Generating graphs that approach a prescribed modularity

Trajanovski, S; Kuipers, FA; Martin Hernandez, J; Van Mieghem, PFA

DOI
10.1016/j.comcom.2012.10.004

Publication date
2013

Document Version
Accepted author manuscript

Published in
Computer Communications

Generating graphs that approach a prescribed modularity

S. Trajanovski*, F.A. Kuipers, J. Martín-Hernández, P. Van Mieghem

Delft University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science, P.O. Box 5031, 2600 GA Delft, The Netherlands

Abstract

Modularity is a quantitative measure for characterizing the existence of a community structure in a network. A network's modularity depends on the chosen partitioning of the network into communities, which makes finding the specific partition that leads to the maximum modularity a hard problem. In this paper, we prove that deciding whether a graph with a given number of links, number of communities, and modularity exists is NP-complete and subsequently propose a heuristic algorithm for generating graphs with a given modularity. Our graph generator allows constructing graphs with a given number of links and different topological properties. The generator can be used in the broad field of modeling and analyzing clustered social or organizational networks.

Keywords: Modularity, Graph generator, Modeling community structure

1. Introduction

Community structure is observed in many real-world networks, such as (online) social networks, where groups of friends of a certain person are often also friends of each other. For instance, one group of friends could originate from the school community, another from the sports community, and yet another group could be living in the same neighborhood.

Community detection or characterizing the level of community structure in a network is difficult. The *modularity* metric, initially proposed by Newman and Girvan [1] to detect network communities, has attracted significant attention, e.g. see [2, 3, 4]. The maximum modularity expresses *how clustered* the network is and gives *the resulting partitioning* into the corresponding clustered communities. Modularity has its limitations in detecting community structure, for instance communities smaller than a certain *resolution limit* may be undetectable [5], while larger sub-graphs may be partitioned even if they are random graphs [6]. Additionally, computing the maximum modularity of a given graph is an NP-complete problem, as was proved by Brandes *et al.* [2]. Nonetheless, has remained a popular metric for representing community structure and several heuristic algorithms for detecting maximum modularity [7, 4, 8] have been proposed.

Ever since the seminal work of Erdős and Rényi [9] on modeling and analyzing random graphs, various graph generators have been proposed. Graph generators are predominantly used to mimic existing networks, such that either a proper network abstraction can be analyzed or simply to test new algorithms and applications when the actual network is too big or not completely known. Popular graph generators include the:

- Erdős-Rényi random graph generator [9, 10] that generates networks with a binomial degree distribution and where links exist with a fixed probability p.
- Barabási-Albert power-law graph generator [11] and its variations [12, 13] that produce graphs with a power-law degree distribution. Power-law graphs are for instance used to reflect the Internet AS topology [14].
- Watts and Strogatz small-world graph generator [15], which was proposed to generate networks with high clustering coefficient and small diameter.

However, the proposed models produce graphs with low modularity, thus failing to match the strong community structure of social networks. To date, there does not exist any generator that produces graphs with a given number of communities and fixed modularity. This paper aims to fill this gap by proposing such a generator. Artificially generated graphs with a required modularity would offer the possibility to analyze community detection, information spreading, or robustness properties on an appropriate scale.

We study the problem of finding a graph G with a given modularity m, number L of links and number c of communities. As it is shown in the paper, the modularity m taken together with the number of communities c quantitatively shows community presence or absence. Our main contributions are:

(a) We prove that deciding whether a graph, with a modularity m, number L of links, and partitioning into c communities exists, is NP-complete.

(b) We analyze the influence of link rewiring strategies on the modularity of a graph.

(c) We propose a novel graph generator that produces graphs with a given number of communities and a modularity close to that of a given modularity.

*Corresponding author

Email addresses: S.Trajanovski@tudelft.nl (S. Trajanovski), F.A.Kuipers@tudelft.nl (F.A. Kuipers), J.MartinHernandez@tudelft.nl (J. Martín-Hernández), P.F.A.VanMieghem@tudelft.nl (P. Van Mieghem)

The paper is organized as follows. A short overview of the state-of-the-art on modularity, community detection and related graph generators is given in Section 2. The *complexity* of generating graphs with a given modularity is discussed in Section 3. Section 4 analyzes the effect of link rewiring on the modularity of a graph. Section 5 proposes a heuristic algorithm for generating network structures with a given modularity and number of communities. The properties of the generated graphs are discussed in Section 6. We conclude in Section 7.

