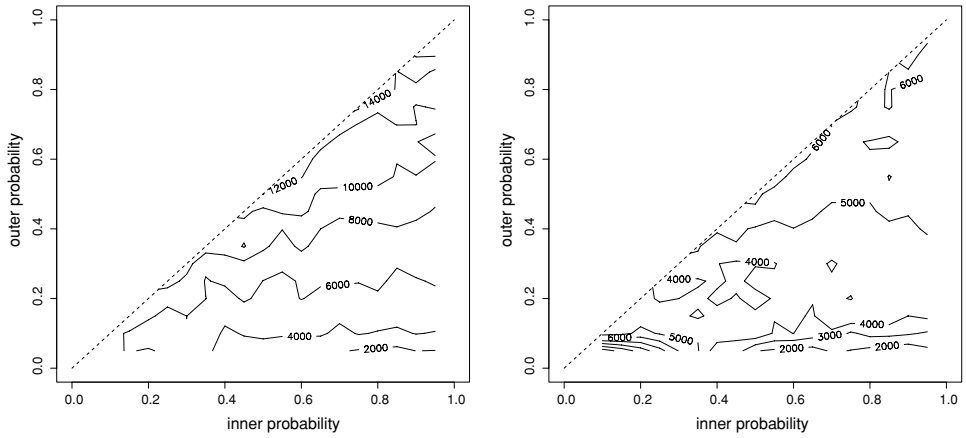
4.3.1 *Running Time.* All presented clustering algorithms were implemented using sophisticated data structure and software engineering techniques. However, there are certain limitations, especially with respect to runtime measurements in Java, which are very difficult. Since such measurements are rarely significant on small scales, none of the implementations were especially optimized with respect to running time. Nevertheless, the following results show certain tendencies.

---

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

[2] `http://www.yworks.com`

[3] `http://hoschek.home.cern.ch/hoschek/colt/`



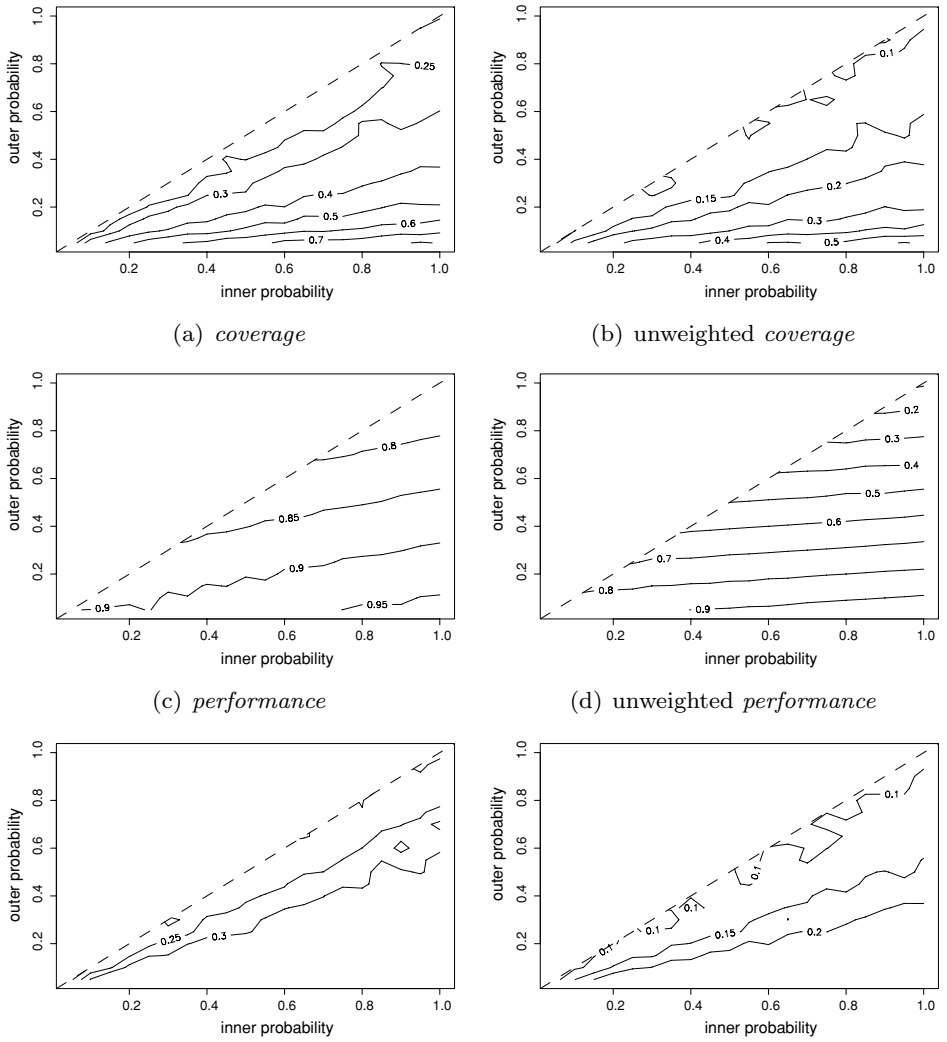
(a) GMC (b) ICC

Fig. 1. Running time of GMC and ICC, where the x axis represents the inner probability p_{in} and the y axis shows the outer probability p_{out}.

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 $\alpha^* = 0.2$ than for $\alpha^* = 0.4$. Note the coarseness of the clustering computed by ICC depends on the value of α^*. In contrast, all versions of GMC were equally fast, except those versions that included intracluster conductance.

