Experiments on Density-Constrained Graph Clustering*

Robert Görke† robert.goerke@kit.edu
Andrea Schumm† andrea.schumm@kit.edu
Dorothea Wagner† dorothea.wagner@kit.edu

Abstract

Clustering a graph means identifying internally dense subgraphs which are only sparsely interconnected. Formalizations of this notion lead to measures that quantify the quality of a clustering and to algorithms that actually find clusterings. Since, most generally, corresponding optimization problems are hard, heuristic clustering algorithms are used in practice, or other approaches which are not based on an objective function. In this work we conduct a comprehensive experimental evaluation of the qualitative behavior of greedy bottom-up heuristics driven by cut-based objectives and constrained by intracluster density, using both real-world data and artificial instances. Our study documents that a greedy strategy based on local movement is superior to one based on merging. We further reveal that the former approach generally outperforms alternative setups and reference algorithms from the literature in terms of its own objective, while a modularity-based algorithm competes surprisingly well. Finally, we exhibit which combinations of cut-based inter- and intracluster measures are suitable for identifying a hidden reference clustering in synthetic random graphs. Our results serve as a guideline to the usage of bicriterial, cut-based measures for graph clusterings.

1 Introduction

Graph clustering aims at finding subsets of vertices that are densely connected with each other but sparsely connected with the remainder of the graph. In the last decades, interest in graph clustering algorithms has grown rapidly, with applications ranging from customer recommendation systems to the analysis of networks describing social ties or protein-protein interaction. A variety of measures have been proposed, which are used to assess and compare different clusterings and to guide the design of algorithms. Traditional methods from algorithmics often focus on sparse cuts with respect to measures like conductance [18] or expansion [16], while, independent from that, a measure called modularity [22] proved to yield meaningful clusterings on a wide range of application data.

Recently, we systematically assembled a range of self-evident intracluster density and intercluster sparsity measures for clusterings, where the latter are based on conductance , expansion and density of the cuts induced by the clusters [13]. We further formally stated the problem Density-Constrained Clustering (DCC), where the objective is to optimize intercluster sparsity with the constraint that the intracluster density must exceed a given threshold. As optimal polynomial-time algorithms for DCC are unknown, we investigated how different combinations of intracluster sparsity and intercluster density measure influence the efficiency of a greedy optimization strategy based on cluster merging. However, little is known about its qualitative behavior in practical scenarios, and an experimental evaluation of DCC has yet been missing.

Our Contribution. We provide a comprehensive study of the practical behavior of greedy graph clustering heuristics driven by cut-based objectives and constrained by intracluster density. We give evidence that, in general, greedy algorithms based on local vertex moves lead to better quality than the corresponding merge-based algorithm. We then compare the move-based algorithm to a set of reference algorithms from the literature, both with respect to the objective of DCC and their ability to reconstruct planted partitions in a family of synthetic graphs. We find that the greedy move algorithm compares favorably to most reference algorithms in the context of DCC, while a comparison with the modularity-based algorithm shows that optimizing modularity implicitly yields good results for some variants of DCC. Experiments with planted partition graphs suggest that certain combinations of inter- and intracluster measures are effective in finding the hidden clustering, while others clearly fail. Together with observations about the number of identified clusters, this yields valuable insights about the behavior of the respective intra- and intercluster density measures.

*This work was partially supported by the DFG under grant WA 654/19-1

†Institute of Theoretical Informatics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany

Table 1: Density measures

intracluster density		
global	gid	$\frac{\sum_{C\in\mathcal{C}} m_C}{\sum_{C\in\mathcal{C}} \binom{n_C}{2}}$
minimum	mid	$\min_{C\in\mathcal{C}} \frac{m_C}{\binom{n_C}{2}}$
average	aid	$\frac{1}{\lvert\mathcal{C}\rvert} \sum_{C\in\mathcal{C}} \frac{m_C}{\binom{n_C}{2}}$
intercluster density		
global	gxd	$\frac{\sum_{A\neq B\in\mathcal{C}} m_{A,B}}{\sum_{A\neq B\in\mathcal{C}} n_A n_B}$
maximum	mixd	$\max_{C\in\mathcal{C}} \frac{x_C}{n_C n_{V\setminus C}}$
average	aixd	$\frac{1}{\lvert\mathcal{C}\rvert} \sum_{C\in\mathcal{C}} \frac{x_C}{n_C n_{V\setminus C}}$
intercluster conductance		
maximum	mixc	$\max_{C\in\mathcal{C}} \frac{m_{C,V\setminus C}}{\min\{v_C, v_{V\setminus C}\}}$
average	aixc	$\frac{1}{\lvert\mathcal{C}\rvert} \sum_{C\in\mathcal{C}} \frac{m_{C,V\setminus C}}{\min\{v_C, v_{V\setminus C}\}}$
intercluster expansion		
maximum	mixe	$\max_{C\in\mathcal{C}} \frac{m_{C,V\setminus C}}{\min\{n_C, n_{V\setminus C}\}}$
average	aixe	$\frac{1}{\lvert\mathcal{C}\rvert} \sum_{C\in\mathcal{C}} \frac{m_{C,V\setminus C}}{\min\{n_C, n_{V\setminus C}\}}$
intercluster edges		
global	nxe	$\sum_{A\neq B\in\mathcal{C}} m_{A,B}$
modularity		
global	mod	$\frac{\sum_{C\in\mathcal{C}} m_C}{m} - \frac{\sum_{C\in\mathcal{C}} v_C^2}{4m^2}$

Related Work. Related clustering algorithms are Iterative Conductance Cutting [18], Markov-Clustering [9], Geometric MST Clustering [5] and a modularity-based greedy algorithm based on vertex moves [23]; we use these as reference algorithms. Kannan et al. propose to minimize the cut between, subject to a guaranteed conductance within clusters [18], which is closely related to the DCC. They further show that Iterative Conductance Cutting has polylogarithmic approximation guarantees on both of these measures. Brandes et al. conduct an experimental study on the performance of Iterative Conductance Cutting, Markov-Clustering and Geometric MST Clustering, both with respect to quality and running times [6]. A similar, but more recent study can be found in [19]. Flake et al. give a clustering algorithm with provable, but interdependent bounds on both intra- and a variant of intercluster expansion. The notion of modularity was introduced in [22], an extensive and recent overview of the research on it can be found in [11]. Apart from these, there is a huge number of publications on graph clustering, for an overview see [17, 3].

