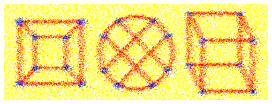


Journal of Graph Algorithms and Applications
http://jgaa.info/ vol. 14, no. 1, pp. 165–197 (2010)

Computational Aspects of Lucidity-Driven Graph Clustering

Robert Görke Marco Gaertler Florian Hübner Dorothea Wagner

Institute of Theoretical Informatics, Fakultät für Informatik,
Universität Karlsruhe (TH), Karlsruhe Institute of Technology, Germany

Abstract

We formally state and investigate the *lucidity* paradigm for graph clusterings. The rationale that substantiates this concept is the trade-off between the achieved quality and the expected quality of a graph clustering. A recently defined quality measure for graph clusterings, *modularity*, is one specific realization of this paradigm, in fact the pioneering one. On a theoretical side, we state a sound probability space for *lucidity* and thus for *modularity* and prove that in this paradigm of *lucidity*, using a subtractive trade-off and either the index *coverage* (yields *modularity*) or *performance* leads to equivalent indices. Furthermore, we show that the NP-hardness of optimizing these indices yields the NP-hardness of the problem MinMixedMultiPartition. Moreover, we describe an efficient maximization algorithm for a *divisive* trade-off between quality and expectation. We experimentally evaluate four realizations of this paradigm systematically and confirm their feasibility in a first methodic analysis of the behavior of these realizations on both artificial and on real-world data, arriving at good results of community assessment and detection.

Submitted:	Reviewed:	Revised:	Accepted:	Final:
March 2009	September 2009	October 2009	December 2009	December 2009
		Published: January 2010		
	Article type: Regular paper		Communicated by: U. Brandes	

This work was partially supported by the DFG under grants WA 654/14-3 and WA 654/15-1 and by the EU under grant DELIS (contract no. 001907) and grant CREEN (contract no. 012684).

E-mail addresses: robert.goerke@kit.edu (Robert Görke) marco.gaertler@kit.edu (Marco Gaertler) florian.h@gmail.com (Florian Hübner) dorothea.wagner@kit.edu (Dorothea Wagner)

1 Introduction

The discovery of natural groups and large scale inhomogeneities is a crucial task in the exploration and analysis of large and complex networks. This task is usually realized with *graph clustering* methods. The majority of algorithms for graph clustering are based on the paradigm of intra-cluster edge-density versus inter-cluster edge-sparsity. Several formalizations have been proposed and evaluated, an overview of such techniques is given in [5] and [9]. Among the many approaches which incorporate this paradigm, the quality index *modularity* introduced in [26] has attained much attention lately, in particular as an objective function for optimization [24, 11]. Roughly speaking, *modularity* evaluates a clustering by comparing the fraction of intra-cluster edges to the expected value of this number. The predominant usage of *modularity* employs a simple greedy agglomerative algorithm for heuristic maximization, proposed in [11]. A range of alternative algorithmic approaches have been proposed, based on spectral division [25, 33], simulated annealing [18, 30], extremal optimization [14], and fast iterative local optimization [2]. Applications range from protein interaction dependencies to recommendation systems, social network analysis and even embeddings of neural networks (see, e.g., [35, 21, 29]). While evaluations and comparisons of clustering methods can, at least to some extent, be found in the majority of publications on new clustering algorithms (see, e.g., [7, 13, 2]), only few theoretical analyses of *modularity* and its optimization exist. Recently, it has been proven, that it is NP-hard to optimize *modularity* [[4, 3]], which justifies the need for heuristics and approximations. Also worth mentioning are the results on resolution limits of *modularity*-based methods by [15], who describe a restrictive tendency of *modularity* to detect communities of specific size categories that depend on the size of the input.

In this work, we formally state and investigate the founding paradigm for *modularity*, being the *lucidity* of a clustering, as the trade-off between the achieved quality and the expected quality for random networks incorporating the intrinsic properties of the original network. We explore a probability space for random networks that fulfills the underlying assumptions. Since *performance* is known to be more robust than *coverage*, it seems promising to utilize it in combination with *subtraction* (as for *modularity*). We prove that this yields a measure of *lucidity* equivalent to *modularity*, which corroborates its results. An alternative interpretation, motivated by an ILP formulation, ultimately leads us to the NP-hardness of MinMixedMultiPartition. We systematically evaluate how well *lucidity* complies with human intuition of clustering quality and with established indices. Furthermore, we present an algorithm that efficiently optimizes a promising realization, *relative performance lucidity*, in $O(n^2 \log n)$ time and using geometry. We compare the goodness of these algorithms in terms of clustering quality to that of other algorithms, on a set of random preclustered graphs and complement our findings with results on real data. Our results indicate the feasibility of the paradigm in that our algorithms surpass the benchmark algorithms, and in that the generality of the approach is justified by specific realizations excelling on real-world data.

This paper is organized as follows: After introducing the necessary preliminaries for graph clustering and some quality measures (Section 2), we give the formal definition of our *lucidity* paradigm, explore appropriate probabilistic setups and deduce four realizations of *lucidity* (Section 3). Section 4 scrutinizes the greedy algorithms which are employed to obtain clusterings with high *lucidity* score, including an efficient implementation for a quick stepwise update. The setup and the results of the experimental evaluation are described in Section 5 which are followed by a conclusion. A precursory paper on this work appeared in the *Proceedings of the 3rd AAIM'08 Conference* [16].

2 Preliminaries

Throughout this paper, we will use the notation of [5]. We assume that $G = (V, E, \omega)$ is an undirected, weighted graph and $\omega\colon E \to [0, 1]$. Although considering only simple graphs suffices for most insights, we require loops and parallel edges later and thus start out general straight away. We often consider only unweighted graphs but will say so explicitly. For a node v, we define $\omega(v)$ as the sum of the weights of incident edges, and $\deg(v)$ as the number of incident edges, doubly counting loops in both terms. We set $|V| =: n, |E| =: m$ and $\mathcal{C} = \{C_1, \ldots, C_k\}$ to be a partition of V. Since we allow non-simple graphs, we write both edges and edge sets E as multisets, such that $\{v, v\} \in E$ is allowed (a loop) and $E = \{\{u, v\}, \{u, v\}\}$ (two parallel edges). Edges are undirected but denoted as both $\{u, v\}$ (unordered) or (u, v) (as an ordered set) for convenience, depending on how we enumerate over edges. For simplicity in enumerations we assume that V is ordered. We denote the set of multisets of two nodes that can be connected by an edge in a non-simple graph as $V^\times = \{\{u, v\} \mid u \geq v, u \in V, v \in V\}$. The set of all 2-tuples from V is $V^2 = \{(u, v) \mid u \in V, v \in V\}$, these can then be translated to edges, but the set and the enumeration are different from $V^\times$. Furthermore, $\omega(e)$ is the weight of a single edge e, whereas $\omega(\{u, v\})$ (or $\omega((u, v))$) is the sum of weights of edges between nodes u and v. We abbreviate $\omega(\{u, v\}) = \omega(u, v)$.

We call $\mathcal{C}$ a *clustering* of G and the C_i *clusters*. The cluster which contains node v is denoted by $\mathcal{C}(v)$. We identify a cluster C_i with the induced subgraph of G, i.e., the graph $G[C_i] := (C_i, E(C_i), \omega_{|E(C_i)})$, where $E(C_i) := \{\{v, w\} \in E : v, w \in C_i\}$. Then $E(\mathcal{C}) := \bigcup_{i=1}^{k} E(C_i)$ is the set of *intra-cluster* edges and $E \backslash E(\mathcal{C})$ the set of *inter-cluster* edges, with $|E(\mathcal{C})| =: m(\mathcal{C})$ and $|E \backslash E(\mathcal{C})| =: \overline{m}(\mathcal{C})$. The set $E(C_i, C_j)$ denotes the set of edges connecting nodes in C_i to nodes in C_j. We denote the number of non-adjacent intra-cluster pairs of nodes as $m(\mathcal{C})^c$, and the number of non-adjacent inter-cluster pairs as $\overline{m}(\mathcal{C})^c$. We summarize the heavily used term $\sum_{v \in C} \deg(v) =: \mathrm{vol}(C)$.

We measured the quality of clusterings with a range of quality indices, discussed, e.g., in [5], however, we set our focus on the indices *inter-cluster conductance* [7], *coverage* [5] and *performance* [31] in this work, since they are the most studied ones. For unweighted graphs, in brief, *coverage* is the fraction of edges which are intra-cluster edges and *performance* is the fraction of correctly

classified node pairs with respect to their connectedness. *Inter-cluster conductance* (or *inter-cc*) measures the worst bottleneck constituted by cutting off a cluster from the graph, normalized by the degree sums thereby cut off. For a profound discussion of these indices we refer the reader to the given references and to further pointers therein, and simply state their formal definitions:

$$\mathrm{cov}(\mathcal{C}) := \frac{m(\mathcal{C})}{m(\mathcal{C}) + \overline{m}(\mathcal{C})} \qquad (1) \qquad \mathrm{perf}(\mathcal{C}) := \frac{m(\mathcal{C}) + \overline{m}(\mathcal{C})^c}{\frac{1}{2}n(n-1)} \qquad (2)$$

$$\mathrm{icc}(\mathcal{C}) := 1 - \max_{C \in \mathcal{C}} \frac{\sum_{v \in C} \deg(v) - 2|E(C)|}{\min\Big(\sum_{v \in C} \deg(v), \sum_{v \in V \setminus C} \deg(v)\Big)} \qquad (3)$$

Note that while *coverage* and *inter-cc* are bounded by $[0, 1]$, *performance* can take values beyond 1 in non-simple graphs. *Inter-cc* is based on the measure *conductance* [20], which seeks the "cheapest" cut $(S, V \setminus S)$ (with $S \subseteq V$) in a graph (measured by the fractional term of Equation 3. The conductance of a clustering is then defined as the minimum conductance of each cluster. However, determining the minimum *conductance* cut in a graph is NP-hard [1], and thus this measure is ill-suited for measuring clustering quality. In turn, the cut induced by a cluster should have a very low *conductance* in a good clustering. Following [7] we can thus examine how good cuts induced by clusters are (instead of all cuts inside a cluster), which yields the meaningful (and computable) formula in Equation 3. We shape this measure as to yield 1 for good clusterings.

All previous definitions generalize in a natural way as to take edge weights $\omega(e)$ into account, which we indicate by a subscript as in, e.g., cov_ω. Thus, $\omega(\mathcal{C})$ ($\overline{\omega}(\mathcal{C})$) denotes the sum of the weights of all intra-cluster (inter-cluster) edges, W denotes the sum of all edge weights and $\mathrm{vol}_\omega(C) := \sum_{v \in C} \omega(v)$. The maximum edge weight in a graph is called $\omega_{\max}$. The fact that *modularity* can be expressed as *coverage* minus the expected value of *coverage* (see Section 3.1 and [11]) motivates the general paradigm we state in the next section.

3 The Lucidity Paradigm

In the *lucidity paradigm* a good clustering is characterized by having a high quality compared to the value the clustering obtains for a random network that reflects specific *structural properties* that are expected to be present in the graph, as predefined in an appropriate null hypothesis. Every realization of the *lucidity* paradigm requires a quality measure, a null hypothesis based on a set of such characteristics, and a mode of comparison of these. The concept of *lucidity* is related to the notion of p-values in statistical hypothesis testing. The p-value of a value t observed for a random variable T is the probability that under the assumption of a given null hypothesis, T assumes a value at least as unfavorable

to the null hypothesis as the observed value t. In general, the null hypothesis is rejected, if the p-value is smaller than the statistical significance level of the measurement. However, in our concept we do not reject a null hypothesis, which we assume to reasonably describe observed graphs. Instead, we compare the quality of a clustering to the expected value, in order to judge its relevance.

Definition 1 *Given a quality index $\mathcal{M}$ and a clustering $\mathcal{C}$, we define the lucidity $L^{\odot}_{\mathcal{M}}$ of a clustering $\mathcal{C}$ as the corresponding quality index as follows:*

$$L^{\odot}_{\mathcal{M}}(\mathcal{C}) := \mathcal{M}(\mathcal{C}) \odot \mathbb{E}_{\Omega}[\mathcal{M}(\mathcal{C})] \ , \tag{4}$$

where $\mathbb{E}_{\Omega}[\mathcal{M}]$ is the expected value of the quality index $\mathcal{M}$ for the clustering $\mathcal{C}$ with respect to a suitable probability space Ω and $\odot$ is a binary operator.

