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.
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 pre-clustered 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 $\omega: E \rightarrow [0, 1]$. 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 $C = \{C_1, \ldots, C_k\}$ 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 $V^\times = \{(u, v) \mid u \geq v, u, v \in V\}$. The set of all 2-tuples from $V$ is $V^2 = \{(u, v) \mid u, v \in V\}$, 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 $\omega((u, v)) = \omega(v, u)$.

We call $C$ a clustering of $G$ and the $C_i$ clusters. The cluster which contains node $v$ is denoted by $C(v)$. 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 $E(C) := \bigcup_{i=1}^{k} E(C_i)$ is the set of intra-cluster edges and $E \setminus E(C)$ the set of inter-cluster edges, with $|E(C)| = m(C)$ and $|E \setminus E(C)| = \overline{m}(C)$. 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 $m(C)^c$, and the number of non-adjacent inter-cluster pairs as $\overline{m}(C)^c$. We summarize the heavily used term $\sum_{v \in C} \deg(v) = \text{vol}(C)$.

We measured the quality of clusterings with a range of quality indices, discussed, e.g., in [3], however, we set our focus on the indices inter-cluster conductance [7], coverage [3] 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:

\[ \text{cov}(C) := \frac{m(C)}{m(C) + \overline{m}(C)} \]  
\[ \text{perf}(C) := \frac{m(C) + \overline{m}(C)^c}{\frac{1}{2}n(n-1)} \]  
\[ \text{icc}(C) := 1 - \max_{C \in \mathcal{C}} \frac{\sum_{v \in C} \deg(v) - 2|E(C)|}{\min\left(\sum_{v \in C} \deg(v), \sum_{v \in V \setminus C} \deg(v)\right)} \]  

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 \(\text{NP}\)-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 \(w(e)\) into account, which we indicate by a subscript as in, e.g., \(\text{cov}_\omega\). Thus, \(\omega(C) (\overline{\omega}(C))\) denotes the sum of the weights of all intra-cluster (inter-cluster) edges, \(W\) denotes the sum of all edge weights and \(\text{vol}_\omega(C) := \sum_{v \in C} \omega(v)\). The maximum edge weight in a graph is called \(\omega_{\text{max}}\). 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 \( M \) and a clustering \( C \), we define the *lucidity* \( L_M^\circ \) of a clustering \( C \) as the corresponding quality index as follows:

\[
L_M^\circ (C) := M(C) \odot E_{\Omega}[M(C)],
\]

where \( E_{\Omega}[M] \) is the expected value of the quality index \( M \) for the clustering \( C \) with respect to a suitable probability space \( \Omega \) 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]):

\[
\text{mod}(C) := \frac{m(C)}{m} - \frac{1}{4m^2} \sum_{C \in \mathcal{C}} \left( \sum_{v \in C} \deg(v) \right)^2,
\]

or equivalently:

\[
\text{mod}(C) = \sum_{\{u, v\} \in V^2} \left( \frac{A(u, v)}{m} \delta_{uv} \right) - \sum_{\{u, v\} \in V^2} \left( \frac{\deg(u) \cdot \deg(v)}{4m^2} \delta_{uv} \right),
\]

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

and \( A(u, v) = \text{number of (parallel) edges between } u \text{ and } v \).

using Kronecker’s symbol \( \delta_{uv} \) 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 \( (\Omega, \mathcal{F}, \mathbb{P}) \) 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 \( \frac{\deg(v) \deg(v)}{2m^2} \). 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., \( \frac{\deg(x)}{2m} \), see Figure 1. However, note that this model also assigns a probability of \( \frac{\deg(v)^2}{4m^2} \) to a self loop on \( v \). In this, the model differs from those used in [27, 10]. Thus we obtain

\[
p(e) = \begin{cases} 
\frac{\deg u \deg v}{(\deg v)^2} & \text{if } e = \{u, v\}, \, u \neq v \\
\frac{\deg v}{4m} & \text{if } e = \{v, v\}
\end{cases}
\]

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

\[
\sum_{\{u,v\} \in V \times V} p(e = \{u, v\}) = \sum_{v > w \in V} \frac{\deg v \deg w}{2m^2} + \sum_{v \in V} \frac{(\deg v)^2}{4m^2} = \sum_{v \in V} \frac{\deg v^2}{4m^2} \left(\sum_{v \in V} \deg v\right)^2 = \frac{1}{4m^2} (2m)^2 = 1
\]