On sparse graphs, GMC and ICC perform equally will, while ICC was up to two times faster on dense graphs. The complete results are given in Figure 1. ICC performs on dense graphs so well since the approximation of intracluster conductance yield large values and, thus, only a few number of cuts are calculated. In other words, the divisive structure of the ICC is more suitable for dense graphs than for sparse ones, while the agglomerative GMC benefits from a sparse edge set. Not very surprisingly the runtime depends much more on the outer probability p_{out} than on the inner probability p_{in}, which results from the fact that the number of potential intercluster edges is much larger than the number of potential intracluster edges (for most values of k).

4.3.2 *Indices for the Initial Clustering.* Studying *coverage*, *performance*, intra- and intercluster 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} (Figure 2a–d). 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_{out}. This is because *performance* considers both, edges inside and nonedges between



(a) *coverage* (b) unweighted *coverage*

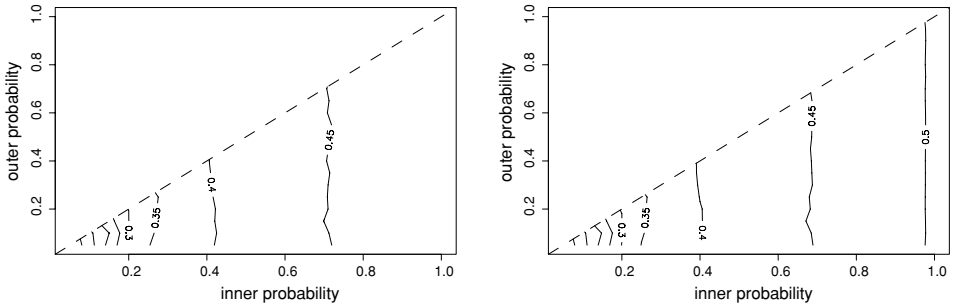
(c) *performance* (d) unweighted *performance*

(e) intercluster conductance (f) unweighted intercluster conductance

Fig. 2. Indices of the initially generated clustering I/II.

clusters, while *coverage* measures only the fraction of intracluster edges within all edges.

Both conductance versions have a very different behavior. Intercluster conductance is very homogenous for large ranges of the parameters. However, it still performs according to the general intuition, i.e., it has smaller values for almost uniformly random graphs (instances close to the dashed lines) than those instances with significant clustering. In contrast to the other three indices, intracluster conductance shows a completely different behavior with respect to the choices of p_{in} and p_{out}. Actually, intracluster conductance does not depend



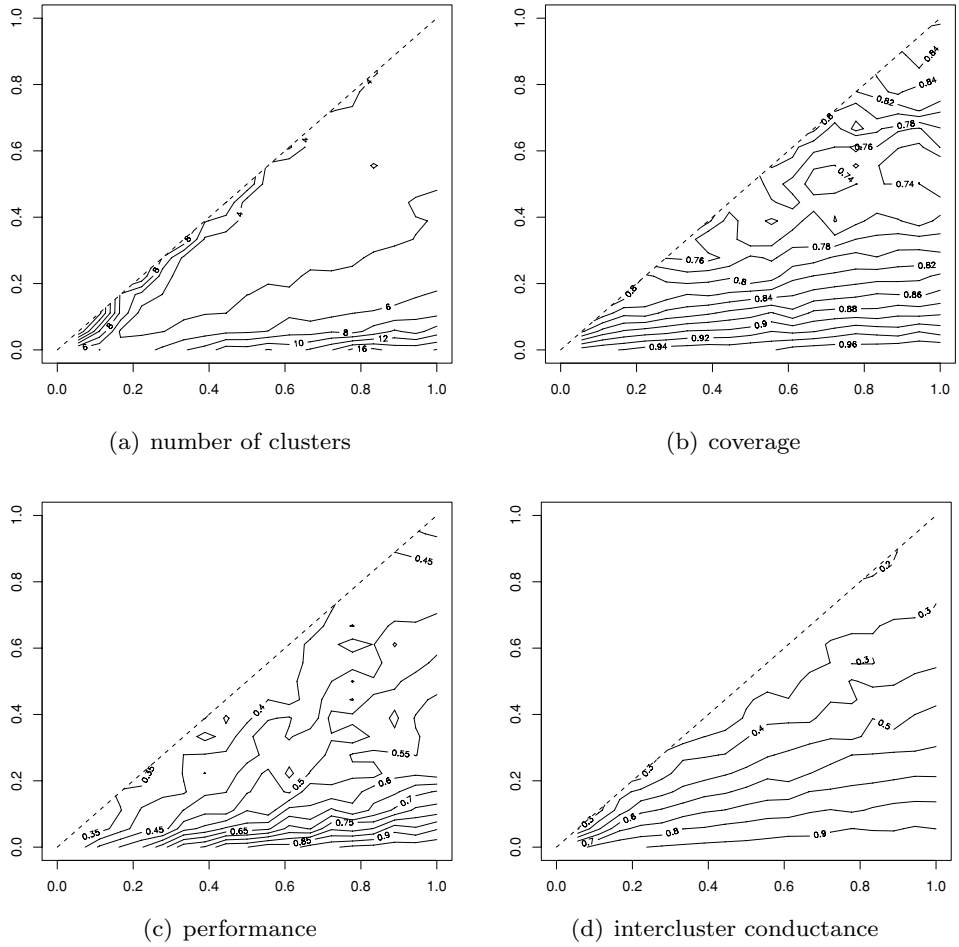
(a) intracluster conductance (b) unweighted intracluster conductance

Fig. 3. Indices of the initially generated clustering II/II.

on p_{out} (Figure 3a and b). This is not very surprising, since the bottleneck cuts of the clusters should be independent of p_{out}. Although the two indices show some artifical behavior with respect to the generated instances, it does not deny their usability for clusterings and qualitative evaluation.