As, in this paradigm, *modularity* (Equation 5) employs *coverage* (Equation 1) and *subtraction*, the concept of *lucidity* is a true generalization of *modularity*. For unweighed graphs *modularity* has been defined as (see [26] and [11]):

$$\mathrm{mod}(\mathcal{C}) := \frac{m(\mathcal{C})}{m} - \frac{1}{4m^2} \sum_{C \in \mathcal{C}} \left(\sum_{v \in C} \deg(v) \right)^2 \ , \text{ or equivalently:} \tag{5}$$

$$\mathrm{mod}(\mathcal{C}) = \sum_{\{u,v\} \in V^{\times}} \left(\frac{A(u,v)}{m} \delta_{uv} \right) - \sum_{(u,v) \in V^2} \left(\frac{\deg(u) \cdot \deg(v)}{4m^2} \delta_{uv} \right) \ , \tag{6}$$

$$\text{with } \delta_{uv} = \begin{cases} 1 & \text{if } \mathcal{C}(u) = \mathcal{C}(v) \\ 0 & \text{otherwise} \end{cases} \ ,$$

and $A(u,v) =$ number of (parallel) edges between u and v.

using Kronecker's symbol δ_{uv} as an indicator function. Originally, loops and parallel edges were disregarded, and the original definitions are inconsistent, if such were allowed. However, since the founding probabilistic assumptions are not sound without them, as we shall see below, it is meaningful to faithfully generalize the formulations in [26] and [11], as in Equations 5 and 6. Restricting the above formulations to simple input graphs yields the original terms.

3.1 A Probabilistic Setup

The question that motivates this subsection is: Is there a sound probability space underlying the definition of *modularity*? The random models proposed below are thus not intended to be particularly elegant or universal, but they serve as a support for *modularity* and *lucidity*. In the following we discuss a suitable probability space (Ω, p) required for Definition 1, which we use throughout this paper. We restrict ourselves to the unweighted case for now and discuss a weighted setup later. In their definition of *modularity*, the authors of [26] and [11] suggest

setting the probability of a randomly inserted edge to become $\{u, v\}$ $(u \neq v)$ to $\deg(v)\deg(w)/2m^2$. The motivation for this, and thus the assumed underlying principle by which the graph is built, is a random process that inserts m edges into the disconnected set of n nodes, of which both ends then connect to node x with probability proportional to the degree $\deg(x)$ of x in the input, i.e., $\deg(x)/2m$, see Figure 1. However, note that this model also assigns a probability of $(\deg(v))^2/4m^2$, to a self loop on v. In this, the model differs from those used in [27, 10]. Thus we obtain

$$p(e) := \begin{cases} \frac{\deg u \deg v}{2m} & \text{if } e = \{u, v\}, u \neq v \\ \frac{(\deg v)^2}{4m} & \text{if } e = \{v, v\} \end{cases} . \quad (7)$$

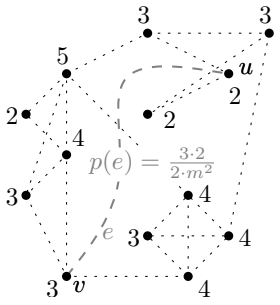


Figure 1: Input graph (dashed) with original node degrees inducing edge probabilities.

As follows, this setup is unbiased, i.e., probability masses of edges add up to 1:

$$\sum_{\{u,v\} \in V^{\times}} p[e = \{u, v\}] = \underbrace{\sum_{v > w \in V} \frac{\deg v \deg w}{2m^2}}_{\text{non-loops}} + \underbrace{\sum_{v \in V} \frac{(\deg v)^2}{4m^2}}_{\text{loops}} \quad (8)$$

$$= \sum_{v,w \in V} \frac{\deg v \deg w}{4m^2} = \frac{1}{4m^2} \left(\sum_{v \in V} \deg v \right)^2 = \frac{1}{4m^2} (2m)^2 = 1$$

In the case that edges are not allowed to form loops, the above assumptions are incorrect and overestimate the number of intra-cluster edges, since the intra-cluster edge mass contributed by loops has to be distributed elsewhere. Thus, the discrete probability space (Ω_E, p) for edge insertions uses as Ω_E all unordered pairs (two-element multisets) $\{u, v\} \in V^{\times}$, and $p(\{u, v\})$ is defined as above. Clearly, the probability function p is nonnegative, and the sample space Ω_E is normed to 1 by Equation 10. A trial consisting of m edges being drawn independently as elementary events from Ω_E, by symmetry and using the above probabilities, yields an expected number of $(\deg(u)\deg(v))/(2m)$ (parallel) edges between u and v, for two nodes $u \neq v$, and an expected number $(\deg(v))^2/(4m)$ of self loops on v. These very values are used in the definition of *modularity* in Equation 6, and induce a probability space for graphs.

By fixing the number m of edges and expected node degrees, the above setup is rather restrictive. In the *lucidity* paradigm, different random models are conceivable, if other or less properties of a graph are considered to be fixed according to the application. However, given the ideas of the founders of *modularity* and the fact that a sound probability space for graphs in accordance with its formula can be given, we restrict ourselves to that setup in this work.

From Edges to Graphs. Since an edge set E' is a multiset of elementary events in Ω_E, we may build upon this setup and define the discrete probability

space (Ω, p) for graphs as follows. Let Ω consist of all m-element multisets of elementary events in (Ω_E, p), which is a subset of the set of all multisets over Ω_E. We can now trivially identify the family of all graphs on n (labeled) nodes and m (unlabeled) edges with Ω. The probability for a specific graph $H = (V, E')$ in this family can then be chosen to directly reflect the edge probabilities (see Equation 7) in the definition of *modularity*: Using edge probabilities $p(e)$ as defined in (Ω_E, p) (see Equation 7), in space (Ω, p), let

$$p(H) := \underbrace{\prod_{e \in E'} p(e)}_{\substack{\text{prob. of } \textit{one} \text{ ordering} \\ \text{of the events in } E'}} \cdot \underbrace{\frac{m!}{\prod_{e \in E'} s_e!}}_{\substack{\text{number of orderings} \\ \text{which yield } E'}} \qquad (s_e = \text{multipl. of } e \text{ in } E') \tag{9}$$

be the probability of the event that the m elementary events from Ω_E result in the multiset E' and thus induce H.[1] From the above construction of (Ω, p) a random process for graph creation is immediate: draw m edges independently, each according to (Ω_E, p). Equations 7 and 8 yield that this model is unbiased, yielding m expected edges. Again, since edges are drawn independently, it is easy to see that this probability space is sound, i.e., that $p(H) \geq 0$ and that $\sum_{H \in \Omega} p(H) = 1$. The former claim is trivial by Equations 9 and 7, and the latter can be seen as follows. As opposed to the above, suppose for now the drawings to be *labeled*, i.e., it matters in which order edges are drawn, and let this setup be $(\dot{\Omega}, p)$. Then we obtain $|V^\times|^m = \tilde{m}^m$ different elementary events $\dot{\delta}$ in $\dot{\Omega}$ (some of which represent identical graphs, merely with edges added in a different order). Analogous to (Ω, p) (Equation 9), we may now define $p(\dot{\delta}) = \prod_{e \in \dot{\delta}} p(e)$ for all $\dot{\delta} \in \dot{\Omega}$, and get the following lemma:

Lemma 1 *The probability spaces* $(\dot{\Omega}, p)$ *and* (Ω, p) *are normed to* 1.

Proof:

$$\sum_{\dot{\delta} \in \dot{\Omega}} \prod_{e \in \dot{\delta}} p(e) = \sum_{\substack{E' \in \\ (V^\times)^m}} \prod_{e \in E'} p(e) = \Big(\sum_{e \in V^\times} p(e) \Big)^m = 1^m = 1 \ . \tag{10}$$

The first two equalities exploit the independence of $p(e)$ and reorder terms, and the third equality holds by Equation 8. Given that $(\dot{\Omega}, p)$ is normed to 1, for (Ω, p) we can summarize terms that represent the same unordered multiset (graph) as shown in Equation 9 and obtain that (Ω, p) is normed to 1. □

What is left to be shown is that for any given graph G and clustering $\mathcal{C}(G)$, $\mathbb{E}(\text{cov}(\mathcal{C}))$ in (Ω, p) equals the term in *modularity* (see Equation 6):

[1] This multiplicity is accounted for by the second factor in Equation 9. This factor can be seen as follows: there are $m!$ possibilities to order m events, but since the s_i drawings of event i are indistinguishable, $s_i!$ of these $m!$ orderings are identical; as this applies to the multiplicities of all events, we obtain the given factor. It equals $m!$ iff $s_e = 1$ for all $e \in E'$, and 1 iff $s_e = m$ for some $e \in E'$.

Lemma 2 *For any given graph G and clustering $\mathcal{C}(G)$, in (Ω, p) it holds that:* $\mathbb{E}(\mathrm{cov}(\mathcal{C})) = \sum_{(u,v)\in V^2} \frac{\deg(u)\deg(v)}{4m^2}\delta_{uv}$ *. (As above s_e denotes e's multiplicity.)*

Proof:

$$\mathbb{E}(\mathrm{cov}(\mathcal{C})) = \mathbb{E}\left(\frac{\sum_{e\in E(\mathcal{C})} s_e}{m}\right) = \frac{1}{m}\mathbb{E}\left(\sum_{e\in E(\mathcal{C})} s_e\right) = \frac{1}{m}\sum_{e\in E(\mathcal{C})}\mathbb{E}(s_e)$$

$$= \frac{1}{m}\left(\sum_{\substack{e=\{u,v\}\in E(\mathcal{C})\\ u\neq v}} \frac{\deg(u)\deg(v)}{2m} + \sum_{\substack{e\in E(\mathcal{C})\\ e=\{v,v\}}} \frac{(\deg(v))^2}{4m}\right) = \sum_{(u,v)\in V^2} \frac{\deg(u)\deg(v)}{4m^2}\delta_{uv}$$

□

Examining the above proof we can see that any distribution that (i) fulfills Equation 7 and (ii) surely uses a total number m of edges, has the property described in Lemma 2. Moreover we can immediately see that the additional postulation that expected node degrees should be fixed is also fulfilled.

Corollary 2 *The expected edge degree of node v in (Ω, p) is* $\deg(v)$ *(from G).*

Proof:

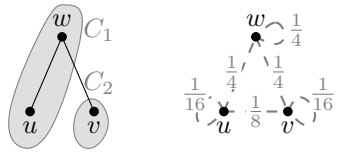
$$\mathbb{E}_\Omega(\deg(v)) = \sum_{\substack{u\in V\\ u\neq v}} \frac{\deg(u)\deg(v)}{2m}\cdot 1 + \frac{(\deg(v))^2}{4m}\cdot 2$$

$$= \deg(v)\left(\frac{2m-\deg(v)}{2m} + \frac{\deg(v)}{2m}\right) = \deg(v)$$

□

The above proof uses the discussed edge probabilities; note that a self loop (second summand) contributes 2 to $\deg(v)$. Concluding, we now have a sound probabilistic setup for unweighted graphs for the *lucidity* paradigm.

An Instructive Example. The following tiny example illustrates this model. Let graph $G = (V, E)$ in the righthand Figure 2a be given, with $n = 3, m = 2$, alongside a clustering $\mathcal{C}$. Figure 2b states the edge probabilities according to Equation 7, comprising $\tilde{m} = \binom{n}{2} + n = 6$ possible edges. We first consider only one random edge: This yields the family $\Omega_1 = \mathcal{H}_1$ of the 6 graphs on three nodes and one edge, their probabilities match the corresponding edge probabilities in Figure 2b, which completes space (Ω_1, p). Due to the independence of edge drawings, we can now build the required probability space (Ω_2, p) inducing the family $\mathcal{H}_2$ of graphs on 3 nodes and 2 edges by building the Cartesian product $\Omega_1 \times \Omega_1$. This yields 6^2 outcomes in Ω_2, whose probabilities are obtained by multiplying those of the participating members of Ω_1. Of these outcomes $\tilde{m}$ occur once (two parallel edges) and $\binom{\tilde{m}}{2}$ occur twice (different insertion orders lead

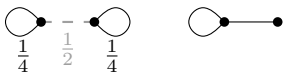


(a) G, $\mathcal{C}(G)$ (b) $p(e)$

Figure 2: Given G (a), Equation 7 yields probabilities $p(e)$ (b)

to the same graph). Consider now the clustering $\mathcal{C}$ of G depicted in Figure 2a. Equation 5 yields $\mathrm{mod}(\mathcal{C}) = \frac{1}{2} - \frac{3^2+1^2}{4\cdot 2^2} = -\frac{1}{8}$, and in particular $\mathbb{E}(\mathrm{cov}) = \frac{5}{8}$. To see that this coincides with the expected coverage in Ω_2 (i.e., $\mathcal{H}_2$) regarding $\mathcal{C}$, as theoretically proven in Lemma 2, in Figure 4 we list those 18 (of 21) members of $\mathcal{H}_2$ with positive *coverage* and check that $\sum_{H\in\mathcal{H}_2} p(H)\mathrm{cov}(\mathcal{C})_H = \frac{5}{8}$.

As an interesting side note, the example in Figure 3 shows that this setup does not necessarily grant the highest probability to the very graph used as the blueprint for the probability space. In Figure 3, probabilities are $p(H) = \frac{1}{4}$ and $p(G) = \frac{1}{8}$.



(a) G with $p(e)$ (b) $p(H) = \max$

Figure 3: A graph G and one of its most likely random variants H.