2 Preliminaries

Notation. Let $G = (V, E)$ be an undirected, unweighted, and simple graph, i.e. G is loopless and has no parallel edges. In the following, n will always denote the number of vertices and m the number of edges in G. For two subsets A and B of V, $m_{A,B} := |\{\{u, v\} \in E \mid u \in A, v \in B\}|$ is the number of edges between A and B, $n_A := |A|$ is the number of vertices in A, $m_A := |E(A)|$ is its number of intracluster edges and $x_A := m_{A,V\setminus A}$ the number of intercluster edges incident to A. Further, the *volume* v_A of A is defined as $v_A := \sum_{v\in A} \deg(v)$. The *conductance* of a cut (S, T) measures the bottleneck between S and T, defined as $\frac{m_{S,T}}{\min\{v_S, v_T\}}$; *expansion* substitutes volume by cardinality: $\frac{m_{S,T}}{\min\{n_S, n_T\}}$. The *density* (or *sparsity*) of a cut is $\frac{m_{S,T}}{n_S n_T}$, which equals the *uniform minimum-ratio cut*. We restrict ourselves to disjoint clusters in this work, this means, if $\mathcal{C} = \{C_1, \ldots, C_k\}$ is a partition of V, we call $\mathcal{C}$ a *clustering* of G and the sets C_i *clusters*. The cluster containing vertex v is $\mathcal{C}(v)$ and the clustering that results from moving vertex v to cluster D, i.e. $\big(\mathcal{C}\setminus\{\mathcal{C}(v), D\}\big) \cup \{\mathcal{C}(v)\setminus v, D\cup\{v\}\}$, is abbreviated by $\mathcal{C}_{v\to D}$. A clustering is *trivial* if either $k = 1$ (*all-clustering*), or each cluster contains only one element (*singletons*). We identify a cluster C with the set of nodes it constitutes and with its vertex-induced subgraph of G. Then $E(\mathcal{C}) := \bigcup_{C\in\mathcal{C}} E(C)$ are called *intracluster* edges and $E \setminus E(\mathcal{C})$ *intercluster* edges. A *clustering measure* is a function that maps clusterings to real numbers, thereby assessing the quality of a clustering. We define high quality to correspond to high (low) values of intracluster (intercluster) measures and will always denote intracluster density measures with i and intercluster density measures with x, unless otherwise stated.

Intracluster Density and Intercluster Sparsity Measures. All intercluster measures we use are based on *cuts* or *k-way cuts*. Separating a single cluster from the remaining vertices induces a cut, whose sparsity can be evaluated using density, conductance or expansion. This defines a set of sparsity values for the whole clustering, from which we can either compute the average or the maximum, yielding *maximum/average intercluster density/conductance/expansion* (mixd, aixd, mixc, aixc, mixe and aixe)[1]. Another point of view is to evaluate the clustering as a whole, i.e. to assess the sparsity of the induced k-way cut directly. We do this by either counting the number of intercluster edges (nxe) or by dividing the number of intercluster edges by the maximum possible number, i.e. the number of intercluster pairs (gxd). It is possible to use similar, cut-based measures for intracluster density. However, even evaluating these measures for a given clustering is $\mathcal{NP}$-hard, such

[1] Note that we keep the i in the abbreviations, although in contrast to [13], we do not distinguish between pairwise and isolated measures

that clustering algorithms usually work with approximations or bounds [18, 10, 6]. As we intend to use intracluster density measures as constraints in greedy bottom-up algorithms, it is crucial to be able to evaluate them efficiently. We therefore use a more practical approach and define intracluster density as the ratio of the number of intracluster edges and the number of intracluster pairs. Evaluating this globally leads to *global intracluster density* (gid), whereas the average and minimum of all clusters yields *average* and *minimum intracluster density* (aid and mid).

Table 1 summarizes the formalizations of all measures considered. Note that, in contrast to the set of measures used in [13], we omit the notions of *pairwise* densities as they turned out to be very prone to local minima if used with greedy bottom-up algorithms. Although it does not quite fit into this classification, Table 1 also includes the objective used by one of the reference algorithms, *modularity*, which simultaneously assesses intracluster density and intercluster sparsity by subtracting from the fraction of intracluster edges the expectation of this value in a random graph (note that high modularity corresponds to high quality).

Density-Constrained Clustering. Density-Constrained Clustering (DCC) is the problem of optimizing intercluster density while retaining guarantees on the intracluster density. Considering each combination of intracluster and intercluster measure listed in Table 1 leads to a family of optimization problems. Slightly abusing the notation, we consider modularity as an intercluster density objective in this context.

Problem 1. (Density-Constrained Clustering) *Given a graph $G = (V, E)$, among all clusterings with an intracluster density of no less than α, find a clustering $\mathcal{C}$ with optimum intercluster quality.*

3 Greedy Algorithms for Density-Constrained Clustering

The following generic greedy algorithms heuristically minimize (maximize) the objective function of DCC for all density measures considered.

Greedy Merge (GM). Starting from singletons, the algorithm greedily merges pairs of clusters. In each step, among all pairs of clusters whose merge does not violate the constraint on the intracluster density, the merge with the largest benefit to the intercluster density is performed. We recently proposed this algorithm in the context of DCC [13] and classified combinations of intercluster and intracluster density with respect to the question how efficiently this algorithm can be implemented. Algorithms of these kind are common in the context of clustering point sets in d-dimensional space, where a basic constraint is that the number of clusters must not fall below a certain threshold. In the field of graph clustering, this algorithm is used to optimize modularity [7].

Algorithm 1: Greedy Vertex Moving (GVM)

Input : graph $G = (V, E)$, inter, intra, α
Output: clustering $\mathcal{C}_0$ of G
$G^0 \leftarrow G$, $h \leftarrow 0$
repeat
  $\mathcal{C}^h \leftarrow \mathrm{LM}(G^h, \mathrm{Singletons}(G^h), \mathsf{intra}, \mathsf{inter}, \alpha)$;
  $G^{h+1} \leftarrow \mathrm{contract}(G^h, \mathcal{C}^h)$
  $h \leftarrow h + 1$
until *no more real contractions*
while $h \geq 0$ **do**
  $h \leftarrow h - 1$
  $\mathcal{C}^h \leftarrow \mathrm{project}(\mathcal{C}^{h+1}, G^h)$
  $\mathcal{C}^h \leftarrow \mathrm{LM}(G^h, \mathcal{C}^h, \mathsf{inter}, \mathsf{intra}, \alpha)$
end
return $\mathcal{C}^0$

Algorithm 2: Local Moving (LM)

Input : graph $G = (V, E)$, clustering $\mathcal{C}_{\text{init}}$ of G, inter, intra, α
Output: clustering $\mathcal{C}$ of G
$\mathcal{C} \leftarrow \mathcal{C}_{\text{init}}$
repeat
  forall the $v \in V$ **do**
    $\mathcal{A} \leftarrow \{C \in \mathcal{C} \mid \mathsf{intra}(\mathcal{C}_{v \to C}) \geq \alpha \text{ and } |E(v, C)| > 0\}$
    $N \leftarrow \arg \min_{C \in \mathcal{A} \cup \{\}} \{\mathsf{inter}(\mathcal{C}_{v \to C})\}$
    if $\mathsf{inter}(\mathcal{C}_{v \to N}) < \mathsf{inter}(\mathcal{C})$ **then** move(v, N)
  end
until *no more changes*
return $\mathcal{C}$

Greedy Vertex Moving (GVM). The key ingredient of GVM (Algo. 1) is a subprocedure that tries to greedily improve the objective function by letting vertices move to neighboring clusters (Algo. 2). This subprocedure repeatedly iterates through the vertex set and, for each vertex, performs the most improving move (subject to the constraint), potentially isolating a vertex, or leaving it where it was, until a local optimum is reached. Starting with singletons, GVM first calls this subprocedure and contracts the resulting preliminary clustering into a super-graph, i.e. each cluster becomes a vertex weighted with the number of vertices it represents, and edges are summarized such that edge weights reflect the number of edges in the original graph. This whole process is iterated until local moving does not yield any further improvement, and results in a hierarchy of graphs with increasing coarseness. In the second phase (refinement),

the hierarchy is unfurled step by step by projecting the clustering of the $i+1$-th level of the hierarchy to level i, i.e. the clusters in level i are merged according to the clustering in level $i+1$. After each step, LM is called again on the current level of the hierarchy to potentially improve the objective function further, until a clustering for the finest level, i.e. the original graph, is obtained.