4.3.3 *Comparing the Algorithms.* Figures 4, 5, and 6 show the different quality indices for the different algorithms for the first group of experiments. All diagrams show the inner probability p_{in} as x axis and the outer probability p_{out} as y axis. A significant observation when comparing the three algorithms with respect to the quality indices regards their behavior for dense graphs. All algorithms (Figure 4a, 5a, and 6a) have a tendency to return trivial or very coarse clusterings containing only few clusters. As mentioned previously, this is because of the fact that the number of potential intercluster edges is much larger than the number of potential intracluster edges. In contrast for sparse graphs, ICC and MCL only find clusterings with many clusters. This suggests modifications to at least incorporate bound for the number of clusters in order to avoid too coarse 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 nontrivial clusterings. In contrast, it is easy to modify GMC in such a way that only clusterings with bounded (from below, above, or both) numbers of clusters are computed. This is accomplished by limiting the search space of τ.

Both ICC and MCL are comparably good with respect to *performance*, although neither of them optimizes it explicitly. While GMC is not as good with respect to *performance*, it outperforms MCL with respect to intercluster conductance and ICC with respect to *coverage*. Still, all three algorithms find clusterings with acceptable quality. Further more, the calculated clusterings similarly react to changes in the generation parameters, i.e., the quality drops



(a) number of clusters

(b) coverage

(c) performance

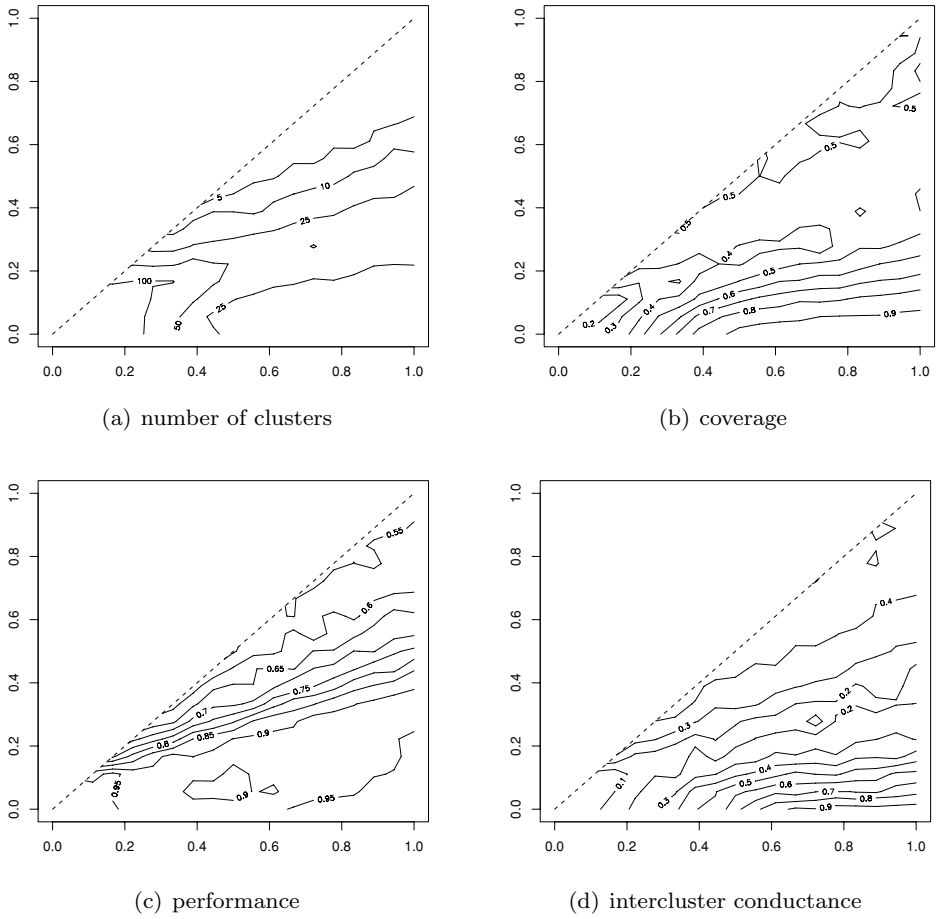
(d) intercluster conductance

Fig. 4. GMC using geometric mean of *coverage*, *performance*, and intercluster conductance.

when approaching random graphs (diagonal). More precisely, GMC (Figure 4d) and ICC (Figure 5d) are more sensitive (with respect to intercluster conductance) than the initial clustering (Figure 2e).

Further variations of the algorithms, e.g., ICC with $\alpha = 0.2$ and different versions of the GMC can be found in Appendix A.

The results of the second group of experiments are shown in Figures 7–9. All diagrams show the inner probability p_{in} as x axis and the scaling parameter f as y axis. Recall that f roughly estimates the ratio of (expected) inter- to (expected) intracluster edges. Intuitively speaking, the parameter f is inversely proportional to the significance of the initial clustering. Figures 7–9 clearly illustrate that both GMC and ICC find a clustering that is very similar to the initial one with respect to quality.