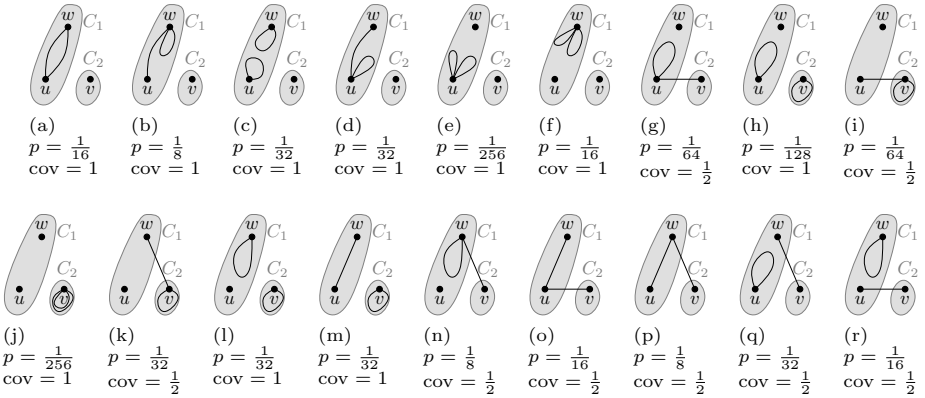


Figure 4: All graphs in $\mathcal{H}^2$ with positive *coverage* for $\mathcal{C}$, yielding $\mathbb{E}(\mathrm{cov}(\mathcal{C})) = \frac{5}{8}$. Note that graphs with non-parallel edges occur twice in Ω_2, hence their double probability.

The weighted case. A generalization of *modularity* to weighted edges, such that its restriction to weights 0 and 1 yields the unweighted version, is straightforward, as proposed in [23]. We again state the formula we use, in order to disambiguate between formulations in previous works:

$$\mathrm{mod}_\omega(\mathcal{C}) := \underbrace{\frac{\omega(\mathcal{C})}{W}}_{\mathrm{cov}_\omega} - \underbrace{\frac{1}{4W^2}\sum_{C\in\mathcal{C}}\left(\sum_{v\in C}\omega(v)\right)^2}_{\mathbb{E}(\mathrm{cov}_\omega)} \tag{11}$$

Analogous to unweighted edges, this formula assumes for expected edge weights

$$\mathbb{E}(\omega(e)) := \begin{cases} \frac{\omega(u)\omega(v)}{2W} & \text{if } e = \{u,v\}, u \neq v \\ \frac{(\omega(v))^2}{4W} & \text{if } e = \{v,v\} \end{cases} . \tag{12}$$

Note that for our view parallel edges are obsolete (even disruptive, notationally) in this setting, if we allow the edge weight function ω to go beyond 1 as $\omega : E \to$

$\mathbb{R}_0^+$ and simply summarize parallel edges to one "heavier" edge. For simplicity we shall do this in the following. Analogous to Equation 8 we can see that the choices in Equation 12 are unbiased, as the expected total edge mass $\mathbb{E}(W)$ equals W. However, now we cannot simply draw m edges independently, but have to continuously distribute an edge mass W. Analogous to Lemma 2 we can prove the following:

Lemma 3 *A probability distribution for weighted graphs will justify Equation 11 if it fulfills the following two properties:*[2]

(i) expected edge weights are as in Equation 12,

(ii) random graphs surely have a total edge mass equal to W.

We will not define a probability distribution for weighted graphs, but describe a rather simple random process, which produces a distribution which fulfills these two properties. This process starts with expected weights (as in Equation 12). Then in an arbitrary number of handshakes between random edges, two participants contest about their combined edge mass. The mass is divided up in a new random way between the two, but such that the expected ratio of the two halves matches the ratio of their respective expected weights. Suppose the two handshaking edges are e_ℓ and e_r with expected weights a_ℓ and a_r, and actual edge weights x_ℓ and x_r, respectively. Let $c := a_\ell/(a_\ell + a_r)$ be the fraction that e_ℓ expects to get. We define a piecewise uniform density function $d(x)$ as depicted in Figure 5 as follows:[3]

Figure 5: density d

$$d(x) = \begin{cases} \frac{1-c}{c} =: f_\ell & \text{if } 0 \le x \le c \\ \frac{c}{1-c} =: f_r & \text{if } c < x \le 1 \end{cases} . \quad (13)$$

Having drawn x from $d(x)$, the available weight $x_\ell + x_r$ is divided up such that e_ℓ gets a part of size $x \cdot (x_\ell + x_r)$ and e_r gets a part of size $(1-x) \cdot (x_\ell + x_r)$. Algorithm 1 summarizes this procedure.

Lemma 4 *Given a weighted graph G and a clustering $\mathcal{C}(G)$. Algorithm 1 yields a distribution of graphs with $\mathbb{E}(\mathrm{cov}_\omega)$ as used in Equation 11.*

Proof: We use Lemma 3 for the proof. Property (ii) is trivially fulfilled as edge mass W is introduced in line 1 and only moved between edges later. To see property (i) we use induction over the number of runs as coupled experiments. Ind. start: In the beginning $x_i = a_i$ for all e_i by line 1.

[2] As in the unweighted case this does not rule out the existence of different setups.

[3] In practice, random draws with density d can be done, e.g., as follows. First decide which side of c to use with the help of a single Bernoulli trial that chooses ℓ with prob. $p(\ell) = c \cdot f_\ell = 1 - c$ (and r with prob. $p(r) = (1-c) \cdot f_r = c$). Then, choose a value x uniformly at random within the chosen interval.

Algorithm 1: Random Process for Weighted Graphs

Set a_i and x_i as in Eq. 12 $\forall e_i = \{u, v\} \in V \times V$
for $\#T$ *runs* **do**
  unif. at rand. choose edges $\{\ell, r\} \in \binom{V^\times}{2}$ `// choose contestants`
  $c \leftarrow \frac{a_\ell}{a_\ell + a_r}$ `// ℓ's expected fraction`
  draw[3] $x \sim d(x)$ as in Equation 13 `// see Figure 5 for` $d(x)$
  $x_\ell \leftarrow x(x_\ell + x_r)$ and $x_r \leftarrow (1-x)(x_\ell + x_r)$ `// distribute` $x_\ell + x_r$
return Graph G with edge weights x_i

Ind. hypothesis: For all t' up to some $t \leq T$: $\mathbb{E}(x_i) = a_i$ for all e_i.
Ind. step: Given $\mathbb{E}(x_i) = a_i \; \forall \; e_i$ after run t. For the expected value of x we get:

$$\mathbb{E}(x) = \int_0^1 x d(x) dx = \int_0^c x f_\ell dx + \int_c^1 x f_r dx = f_\ell \frac{c^2}{2} + f_r \frac{1-c^2}{2} = c$$

For all e_i not chosen in run $t+1$ we get $\mathbb{E}^{t+1}(x_i) = \mathbb{E}^t(x_i)$ after $t+1$, and for the two affected edges we get (x_r analogously):

$$\mathbb{E}(x_\ell^{t+1}) = \mathbb{E}(x \cdot (x_\ell + x_r)) = \mathbb{E}(x) \cdot (\mathbb{E}^t(x_\ell) + \mathbb{E}^t(x_r)) = \frac{a_\ell \cdot (a_\ell + a_r)}{a_\ell + a_r} = a_\ell$$

□

Certainly, this process is nowhere close to yielding a uniform distribution, however it serves our particular purpose. Open questions for it include how T needs to be chosen, in order to have drawn graphs be more or less independent.

The Loop-Free Case. Suppose now we disallow loops, but still adopt the intuition that a randomly inserted edge should become incident with node v with probability proportional to $\deg(v)$ in the unweighted case. Analogous to Figure 1 and the derivation of *modularity*, we can now derive the probability of a random edge in a loop-free setup to become:

$$p_\varnothing(\{u,v\}) = \underbrace{\frac{\deg(u)}{2m} \cdot \frac{\deg(v)}{2m - \deg(u)}}_{\substack{=p_\varnothing((u,v)) \\ \text{“first connect to } u \text{ then to } v\text{”}}} + \underbrace{\frac{\deg(v)}{2m} \cdot \frac{\deg(u)}{2m - \deg(v)}}_{\substack{=p_\varnothing((v,u)) \\ \text{“first connect to } v \text{ then to } u\text{”}}}$$

$$= \frac{\deg(u)\deg(v) \cdot (\overline{\deg}(u) + \overline{\deg}(v))}{2m \cdot \overline{\deg}(u)\overline{\deg}(v)} \qquad \text{using } \overline{\deg}(v) = 2m - \deg(v) \quad (14)$$

Analogous to Equation 8 we can observe that this setup is normed to 1. For easier summation we suppose for a moment that the graph was directed, and we write the above probability for edge $\{u, v\}$ as the sum of the probabilities of the two directed edges (u, v) and (v, u), as in the derivation of Equation 14.

$$\sum_{\substack{\{u,v\}\in V\\ u\neq v}} p_\emptyset(\{u,v\}) = \sum_{\substack{\{u,v\}\subseteq V\\ u\neq v}} (p_\emptyset((u,v)) + p_\emptyset((v,u))) = \sum_{v\in V}\sum_{\substack{u\in V\\ u\neq v}} p_\emptyset((v,u))$$
$$= \sum_{v\in V}\sum_{\substack{u\in V\\ u\neq v}} \frac{\deg(v)\deg(u)}{2m\cdot\overline{\deg}(v)} = \frac{1}{2m}\sum_{v\in V}\frac{\deg(v)}{\overline{\deg}(v)}\overline{\deg}(v) = 1$$

Using arguments from the previous sections we can now setup a discrete probability space $(\Omega_\emptyset, p_\emptyset)$ for loop-free graphs in an analogous way. By drawing m independent edges according to Equation 14 we obtain probabilities for graphs similar to Equation 9 and a lemma analogous to Lemma 1. Even our arguments concerning a weighted version (using total weight W) and a random process for weighted graphs in Section 3.1 carry over, yielding an expected edge weight of $\mathbb{E}_\emptyset(\omega(u,v)) = \omega(u)\omega(v)\cdot(\overline{\omega}(u)+\overline{\omega}(v))/(2\overline{\omega}(u)\overline{\omega}(v))$, using $\overline{\omega}(v) := 2W - \omega(v)$ (compare to Eq. 14). A variant $\mathit{modularity}_\emptyset$ for loop-free graphs could thus be defined as (compare to Formulas 6 and 11):

$$\mathrm{mod}_\emptyset(\mathcal{C}) := \sum_{\substack{\{u,v\}\subseteq V\\ u\neq v}} \left(\frac{\omega(u,v)}{W} - \frac{\omega(u)\omega(v)\cdot(\overline{\omega}(u)+\overline{\omega}(v))}{2W\cdot\overline{\omega}(u)\overline{\omega}(v)}\right)$$

It is important to note that this formulation does *not* fulfill Corollary 2: the intuition of making random edges incident to v with probability proportional to $\deg(v)$ does not generally preserve expected node degrees in the loop-free model. Devising such a model is much harder, simply using $p(\{u,v\}) = \deg(u)\deg(v)$, normalized, does not work. Excluding parallel edges makes issues even worse, thus we stop here and postpone such thoughts to future work. While sophisticated random processes have been proposed for simple random graphs with a predefined degree sequence, e.g., in [32], we require a formulation which yields edge probabilities in a closed form, in order to support a formula for *modularity*.

3.2 Implementations of the Lucidity Paradigm

The building blocks presented above enable us to study four implementations of the *lucidity* paradigm, namely, *coverage* and *performance* as quality indices and *subtraction* and *division* as the binary operators. Using *coverage* and subtraction, *modularity* is one of these implementations. For a discussion of *performance* [31] in weighted graphs we refer the reader to [5]. However, one aspect needs particular attention: *Performance* evaluates node pairs based on their being connected or not. Switching to weighted edges now requires a meaningful assumption (see [5]) about a maximum edge weight M to compare to, in order to measure, e.g., how missing inter-cluster edges contribute. The above main references for *performance* are not specific about M and thus three possible choices for M are immediate: $\omega_{\max}$ of G, 1 (being the maximum allowed edge weight), or W. The canonic formulation is (compare Equation 2)

$$\text{perf}_\omega = \frac{\omega(\mathcal{C}) + M\overline{m}(\mathcal{C})^c + M\overline{m}(\mathcal{C}) - \overline{\omega}(\mathcal{C})}{\frac{1}{2}n(n-1)M} \quad . \tag{15}$$

Any choice for M which is a parameter dependent on G (such as $\omega_{\max}$) becomes a random variable in (Ω, p). However, there is a more fundamental objection against using $\omega_{\max}$: in a fixed random model, there should be a fixed maximum weight to compare to. On the other hand, keeping the value of $\omega_{\max}$ in G, being one specific draw, as a fixed constant for all draws, seems equally inappropriate. As a better choice for M, the range of the weight function ω should be used, which will often be 1. Thus, in the following we assume M to be some choice of a constant, which leads to the following lemma:

Lemma 5 *Using the probability space described in Section 3.1 and an arbitrary but fixed constant M, the expected value of* performance *is*

$$\frac{\sum_{C\in\mathcal{C}}(\sum_{v\in C}\omega(v))^2/W + M(n^2 - \sum_{C\in\mathcal{C}}|C|^2) - 2W}{n(n-1)M}$$

Proof: We split Equation 15 into edges (first term in numerator) and non-edges (remains). Again we use (Ω, p), i.e., Equation 12 for expected edge weights.