GVM is closely related to algorithms in the context of graph partitioning and has previously been used for modularity-based clustering without constraints [4, 23]. Neither approximation guarantees nor subexponential bounds on the running time are known, but experimentally it has been shown to outperform the corresponding greedy merge algorithm with respect to both quality and efficiency. For modularity, it can easily be shown that moving a vertex to a cluster it is not linked with is never the best choice, therefore it suffices to consider neighboring clusters. Together with the observation that the change in modularity can be determined in constant time for each move if some information about the clustering is maintained, this yields a running time in $O(m)$ for each round in LM. This latter observation on running time also holds for all intracluster density and intercluster sparsity measures except for mixd, mixc and mixe, whose values are expensive to maintain.

Ensuring Strict Improvements. Another issue with a direct application of GVM to maximum-based measures is that iteratively traversing the whole vertex set is inefficient if only very few vertex moves potentially decrease the cut of the cluster with the currently worst value. Even worse, if this cluster is not unique, it is likely that the search is stuck in a local minimum, as vertex moves generally can only improve the value for one of these cluster, not for all of them simultaneously. If we try to prevent this by allowing vertex moves that are not strictly improving, we somehow have to ensure that the algorithm terminates after a finite number of operations. We do this in a similar way as proposed in [13] for GM by greedily optimizing the lexicographical order of the intercluster sparsity values of all the clusters. Let $L(\mathcal{C}) := \big(f(C_1), \ldots, f(C_k)\big), C_i \in \mathcal{C}$, be the sequence of these values with decreasing intercluster density, i.e. $f(C_i) \geq f(C_{i+1})$ for $i \in \{1, \ldots, k-1\}$. Then a clustering $\mathcal{C}$ is *L-better* than $\mathcal{C}'$ if $L(\mathcal{C})$ is lexicographically less than $L(\mathcal{C}')$. We now determine for each vertex the set of clusterings that can be reached by moving it. If one of these clusterings is L-better than the current clustering, the move that results in the L-best sequence is performed. As we strictly improve the lexicographical order in each step, termination is guaranteed. This means, we greedily optimize the maximum value but are also allowed to improve the intercluster sparsity of clusters more locally, yielding better efficiency and the possibility to escape local minima.

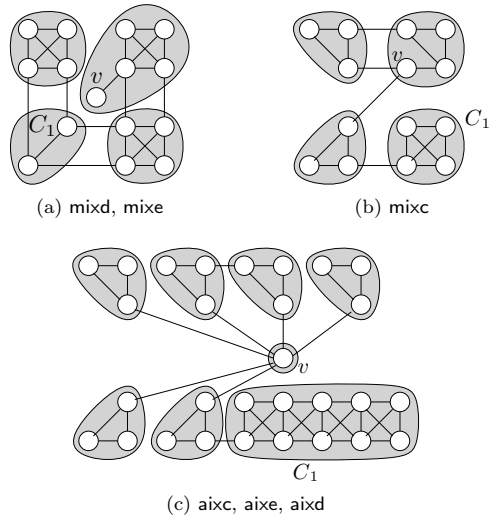

Figure 1: Examples illustrating that most measures considered do not enforce connected moves. Given the clusterings indicated by the gray areas, among all moves involving v, moving v to cluster C_1 yields the largest decrease in the objective function.

Determining the Best Move in $O(\deg(v))$ Time. Using the following observation, it can be seen that any two clusterings resulting from leaving vertex v untouched or from moving v to a different (or new) cluster can be L-compared in constant time.

Observation 1. *For three distinct clusters C, A and B in $\mathcal{C}$ and $v \in C$ it holds that:*

- *$\mathcal{C}_{v \to A}$ is L-better than $\mathcal{C} \Leftrightarrow \big\{C \setminus \{v\}, A \cup \{v\}\big\}$ is L-better than $\{C, A\}$*
- *$\mathcal{C}_{v \to A}$ is L-better than $\mathcal{C}_{v \to B} \Leftrightarrow \big\{A \cup \{v\}, B\big\}$ is L-better than $\big\{B \cup \{v\}, A\big\}$*

If the volume, size and number of out-going edges of the clusters A, B and C are maintained by the algorithm, the density/conductance/expansion of $C, A, B, C\setminus\{v\}, A\cup\{v\}$ and $B\cup\{v\}$ can be determined in constant time. Hence, the conditions on the right-hand side can be evaluated in constant time, which can be used to determine the best move for a vertex efficiently.

Furthermore, it is immediate that moving a vertex to a cluster it is not linked to can never decrease the number of intercluster edges (nxe). This does not hold for gxd, however, the following equation shows that GVM never has to consider non-neighboring clusters for gxd, as isolating the respective vertex is always more

beneficial. Let $v \in V$, $A := C(v) \setminus \{v\}$ and $B \in \mathcal{C}$ such that $m_{\{v\},B} = 0$, then:

$$\begin{aligned}\mathsf{gxd}(\mathcal{C}_{v\to\{\}}) &= \frac{\sum_{C_i,C_j,j>i} m_{C_i,C_j} + m_{\{v\},A}}{\sum_{C_i,C_j,j>i} |C_i||C_j| + |A|} \\ &< \frac{\sum_{C_i,C_j,j>i} m_{C_i,C_j} + m_{\{v\},A} - \overbrace{m_{\{v\},B}}^{=0}}{\sum_{C_i,C_j,j>i} |C_i||C_j| + |A| - \underbrace{|B|}_{>0}} \\ &= \mathsf{gxd}(\mathcal{C}_{v\to B})\end{aligned}$$

For all other intercluster density measures this does not hold as can be seen in the examples in Fig. 1. As configurations like these are only expected in degenerate cases, the impact on efficiency is large on sparse graphs, and unconnected clusters are not desirable in the context of graph clustering, we chose to restrict the set of feasible moves to neighboring clusters. Together with the possibility to compare different moves in constant time, we get a time complexity of $O(m)$ for each round of the local move procedure for each of the combinations considered.