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 \( (\Omega_E, p) \) for edge insertions uses as \( \Omega_E \) all unordered pairs \( \text{two-element multisets} \{u, v\} \in V^\times \), and \( p(\{u, v\}) \) is defined as above. Clearly, the probability function \( p \) is nonnegative, and the sample space \( \Omega_E \) is normed to 1 by Equation 10. A trial consisting of \( m \) edges being drawn independently as elementary events from \( \Omega_E \), by symmetry and using the above probabilities, yields an expected number of \( \frac{(\deg(u) \deg(v))}{2m} \) (parallel) edges between \( u \) and \( v \), for two nodes \( u \neq v \), and an expected number \( \frac{(\deg(v))^2}{4m} \) 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 \( \Omega_E \), we may build upon this setup and define the discrete probability
space $(\Omega, p)$ for graphs as follows. Let $\Omega$ consist of all $m$-element multisets of elementary events in $(\Omega_E, p)$, which is a subset of the set of all multisets over $\Omega_E$. We can now trivially identify the family of all graphs on $n$ (labeled) nodes and $m$ (unlabeled) edges with $\Omega$. 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 $(\Omega_E, p)$ (see Equation 7), in space $(\Omega, p)$, let

$$p(H) := \prod_{e \in E'} p(e) \cdot \frac{m!}{s_e!}$$  \hspace{1cm} (9)

be the probability of the event that the $m$ elementary events from $\Omega_E$ result in the multiset $E'$ and thus induce $H$. From the above construction of $(\Omega, p)$ a random process for graph creation is immediate: draw $m$ edges independently, each according to $(\Omega_E, p)$. 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 $(\tilde{\Omega}, p)$. Then we obtain $|V|^m = \tilde{m}^m$ different elementary events $\delta$ in $\tilde{\Omega}$ (some of which represent identical graphs, merely with edges added in a different order). Analogous to $(\Omega, p)$ (Equation 6), we may now define $p(\delta) = \prod_{e \in \delta} p(e)$ for all $\delta \in \tilde{\Omega}$, and get the following lemma:

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

**Proof:**

$$\sum_{\delta \in \tilde{\Omega}} \prod_{e \in \delta} p(e) = \sum_{E' \in (V^*)^m} \prod_{e \in E'} p(e) = \left( \sum_{e \in V^*} p(e) \right)^m = 1^m = 1 .$$  \hspace{1cm} (10)

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

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

\[ \text{modularity} \]

\[ 1 \text{ iff } s_e = 1 \text{ for all } e \in E', \]
Lemma 2  For any given graph $G$ and clustering $C(G)$, in $(\Omega, p)$ it holds that:  
$$E(\text{cov}(C)) = \sum_{(u,v) \in V^2} \frac{\deg(u) \deg(v)}{4m^2} \delta_{uv}. \text{ (As above } s_e \text{ denotes } e \text{'s multiplicity.)}$$

Proof:

$$E(\text{cov}(C)) = E\left(\sum_{e \in E(C)} s_e/m\right) = \frac{1}{m} E\left(\sum_{e \in E(C)} s_e\right) = \frac{1}{m} \sum_{e \in E(C)} E(s_e)$$

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

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 $(\Omega, p)$ is $\deg(v)$ (from $G$).

Proof:

$$E_{\Omega}(\deg(v)) = \sum_{u \in V, u \neq v} \frac{\deg(u) \deg(v)}{2m} \cdot 1 + \frac{(\deg(v))^2}{4m} \cdot 2$$

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

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 $C$. Figure 2b states the edge probabilities according to Equation 7 comprising $\tilde{m} = \binom{3}{2} + n = 6$ possible edges. We first consider only one random edge: This yields the family $\Omega_1 = H_1$ of the 6 graphs on 3 nodes and one edge, their probabilities match the corresponding edge probabilities in Figure 2b, which completes space $(\Omega_1, p)$. Due to the independence of edge drawings, we can now build the required probability space $(\Omega_2, p)$ inducing the family $H_2$ of graphs on 3 nodes and 2 edges by building the Cartesian product $\Omega_1 \times \Omega_1$. This yields $6^2$ outcomes in $\Omega_2$, whose probabilities are obtained by multiplying those of the participating members of $\Omega_1$. Of these outcomes $\tilde{m}$ occur once (two parallel edges) and $\binom{6}{2}$ occur twice (different insertion orders lead
to the same graph). Consider now the clustering $C$ of $G$ depicted in Figure 2A. Equation 3 yields $\text{mod}(C) = \frac{1}{2} - \frac{5^2 + 1^2}{2^2} = -\frac{1}{8}$, and in particular $E(\text{cov}) = \frac{5}{8}$. To see that this coincides with the expected coverage in $\Omega_2$ (i.e., $\mathcal{H}_2$) regarding $C$, 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 $\sum_{H \in \mathcal{H}_2} p(H) \text{cov}(C)_H = \frac{5}{8}$.