2. Related Work

The modularity metric has been proposed by Newman and Girvan [1] as a global metric for quantifying community existence in networks. Subsequently, modularity has been explored as a metric for community detection in graphs and networks [16, 7, 8, 17, 18]. A thorough summary of the state-of-the-art in community detection in general and modularity in particular has been provided by Fortunato [19]. Brandes *et al.* [2] proved that finding the maximum modularity is an NP-complete problem. In addition, they proposed a *linear programming* (LP) technique for finding the maximum modularity. A similar LP-based approach for modularity maximization was proposed in [20]. In our previous work [21], we have determined a tight bound and the properties of the maximum modular graphs for a given number of links. An algorithm that seeks for the local maxima, based on a *greedy* technique has been given in [16]. Fast modularity based community detection algorithms on very large networks have been proposed in [17, 8, 22]. Some weaknesses in modularity optimization have also been determined, such as the incapability to detect communities smaller than a *resolution limit* [5] or the breaking up of large random sub-graphs into separate communities [6]. A spectral analysis of the modularity as well as correlation with other metrics, such as assortativity [23, 24], has been conducted in [25].

Orman *et al.* [26] have made a qualitative comparison of community detection algorithms and surveyed the models for generating graphs with community structure. The model presented by Girvan and Newman [27] generates a network consisting of a small number of Erdős-Rényi graphs [9] that are weakly connected. Few other models with a larger number of communities have been proposed that lead to more realistic (e.g., power-law) degree distributions [28, 29]. Finally, models that produce weighted and undirected graphs with community overlap have been proposed by Lancichinetti and Fortunato [30].

Unlike previous work, we first prove the NP-completeness of deciding whether a graph with a given modularity, number of links and number of communities exists. To the best of our knowledge, our generator is the first in producing graphs with a given modularity, number of links and number of communities. Moreover, our generator returns the number of links per community, leaving space for leveraging other structural properties per community, such as the degree distribution.

3. Complexity of modular graph generation

For a certain partitioning of a network G of N nodes into c communities, *modularity* has been defined by Newman and Girvan [1] as a function of the graph's adjacency matrix values a_{ij} and its node degrees d_i for $i, j = 1, 2, ..., N$ as

$$m = \frac{1}{2L} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(a_{ij} - \frac{d_i d_j}{2L} \right) 1_{\{i,j \in \text{ the same community}\}} \qquad (1)$$

where we follow the notation introduced in [31].

By considering the *cumulative degree* D_{C_i}, which is the sum of all the nodal degrees in community C_i; the total number L_{C_i} of links within C_i; and the number L_{inter} of links that connect nodes in different communities, the original form for the modularity (1) can be modified [25] into

$$m = 1 - \frac{1}{c} - \frac{L_{\text{inter}}}{L} - \frac{1}{2c} \sum_{j=1}^{c} \sum_{k=1}^{c} \left(\frac{D_{C_j} - D_{C_k}}{2L} \right)^2 \qquad (2)$$

We use the term *inter-community links* to refer to links that connect nodes in different communities and the term *intra-community links* for those links, where both end-points reside in the same community. For each community C_i ($i = 1, ..., c$), the number of inter-community links, where exactly one node is in C_i, is denoted as $L_{\text{out}}^{C_i}$ and the number of intra-community links within C_i as $L_{\text{in}}^{C_i}$. Because, from a degree perspective, all inter-community links in C_i are counted twice, we have

$$D_{C_i} = 2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}$$

Over all possible partitions of G, the partitioning that leads to highest modularity m is of general interest. Based on (2), an immediate conclusion is that maximum modularity is achieved by minimizing the number L_{inter} of links that connect nodes in different communities, while keeping the cumulative degrees of the communities as equal as possible.

In order to gain more control over modularity-based community structure (and its weaknesses as exposed in [5, 6]), we consider the modularity m and the number of communities c as joint indicators for the community existence in a graph. For a fixed number c of communities, a rough upper bound for the modularity is $(1 - \frac{1}{c})$. The modularity value should therefore be interpreted based on the number of communities. For instance, for $c = 2$, a modularity value $m = 0.48$ would constitute a "highly clustered" network, while the same value for $c = 5$ could be interpreted as "medium clustered." Theoretically, $m < 1$ and the asymptotic value of 1 is only achieved for an infinite number of fully isolated communities. However, we are interested in modularity maximization in *connected networks*.