4 Experiments

Qualitative Comparison of Greedy Merge and Greedy Vertex Moving. Our first experiments address the question which flavor of greedy algorithm is better suited for DCC. As test instances, we used all graphs listed in Table 2 with less than 1000 vertices, these are real-world networks taken from the websites of Mark Newman [21] and Alex Arenas [2]. For all proposed combinations of measures, Figure 2 shows the ratio of the intercluster density obtained by using GVM and GM, averaged over all graphs. For modularity, this ratio is always greater than one, confirming that local moving yields better results, regardless of the choice and strength of the constraint. In combination with gid and mid, this similarly holds for all other objectives except for nxe, note that, in contrast to modularity, we aim to minimize these measures and therefore a value below one means that GVM attains better results. For nxe, the outcome depends on the value of α chosen. In combination with aid, the outcome is less clear, the results for nxe are out of bounds as the ratio for some configurations exceeds 300 percents. This can be explained by the observation that aid happily allows (and thereby encourages) unbalanced clusterings, as bad intracluster density values of large clusters can easily be compensated by a set of small and dense clusters, and GM is known to have a tendency to produce unbalanced partitions. As this most often leads to unintuitive clusterings, we deem aid less suitable in the context of graph clustering. Disregarding aid for these reasons, in a vast majority of configurations, GVM outperforms GM. For tackling DCC, we thus solely use GVM, putting aside the algorithm based on greedy merging.

Effectiveness of Different Objective Functions. The next question we pose is, if each of the intercluster density measures is effective in optimizing itself when used as inter in GVM. To answer this question, we conducted the following experiment on the set of graphs listed in Table 2. In the following, let $\text{GVM}_{i,\alpha,x}$ denote GVM incorporating the constraint $i(\mathcal{C}) \geq \alpha$ and the objective $x(\mathcal{C})$. For each setup of DCC, i.e. intracluster measure i, intercluster measure x and $\alpha \in \{0.0, 0.1, \ldots, 1.0\}$, we ranked the clusterings obtained by $\text{GVM}_{i,\alpha,y}$ by their performance with respect to x, using all possible objectives y for GVM. Figure 3 shows the distribution of these ranks over all configurations involving gid, grouped by x. The outcome of this experiment is less clear than what might be expected—none of the intercluster measures, not even modularity, scores the best quality with respect to itself in all configurations. Nonetheless, in general, except for nxe which is clearly dominated by gxd, each objective optimizes itself quite well. This also holds for mid, while for aid, the outcome is even less clear, as can be seen in Figures 6, 7 in App. A.

Reference Algorithms. For a more comprehensive assessment of GVM as a means to address DCC, we use the following reference algorithms:

- *Iterative Conductance Cutting (ICC)* [18]: This top-down algorithm iteratively splits the input graph into two subgraphs based on a cut with low conductance. The process stops when the conductance of the cut exceeds a given threshold, which we set to 0.4 in our experiments.
- *Markov-Clustering (MCL)* [9]: Emulating a random walk, the matrix of transition probabilities is alternately taken to to the power of e and renormalized after taking each entry to the power of r, where e and r are input parameters. In our experiments, we set r and e to 2 (this equals the parameter setting used in [5]).
- *Geometric MST Clustering (GMC)* [5]: First, a spectral embedding of the graph in d-dimensional space is built. Then the algorithm constructs a Euclidean minimum spanning tree and successively deletes the heaviest edge. This defines a sequence of forests whose connected components induce a set of clusterings. Among these clusterings, the one with the best value according to some given objective function is chosen.

graph	n	m	graph	n	m	graph	n	m
karate(N)	34	78	celegansneural(N)	297	2148	astro-ph(N)	16706	121251
dolphins(N)	62	159	celegans_metabolic(A)	453	2039	cond-mat(N)	16726	47594
lesmis(N)	77	254	polblogs(N)	1490	16718	as-22july06(N)	22963	48436
polbooks(N)	105	441	netscience(N)	1589	2742	cond-mat-2003(N)	31163	120029
adjnoun(N)	112	425	power(N)	4941	6594	cond-mat-2005(N)	40421	175693
football(N)	115	616	hep-th(N)	8361	15751			
jazz(A)	198	2742	PGPgiantcompo(A)	10680	24316			

Table 2: List of the real world test instances ordered by increasing number of vertices. These are taken from the webpages of Arenas(A) [2] and Newman(N) [21] and are often used to compare clustering algorithms.

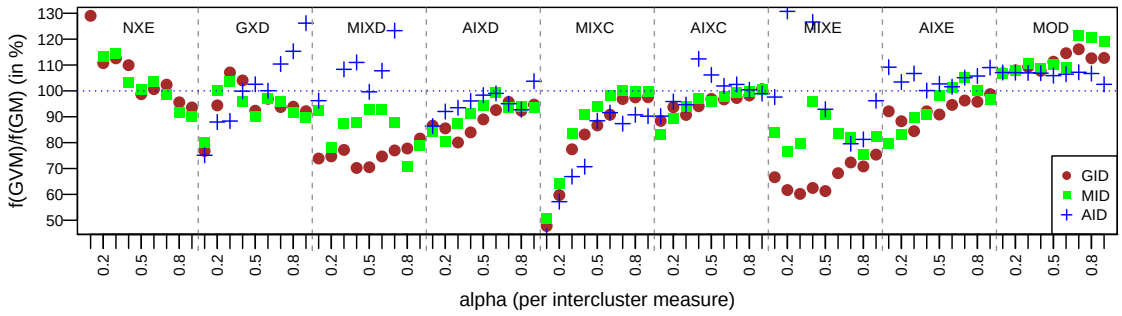

Figure 2: Qualitative comparison of GVM and GM.

- *Multi-Level Modularity (ML-MOD)* [23]: This is the GVM-algorithm based solely on modularity without using any constraint. This algorithm has been shown to perform very well in the context of modularity optimization [23].

Comparison Based on Intracluster Density Found by Reference Algorithms. ICC, MCL and ML-MOD do not incorporate constraints on the intracluster density of the resulting clustering. Nonetheless, it is still possible to evaluate them with respect to those variants of DCC, where α is set to the intracluster density found by these algorithms. In other words, given the same constraint a reference algorithm $\mathcal{A}$ implicitly adheres to, how well does GVM compare to $\mathcal{A}$ wrt. DCC?

We first ran ICC, MCL and ML-MOD on all test instances in Table 2 and recorded the intracluster density values of the resulting clusterings. Then, for each reference algorithm $\mathcal{A}$, i, recorded corresponding intracluster density α and x, we compare the clustering obtained by $\text{GVM}_{i,\alpha,x}$ to the clustering of $\mathcal{A}$ with respect to x. For GMC the experiments slightly differ as GMC requires an objective function. We filled this degree of freedom by choosing $f(\mathcal{C}) = i(\mathcal{C}) - x(\mathcal{C})$ as the objective function for the experiments using i as intracluster and x as intercluster density measure. This seemed to be the fairest way of comparison and in almost all cases led to non-trivial clusterings.