$$\underbrace{\mathbb{E}\left(\frac{\omega(\mathcal{C})}{\frac{1}{2}n(n-1)M}\right)}_{\mathbb{E}_1} = \frac{\frac{1}{4W}\sum_{C\in\mathcal{C}}\left(\text{vol}_\omega(C)\right)^2}{\frac{1}{2}n(n-1)M} = \frac{\frac{1}{2W}\sum_{C\in\mathcal{C}}\left(\text{vol}_\omega(C)\right)^2}{n(n-1)M}$$

$$\underbrace{\mathbb{E}\left(\frac{M\overline{m}(\mathcal{C})^c + M\overline{m}(\mathcal{C}) - \overline{\omega}(\mathcal{C})}{\frac{1}{2}n(n-1)M}\right)}_{\mathbb{E}_2} = \frac{\mathbb{E}\left(\sum\limits_{\substack{e=\{u,v\}\in E\\ \mathcal{C}(u)\neq\mathcal{C}(v)}}(M-\omega(e)) + \sum\limits_{\substack{\{u,v\}\notin E\\ \mathcal{C}(u)\neq\mathcal{C}(v)}} M\right)}{\frac{1}{2}n(n-1)M}$$

$$= \frac{\frac{1}{2}\sum\limits_{C\in\mathcal{C}}\sum\limits_{C'\in\mathcal{C}\setminus C}\sum\limits_{(v,w)\in C\times C'}\left(M - \frac{\omega(v)\omega(w)}{2W}\right)}{\frac{1}{2}n(n-1)M}$$

$$= \frac{M(n^2 - \sum_{C\in\mathcal{C}}|C|^2)}{n(n-1)M} - \frac{\frac{1}{4W}\sum_{C\in\mathcal{C}}\sum_{v\in C}\omega(v)(2W - \text{vol}_\omega(C))}{\frac{1}{2}n(n-1)M}$$

$$= \frac{M(n^2 - \sum_{C\in\mathcal{C}}|C|^2)}{n(n-1)M} - \frac{\frac{1}{4W}4W^2 - \frac{1}{4W}\sum_{C\in\mathcal{C}}(\text{vol}_\omega(C))^2}{\frac{1}{2}n(n-1)M}$$

$$\mathbb{E}(\text{perf}_\omega) = \mathbb{E}_1 + \mathbb{E}_2 = \frac{\frac{1}{W}\sum\limits_{C\in\mathcal{C}}\left(\sum\limits_{v\in C}\omega(v)\right)^2 - 2W + M\left(n^2 - \sum\limits_{C\in\mathcal{C}}|C|^2\right)}{n(n-1)M} \tag{16}$$

□

measure	*coverage*	*performance*
$\mathcal{M}$	$\frac{m(\mathcal{C})}{m}$	$\frac{m(\mathcal{C})+\overline{m}(\mathcal{C})^c}{0.5\cdot n(n-1)}$
$\mathbb{E}[\mathcal{M}]$	$\sum_{C\in\mathcal{C}}\left(\frac{\sum_{v\in C}\deg(v)}{2m}\right)^2$	$\frac{\sum_{C\in\mathcal{C}}((\sum_{v\in C}\deg(v))^2/m-(\sum_{v\in C}1)^2)+n^2-2m}{n(n-1)}$
$\mathcal{M}_\omega$	$\frac{\omega(\mathcal{C})}{W}$	$\frac{\omega(\mathcal{C})+M\overline{m}(\mathcal{C})^c+(M\overline{m}(\mathcal{C})-\overline{\omega}(\mathcal{C}))}{0.5\cdot n(n-1)M}$
$\mathbb{E}[\mathcal{M}_\omega]$	$\sum_{C\in\mathcal{C}}\left(\frac{\sum_{v\in C}\omega(v)}{2W}\right)^2$	$\frac{\sum_{C\in\mathcal{C}}(\sum_{v\in C}\omega(v))^2/W+M(n^2-\sum_{C\in\mathcal{C}}\lvert C\rvert^2)-2W}{n(n-1)M}$

Table 1: Quality indices and expected values (M: maximum edge weight in the model). The subscript "ω" indicates edge-weighted versions.

We can now state an overview summarizing the formulas of the resulting four implementations of the *lucidity* paradigm in Table 1. The straightforward weighted variant of $\mathrm{L}^-_{\mathrm{cov}}$ has been described by [23]. Based on Table 1 we now define the following implementations:

$$\mathrm{L}^-_{\mathrm{cov}} := \mathrm{cov} - \mathbb{E}[\mathrm{cov}] \quad \text{(equals modularity)} \qquad \mathrm{L}^{\div}_{\mathrm{cov}} := \frac{\mathrm{cov}}{\mathbb{E}[\mathrm{cov}]} \tag{17}$$

$$\underbrace{\mathrm{L}^-_{\mathrm{perf}} := \mathrm{perf} - \mathbb{E}[\mathrm{perf}]}_{\text{absolute variants (subtractive)}} \qquad \underbrace{\mathrm{L}^{\div}_{\mathrm{perf}} := \frac{\mathrm{perf}}{\mathbb{E}[\mathrm{perf}]}}_{\text{relative variants (divisive)}} \tag{18}$$

Corollary 3 *A constant M for weighted $\mathrm{L}^-_{\mathrm{perf}}$ is a scaling factor, which means that an observation $\mathrm{L}^-_{\mathrm{perf}}(\mathcal{C}(G)) \geq \mathrm{L}^-_{\mathrm{perf}}(\mathcal{C}'(G))$ is M-invariant.*

Proof: From Lemma 5 it it not hard to see, that some terms from perf_ω have survived in $\mathbb{E}(\mathrm{perf}_\omega)$, which for simplicity we denote:

$$\Phi = M\overline{m}(\mathcal{C})^c + M\overline{m}(\mathcal{C}) = \frac{1}{2}M(n^2 - \sum_{C\in\mathcal{C}}|C|^2) \tag{19}$$

Rewriting and summarizing $\mathrm{L}^-_{\mathrm{perf}}$ yields the following term, which uses M only as a factor in the denominator, as an inverse scaling factor.

$$\begin{aligned} \mathrm{L}^-_{\mathrm{perf}} &= \mathrm{perf}_\omega - \mathbb{E}(\mathrm{perf}_\omega) \\ &= \frac{\omega(\mathcal{C}) + \Phi - \overline{\omega}(\mathcal{C})}{\frac{1}{2}\cdot n(n-1)M} - \frac{\sum_{C\in\mathcal{C}}(\mathrm{vol}_\omega(C))^2/W + 2\Phi - 2W}{n(n-1)M} \\ &= \frac{\omega(\mathcal{C}) - \overline{\omega}(\mathcal{C}) - \frac{1}{2W}\sum_{C\in\mathcal{C}}(\mathrm{vol}_\omega(C))^2 - W}{\frac{1}{2}n(n-1)M} \end{aligned} \tag{20}$$

□

We refrain from a discussion of the usage of *lucidity* on graphs with a fuzzy clustering, which allows clusters to overlap, i.e., nodes may belong to several

clusters. However we point the reader to two recent works which consistently generalize *modularity* to the overlapping case. These are [28], which also proposes a generalization to directed graphs, and [22] which discusses the former and proposes sound improvements. Summarizing, the introduction of *belonging factors* of nodes to clusters, as proposed by these two works, can immediately be applied to *coverage* and *performance* and thus also to the implementations of *lucidity* discussed herein, but not necessarily to any implementation.

3.3 The Equivalence of $\mathrm{L}^-_{\mathrm{perf}}$ and $\mathrm{L}^-_{\mathrm{cov}}$

As is well known from [4, 3], $\mathrm{L}^-_{\mathrm{cov}}$ can be optimized via ILP[4] formulation: Constraints ensure a consistent partition of the nodes by formalizing an equivalence relation on the nodes, deciding whether two nodes are in the same cluster. The linear target function follows directly from the weighted version of Equation 6:

$$\text{weighted } \mathrm{L}^-_{\mathrm{cov}} = \sum_{\{u,v\}\in V^{\times}} \left(\frac{\omega(u,v)}{W} X_{uv}\right) - \sum_{(u,v)\in V^2} \left(\frac{\omega(u)\omega(v)}{4W^2} X_{uv}\right) \tag{21}$$

$$\text{with } X_{uv} = [\delta_{uv} =] \begin{cases} 1 & \text{if } u,v \text{ in same cluster} \\ 0 & \text{otherwise} \end{cases}$$

A similar formulation is possible for $\mathrm{L}^-_{\mathrm{perf}}$. We first build upon the formula for $\mathrm{L}^-_{\mathrm{perf}}$ derived in Equation 20, and then rewrite it:

$$\begin{aligned}
\text{weighted } \mathrm{L}^-_{\mathrm{perf}} &= \frac{\omega(\mathcal{C}) - \overline{\omega}(\mathcal{C}) - \frac{1}{2W}\sum_{C\in\mathcal{C}}(\sum_{v\in C}\omega(v))^2 - W}{\frac{1}{2}n(n-1)M} \\
&= \frac{\sum_{\{u,v\}\in V^2}\omega(u,v)X_{uv} - \sum_{\{u,v\}\in V^2}\omega(u,v)(1-X_{uv}) - \sum_{(u,v)\in V^2}\frac{\omega(u)\omega(v)}{2W}X_{uv} - W}{\frac{1}{2}n(n-1)M} \\
&= \frac{2\sum_{\{u,v\}\in V^2}\omega(u,v)X_{uv} - \frac{1}{2W}\sum_{(u,v)\in V^2}\omega(u)\omega(v)X_{uv} - 2W}{\frac{1}{2}n(n-1)M} \\
&= \frac{\sum_{\{u,v\}\in V^2}\frac{\omega(u,v)}{W}X_{uv} - \sum_{(u,v)\in V^2}\frac{\omega(u)\omega(v)}{4W^2}X_{uv}}{\underbrace{\frac{1}{4W}n(n-1)M}_{a}} - \underbrace{\frac{1}{\frac{1}{4W}n(n-1)M}}_{b}
\end{aligned} \tag{22}$$

We now trim Formula 22 by removing the second summand (b) and the (main) denominator (a), which are both invariant under X_{uv} and obtain Formula 21. This yields the following lemma:

[4]ILP stands for integer linear program.

Lemma 6 (Equivalence of $\mathrm{L}^-_{\mathrm{perf}}$ and $\mathrm{L}^-_{\mathrm{cov}}$) *The problem of optimizing $\mathrm{L}^-_{\mathrm{perf}}$ and that of optimizing $\mathrm{L}^-_{\mathrm{cov}}$ are equivalent, furthermore*

$$\mathrm{L}^-_{\mathrm{cov}}(G,\mathcal{C}_1) > \mathrm{L}^-_{\mathrm{cov}}(G,\mathcal{C}_2) \quad \Longleftrightarrow \quad \mathrm{L}^-_{\mathrm{perf}}(G,\mathcal{C}_1) > \mathrm{L}^-_{\mathrm{perf}}(G,\mathcal{C}_2) \tag{23}$$

This lemma together with the NP-completeness of optimizing *modularity* [3], immediately gives us the following corollary:

Corollary 4 *Given a graph G (weighted or unweighted) and a real L. It is NP-complete to decide whether there is a clustering $\mathcal{C}(G)$ with $\mathrm{L}^-_{\mathrm{perf}}(\mathcal{C}(G)) \geq L$.*

The deduction of the equivalence in Lemma 6 implies that a linear relation between the values of $\mathrm{L}^-_{\mathrm{perf}}$ and $\mathrm{L}^-_{\mathrm{cov}}$ for a given instance G and an arbitrary clustering $\mathcal{C}(G)$ can be given in the form $\mathrm{L}^-_{\mathrm{perf}} = a(G) \cdot \mathrm{L}^-_{\mathrm{cov}} + b(G)$. Coefficients a and b both depend on the instance G and are the very terms mentioned above (see Equation 22). Together with the fact that both $\mathrm{L}^-_{\mathrm{perf}}$ and $\mathrm{L}^-_{\mathrm{cov}}$ can attain the value 0, even for the respective optimum clusterings, this yields that relative approximation guarantees do not easily carry over in either direction. In any way, to our best knowledge, no positive results on the approximability of either $\mathrm{L}^-_{\mathrm{perf}}$ or $\mathrm{L}^-_{\mathrm{cov}}$ exist. Note that Formula 20 can be trimmed further, such that in Formula 24 we obtain a very simple but equivalent target function for maximizing $\mathrm{L}^-_{\mathrm{cov}}$ (or $\mathrm{L}^-_{\mathrm{perf}}$) in, e.g., an ILP:

$$\mathrm{L}^-_{\mathrm{cov}} \cong \sum_{\{u,v\} \in \binom{V}{2}} \left(\left(\omega(u,v) - \frac{\omega(u)\omega(v)}{2W} \right) X_{uv} \right) \tag{24}$$

3.4 The Relation to MinMixedMultiPartition

Note that the ILP formulation in Equation 24 has an equivalent *metric* version, i.e., $X_{uv} = 1$ iff nodes u and v are in *different* clusters. The problem thus changes to minimizing the same target function: Instead of maximizing the edge contributions inside clusters, we minimize those in between. This is equivalent to finding the minimum weight edge set inducing a (multi-)partition on the complete graph $\mathcal{K}$ on V, where edge weights g are equal to the (simplified) term in brackets in Equation 24. Given an unweighted instance of $\mathrm{L}^-_{\mathrm{cov}}$ (i.e., *modularity*), edge weights in $\mathcal{K}$ are multiples of $1/m$, thus we can assume $g \in \mathbb{Z}$. We formalize the general form of this problem as follows:

Definition 5 (MinMixedMultiPartition**)** *Consider an undirected graph $\mathcal{K} = (V,E)$, an edge weight function $g : E \to \mathbb{Z}$ and a rational number L. Is there a partition of V into disjoint subsets $V_1, \ldots, V_m$ ($m \geq 1$) such that the sum of weights of edges whose endpoints lie in different subsets is at most L?*

By the fact that optimizing $\mathrm{L}^-_{\mathrm{cov}}$ is NP-hard, we obtain the following corollary:

Corollary 6 *The problem* MinMixedMultiPartition *is NP-hard.*

For a weighted L^{-}_{cov} instance, a similar observation holds, if we assume that original edge weights are rational, $\omega(e) \in \mathbb{Q}$. Although many similar hardness results on cuts in graphs exist, we are not aware of a proof of this particular variant. Well known hardness results in this context have been presented, e.g., by [17] for GraphPartition or MaxCut, and by [12] for MinimumMultiwayCut, where in a positively weighted graph a set of terminals $T \subseteq V$ has to be separated. Note that MinMixedMultiPartition is not, as it might seem, a straightforward generalization of the NP-hard problem MixedMinCut (i.e., MaxCut), in that the set of cut edges of a MixedMinCut is a subset of the set of edges cut by MinMixedMultiPartition, as can be disproven by the simple example in Figure 6. Moreover instances exist where the solution to MinMixedMultiPartition is the trivial partition $\{V\}$ (e.g., in the case of exclusively positive weights), such that for obvious reasons no MixedMinCut can be deduced. This emphasizes the relevance of Corollary 6.