(a) number of clusters

(b) coverage

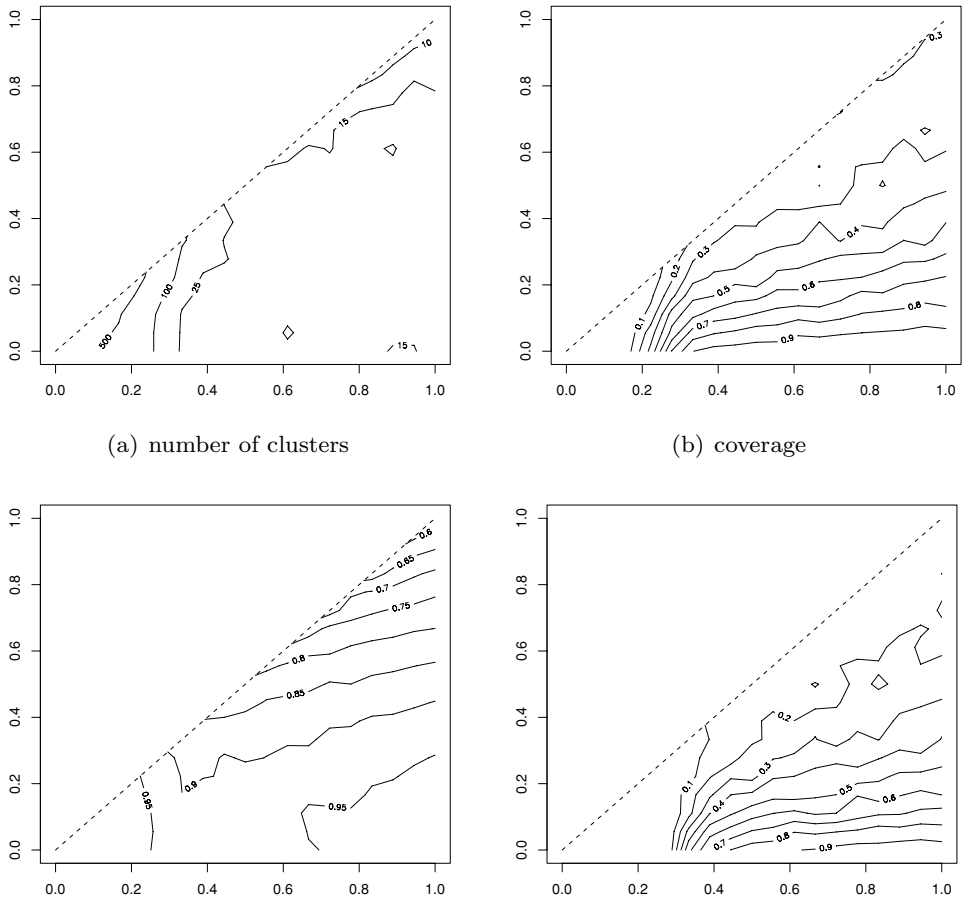
(c) performance

(d) intercluster conductance

Fig. 5. ICC with $\alpha = 0.4$.

5. CONCLUSION

The experimental study confirms the promising expectations about MCL, i.e., in many cases MCL seems to perform well with respect to quality. However, MCL often generates clusterings of inappropriate size. Moreover, MCL is very slow. The theoretical result on ICC is reflected by the experimental study, i.e.,ICC computes clusterings that are also good with respect to other indices. However, there is the suspicious that the index intracluster conductance does not measure the quality of a clustering appropriately. Indeed, the experimental study shows that all four cluster indices have weaknesses. Comparing the original versions of the measures to the new weighted formulations, which need not to be straight forward or unique, the study further demonstrates that both sets exhibit a similar behavior. 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. Although the indices exhibited weaknesses,