Table 3 shows both the percentage of graphs where the greedy algorithm for x compares favorably and the arithmetic mean of the ratio of x obtained with GVM and with the reference algorithm. As we aim to minimize intercluster density, a value below 1 indicates that the greedy algorithm succeeds in beating the reference algorithm and vice versa. Compared to ICC and MCL, GVM clearly yields better results. The same holds for GMC, except if used in combination with aid, where GMC sometimes produces far better results. This can be explained by the fact that aid does not punish unbalancedness and GMC naturally leads to very unbalanced clusterings in most instances. The outcome of the comparison with the modularity-based algorithm is less clear. For aid, GVM performs better, which is not surprising as modularity strongly discourages unbalanced clusterings. For mid, GVM still beats ML-MOD in the majority of configurations, while for gid, this only holds for slightly less than half of the configurations. Furthermore, it is worth mentioning that especially for aixd and aixe there are instances where modularity minimizes these functions far better than the respective greedy algorithms. Altogether, the comparison with ICC, MCL and GMC suggests that GVM effectively addresses DCC, while the comparison with ML-MOD shows that optimizing modularity is similarly effective in minimizing cut-based intercluster

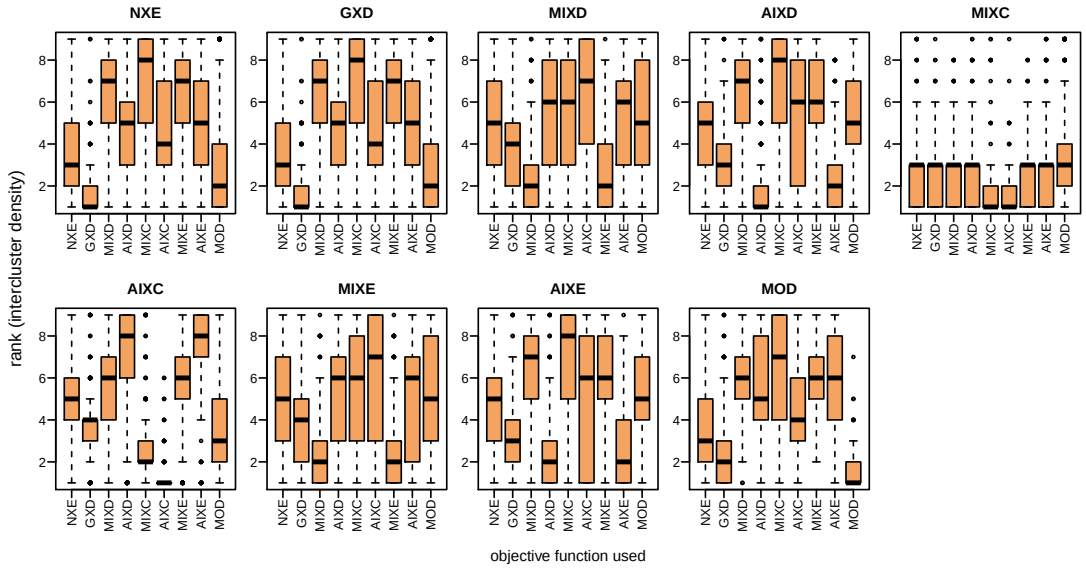

Figure 3: Ranks for different intercluster density measures as objectives in the GVM-algorithm using gid as constraint, evaluated by the intercluster density of the resulting clustering.

	gid				mid				aid			
	ICC	MCL	ML-MOD	GMC	ICC	MCL	ML-MOD	GMC	ICC	MCL	ML-MOD	GMC
nxe	84/0.67	95/0.52	16/1.17	63/1.26	89/0.42	95/0.08	63/0.97	74/0.88	95/0.03	100/0.06	100/0.05	63/8.07
gxd	84/0.64	100/0.50	42/1.07	100/0.11	95/0.40	100/0.09	84/0.89	100/0.10	95/0.07	100/0.10	100/0.13	84/0.76
aixd	84/0.47	100/0.32	42/5.30	100/0.25	89/0.34	100/0.06	37/5.08	95/0.23	95/0.18	100/0.12	100/0.22	84/0.61
aixc	84/0.57	100/0.29	21/2.17	53/0.28	95/0.39	100/0.05	79/0.81	42/0.27	95/0.41	95/0.27	100/0.37	63/7.87
aixe	84/0.49	95/0.39	42/5.55	89/0.31	89/0.36	95/0.14	42/5.22	95/0.31	95/0.19	95/0.13	95/0.24	95/1.45
mixd	84/0.45	95/0.34	53/0.96	84/0.41	89/0.39	100/0.07	74/1.27	89/0.30	89/0.21	95/0.18	89/0.32	74/3.17
mixc	89/0.69	95/0.58	42/1.15	37/0.34	89/0.47	95/0.15	63/1.09	37/0.30	89/0.44	95/0.39	84/0.46	21/1.60
mixe	89/0.48	95/0.26	58/1.25	89/0.57	84/0.39	95/0.14	47/1.28	79/0.63	95/0.13	95/0.16	89/0.28	63/3.02

Table 3: Comparison of GVM and reference algorithms. Entries are of the form p/r, where p is the percentage of graphs GVM compares favorably and r is the mean ratio of x obtained by GVM and reference algorithm.

sparsity measures.

Recovering Planted Partitions. To compare the different objective functions qualitatively, we evaluated how well the corresponding GVM-algorithms are able to reconstruct planted partitions in random graphs. As a comparison, we also give the results obtained by ML-MOD. Due to higher running times and large numbers of experiments, we omit a comparison with ICC, MCL and GMC.

Random Graphs Generated. We use an adapted Erdős-Rényi-model, where, starting from a given reference partition, the probability that vertices in the same set (in different sets) are connected equals p_{in} (p_{out}). The number of vertices (n) and clusters (k) as well as the skewness of the distribution of cluster sizes (β) of the planted partition are input parameters. Setting β to 1.0 corresponds to uniform cluster sizes, values below and above 1 cause this distribution to be skewed, for more details see [15]. As configurations, we fixed n to 10000 and chose p_{in} and p_{out} such that the average number of intracluster (intercluster) edges a vertex is incident to equals 5 (3). To determine the reference partition, we used all combinations of $k \in \{10, 100, 300\}$ and $\beta \in \{0.3, 1.0, 2.0\}$. For each configuration, we generated 100 instances and always averaged obtained values.

Distance Measures. To compare the clusterings obtained with the different algorithms with the reference clustering, we use the following graph-based distance measures taken from [8]:

- *Graph-based Rand Index (R_g)*: Let $\mathcal{C}_1$ and $\mathcal{C}_2$ be clusterings and e_{11} (e_{00}) the number of edges which are intracluster (intercluster) wrt. both $\mathcal{C}_1$ and $\mathcal{C}_2$.