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 $p(G) = \frac{1}{8}$.

![Figure 4: All graphs in $\mathcal{H}_2$ with positive coverage for $C$, yielding $E(\text{cov}(C)) = \frac{5}{8}$. Note that graphs with non-parallel edges occur twice in $\Omega_2$, hence their double probability.](image)

**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:

$$\text{mod}_\omega(C) := \frac{\omega(C)}{W} - \frac{1}{4W^2} \sum_{C \in \mathcal{C}} \left( \sum_{v \in v} \omega(v) \right)^2$$

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

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

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

1. expected edge weights are as in Equation 12,
2. 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 $e_\ell$ and $e_r$ with expected weights $a_\ell$ and $a_r$, and actual edge weights $x_\ell$ and $x_r$, respectively. Let $c := a_\ell/(a_\ell + a_r)$ be the fraction that $e_\ell$ expects to get. We define a piecewise uniform density function $d(x)$ as depicted in Figure 5 as follows:

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

Having drawn $x$ from $d(x)$, the available weight $x_\ell + x_r$ is divided up such that $e_\ell$ 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 $C(G)$. Algorithm 1 yields a distribution of graphs with $\mathbb{E}(\text{cov}_\omega)$ 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 $\ell$ 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

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

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

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

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

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

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:

$$p_a(\{u, v\}) = \frac{\deg(u)}{2m} \cdot \frac{\deg(v)}{2m - \deg(u)} + \frac{\deg(v)}{2m} \cdot \frac{\deg(u)}{2m - \deg(v)}$$

```
= p_a(\{u, v\})
```

```
= p_a(\{v, u\})
```

```
= \frac{\deg(u) \cdot \deg(v)}{2m \cdot \deg(u) \cdot \deg(v)}
```

```
using $\deg(v) = 2m - \deg(v)$ (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.
\[
\sum_{\{u,v\} \in V, u \neq v} p_{uv} = \sum_{\{u,v\} \subseteq V, u \neq v} (p_{uv} + p_{vu}) = \sum_{v \in V} \sum_{u \in V, u \neq v} p_{uv} = \sum_{v \in V} \sum_{u \in V, u \neq v} \deg(v) \deg(u)
\]

\[
2m \cdot \deg(v) = 1
\]

Using arguments from the previous sections we can now setup a discrete probability space \((\Omega_\phi, p_\phi)\) 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

\[
E_\phi(\omega(u,v)) = \omega(u)\omega(v) \cdot (\overline{\omega}(u) + \overline{\omega}(v))/(2\overline{\omega}(u)\overline{\omega}(v)), \text{ using } \overline{\omega}(v) := 2W - \omega(v)
\]

(14). A variant modularity \(\phi\) for loop-free graphs could thus be defined as (compare to Formulas 6 and 11):

\[
\text{mod}_\phi(C) := \sum_{\{u,v\} \subseteq V, u \neq v} \frac{\omega(u,v)}{W} - \frac{\omega(u)\omega(v) \cdot (\overline{\omega}(u) + \overline{\omega}(v))}{2W \cdot \overline{\omega}(u)\overline{\omega}(v)}
\]

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: \(\omega_{\text{max}}\) of \(G\), 1 (being the maximum allowed edge weight), or \(W\). The canonic formulation is (compare Equation 2)
\[
\text{perf}_\omega = \frac{\omega(C) + M\overline{m}(C)^c + M\overline{m}(C) - \overline{\omega}(C)}{\frac{1}{2}n(n-1)M}.
\] (15)

Any choice for \(M\) which is a parameter dependent on \(G\) (such as \(\omega_{\text{max}}\)) becomes a random variable in \((\Omega, p)\). However, there is a more fundamental objection against using \(\omega_{\text{max}}\): in a fixed random model, there should be a fixed maximum weight to compare to. On the other hand, keeping the value of \(\omega_{\text{max}}\) 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 \(\omega\) 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

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

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

\[
E \left( \frac{\omega(C)}{\frac{1}{2}n(n-1)M} \right) = \frac{1}{2W} \sum_{C \in \mathcal{C}} (\text{vol}_\omega(C))^2 = \frac{1}{2W} \sum_{C \in \mathcal{C}} (\text{vol}_\omega(C))^2
\]

\[