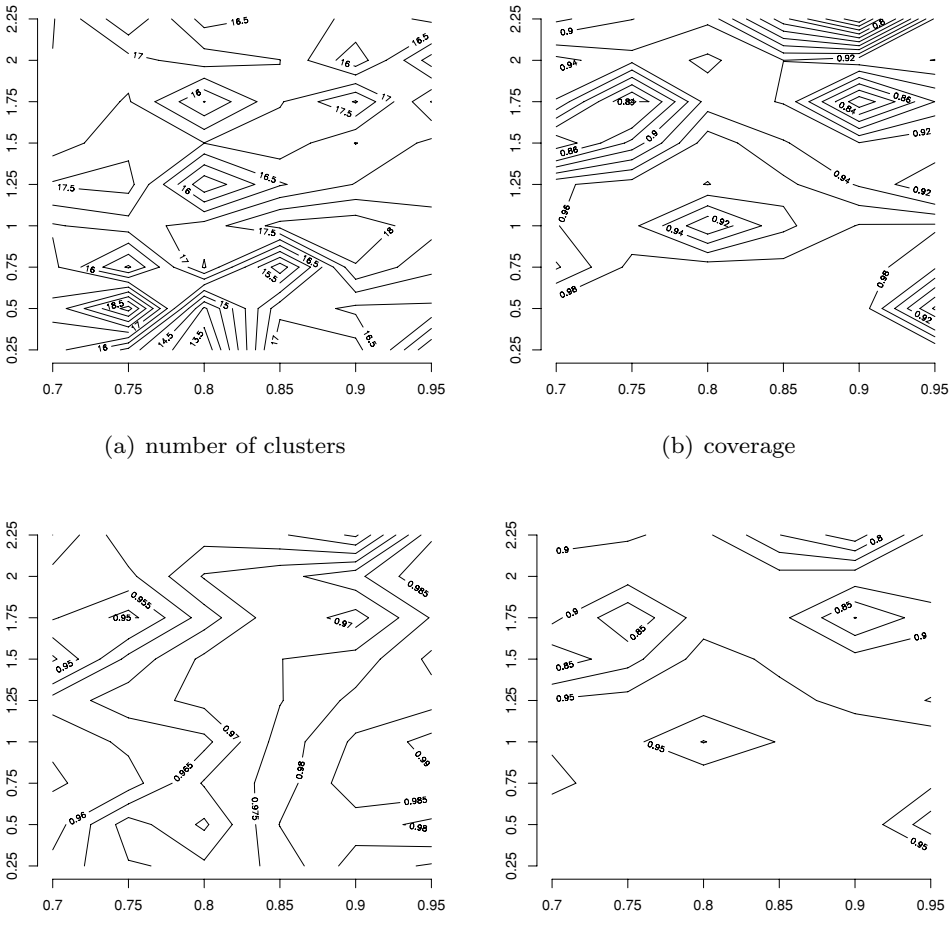
(a) number of clusters (b) coverage

(c) performance (d) intercluster conductance

Fig. 6. MCL.

GMC (with different quality functions) performed comparably well with respect to the other algorithms. More precisely, the obtain evaluation with the original four indices partially reflected the different optimization criteria. Moreover, refinement of the embedding used by GMC offers additional potential. Thus, 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, yet sparse, graphs.



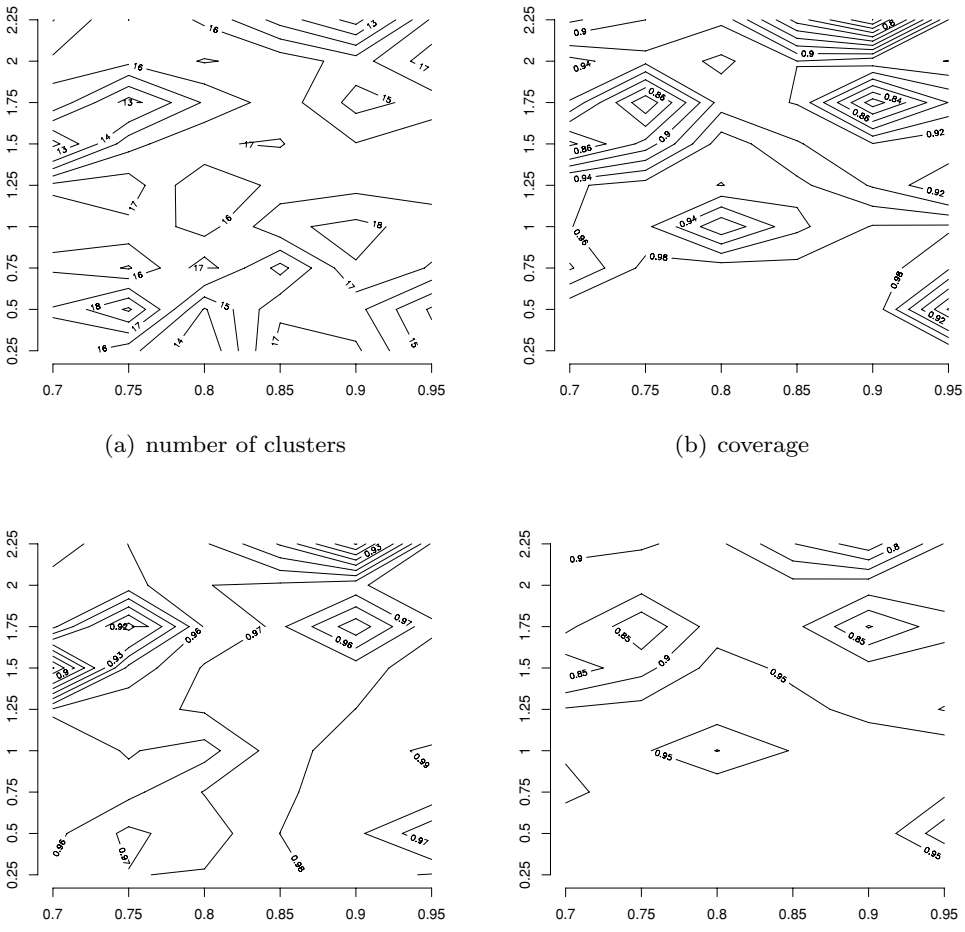
(a) number of clusters

(b) coverage

(c) performance

(d) intercluster conductance

Fig. 7. initial clustering.