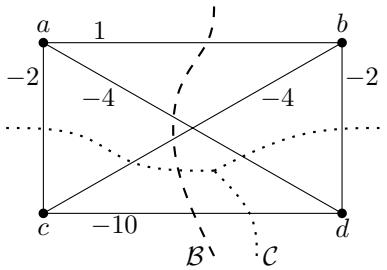


Figure 6: The (unique) minimum multipartition $\mathcal{C}$ with $\text{cost}(\mathcal{C}) = -22$ does not directly induce the (unique) minimum bipartition $\mathcal{B}$ with $\text{cost}(\mathcal{B}) = -17$.

4 Lucidity-Clustering Algorithms

The optimization of L^{-}_{cov} being NP-complete ([4], [3]) encourages the usage of heuristics or approximations. In this section, we briefly describe the algorithms we use for *lucidity* maximization. Throughout our experiments, we employ a greedy heuristic approach, allowing for a consistent evaluation of the four variants of lucidity, as follows. For a given *lucidity* measure L the greedy algorithm starts with the singleton clustering and iteratively merges those two clusters that yield the largest increase or the smallest decrease in L. After a maximum of $n-1$ merges the intermediate clustering that achieved the highest value of L is returned. The algorithm maintains a symmetric matrix ΔL with entries $\Delta L_{i,j}$ equaling $L(\mathcal{C}_{i,j}) - L(\mathcal{C})$, where $\mathcal{C}$ is the current clustering and $\mathcal{C}_{i,j}$ is obtained from $\mathcal{C}$ by merging clusters C_i and C_j. The pseudo-code for the greedy algorithm is given in Algorithm 2. In this work, we refrain from delving into the many variants of this pure greedy approach that can be found in the literature, for the sake of brevity.

Let ΔL be defined by matrices $\Delta\mathcal{M}$ and $\Delta\mathbb{E}[\mathcal{M}]$, denoting the additive changes in $\mathcal{M}$ and in $\mathbb{E}[\mathcal{M}]$, respectively. Then, when merging C_i and C_j, entries $\Delta\mathcal{M}_{pq}$ of unaffected clusters do not change, while entries in rows and columns i and j are updated as follows: $\Delta\mathcal{M}_{k,(ij)} := \Delta\mathcal{M}_{k,i} + \Delta\mathcal{M}_{k,j}$, where $C_{(ij)} = C_i \cup C_j$ (and $\Delta\mathbb{E}[\mathcal{M}]$ is updated analogously). From Equations 17 and 18 it becomes clear that for the absolute variants this transfers to ΔL, while for the relative variants all entries of ΔL change, even those of unaffected clusters.

Algorithm 2: Greedy Lucidity

Input: Graph $G = (V, E, \omega)$
Output: Clustering $\mathcal{C}$ of G
1 $\mathcal{C} \leftarrow$ Singletons, initialize L
2 Initialize matrix ΔL (with $\Delta L_{ij} \cong$ change in L when merging C_i and C_j)
3 **while** $|\mathcal{C}| > 1$ **do**
4 Find $\{i, j\}$ with $\Delta L_{i,j} = \arg\max \Delta L_{ij}$
5 Merge clusters C_i and C_j
6 Update ΔL
7 $L \leftarrow L + \Delta L_{i,j}$
8 Return intermediate clustering with highest *lucidity*

4.1 Runtime Analysis

Both absolute variants of *lucidity* share the same asymptotic running time. Employing a standard data structure for clusterings, one observes that Step 1 and 8 run in $O(n)$ time. Matrix ΔL is initialized in $O(n^2)$ time. The loop at Step 3 is executed $n - 1$ times. Step 4 runs in $O(n)$ time, if we store the rows of ΔL as heaps. Merging two clusters (Step 5) and updating L (Step 7) require at most linear time. Thus, updating ΔL dominates, which consists of $O(n)$ insertions and deletions from heaps, requiring $O(\log n)$ each. This yields the following lemma:

Lemma 7 *Algorithm 2 runs in $O(n^2 \log n)$ time for the absolute variants.*

Adapting Lemma 7 to the relative variants yields a runtime of $O(n^3)$, since a merge entails an update of n^2 matrix entries. However, in Lemma 8 and Algorithm 3 we improve on this. It is not hard to see that the first local optimum of L, that the absolute greedy heuristic attains, is its global optimum. In case the number of clusters is dependent on n, i.e., $|\mathcal{C}| \in \omega(1)$, this may result in an asymptotic decrease in running time. Since only merges of connected clusters can increase L, for sparse graphs a runtime of $O(md \log(n))$ can then be seen (with d being the maximum number of merges per node) as shown in [11].

4.2 Quick Divisive Merge

In this section we describe how for relative variants, the running time for updating ΔL can be reduced by avoiding explicit matrix updates. We give an algorithm that updates ΔL in $O(n \log n)$ amortized time using a geometric embedding. We store matrix ΔL by a point set P in the plane as follows and as depicted in Figure 7. Each entry $\{i, j\}$ is represented by a point p_{ij} with coordinates $p_{ij} := (\mathcal{M}(\mathcal{C}_{i,j}), \mathbb{E}[\mathcal{M}(\mathcal{C}_{i,j})]) = (\mathcal{M}+\Delta\mathcal{M}_{ij}, \mathbb{E}[\mathcal{M}]+\Delta\mathbb{E}[\mathcal{M}]_{ij})$. Thus, each point encodes the measure (y-axis) of a clustering and its expectation (x-axis). Since these are both non-negative, all points are in quadrant one. We additionally insert one point $R = (\mathcal{M}(\mathcal{C}), \mathbb{E}[\mathcal{M}(\mathcal{C})])$ that represents the current clustering.

Since $\mathcal{M}$ and $\mathbb{E}[\mathcal{M}]$ update additively, we can update each point p in the plane, after merging two clusters C_k and C_l, as follows: First, for a linear number of points p (i.e., those involving C_k and C_l), we set $p_{i,(kl)} \leftarrow p_{i,k} + p_{i,l} - R$. By doing so we actually both delete and introduce a linear number of points. Second, for all points P, including those newly introduced, we set $p \leftarrow p+(p_{kl}-R)$. These steps maintain the data structure.

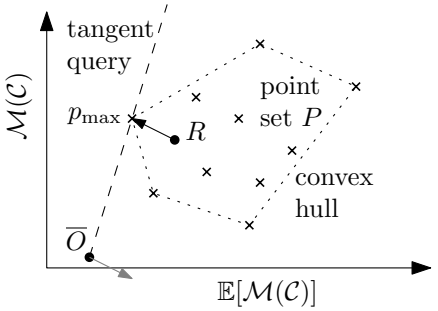


Figure 7: Each cross encodes the quality of some merge, with $p_{\max}$ yielding the highest quotient. Instead of all crosses, $\overline{O}$ moves antipodally by $R - p_{\max}$ (gray arrow). Due to some earlier step, $\overline{O}$ has already been shifted away from $(0, 0)$.

There are two crucial observations: First, instead of uniformly setting $p \leftarrow p + (p_{kl} - R)$, we can save $\Omega(n^2)$ such updates by only shifting the origin: $\overline{O} \leftarrow \overline{O} - (p_{kl} - R)$. Second, at any time, the merge maximizing $L_*^{\div}$ corresponds to the point $p_{\max}$ that maximizes $y(p)/x(p)$. Point $p_{\max}$ must lie on the convex hull of P, and can be found by a *tangent query* through the origin $\overline{O}$. Such a query reports the tangents on the hull that pass through a given point. Initially $\overline{O}$ is set to $(0, 0)$, but each merge shifts this imaginary origin, which serves as the vantage point of the tangent queries. Figure 7 illustrates these observations. We thus need a data structure that maintains the convex hull of a fully dynamic point set P and that allows for quick tangent queries.

In fact [8] present such a data structure, using so-called *kinetic heaps*. It uses linear space (i.e., $O(n^2)$, in our case), handles both insertions into and deletions from P, as well as tangent queries to the convex hull in amortized time $O(\log n)$. This data structure is described more extensively in the dissertation of [19], where, among other things, it is proven that the amortized performance of this data structure is in fact optimal. Since detailing this data structure is far beyond the scope of this paper, we just give a rough idea and use it as a black box. The points are stored in several instances of a semi-dynamic data structure that supports deletions. Insertions result in new instances, which are merged with *rank degree* $\log n$ by a semi-dynamic data structure that supports constant time deletions. Then, the core data structure is built which maintains the *convex hull* of two such merged sets. On top of that data structure, a *kinetic heap* is then built, which finally handles queries and operations. A *kinetic* (or *parametric*) *heap* is a generalization of a *priority queue*, such that the entries are linear functions that change over time. The authors use *interval trees* as secondary structures for answering containment queries.

Given this data structure, Algorithm 3 performs the update in Line 6 of Algorithm 2 in time $O(n \log n)$. First, a tangent query from $\overline{O}$ to the convex hull of P finds $p_{\max}$ (Line 1) in time $O(\log n)$. Then, after storing the merge of C_l and C_k (Line 2) in at most linear time, a linear number of points $p_{i,k}$ and $p_{i,l}$ are replaced by a new point $p_{i,(kl)}$ (Line 3). After each such replacement the

Algorithm 3: Quick Divisive Merge

Input: $\Delta\mathcal{M}, \Delta\mathbb{E}[\mathcal{M}]$, data structure with of points P as described above, reference R, (shifted) origin $\overline{O}$
Output: Best merge, updated matrices $\Delta'\mathcal{M}, \Delta'\mathbb{E}[\mathcal{M}]$

1 Find $p_{\max} = p_{kl}$ with tangent query through R
2 Merge clusters C_k and C_l of point $p_{kl} = p_{\max}$
3 For all clusters C_i insert $p_{i,(kl)} := p_{i,k} + p_{i,l} - R$, delete $p_{i,k}$, $p_{i,l}$
4 $R \leftarrow p_{\max}$
5 $O \leftarrow O - (p_{kl} - R)$

convex hull of P is maintained in time $O(\log n)$ by the data structure. Finally, reference point R is set to the newly improved coordinates (Line 4), and origin $\overline{O}$ is shifted (Line 5) in constant time, saving the update of *all* $\Omega(n^2)$ points in P. Thus, we arrive at the following lemma, which generalizes to all implementations of lucidity where a merge of two clusters entails an addition of corresponding entries of ΔL (or of $\Delta\mathcal{M}_w$ and $\Delta\mathbb{E}[\mathcal{M}_w]$).

Lemma 8 *By employing* quick divisive merge *(Algorithm 3), Algorithm 2 runs in $O(n^2 \log n)$ time for the relative variants.*

5 Experimental Evaluation

The aim of this section is to experimentally evaluate the behavior of *lucidity* and of *lucidity*-based clustering algorithms in a systematic way. We proceed in two steps and start with the measure *lucidity* itself:

1. ***Lucidity* vs. Human Intuition.** The key idea of this part is to evaluate how well *lucidity* quantifies the human intuition of the quality of a graph clustering. In a first step we examine the behavior of *lucidity* on generated *ground-truth* clusterings. These generated clusterings are built by a basic random generator which features an unarguable and intuitive mechanism for tuning the clarity of the implanted clustering. We thereby check whether our implementations of *lucidity* yield results that are in accordance with human intuition of "better" or "worse" clusterings.

   We then cluster the generated graphs with established clustering algorithms and repeat our measurements of *lucidity*. This second set of experiments is less controlled than that which uses the generator, but reduces the dependency of our findings on the generator's clustering.