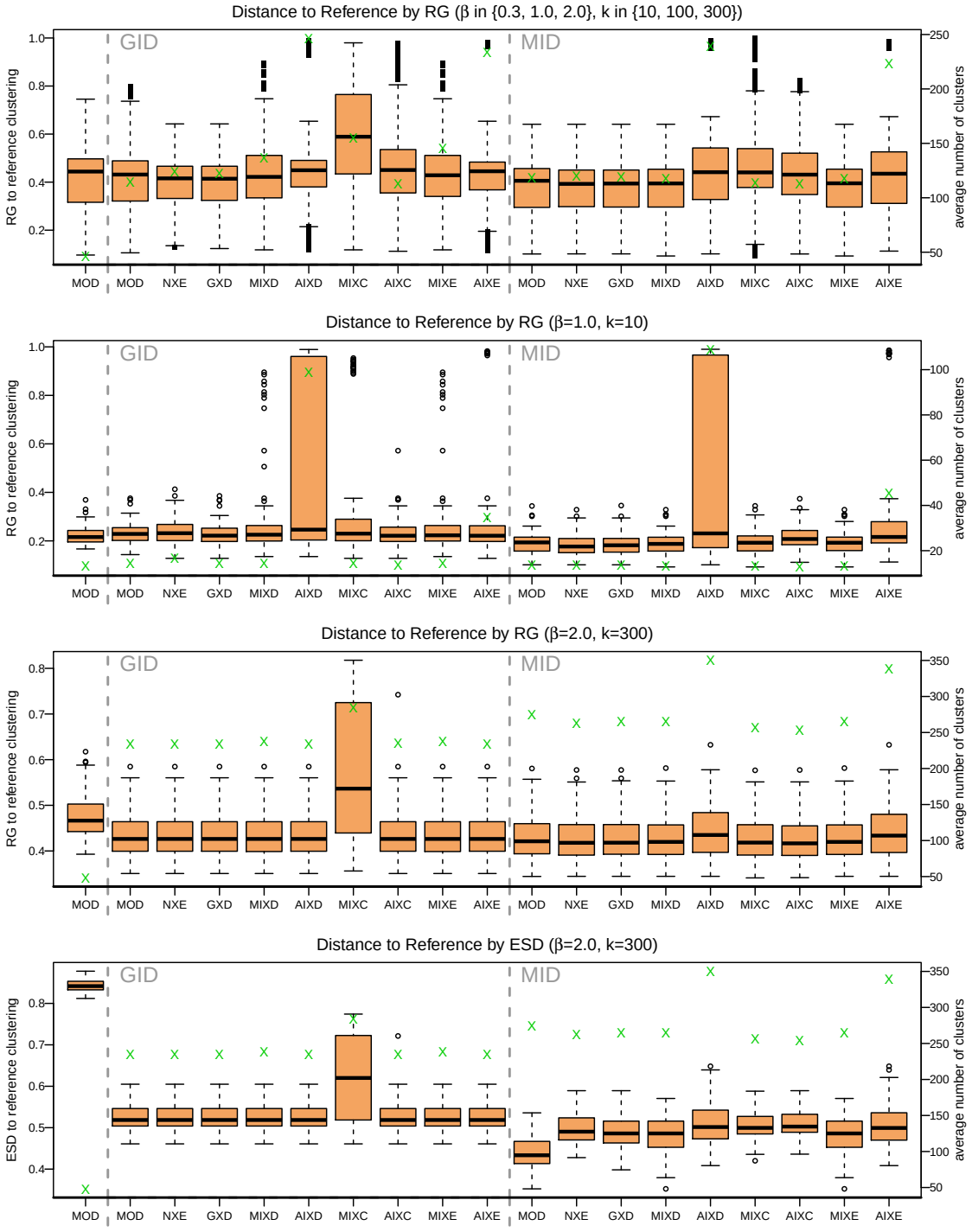

Figure 4: Distance to reference clustering (boxplots, left-hand y-axis) and number of clusters discovered in planted partition graphs (green $\times$-marks, right-hand y-axis), different configurations

Then, $R_g(\mathcal{C}_1, \mathcal{C}_2) = 1 - (e_{11} + e_{00})/m$.

- *Editing Set Difference (ESD)*: For a clustering $\mathcal{C}$, its editing set $F_\mathcal{C}$ is the set of edges requiring insertion or removal such that the clusters in $\mathcal{C}$ form disjoint cliques. Then, for clusterings $\mathcal{C}_1$ and $\mathcal{C}_2$, their editing set difference is defined as $ESD(\mathcal{C}_1, \mathcal{C}_2) = 1 - |F_{\mathcal{C}_1} \cap F_{\mathcal{C}_2}|/|F_{\mathcal{C}_1} \cup F_{\mathcal{C}_2}|$.

Parameters and Evaluation. As an exhaustive parameter search for all configurations would be far too expensive, we always set α to 75 percent of the expected global intra-cluster density p_{in}. We deemed taking the actual value of p_{in} too strict, as, especially for mid, even the reference clustering of the generator most likely does not meet this constraint. The previous experiments indicate that there are configurations where particular objective functions used in GVM do not score the best results with respect to themselves. As our goal is to compare good clusterings with respect to different combinations of i and x, independent of artifacts of GVM, we chose the following approach: For a combination i, α, x, we evaluated the clustering that, among all results obtained with GVM using $i \geq \alpha$ as constraint, is best with respect to x (as opposed to simply evaluating $\text{GVM}_{i,\alpha,x}$). Furthermore, preliminary experiments confirmed that constraining aid leads to very unintuitive and unbalanced clusterings, which is mirrored by the fact that the corresponding versions of DCC are far less effective in finding the hidden clustering. For this reason, we excluded aid in the discussion of the results.

Results on Planted Partition Graphs. Figure 4 shows the results for selected configurations, the results for the whole set of experiments with respect to R_g can be found in App. B, for additional plots evaluating ESD see the extended version [14]. In the first plot it can be seen that, in general, the clusterings that are ranked best with respect to mod, nxe and gxd are most similar to the reference.

Constraining modularity by mid *improves its results.* This especially holds for the experiments with high skewness ($\beta = 2$) and $k = 300$. In these experiments, modularity finds far less clusters than expected, partially due to its known resolution limit [12], which can be circumvented by steering the coarseness of the clustering by constraining the intracluster density. Another interesting fact is that ESD punishes the coarse clustering obtained by pure modularity far more than R_g.

Fine reference clusterings disbalance maximum objectives. Compared to the above, especially mixc in combination with gid yields worse similarity values. This, and the slightly increased cluster count can be explained by a tendency of mixc to favor unbalanced clusterings if the expected number of clusters is high ($k = 300$), which also explains why this effect does not happen in combination with mid that does not allow very unbalanced clusterings. To a smaller extent, the same observation also holds for the other maximum measures, as can be seen for $k = 300$ and $\beta = 1.0$.

aixe *and especially* aixd *identify many clusters.* Another striking observation is that the average number of clusters in clusterings found by aixd and aixe, indicated by the green $\times$-marks, is much higher than the average number of clusters in the reference. This especially stems from the experiments with few clusters. In the configuration with $\beta = 1$ and $k = 10$, it can also be seen that these measures differ the more, the coarser the expected clustering gets. This is not unexpected, as the denominator of aixd grows more slowly with the number of vertices in the cluster than the denominator of aixe, meaning that aixd is less eager to produce very large clusters. Additionally, in [13] it was proven that with the exception of aixd, all inter-cluster measures considered here can always be ameliorated by merging two existing clusters (unboundedness), which is also a hint that aixd is less likely to produce coarse clusterings than the other measures.