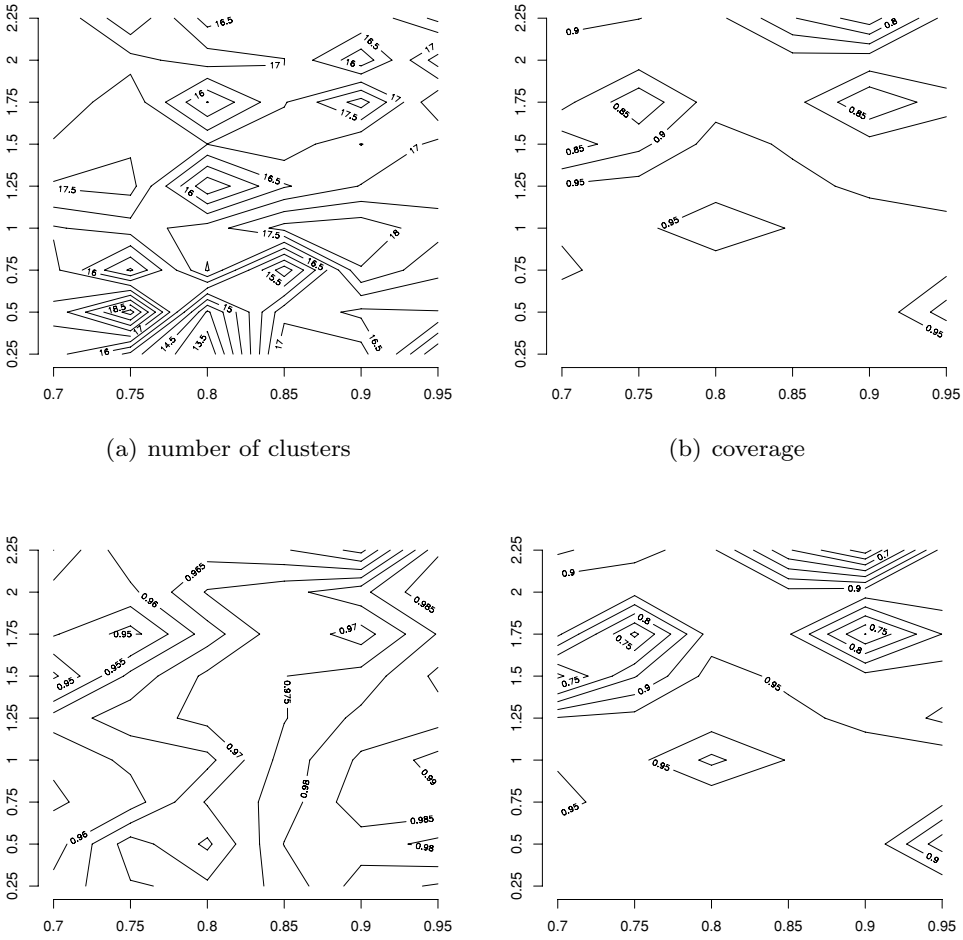
(a) number of clusters

(b) coverage

(c) performance

(d) intercluster conductance

Fig. 8. GMC using geometric mean of *coverage*, *performance*, and intercluster conductance.



(a) number of clusters

(b) coverage

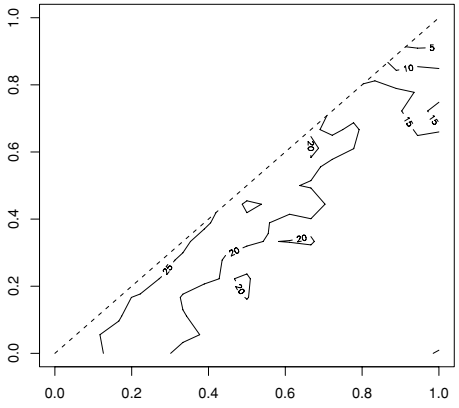
(c) performance

(d) intercluster conductance

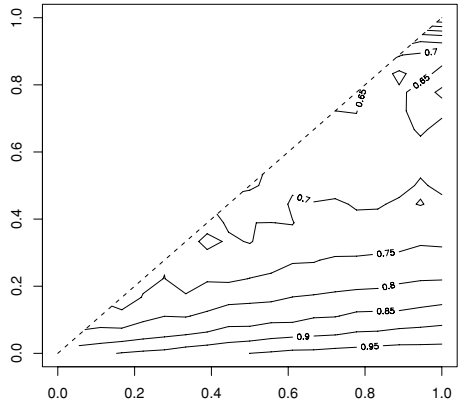
Fig. 9. ICC with $\alpha = 0.4$.

APPENDIX. COMPARING FURTHER ALGORITHMS

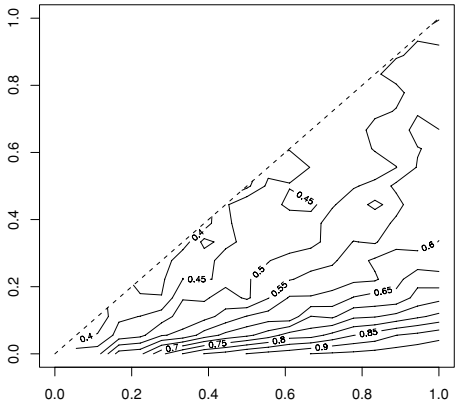
This section contains results of further parameter variations of the presented algorithms.