2. **Quality of Greedy *Lucidity*.** The experiments described above serve to corroborate that *lucidity* may be used to quantify the goodness of a graph clustering. In this second setup, we then try to find out how well *lucidity*-driven algorithms, and the proposed greedy agglomerative algorithms in particular, work in practice. To this end, we use three established quality

indices and *lucidity* itself, and systematically measure the quality of the clusterings found by our *lucidity*-based clustering algorithms from Section 4. Here we again use our generator for clustered random networks with scalable clarity. We thereby compare our algorithms to three established ones which serve as benchmarks. The question we want to answer is: How well do *lucidity*-based clustering algorithms compete with other algorithms in terms of quality?

5.1 The Experimental Setup

We employ an adaption of the benchmark used in [6, 7]. For further details on this experimental setup we refer the reader to these references and restrict ourselves to a brief sketch at this point.

Starting with a fixed set $V = \{1, \ldots, n\}$ of nodes, a random partition generator $\mathcal{P}(n, s, \nu)$ partitions V into $(P_1, \ldots, P_k)$. For the distribution of $|P_i|$ we use $|P_i| \sim \mathcal{N}(s, \frac{s}{\nu})$, with $s = n/k$. This simple process constrains $|P_k|$ (and possibly even predecessors); this is dealt with by setting $|P_k| = n - \sum_{i<k} |P_i|$ iff this yields $||P_k| - s| < s/3$, otherwise the partition is rejected and a new one drawn. Given a partition, this is used as the clustering. Then, for all $e \in \binom{V}{2}$ edges are introduced inside and between clusters with probabilities p_{in} and p_{out}, respectively. Finally a random weight ω is assigned to each edge with $\omega \sim \mathcal{U}([0, p_{\text{in}}])$[5] or $\omega \sim \mathcal{U}([0, p_{\text{out}}])$, respectively. In case the resulting graph is disconnected, additional edges between random nodes of disconnected components are drawn. In our experiments we used $n = 100$ and $n = 1000$, and choose $k \sim \mathcal{U}([\log n, \sqrt{n}])$, $\nu = 4$. We roughly refer to combinations of p_{in} and p_{out} supporting *dense*, *sparse*, *strong* and *random* community structure by A_{dense}, A_{sparse}, A_{strong} and $A_{\text{rand.}}$, respectively, as sketched out in Figure 8.

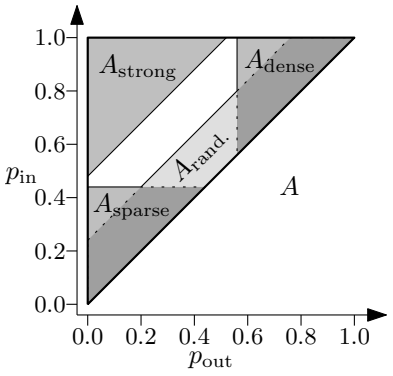


Figure 8: Combinations of p_{in}, p_{out} and their rough naming.

We then let the *lucidity* algorithms, based on Algorithm 2 and on the four variants (see Section 3.2) compete with reference algorithms on these instances. We restrict ourselves to *Markov Clustering* (MCL) [31], *Geometric MST Clustering* (GMC) [7] and *Iterative Conductance Cutting* (ICC)[6] [20] for comparison and to *lucidity*, *coverage*, *performance* and *inter-cc* (see [5]) for measuring clustering quality alongside structural aspects, such as the number of clusters. We keep the number of algorithms for compari-

[5] The uniform distribution over the interval $[a, b]$ is denoted as $\mathcal{U}([a, b])$

[6] *ICC* uses a threshold which determines when the cutting strategy of the algorithm should stop, we use 0.4; for *GMC* we use embedding dimension 2 and the geometric mean of *coverage*, *performance* and *inter-cc* as the objective function; for *MCL* we use *expansion* = 2 and *reduction* = 2; note that while for *ICC* the threshold directly influences the number of clusters, the other two algorithms automatically determine this number.

son limited as they only serve as a benchmark. Although there exist a number of alternative approaches that work towards the maximization of *modularity* (and could thus also be applied to *lucidity*), we refrain from including any of them in our study as it has been repeatedly shown (see Section 1 and the references therein) that all these largely similar approaches only marginally differ in both the structure of the identified clustering and the measured *modularity*.

We systematically conducted experiments using 100 and 1000 nodes, for all combinations of $p_{\text{in}} > p_{\text{out}}$ in steps of 0.05. We repeated each setup, until mean measured qualities were estimated to lie within a confidence interval of length $2 \cdot 0.05$ around the actual mean with an α-level (probability) of 0.95. We separately required this level of significance for each quality index measured. In total about one million total runs were conducted. Effective runtimes of our very basic Java 1.5 implementations ranged from a few milliseconds for 100 nodes using absolute variants to several seconds for 1000 nodes using relative variants, on an AMD Opteron 2.2 GHz. In addition to this systematic evaluation, we show exemplary results on two real-world networks in Section 5.3.

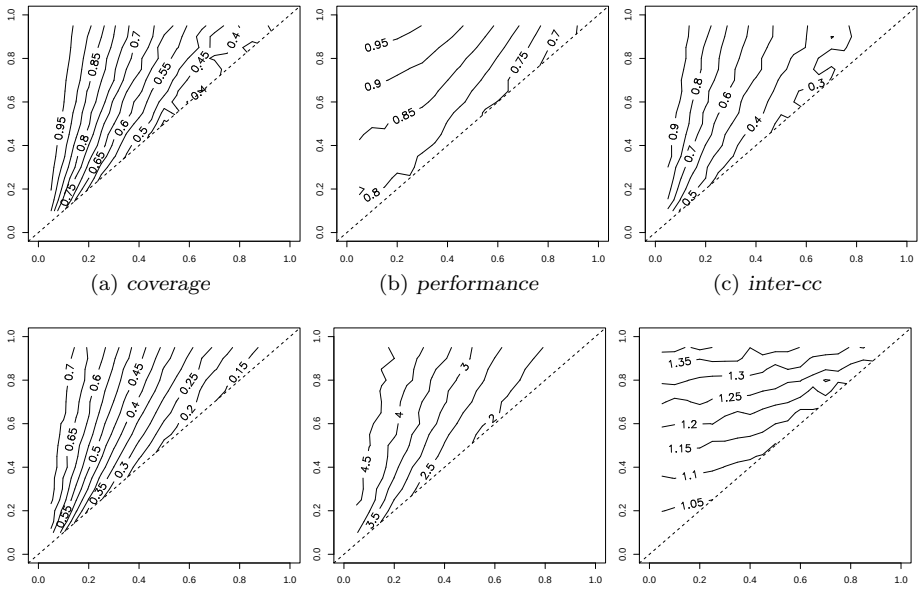
5.2 Computational Results

In this section we discuss the outcomes of our experiments. Since results for $n = 100$ largely agreed with those for $n = 1000$ we chose to focus on the latter (larger) setup here. Due to the equivalence of $\mathrm{L}^{-}_{\text{cov}}$ and $\mathrm{L}^{-}_{\text{perf}}$, we denoted results as L^{-}_{*}. The plots use *isolines* (or *contour lines*), which are curves where the evaluated function has a constant value as denoted by labels on the isolines in the figures. This is comparable to elevation contour lines on topographic maps, giving a good impression of the behavior of a function on two variables. We first conduct experiments that evaluate how well *lucidity* is in accordance with human intuition in terms of the quality of a given clustering, then we evaluate *lucidity*-based algorithms.

5.2.1 *Lucidity*-Scores on Generator and Benchmark Algorithms.

We can assume that the graph generation process described above yields clusterings whose qualities—according to human intuition—clearly scale with p_{in} and (inversely) with p_{out}. Roughly speaking, for our results, one would expect high values in A_{strong}, with some variety of descent towards $A_{\text{rand.}}$. Figure 9 shows the results. As postulated for a reasonable index, all indices clearly attain the highest values for A_{strong}. For most indices, the slope of the quality level decreases with higher p_{out}; since the number of inter-cluster pairs of nodes increases more quickly than the number of intra-cluster nodes in our generator.[7] The slopes for *performance* remain approx. constant, which is a favorable behavior, as it yields a better comparability of clustering qualities of different graphs. This behavior is due to the fact that both edges inside, and non-edges between clusters are considered (as compared to, e.g., *coverage*). By Figure 9f

[7] Roughly speaking, the ratio of intra- to inter-cluster edges is proportional to k. Thus, in our generator and in many real networks, the statement holds.



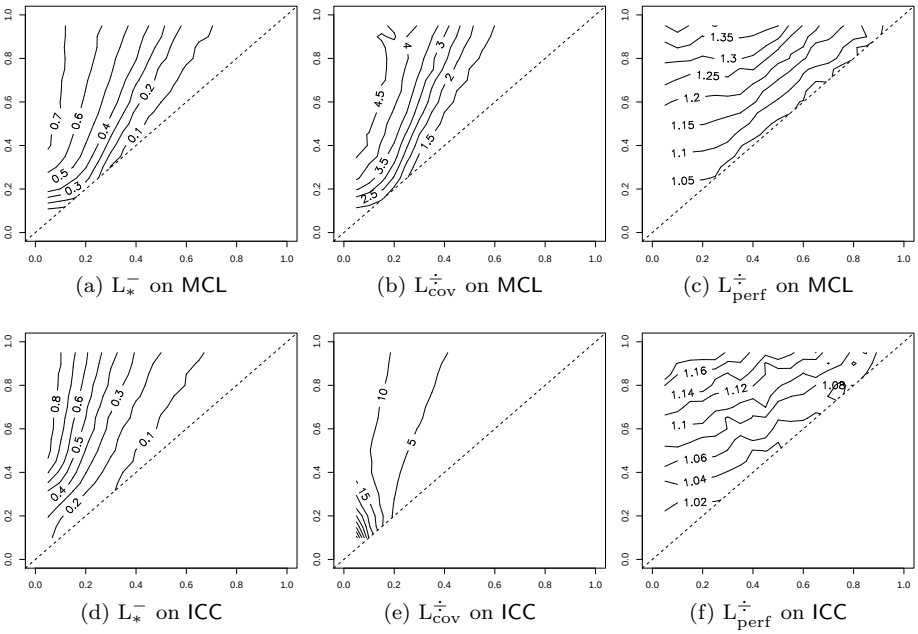
(a) *coverage* (b) *performance* (c) *inter-cc*

(d) L_{*}^{-} (e) $\mathrm{L}_{\mathrm{cov}}^{\div}$ (f) $\mathrm{L}_{\mathrm{perf}}^{\div}$

Figure 9: Plotted results for achieved quality on the underlying generator's clustering. The y-axis shows p_{in}, the x-axis shows p_{out}. The isolines indicate combinations of p_{in} and p_{out} where the same quality (value as label on isoline) has been measured.

$\mathrm{L}_{\mathrm{perf}}^{\div}$ adopts this behavior, a fact that is not obvious from the definition, but a property to keep in mind when using the index. Conversely, L_{*}^{-} does not exhibit this behavior, which is in parts explained by its strong dependence on *coverage* (remember from Section 3.2 that in $\mathrm{L}_{\mathrm{perf}}^{-}$, the terms referring to inter-cluster edges cancel out). As one difference between *coverage* and L_{*}^{-} note that the latter is more discriminative about $A_{\mathrm{rand.}}$, yielding values close to 0. *Inter-cc* yields high values for $A_{\mathrm{strong}}$and low values for $A_{\mathrm{rand.}}$, consistent with the intuition. Again, slopes decrease, but for a different reason: The index *inter-cc* is sensitive to a large cut induced by a single small cluster; since the ratio of inter- to intra-cluster edges for A_{sparse} is lower than for A_{dense}, *inter-cc* generally yields higher values for A_{sparse}. Summarizing, all three implementations of *lucidity* behave consistently in this test on the quality of a "ground truth" clustering with scalable clarity.