We proceed to formalize the problem of graph construction with a given modularity. Using the fact that $\sum_{i=1}^{c} D_{C_i} = 2L$, we transform (2) into

$$\sum_{i=1}^{c} D_{C_i}^2 = \frac{4cL\,(L - L_{\text{inter}} - mL)}{\binom{c}{2} + 1} \qquad (3)$$

We consider two variants of the graph generation problem, namely one where L_{inter} is fixed, and the other in which it is not.

Problem 1. Find a graph G with a given total number L of links and corresponding partitioning into c communities, where the communities are connected by L_{inter} links, for which the modularity of the generated graph equals m, i.e.

$$\begin{cases} \sum_{i=1}^{c} D_{C_i}^2 = \frac{4cL(L-L_{\text{inter}}-mL)}{\binom{c}{2}+1} \\ D_{C_i} = 2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i} \\ \sum_{i=1}^{c} D_{C_i} = 2L \\ \sum_{i=1}^{c} L_{\text{out}}^{C_i} = 2L_{\text{inter}} \end{cases}$$

Problem 1 is equivalent to

Problem 1*. For given L, c, L_{inter} and m, find a non-negative integer vector $\vec{L}_C = \left\{L_{\text{in}}^{C_i}, L_{\text{out}}^{C_i}\right\}_{i=1,\ldots,c}$ of $2c$ elements in total, such that

$$\begin{cases} \sum_{i=1}^{c} \left(2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}\right)^2 = \frac{4cL(L-L_{\text{inter}}-mL)}{\binom{c}{2}+1} \\ \sum_{i=1}^{c} L_{\text{out}}^{C_i} = 2L_{\text{inter}} \\ \sum_{i=1}^{c} L_{\text{in}}^{C_i} = L - L_{\text{inter}} \end{cases}$$

Relaxing the requirement for $\vec{L}_C$ to be an integer valued vector results in a convex quadratically constrained program, which can be solved in polynomial time (i.e., $\sum_{i=1}^{c} \left(2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}\right)^2 = \vec{L}_C^T P \vec{L}_C$, with P a $2c \times 2c$ matrix consisting of the sub-matrix $\begin{bmatrix} 4 & 1 \\ 1 & 1 \end{bmatrix}$ along the diagonal and 0 for the other elements. Since P is positive semi-definite, the quadratic constraint is convex).

Problem 2. Find a graph G with a given number of links L, a corresponding partitioning into c communities, and a given modularity m, such that

$$\begin{cases} 4cLL_{\text{inter}} + \left(\binom{c}{2} + 1\right) \sum_{i=1}^{c} D_{C_i}^2 = 4cL^2(1-m) \\ D_{C_i} = 2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i} \\ \sum_{i=1}^{c} D_{C_i} = 2L \\ \sum_{i=1}^{c} L_{\text{out}}^{C_i} = 2L_{\text{inter}} \end{cases}$$

Problem 2 is equivalent to

Problem 2*. For given L, c, and m, find a non-negative integer vector $\vec{L}_C = \left\{L_{\text{in}}^{C_i}, L_{\text{out}}^{C_i}\right\}_{i=1\ldots c}$ of $2c$ elements in total, such that

$$\begin{cases} 2cL \sum_{i=1}^{c} L_{\text{out}}^{C_i} + \left(\binom{c}{2} + 1\right) \sum_{i=1}^{c} \left(2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}\right)^2 = 4cL^2(1-m) \\ \sum_{i=1}^{c} \left(2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}\right) = 2L \end{cases}$$

Problem 2* is the problem of main interest in this paper and in the remainder we refer to it as the Modular Graph Existence (MGE) problem. A solution to the MGE problem does not constitute a graph, but gives the number of links inside and between communities. Based on this information, various instantiations of graphs might be possible. We will now prove that the MGE problem is NP-complete, even for a fixed partitioning $c = 2$ into two communities. We start with the following Lemma 1.