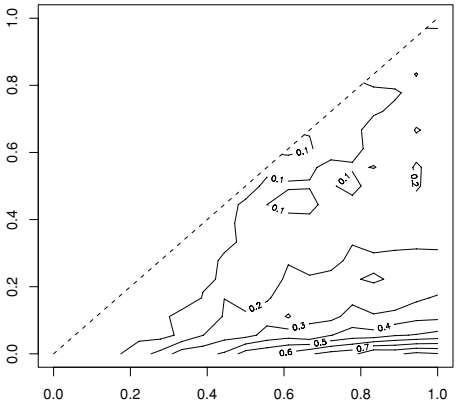
(a) number of clusters



(b) coverage

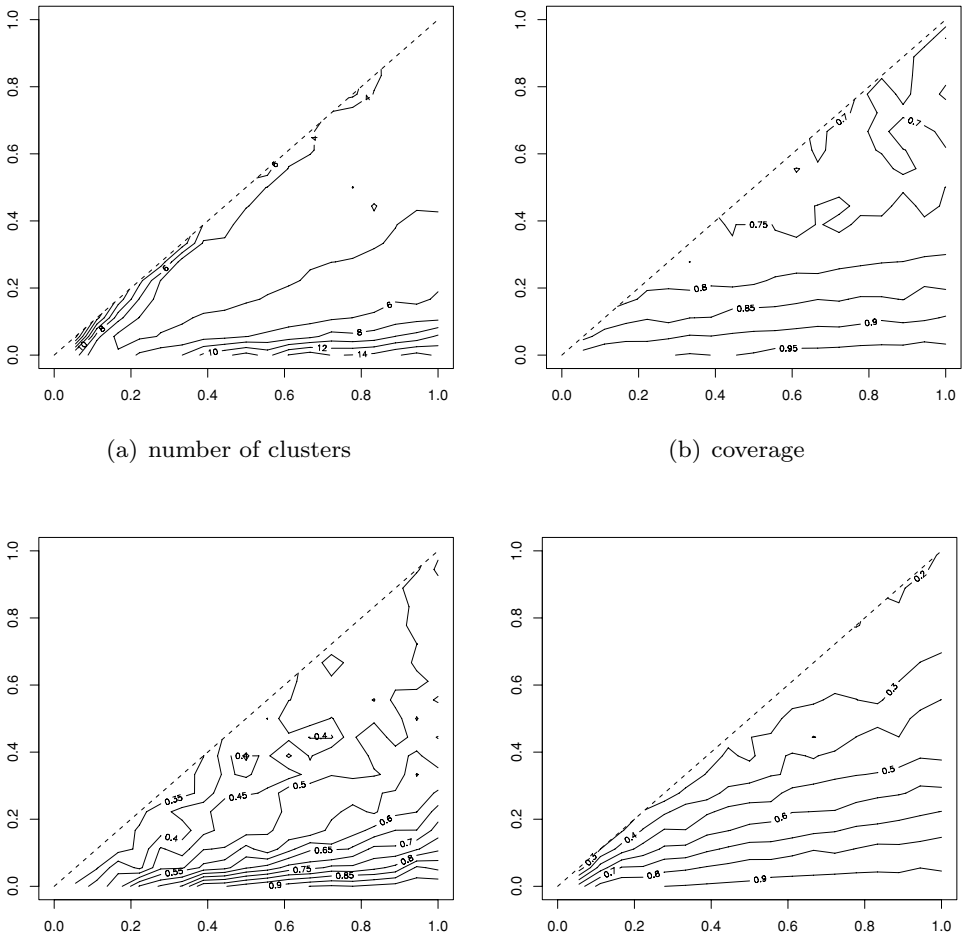


(c) performance



(d) intercluster conductance

Fig. A1. GMC using $\sqrt[3]{coverage \cdot performance^3}$.