So far we know that our implementations of *lucidity* behave in a sound way on the generator's clustering. Figure 10 shows how they assess the results of the algorithms MCL and ICC (here we omit GMC for brevity), being less controlled experiments. For A_{strong} these algorithms probably identify a clustering which is very similar to the generator's. Comparing plots 10(a)-(c) to the corresponding ones in Figure 9 yields strong evidence for this. All three agree about MCL not performing very well for A_{sparse}, a fact *coverage*, *performance* and



(a) L_*^- on MCL (b) $\mathrm{L}_{\mathrm{cov}}^{\div}$ on MCL (c) $\mathrm{L}_{\mathrm{perf}}^{\div}$ on MCL

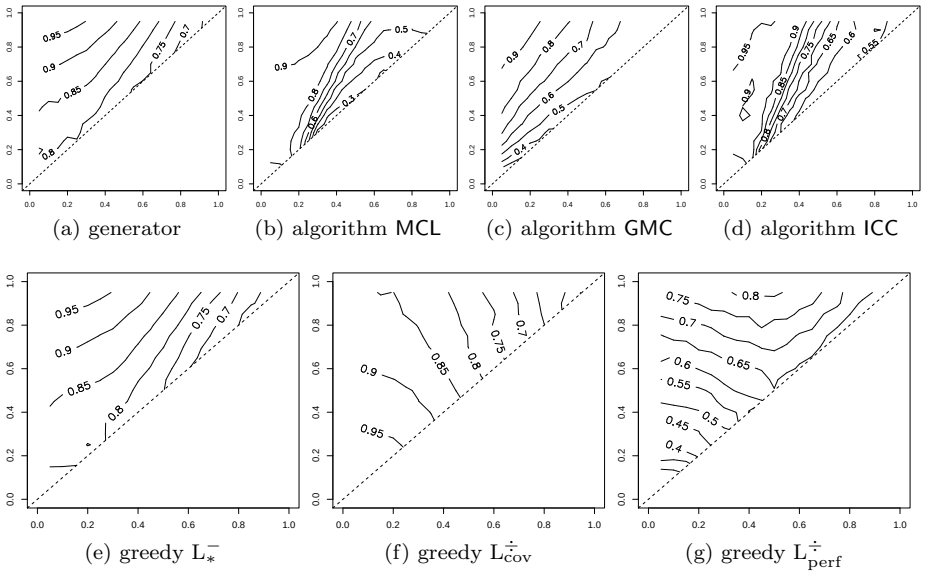
(d) L_*^- on ICC (e) $\mathrm{L}_{\mathrm{cov}}^{\div}$ on ICC (f) $\mathrm{L}_{\mathrm{perf}}^{\div}$ on ICC

Figure 10: Plotted results for achieved *lucidity* on the MCL's and ICC's clusterings (for a comparison to other measures on these clusterings review Figures 11-14)

inter-cc also agree on (see Figures 11b, 12b and 13b, respectively). The main reason for this is that MCL tends to identify a very fine clustering for A_{sparse} (see Figure 14b). Interestingly, L_*^- sees worse quality in MCL's clusterings for A_{dense}, as opposed by $\mathrm{L}_{\mathrm{perf}}^{\div}$.[8] The reason is MCL's rather coarse clustering for that region, something we will see $\mathrm{L}_{\mathrm{perf}}^{\div}$approve of repeatedly below. To briefly discuss the results on ICC note that L_*^-'s values largely agree with those on the generator. Exhibiting a rather exotiv behavior, $\mathrm{L}_{\mathrm{cov}}^{\div}$ seems to approve of the generally rather fine clustering of ICC; note how the number of clusters of ICC (see Figure 14d) correlates with $\mathrm{L}_{\mathrm{cov}}^{\div}$, especially for A_{sparse}. The general shape of the values of $\mathrm{L}_{\mathrm{perf}}^{\div}$ strongly resembles that for the generator, a result all other measures (except $\mathrm{L}_{\mathrm{cov}}^{\div}$) second. However, it does so at a lower absolute level; we shall see the reason for this in Figure 14g, where it becomes obvious that in terms of the number of clusters to be found, this measure disagrees with the behavior of ICC, i.e., that $\mathrm{L}_{\mathrm{perf}}^{\div}$ favors coarse clusterings.

Summary for Question 1. To summarize our findings, we can state that all three implementations of *lucidity* behave very reasonably on the controlled,

[8] Keeping crossreferences rigorous and multiply referring to other figures in almost each sentence massively obfuscates the text. We therefore refrain from most further references to plots in this section and hope that the reader manages to find the relevant ones.



(a) generator (b) algorithm MCL (c) algorithm GMC (d) algorithm ICC

(e) greedy L_{*}^{-} (f) greedy $\mathrm{L}_{\mathrm{cov}}^{\div}$ (g) greedy $\mathrm{L}_{\mathrm{perf}}^{\div}$

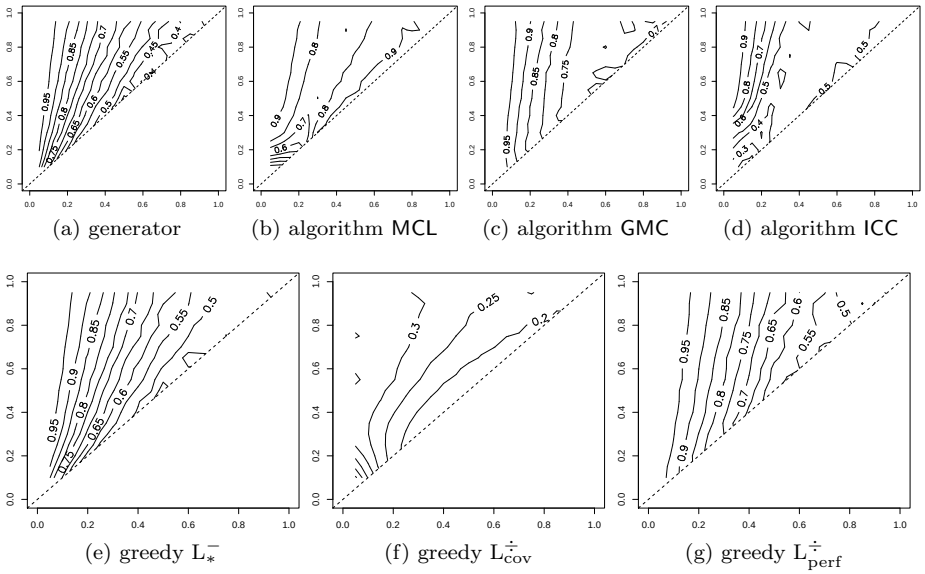
Figure 11: Plotted results for values of *performance* achieved by the generator, the benchmark clustering algorithms and the *lucidity*-based algorithms

pregenerated clusterings, with the small asset for $\mathrm{L}_{\mathrm{perf}}^{\div}$ which seems to react to p_{in} and p_{out} in a largely independent manner. Our experiments on the two benchmark algorithms MCL and ICC partially second these results, but already suggest that $\mathrm{L}_{\mathrm{perf}}^{\div}$ favors coarse clusterings in a mild manner, and that $\mathrm{L}_{\mathrm{cov}}^{\div}$ rather wildly favors fine clusterings. In turn, L_{*}^{-} appears not to depend too strong on this, but instead mildly disfavors both extremes.

5.2.2 *Lucidity*-Based Algorithms

In this section we measure the quality of clusterings identified with *lucidity*-based algorithms with three established indices and with *lucidity* itself. We thereby compare the results with those of three other algorithms which serve as benchmarks. Note that while a structural comparison with the generator's clustering is possible, it is not very meaningful, as this is not a "ground-truth" clustering in the traditional sense: we do not draw samples from an underlying distribution which is to be identified.

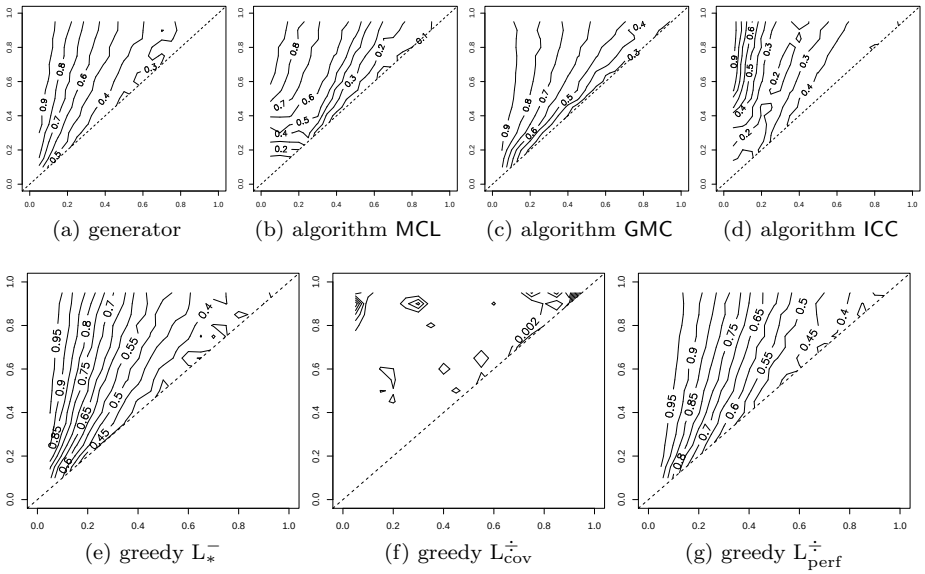
At a first glance, the statistical results for both relative variants ($\mathrm{L}_{\mathrm{cov}}^{\div}$ and $\mathrm{L}_{\mathrm{perf}}^{\div}$) and for L_{*}^{-} essentially differ for all three quality indices. Alongside the disagreement on the quality indices, $\mathrm{L}_{\mathrm{cov}}^{\div}$ tends to identify fine clusterings, i.e., 33 clusters on the average,while $\mathrm{L}_{\mathrm{perf}}^{\div}$ finds clusterings with a coarse granularity, i.e., 2.9 clusters on the average. The absolute variants exhibit a surprisingly similar behavior to the initial clustering with respect to all quality indices. The same holds for $\mathrm{L}_{\mathrm{perf}}^{\div}$ with respect to *coverage* and *inter-cluster conductance*,



(a) generator (b) algorithm MCL (c) algorithm GMC (d) algorithm ICC

(e) greedy L_*^- (f) greedy $L_{\text{cov}}^{\div}$ (g) greedy $L_{\text{perf}}^{\div}$

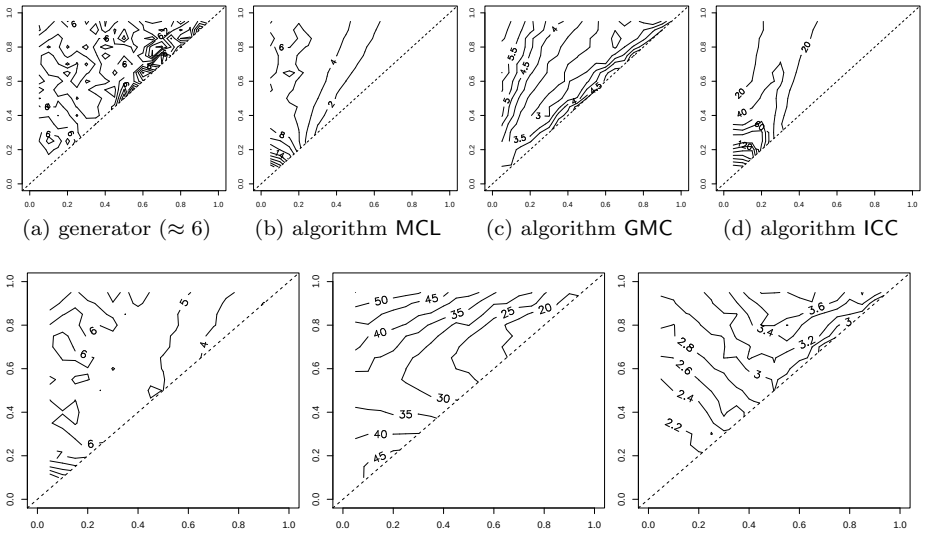
Figure 12: Plotted results for values of *coverage* achieved by the generator, the benchmark clustering algorithms and the *lucidity*-based algorithms



(a) generator (b) algorithm MCL (c) algorithm GMC (d) algorithm ICC

(e) greedy L_*^- (f) greedy $L_{\text{cov}}^{\div}$ (g) greedy $L_{\text{perf}}^{\div}$

Figure 13: Plotted results for values of *inter-cc* achieved by the generator, the benchmark clustering algorithms and the *lucidity*-based algorithms



(a) generator (≈ 6) (b) algorithm MCL (c) algorithm GMC (d) algorithm ICC

(e) greedy L^{-}_{*} (f) greedy $\mathrm{L}^{\div}_{\mathrm{cov}}$ (g) greedy $\mathrm{L}^{\div}_{\mathrm{perf}}$

Figure 14: Average number of clusters identified by the generator and the algorithms

however, the behavior is different for *performance*, but still acceptable scores are attained. In contrast, $\mathrm{L}^{\div}_{\mathrm{cov}}$ clearly fails to achieve high values of *coverage* and *inter-cluster conductance*, while its *performance* score is surprisingly good, a consequence of a very high number of clusters. The benchmark algorithms do not substantially surpass the initial clustering in general. Although the same holds for the *lucidity* algorithms, they shine for $A_{\mathrm{rand.}}$, finding higher quality clusterings than the generator (except $\mathrm{L}^{\div}_{\mathrm{perf}}$ for *coverage*).