Selected clusterings on small example network. Figure 5 demonstrates the differences between intercluster measures on a small network reflecting social interaction of a group of 62 dolphins [20]. As we did not want to introduce an artificial bias towards a particular clustering, the (force-directed) layout of the vertices

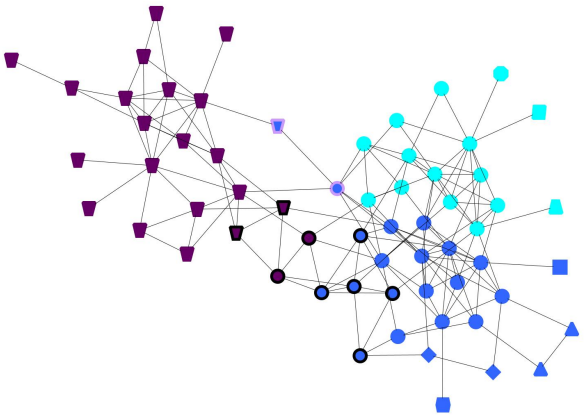

Figure 5: Network of frequent associations between 62 dolphins in a community living of Doubtful Sound, New Zealand [20]. The clusterings displayed are obtained by optimizing the measures mixe (fill color), aixd (vertex shape) and mod (border color) with GVM under the restriction gid > 0.2.

does not use any information about the clustering. With the restriction gid > 0.2, aixd dominates nxe, gxd and mixd in the sense that the clustering obtained by optimizing aixd with GVM yields less intercluster edges and lower values of gxd and mixd than the corresponding clusterings obtained by optimizing these measures directly. Similarly, mixe dominates aixc and mixc, while aixc dominates aixe. Due to this and to retain visual clarity, we only give the clusterings obtained by aixd, mixe and mod. aixc is omitted because the respective clustering is very similar to the one obtained with mixe, they only differ in the assignment of few vertices connecting the left and the right part.

Compared to mixe, the clustering obtained by mod introduces two new clusters that consist of the vertices connecting the left and the right part. The main difference between these clusterings and the one obtained by aixd is the assignment of the nine vertices on the right side that are only sparsely connected to the remainder of the graph; mixe and mod assign them to the only clusters they are connected with while aixd essentially leaves them unclustered. Overall, all clusterings are rather similar in the sense that only few vertices are treated differently, all of them either connecting the two parts or being only loosely connected to the network; a human observer might argue in favor of any of the clusterings considered, as the group affiliation of these vertices seems ambiguous.

The reason why nxe, gxd and mixd are dominated by aixd is that the respective versions of GVM merge the sparsely connected vertices on the right side with their anchor vertices in an early stage of the algorithm. Isolating these vertices later on is not possible, as this would decrease the respective objective function, although isolating these vertices and moving one of the vertices in the middle to the respective cluster would be feasible and improve the objective function.

Implementation and Running Times. The algorithms ICC, MCL, GMC and GM are implemented in Java 1.6.0_22 using the graph library yFiles [24]. GVM (also incorporating ML-MOD as a special case) is implemented in C++ using version 1_42 of the Boost Graph Library [1] and compiled with gcc 4.5.2 with optimization level 4. The focus of this evaluation is on the quality of the resulting clusterings, not on running times. However, to get a rough impression about the latter, clustering cond-mat-2005 on a 2.1 GHz AMD Opteron processor takes about 6 hours with ICC, 1 hour and 50 minutes with MCL, 5 minutes with GMC and 3 to 15 seconds with GVM, depending on the parameter setting. With our prototype implementation (not including the improvements proposed in [13]) of GM, clustering the much smaller celegans_metabolic takes over 2 minutes.

5 Conclusion

This work is an experimental evaluation of algorithms for Density-Constrained Clustering (DCC). We first evaluated two greedy heuristics, vertex moving and cluster merging, against each other and against algorithms from the literature. Vertex moving proved reliably superior to cluster merging and, in many cases, beats the results of the reference algorithms. Our results also show that a well-known modularity-based algorithm implicitly addresses DCC quite well, revealing similarities between cut-based intercluster sparsity measures and modularity. In the second part, we addressed the question whether different combinations of intracluster density and intercluster sparsity measures are suitable to guide algorithms in recovering planted partitions in random graphs. The results suggest that minimizing the average intercluster expansion or density of the clusters overestimates the number of clusters if the expected clustering is coarse, while the maximum intercluster measures lead to unbalanced clusters if the expected clustering is fine and the constraint on the intracluster density does not force the clustering to be balanced. Additionally, it can be seen that the known resolution limit for modularity can be circumvented if the coarseness of the clustering is controlled by an additional constraint on the intracluster density of the clustering.

References

[1] Boost C++ Libraries, http://www.boost.org/

[2] Arenas, A.: Network data sets, http://deim.urv.cat/~aarenas/data/welcome.htm

[3] Berkhin, P.: A Survey of Clustering Data Mining Techniques. In: Kogan, J., Nicholas, C., Teboulle, M. (eds.) Grouping Multidimensional Data: Recent Advances in Clustering, pp. 25–71. Springer (2006), http://www.springerlink.com/content/x321256p66512121/

[4] Blondel, V., Guillaume, J.L., Lambiotte, R., Lefebvre, E.: Fast unfolding of communities in large networks. Journal of Statistical Mechanics: Theory and Experiment 2008(10) (2008), http://dx.doi.org/10.1088/1742-5468/2008/10/P10008

[5] Brandes, U., Gaertler, M., Wagner, D.: Experiments on Graph Clustering Algorithms. In: Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03). Lecture Notes in Computer Science, vol. 2832, pp. 568–579. Springer (2003), http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=2832&spage=568

[6] Brandes, U., Gaertler, M., Wagner, D.: Engineering Graph Clustering: Models and Experimental Evaluation. ACM Journal of Experimental Algorithmics 12(1.1), 1–26 (2007), http://portal.acm.org/citation.cfm?id=1227161.1227162

[7] Clauset, A., Newman, M.E.J., Moore, C.: Finding community structure in very large networks. Physical Review E 70(066111) (2004), http://link.aps.org/abstract/PRE/v70/e066111

[8] Delling, D., Gaertler, M., Görke, R., Wagner, D.: Engineering Comparators for Graph Clusterings. In: Proceedings of the 4th International Conference on Algorithmic Aspects in Information and Management (AAIM'08). Lecture Notes in Computer Science, vol. 5034, pp. 131–142. Springer (June 2008)

[9] van Dongen, S.M.: Graph Clustering by Flow Simulation. Ph.D. thesis, University of Utrecht (2000), http://micans.org/mcl/lit/

[10] Flake, G.W., Tarjan, R.E., Tsioutsiouliklis, K.: Graph Clustering and Minimum Cut Trees. Internet Mathematics 1(4), 385–408 (2004), http://www.internetmathematics.org/volumes/1.htm