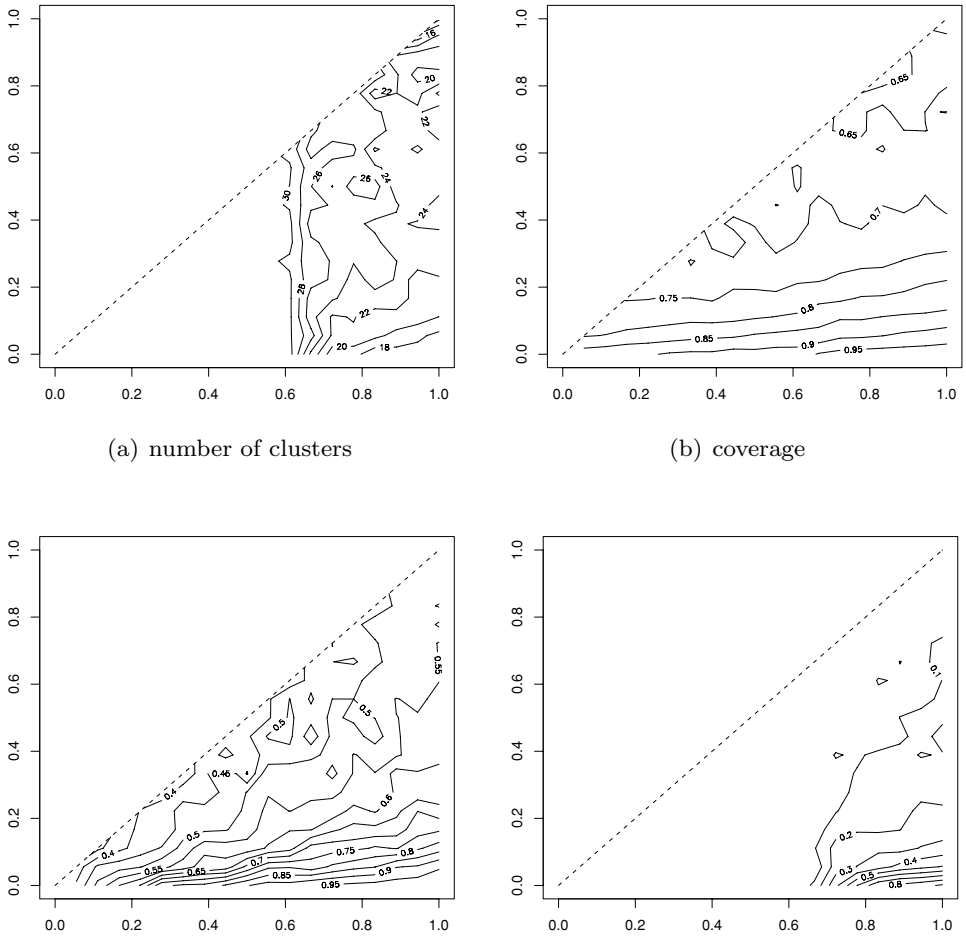
(a) number of clusters

(b) coverage

(c) performance

(d) intercluster conductance

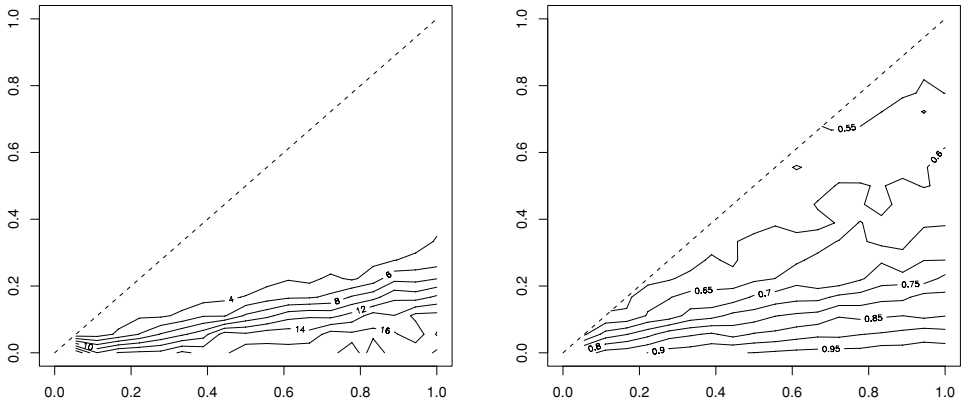
Fig. A2. GMC using $\sqrt[3]{\text{inter-cluster conductance} \cdot \textit{performance}^3}$.



(a) number of clusters (b) coverage

(c) performance (d) intercluster conductance

Fig. A3. GMC using *performance*.



(a) number of clusters (b) coverage

(c) performance (d) intercluster conductance

Fig. A4. ICC with $\alpha = 0.2$.

REFERENCES

AUSIELLO, G., CRESCENZI, P., GAMBOSI, G., KANN, V., AND MARCHETTI-SPACCAMELA, A. 2002. *Complexity and Approximation—Combinatorial Optimization Problems and Their Approximability Properties*, 2nd ed. Springer-Verlagn New York.

BRANDES, U., GAERTLER, M., AND WAGNER, D. 2003. Experiments on graph clustering algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*. Lecture Notes in Computer Science, vol. 2832. 568–579.

CHUNG, F. R. K. AND YAU, S.-T. 1994. A near optimal algorithm for edge separators. In *Proceeding of the 26th Annual ACM Symposium on Theory of Computing*. ACM Press, New York. 1–8.

CHUNG, F. R. K. AND YAU, S.-T. 1997. Eigenvalues, flows and separators of graphs. In *Proceeding of the 29th Annual ACM Symposium on Theory of Computing*. ACM Press, New York. 1–8.

CLAUSET, A., NEWMAN, M. E. J., AND MOORE, C. 2004. Finding community structure in very large networks. *Physical Review E 70,* 066111.

GAERTLER, M. 2005. Clustering. In *Network Analysis: Methodological Foundations*, U. Brandes and T. Erlebach, Eds. Lecture Notes in Computer Science, vol. 3418. Springer-Verlag, New York. 178–215.

HAREL, D. AND KOREN, Y. 2001. On clustering using random walks. In *Proceedings of the 21st Conference on Foundations of Software Technology and Theoretical Computer S*. Lecture Notes in Computer Science, vol. 2245. Springer-Verlag, New York. 18–41.

HARTUV, E. AND SHAMIR, R. 2000. A clustering algorithm based on graph connectivity. *Information Processing Letters 76,* 4-6, 175–181.

HO, T. B., KAWASAKI, S., AND NGUYEN, N. B. 2003. *Documents Clustering Using Tolerance Rough Set Model and Its Application to Information Retrieval*. Physica-Verlag GmbH, Heidelberg. 181–196.

JAIN, A. K. AND DUBES, R. C. 1988. *Algorithms for Clustering Data*. Prentice Hall, Englewood, Cliffs, NJ.

JAIN, A. K., MURTY, M. N., AND FLYNN, P. J. 1999. Data clustering: A review. *ACM Computing Surveys 31,* 3, 264–323.

NEWMAN, M. E. J. AND GIRVAN, M. 2004. Finding and evaluating community structure in networks. *Physical Review E 69,* 026113.

SHAMIR, R., SHARAN, R., AND TSUR, D. 2002. Cluster graph modification problems. In *Proceedings of the 28th International Workshop on Graph-Theoretical Conecpts in Computer Science (WG)*. Lecture Notes in Computer Science, vol. 2573. Springer-Verlag, New York. 379–390.

SPIELMAN, D. A. AND TENG, S.-H. 1996. Spectral partitioning works: Planar graphs and finite element meshes. In *Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science (FOCS'96)*. 96–106.

VAN DONGEN, S. M. 2000. Graph Clustering by Flow Simulation. Ph.D. thesis, University of Utrecht.

VEMPALA, S., KANNAN, R., AND VETTA, A. 2000. On clusterings—good, bad and spectral. In *Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00)*. 367–378.

WAGNER, D. AND WAGNER, F. 1993. Between min cut and graph bisection. In *MFCS '93: Proceedings of the 18th International Symposium on Mathematical Foundations of Computer Science*, A. M. Borzyszkowski and S. Sokolowski, Eds. Lecture Notes in Computer Science, vol. 711. Springer-Verlag, New York. 744–750.

ZAHN, C. T. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers C-20*, 68–86.