E \left( \frac{M\overline{m}(C)^c + M\overline{m}(C) - \overline{\omega}(C)}{\frac{1}{2}n(n-1)M} \right) = \frac{1}{2} \sum_{C \in \mathcal{C}} \sum_{C' \subseteq C} \sum_{(w, v) \in C \times C'} \left( M - \frac{\omega(v)\omega(w)}{2W} \right)
\]

\[
\frac{1}{2}n(n-1)M
\]

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

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

\[
E(\text{perf}_\omega) = E_1 + E_2 = \frac{1}{2W} \sum_{C \in \mathcal{C}} \left( \sum_{v \in C} \omega(v) \right)^2 - 2W + M \left( n^2 - \sum_{C \in \mathcal{C}} |C|^2 \right)
\]

\[
\frac{1}{n(n-1)M}
\]

\[
(16)
\]
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 \( L^{-\text{cov}} \) has been described by [23]. Based on Table 1 we now define the following implementations:

\[
\begin{align*}
L_{\text{cov}}^{-} & := \text{cov} - \mathbb{E}[\text{cov}] \quad \text{(equals modularity)} \\
L_{\text{perf}}^{-} & := \text{perf} - \mathbb{E}[\text{perf}] \\
L_{\text{cov}}^{+} & := \frac{\text{cov}}{\mathbb{E}[\text{cov}]} \\
L_{\text{perf}}^{+} & := \frac{\text{perf}}{\mathbb{E}[\text{perf}]} \tag{18}
\end{align*}
\]

**Relative variants (divisive)**

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

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

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

Rewriting and summarizing \( L_{\text{perf}}^{-} \) yields the following term, which uses \( M \) only as a factor in the denominator, as an inverse scaling factor.

\[
\begin{align*}
L_{\text{perf}}^{-} &= \text{perf}_\omega - \mathbb{E}(\text{perf}_\omega) \\
&= \frac{\omega(C) + \Phi - \overline{\omega}(C)}{\frac{1}{2} \cdot n(n-1)M} - \frac{\sum_{C \in \mathcal{C}} (\text{vol}_\omega(C))^2/W + 2\Phi - 2W}{n(n-1)M} \\
&= \frac{\omega(C) - \overline{\omega}(C) - \frac{1}{2W} \sum_{C \in \mathcal{C}} (\text{vol}_\omega(C))^2 - W}{\frac{1}{2} n(n-1)M} \tag{20}
\end{align*}
\]

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 pro-
poses 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 $L_{\text{perf}}^{-}$ and $L_{\text{cov}}^{-}$

As is well known from [4, 3], $L_{\text{cov}}^{-}$ can be optimized via ILP formulation: Con-
straints 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:

$$\text{weighted } L_{\text{cov}}^{-} = \sum_{\{u,v\} \in V^2} \left( \frac{\omega(u,v)}{W} X_{uv} \right) - \sum_{\{u,v\} \in V^2} \left( \frac{\omega(u)\omega(v)}{4W^2} X_{uv} \right)$$

(21)

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 $L_{\text{perf}}^{-}$. We first build upon the formula
for $L_{\text{perf}}^{-}$ derived in Equation 20, and then rewrite it:

$$\text{weighted } L_{\text{perf}}^{-} = \omega(C) - \bar{\omega}(C) - \frac{1}{2W} \sum_{C \in \mathcal{C}} \left( \sum_{v \in C} \omega(v) \right)^2 - W$$

$$= \sum_{\{u,v\} \in V^2} \omega(u,v) X_{uv} - \sum_{\{u,v\} \in V^2} \omega(u,v) (1 - X_{uv}) - \sum_{\{u,v\} \in V^2} \frac{\omega(u)\omega(v)}{2W} X_{uv} - W$$

$$= \frac{2}{\frac{1}{2W} \sum_{\{u,v\} \in V^2} \omega(u,v) X_{uv} - \frac{1}{2W} \sum_{\{u,v\} \in V^2} \omega(u)\omega(v) X_{uv} - 2W}$$

$$= \frac{\sum_{\{u,v\} \in V^2} \omega(u,v) X_{uv} - \sum_{\{u,v\} \in V^2} \omega(u)\omega(v) X_{uv}}{\frac{1}{2W} n(n-1)M} - \frac{1}{\frac{1}{4W} n(n-1)M}$$

(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 Formu-
la 21. This yields the following lemma:

\text{ILP stands for integer linear program.}
Lemma 6 (Equivalence of $L_{\text{perf}}$ and $L_{\text{cov}}$) The problem of optimizing $L_{\text{perf}}$ and that of optimizing $L_{\text{cov}}$ are equivalent, furthermore

$$L_{\text{cov}}(G, C_1) > L_{\text{cov}}(G, C_2) \iff L_{\text{perf}}(G, C_1) > L_{\text{perf}}(G, C_2) \quad (23)$$

This lemma together with the NP-completeness of optimizing modularity \cite{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 $C(G)$ with $L_{\text{perf}}(C(G)) \geq L$.

The deduction of the equivalence in Lemma 6 implies that a linear relation between the values of $L_{\text{perf}}$ and $L_{\text{cov}}$ for a given instance $G$ and an arbitrary clustering $C(G)$ can be given in the form $L_{\text{perf}} = a(G) \cdot L_{\text{cov}} + b(G)$. 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 $L_{\text{perf}}$ and $L_{\text{cov}}$ 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 $L_{\text{perf}}$ or $L_{\text{cov}}$ 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 $L_{\text{cov}}$ (or $L_{\text{perf}}$) in, e.g., an ILP:

$$L_{\text{cov}} \approx \sum_{(u,v) \in \binom{V}{2}} \left( \left( \frac{\omega(u,v) - \omega(u)\omega(v)}{2W} \right) X_{uv} \right) \quad (24)$$

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 $K$ on $V$, where edge weights $g$ are equal to the (simplified) term in brackets in Equation 24. Given an unweighted instance of $L_{\text{cov}}$ (i.e., modularity), edge weights in $K$ 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 $K = (V, E)$, 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 $L_{\text{cov}}$ 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 MinimumMultiway-Cut, where in a positively weighted graph a set of terminals $T \subseteq V$ has to be separated. Note that MinMixed-MultiPartition is not, as it might seem, a straightforward generalization of the NP-hard problem Mixed-MinCut (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 MinMixed-MultiPartition 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.

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 $\Delta L$ with entries $\Delta L_{i,j}$ equaling $L(C_{i,j}) - L(C)$, where $C$ is the current clustering and $C_{i,j}$ is obtained from $C$ 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 $\Delta L$ be defined by matrices $\Delta M$ and $\Delta E[M]$, denoting the additive changes in $M$ and in $E[M]$, respectively. Then, when merging $C_i$ and $C_j$, entries $\Delta M_{pq}$ of unaffected clusters do not change, while entries in rows and columns $i$ and $j$ are updated as follows: $\Delta M_{k,(ij)} := \Delta M_{k,i} + \Delta M_{k,j}$, where $C_{(ij)} = C_i \cup C_j$ (and $\Delta E[M]$ is updated analogously). From Equations 17 and 18 it becomes clear that for the absolute variants this transfers to $\Delta L$, while for the relative variants all entries of $\Delta L$ change, even those of unaffected clusters.
Algorithm 2: Greedy Lucidity

Input: Graph $G = (V, E, \omega)$
Output: Clustering $C$ of $G$

1. $C \leftarrow$ Singletons, initialize $L$
2. Initialize matrix $\Delta L$ (with $\Delta L_{ij} \equiv$ change in $L$ when merging $C_i$ and $C_j$)
3. while $|C| > 1$
   4. Find $\{i, j\}$ with $\Delta L_{i,j} = \arg \max \Delta L_{ij}$
   5. Merge clusters $C_i$ and $C_j$
   6. Update $\Delta L$
   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 $\Delta L$ 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 $\Delta L$ as heaps. Merging two clusters (Step 5) and updating $L$ (Step 7) require at most linear time. Thus, updating $\Delta L$ 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., $|C| \in \omega(1)$, 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 $\Delta L$ can be reduced by avoiding explicit matrix updates. We give an algorithm that updates $\Delta L$ in $O(n \log n)$ amortized time using a geometric embedding. We store matrix $\Delta L$ 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 $p_{ij} := (\mathcal{M}(C_{i,j}), \mathbb{E}[\mathcal{M}(C_{i,j})]) = (\mathcal{M} + \Delta \mathcal{M}_{ij}, \mathbb{E}[\mathcal{M}] + \Delta \mathbb{E}[\mathcal{M}]_{ij})$. 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 $R = (\mathcal{M}(C), \mathbb{E}[\mathcal{M}(C)])$ that represents the current clustering.
Since \( \mathcal{M} \) and \( E[\mathcal{M}] \) 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 \( p_{i,k(l)} \rightarrow p_i,k + p_{i,l} - R \). 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.

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: \( \mathcal{O} \leftarrow \mathcal{O} - (p_{kl} - R) \). Second, at any time, the merge maximizing \( L^+ \) corresponds to the point \( p_{\text{max}} \) that maximizes \( y(p)/x(p) \). Point \( p_{\text{max}} \) must lie on the convex hull of \( P \), and can be found by a tangent query through the origin \( \mathcal{O} \). 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 \( \mathcal{O} \) to the convex hull of \( P \) finds \( p_{\text{max}} \) (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: $\Delta M, \Delta E[\mathcal{M}]$, data structure with of points $P$ as described above, reference $R$, (shifted) origin $\mathcal{O}$