[11] Fortunato, S.: Community detection in graphs. Physics Reports 486(3–5), 75–174 (2010), http://www.sciencedirect.com/science/journal/03701573

[12] Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proceedings of the National Academy of Science of the United States of America 104(1), 36–41 (2007), http://www.pnas.org/content/104/1/36.full.pdf

[13] Görke, R., Schumm, A., Wagner, D.: Density-Constrained Graph Clustering. In: Dehne, F., Iacono, J., Sack, J.R. (eds.) Algorithms and Data Structures, 12th International Symposium (WADS'11). Lecture Notes in Computer Science, vol. 6844. Springer (August 2011)

[14] Görke, R., Schumm, A., Wagner, D.: Experiments on Density-Constrained Graph Clustering (December 2011), http://arxiv.org/abs/1112.2143v1, arXiv report

[15] Görke, R., Staudt, C.: A Generator for Dynamic Clustered Random Graphs. Tech. rep., ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH) (2009), http://i11www.iti.uni-karlsruhe.de/projects/spp1307/dyngen, informatik, Uni Karlsruhe, TR 2009-7

[16] Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bulletin of the American Mathematical Society 43, 439–561 (2006)

[17] Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice Hall (1988)

[18] Kannan, R., Vempala, S., Vetta, A.: On Clusterings - Good, Bad and Spectral. In: Proceedings of the 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS'00). pp. 367–378 (2000)

[19] Lancichinetti, A., Fortunato, S.: Community detection algorithms: A comparative analysis. Physical Review E 80(5) (November 2009), http://link.aps.org/doi/10.1103/PhysRevE.80.056117

[20] Lusseau, D., Schneider, K., Boisseau, O., Haase, P., Slooten, E., Dawson, S.: The Bottlenose Dolphin Community of Doubtful Sound Features a Large Proportion of Long-Lasting Associations. Behavioral Ecology and Sociobiology 54(4), 396–405 (September 2004), http://www.springerlink.com/content/pepxvj4lu42ur2gw/

[21] Newman, M.: Network data, http://www-personal.umich.edu/~mejn/netdata/

[22] Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E 69(026113), 1–16 (2004), http://link.aps.org/abstract/PRE/v69/e026113

[23] Rotta, R., Noack, A.: Multilevel local search algorithms for modularity clustering. ACM Journal of Experimental Algorithmics 16, 2.3:2.1–2.3:2. (July 2011), http://doi.acm.org/10.1145/1963190.1970376

[24] yWorks GmbH: yFiles for Java (2008), http://www.yworks.com/en/products_yfiles_about.html

A Effectiveness of Different Objective Functions: Additional Plots

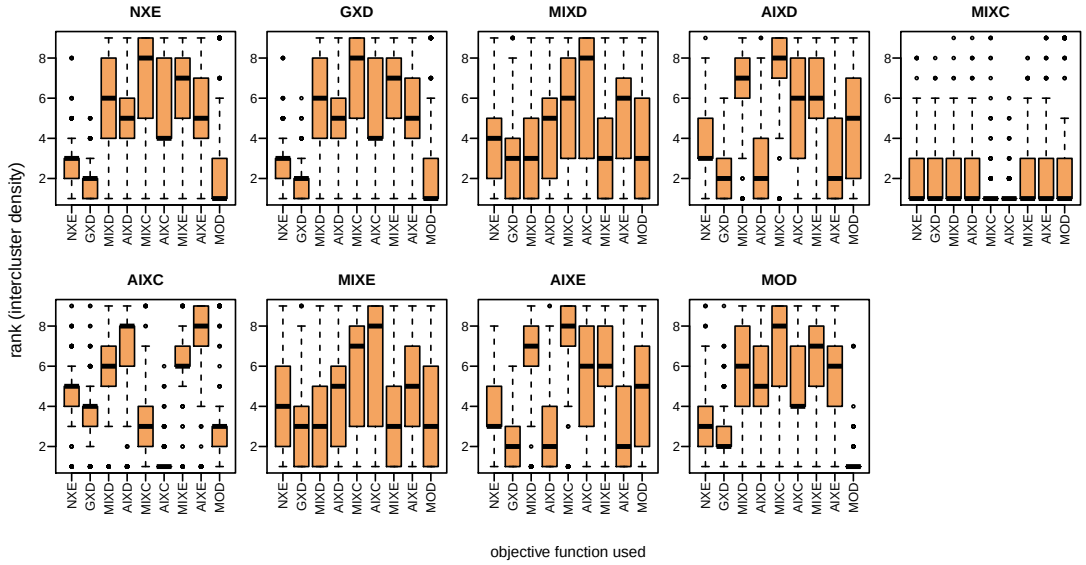

Figure 6: Ranks for different intercluster density measures as objectives in the GVM-algorithm using mid as constraint, evaluated by the intercluster density of the resulting clustering.

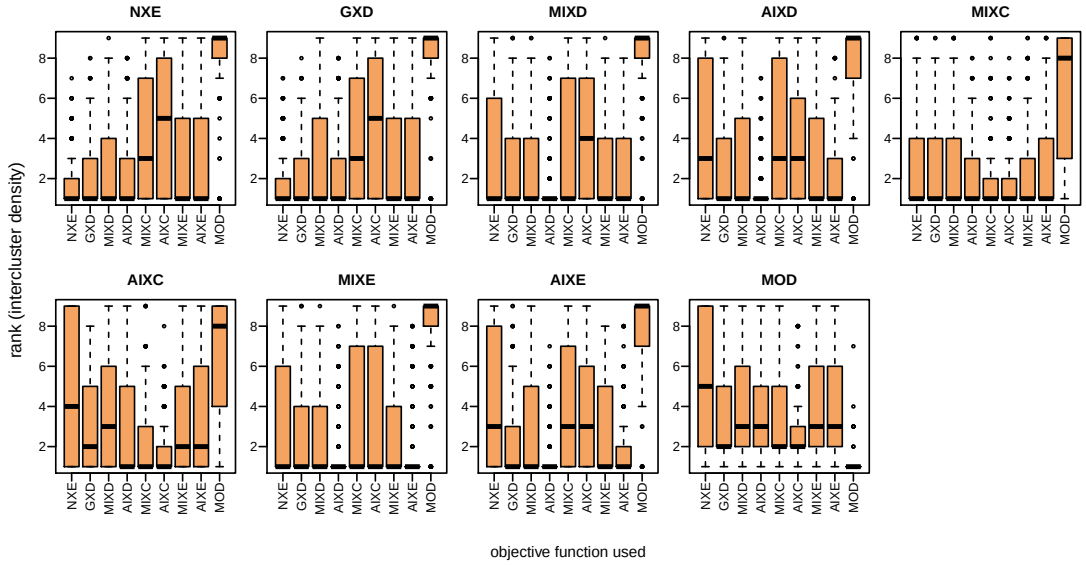

Figure 7: Ranks for different intercluster density measures as objectives in the GVM-algorithm using aid as constraint, evaluated by the intercluster density of the resulting clustering.

B Planted Partition Graphs: Complete Experiments with Respect to R_g