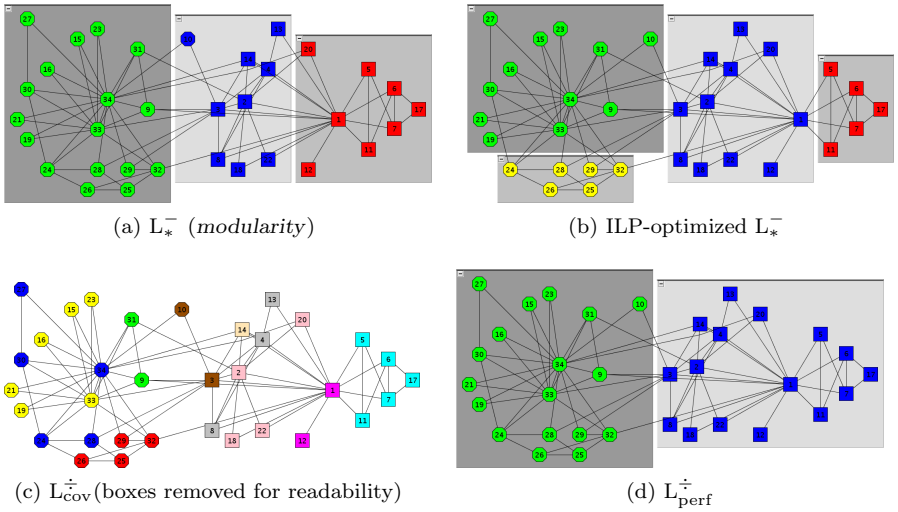
Summary for Question 2. In an overall assessment of the achieved clustering quality, the two absolute variants excel with respect to *performance* for almost all generated instances. This is particularly meaningful since both do not yield inappropriately high numbers of clusters, which would artificially increase *performance*. With respect to *coverage*, the absolute variants are only surpassed by the few algorithms that produced a substantially coarser clustering, among those $\mathrm{L}^{\div}_{\mathrm{perf}}$. An interesting observation is, that, using the *lucidity* measures as quality indices themselves, the greedy algorithms attain the maximum corresponding score for most testsets. However, in the case of A_{strong}, the obtained differences in the *lucidity* measures are small among most algorithms.

Explaining Some Artifacts. The high values of *performance*, attained by greedy $\mathrm{L}^{\div}_{\mathrm{cov}}$ for A_{sparse} are due its fine clusterings. These, in turn, can be explained as follows. Each step of the algorithm increases *coverage* and $\mathbb{E}[\textit{coverage}]$, which are both bounded by 1. These values increase faster, if an already large cluster is enlarged. Thus, the fraction tends to 1 for coarse clusterings, causing

the $\mathrm{L}^{\div}_{\mathrm{cov}}$-algorithm to terminate early, since *coverage* is monotonic in $|\mathcal{C}|$.

5.3 Real Data

Figure 15 shows how the variants of *lucidity* perform on the *karate club* network, studied initially in [34]. The network represents friendship between the 34 members of a university club that, due to an internal dispute, split up into two groups (circular nodes on the left and square-shaped nodes on the right). Clearly, *relative performance lucidity* ($\mathrm{L}^{\div}_{\mathrm{perf}}$) excels here, exactly reproducing the original division and thereby surpassing even *modularity* in precision.



(a) L^{-}_{*} (*modularity*) (b) ILP-optimized L^{-}_{*}

(c) $\mathrm{L}^{\div}_{\mathrm{cov}}$ (boxes removed for readability) (d) $\mathrm{L}^{\div}_{\mathrm{perf}}$

Figure 15: The upper right figure shows the clustering with optimum L^{-}_{*}, In all figures node shapes denote the grouping in reality. While both the greedy and the ILP optimization of L^{-}_{*} are meaningful and close to the real grouping, *relative performance lucidity* (Subfigure 15d) exactly reproduces the real grouping. The clustering $\mathrm{L}^{\div}_{\mathrm{cov}}$ identifies is not unreasonable, but too fine and insensitive for some applications.

Figure 16 shows a graph of the email contacts at our department over a period of three months (approx. 44300 emails). Nodes represent persons and weighted edges represent the number of email contacts between two coworkers. The grouping depicts the department's internal structure while the node colors (gray values) show the findings of the greedy algorithm based on L^{-}_{*}.

6 Conclusion

We formally stated the founding clustering paradigm of *modularity*, a recently introduced quality measure for graph clusterings. This paradigm of *lucidity* $L^{\odot}_{\mathcal{M}}(\mathcal{C})$ considers the trade-off between the achieved quality and the expected

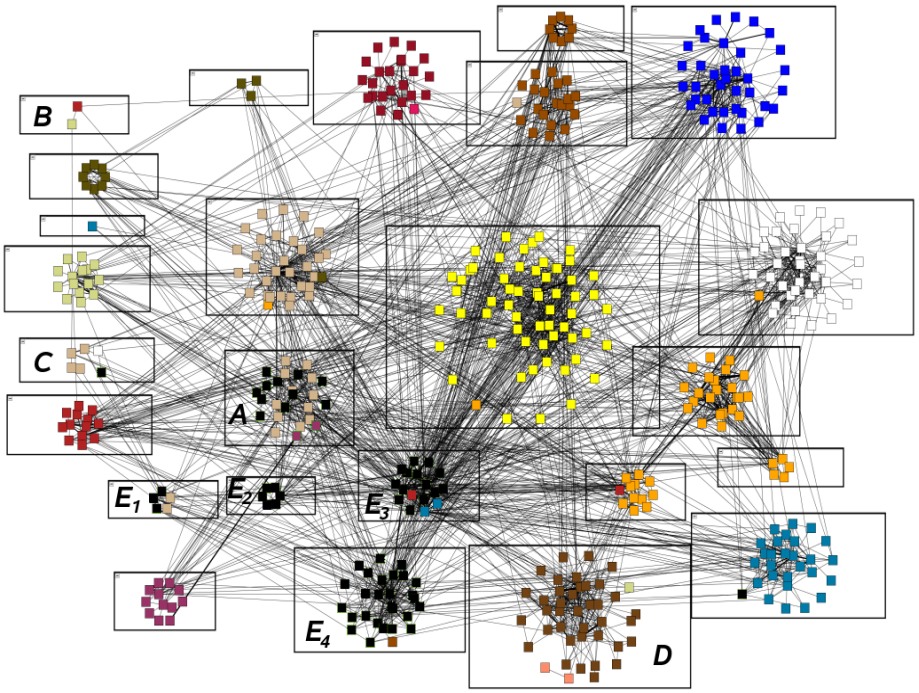


Figure 16: Inside reference clusters, L_*^- misclassifies only 6.8% of nodes, most of which are due to the highly ambiguous reference cluster A, which is split in half by the algorithm. The clustering of L_*^- yields a noticeably higher ($\approx 6\%$) *coverage*, which is partly due to 9 clusters each being merged into other clusters they are strongly connected with. In terms of *inter-cc* and all four realizations of lucidity, L_*^- slightly surpasses the reference. However, the *performance* of the reference clustering is 2.4% higher than for L_*^-. Note the artifact nodes in clusters B, C, D and the strong connections between clusters $A, E_1, \ldots, E_4$, which account for the aggregation done by the algorithm.

quality of a clustering with respect to networks with a similar intrinsic structure. We scrutinize four specific realizations of this paradigm—one being *modularity*—based on the well-known quality indices *performance* and *coverage*, and realizing the trade-off to the expectation by either *subtraction* or *division*. We state and discuss a probabilistic model for the assumptions that *modularity* and *lucidity* are founded upon. We derive and prove that using *performance* and *subtraction* yields a measure $\mathrm{L}_{\mathrm{perf}}^-$ of *lucidity* which is equivalent to *modularity*, which in turn constitutes the special case $\mathrm{L}_{\mathrm{cov}}^-$. Since *performance* is more robust than *coverage*, this observation strengthens *modularity*. The ILP formulation of the corresponding optimization problems leads us to NP-hardness of MinMixedMultiPartition. The performed experimental study is a systematic evaluation of the *lucidity* paradigm. Its evaluation yields that the paradigm is highly feasible, producing meaningful indices and clusterings with good quality. The generality of our approach is substantiated by the good results of using *per-*

formance and *division* on real networks. We additionally show its applicability by giving a general and efficient greedy algorithmic approach for a whole class of relative realizations, including the presented ones.

References

[1] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, and A. Marchetti-Spaccamela. *Complexity and Approximation - Combinatorial Optimization Problems and Their Approximability Properties*. Springer, 2nd edition, 2002.

[2] V. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[3] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Höfer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, February 2008.

[4] U. Brandes, D. Delling, M. Höfer, M. Gaertler, R. Görke, Z. Nikoloski, and D. Wagner. On Finding Graph Clusterings with Maximum Modularity. In A. Brandstädt, D. Kratsch, and H. Müller, editors, *Proceedings of the 33rd International Workshop on Graph-Theoretic Concepts in Computer Science (WG'07)*, volume 4769 of *Lecture Notes in Computer Science*, pages 121–132. Springer, October 2007.

[5] U. Brandes and T. Erlebach, editors. *Network Analysis: Methodological Foundations*, volume 3418 of *Lecture Notes in Computer Science*. Springer, February 2005.

[6] U. Brandes, M. Gaertler, and D. Wagner. Experiments on Graph Clustering Algorithms. In *Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03)*, volume 2832 of *Lecture Notes in Computer Science*, pages 568–579. Springer, 2003.

[7] U. Brandes, M. Gaertler, and D. Wagner. Engineering Graph Clustering: Models and Experimental Evaluation. *ACM Journal of Experimental Algorithmics*, 12(1.1):1–26, 2007.

[8] G. Brodal and R. Jacob. Dynamic Planar Convex Hull. In *Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'02)*, pages 617–626. IEEE Computer Society Press, 2002.

[9] C. Castellano and S. Fortunato. Community Structure in Graphs. To appear as chapter of Springer's Encyclopedia of Complexity and Systems Science; arXiv:0712.2716v1, 2008.

[10] F. R. K. Chung and L. Lu. Connected Components in Random Graphs Graphs with Given Expected Degree Sequences . *Annals of Combinatorics*, 6(2):125–145, 2002.

[11] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(066111), 2004.

[12] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The Complexity of Multiterminal Cuts. *SIAM Journal on Computing*, 23(4):864–894, 1994.

[13] L. Danon, A. Díaz-Guilera, J. Duch, and A. Arenas. Comparing Community Structure Identification. *Journal of Statistical Mechanics: Theory and Experiment*, 09(P09008):1–10, 2005.

[14] J. Duch and A. Arenas. Community Detection in Complex Networks using Extremal Optimization. *Physical Review E*, 72(027104):1–4, 2005.

[15] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Science of the United States of America*, 104(1):36–41, 2007.

[16] M. Gaertler, R. Görke, and D. Wagner. Significance-Driven Graph Clustering. In *Proceedings of the 3rd International Conference on Algorithmic Aspects in Information and Management (AAIM'07)*, Lecture Notes in Computer Science, pages 11–26. Springer, June 2007.

[17] M. R. Garey and D. S. Johnson. *Computers and Intractability. A Guide to the Theory of $\mathcal{NP}$-Completeness.* W. H. Freeman and Company, 1979.

[18] R. Guimerà and L. A. N. Amaral. Functional Cartography of Complex Metabolic Networks. *Nature*, 433:895–900, February 2005.

[19] R. Jacob. *Dynamic Planar Convex Hull.* PhD thesis, BRICS Research Centre, 2002.

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

[21] S. Muff, F. Rao, and A. Caflisch. Local Modularity Measure for Network Clusterizations. *Physical Review E*, 72(056107):1–4, 2005.

[22] S. Nagel. Optimisation of Clustering Algorithms for the Identification of Customer Profiles from Shopping Cart Data. Master's thesis, Universität Karlsruhe (TH), Fakultät für Informatik, October 2008. Diplomarbeit Informatik.

[23] M. E. J. Newman. Analysis of Weighted Networks. *Physical Review E*, 70(056131):1–9, 2004.

[24] M. E. J. Newman. Fast Algorithm for Detecting Community Structure in Networks. *Physical Review E*, 69:066133, 2004.

[25] M. E. J. Newman. Modularity and Community Structure in Networks. *Proceedings of the National Academy of Science of the United States of America*, 103(23):8577–8582, June 2006.

[26] M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(026113), 2004.

[27] M. E. J. Newman and J. Park. Origin of degree correlations in the Internet and other networks. *Physical Review E*, 68(2), 2003.

[28] V. Nicosia, G. Mangioni, V. Carchiolo, and M. Malgeri. Extending the Definition of Modularity to Directed Graphs with Overlapping Communities. *Journal of Statistical Mechanics: Theory and Experiment*, 2009(03):p03024 (23pp), 2009.

[29] A. Philippides, P. Fine, and E. D. Paolo. Spatially Constrained Networks and the Evolution of Modular Control Systems. In *Proc. 9th International Conference on Simulation of Adaptive Behavior*, Lecture Notes in Computer Science, pages 546–557. Springer, 2006.

[30] J. Reichardt and S. Bornholdt. Statistical Mechanics of Community Detection. *Physical Review E*, 74(016110):1–16, 2006.

[31] S. M. van Dongen. *Graph Clustering by Flow Simulation.* PhD thesis, University of Utrecht, 2000.

[32] F. Viger and M. Latapy. Efficient and Simple Generation of Random Simple Connected Graphs with Prescribed Degree Sequence. In *Proceedings of the 11th Annual International Conference on Computing Combinatorics (COCOON'05)*, Lecture Notes in Computer Science, pages 440–449. Springer, 2005.

[33] S. White and P. Smyth. A Spectral Clustering Approach to Finding Communities in Graphs. In *Proceedings of the fifth SIAM International Conference on Data Mining*, pages 274–285. SIAM, 2005.

[34] W. W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. *Journal of Anthropological Research*, 33:452–473, 1977.

[35] E. Ziv, M. Middendorf, and C. H. Wiggins. Information-Theoretic Approach to Network Modularity. *Physical Review E*, 71(046117):1–9, 2005.