Output: Best merge, updated matrices $\Delta' M, \Delta' E[\mathcal{M}]$

1. Find $p_{\text{max}} = p_{kl}$ with tangent query through $R$
2. Merge clusters $C_k$ and $C_l$ of point $p_{kl} = p_{\text{max}}$
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. $R \leftarrow p_{\text{max}}$
5. $O \leftarrow O - (p_{kl} - R)$

The 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 $\mathcal{O}$ 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 $\Delta L$ (or of $\Delta M_w$ and $\Delta E[\mathcal{M}_w]$).

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 $\text{lucidity}$ and of $\text{lucidity}$-based clustering algorithms in a systematic way. We proceed in two steps and start with the measure $\text{lucidity}$ itself:

1. **Lucidity vs. Human Intuition.** The key idea of this part is to evaluate how well $\text{lucidity}$ quantifies the human intuition of the quality of a graph clustering. In a first step we examine the behavior of $\text{lucidity}$ 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 $\text{lucidity}$ 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 $\text{lucidity}$. 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 $\text{lucidity}$ may be used to quantify the goodness of a graph clustering. In this second setup, we then try to find out how well $\text{lucidity}$-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 \( P(n, s, \nu) \) partitions \( V \) into \( (P_1, \ldots, P_k) \). For the distribution of \( |P_i| \) we use \( |P_i| \sim \mathcal{N}(s, s\nu) \), 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 \( e \in (V^2) \) edges are introduced inside and between clusters with probabilities \( p_{\text{in}} \) and \( p_{\text{out}} \), respectively. Finally a random weight \( \omega \) is assigned to each edge with \( \omega \sim U([0, p_{\text{in}}]) \) or \( \omega \sim U([0, p_{\text{out}}]) \), 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 \( k \sim U([\log n, \sqrt{n}]) \), \( \nu = 4 \).

We roughly refer to combinations of \( p_{\text{in}} \) and \( p_{\text{out}} \) supporting dense, sparse, strong and random community structure by \( A_{\text{dense}}, A_{\text{sparse}}, A_{\text{strong}} \) and \( A_{\text{rand}} \), respectively, as sketched out in Figure 8.

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) [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 comparison

---

5The uniform distribution over the interval \([a, b]\) is denoted as \( U([a, b]) \)
6ICC 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 I 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 $p_{in} > p_{out}$ 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 $\alpha$-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 $L_{cov}$ and $L_{perf}$, 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 $A_{\text{strong}}$, 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 $A_{\text{strong}}$. 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.
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.

$L_{perf}^+$ adopts this behavior, a fact that is not obvious from the definition, but a property to keep in mind when using the index. Conversely, $L_{perf}^-$ does not exhibit this behavior, which is in parts explained by its strong dependence on coverage (remember from Section 3.2 that in $L_{perf}^-$, the terms referring to inter-cluster edges cancel out). As one difference between coverage and $L_{perf}^-$ note that the latter is more discriminative about $A_{rand}$, yielding values close to 0. Inter-cc yields high values for $A_{strong}$ and low values for $A_{rand}$, 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
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 $A_{\text{dense}}$, as opposed by $L_{\text{perf}}^{-}\div$. The reason is MCL’s rather coarse clustering for that region, something we will see $L_{\text{perf}}^{-}\div$ approve of repeatedly below. To briefly discuss the results on ICC note that $L_{\text{cov}}^{-}$’s values largely agree with those on the generator. Exhibiting a rather exotic behavior, $L_{\text{cov}}^{+}$ seems to approve of the generally rather fine clustering of ICC; note how the number of clusters of ICC (see Figure 14d) correlates with $L_{\text{cov}}^{+}$, especially for $A_{\text{dense}}$. The general shape of the values of $L_{\text{perf}}^{+}$ strongly resembles that for the generator, a result all other measures (except $L_{\text{cov}}^{+}$) 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 $L_{\text{perf}}^{+}$ 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,

---

8Keeping 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.
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 $L_{\text{per}}^+$ which seems to react to $p_{\text{in}}$ and $p_{\text{out}}$ in a largely independent manner. Our experiments on the two benchmark algorithms MCL and ICC partially second these results, but already suggest that $L_{\text{per}}^+$ favors coarse clusterings in a mild manner, and that $L_{\text{cov}}^+$ rather wildly favors fine clusterings. In turn, $L_{\text{in}}^-$ 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 ($L_{\text{cov}}^+$ and $L_{\text{per}}^+$) and for $L_{\text{in}}^-$ essentially differ for all three quality indices. Alongside the disagreement on the quality indices, $L_{\text{cov}}^+$ tends to identify fine clusterings, i.e., 33 clusters on the average, while $L_{\text{per}}^+$ 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 $L_{\text{per}}^+$ with respect to coverage and inter-cluster conductance,
Figure 12: Plotted results for values of coverage achieved by the generator, the benchmark clustering algorithms and the lucidity-based algorithms.

Figure 13: Plotted results for values of inter-cc achieved by the generator, the benchmark clustering algorithms and the lucidity-based algorithms.
However, the behavior is different for performance, but still acceptable scores are attained. In contrast, \( L_{\text{cov}} \) 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 \( A_{\text{rand}} \), finding higher quality clusterings than the generator (except \( L_{\text{perf}} \) 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 \( L_{\text{perf}} \). 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 \( A_{\text{strong}} \), the obtained differences in the lucidity measures are small among most algorithms.

**Explaining Some Artifacts.** The high values of performance, attained by greedy \( L_{\text{cov}} \) for \( A_{\text{sparse}} \), are due its fine clusterings. These, in turn, can be explained as follows. Each step of the algorithm increases coverage and \( \mathbb{E}[\text{coverage}] \), 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 $L_{\text{cov}}^\perp$-algorithm to terminate early, since coverage is monotonic in $|C|$.

### 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 ($L_{\text{perf}}^\perp$) excels here, exactly reproducing the original division and thereby surpassing even modularity in precision.

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

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 $L_{\text{perf}}^\perp$.

### 6 Conclusion

We formally stated the founding clustering paradigm of modularity, a recently introduced quality measure for graph clusterings. This paradigm of lucidity $L_{\text{mod}}^\perp(M(C))$ 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 $L^-_{\text{perf}}$ of lucidity which is equivalent to modularity, which in turn constitutes the special case $L^-_{\text{cov}}$. 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 Min-MixedMultiPartition. 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 Algorithms, 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 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 De-
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 Com-
puter Science, pages 546–557. Springer, 2006.

[30] J. Reichardt and S. Bornholdt. Statistical Mechanics of Community De-
tection. 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 (CO-
COON’05), Lecture Notes in Computer Science, pages 440–449. Springer,
2005.

[33] S. White and P. Smyth. A Spectral Clustering Approach to Finding Com-
munities in Graphs. In Proceedings of the fifth SIAM International Con-
ference 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 Ap-
proach to Network Modularity. Physical Review E, 71(046117):1–9, 2005.