Lemma 1. *For $x < \lfloor \sqrt{C} \rfloor$, $x^2 \equiv C \pmod B$ is equivalent to $x^2 + By = C$.*

Proof. Let us assume that x is a solution of $x^2 \equiv C \pmod B$, then the pair $(x, y = \frac{C-x^2}{B})$ is a solution of $x^2 + By = C$, since $x^2 = Bk + C$ for some $k \in \mathbb{N}$ and thus $x^2 + By = Bk + C + B\frac{C-Bk-C}{B} = C$. On the other hand, assuming that (x, y) is a solution of $x^2 + By = C$ and taking modulo B on both sides, using $(By) \bmod B = 0$, we arrive at $x^2 \equiv C \pmod B$, hence x is a solution. □

Lemma 1 shows that finding a solution to the quadratic Diophantine equation $x^2 + By = C$ is as hard as finding a solution to $x^2 \equiv C \pmod B$. This problem has been shown to be NP-complete by Manders and Adleman [32] even for few known factors of B, for instance with B an even number[1]. Hence, the quadratic Diophantine problem $x^2 + By = C$ is NP-complete.

Theorem 2. *The MGE problem, i.e. deciding whether a graph, with modularity m, number L of links, and a partitioning into $c = 2$ communities, exists, is NP-complete.*

Proof. Given $c = 2$ and L, a solution to the MGE problem returns two integer numbers, namely $L_{\text{in}}^{C_1}$ and $L_{\text{out}}^{C_1}$ (where $L_{\text{in}}^{C_2} = L - L_{\text{in}}^{C_1} - L_{\text{out}}^{C_1}$ and $L_{\text{out}}^{C_2} = L_{\text{out}}^{C_1}$). Based on (2), it can be verified in polynomial time whether those numbers indeed lead to a modularity m, and hence the problem is in the class NP[2]. To prove that the MGE problem is also NP-hard[3], we demonstrate how solving the modular graph existence problem would present a solution to the NP-complete quadratic Diophantine problem, which asks whether an $x \in \mathbb{N}$ exists for which $x^2 + By = C$ holds with $B, C \in \mathbb{N}$ and B even. We proceed in two steps. First we translate, in polynomial time, the quadratic Diophantine problem into an MGE problem and subsequently demonstrate how a solution to that MGE problem can be translated back, in polynomial time, to a solution of the quadratic Diophantine problem.

1. *Diophantine* to *MGE*. Let us assume that we are looking for a solution (x, y) to $x^2 + By = C$ with B even, where the implicit factor of 2 does not affect the hardness of the problem. This problem translates to deciding whether a graph G exists with $L = \frac{B}{2}$ links and with modularity $m = \frac{1}{2} - \frac{C}{2L^2}$. If indeed a solution (x, y) exists, then a solution to MGE also exists where community C_1 contains $\frac{L-y+x}{2}$ links and community C_2 contains $\frac{L-y-x}{2}$ links, and where both communities are connected via y links. Indeed, based on the expression in (2), such a solution has L links and a

[1] In the same paper [32], Manders and Adleman have also proved that finding a solution to the general quadratic Diophantine equation $Ax^2 + By = C$ is NP-complete.

[2] NP (non-deterministic polynomial time) refers to a class of problems whose solution correctness can be verified in polynomial time [33].

[3] NP-hard problems refer to a class of problems that are "at least as hard as the hardest problems in NP," and it is generally believed that they cannot be solved in polynomial time. NP-hard problems that themselves are in NP are called NP-complete [33].

modularity

$$
\begin{aligned}
m &= 1 - \frac{1}{2} - \frac{y}{L} - \frac{1}{8L^2}(2\frac{L-y+x}{2} - 2\frac{L-y-x}{2})^2 \\
&= \frac{1}{2} - \frac{y}{L} - \frac{1}{8L^2}4x^2 = \frac{1}{2} - \frac{x^2+2Ly}{2L^2} = \frac{1}{2} - \frac{x^2+By}{2L^2} \\
&= \frac{1}{2} - \frac{C}{2L^2}
\end{aligned}
$$

2. *MGE* to *Diophantine*. Let us assume that the constraints of the MGE problem are satisfied, namely

$$
\begin{cases}
4L(L_{\text{out}}^{C_1} + L_{\text{out}}^{C_2}) + 2\left(\left(2L_{\text{in}}^{C_1} + L_{\text{out}}^{C_1}\right)^2 + \left(2L_{\text{in}}^{C_2} + L_{\text{out}}^{C_2}\right)^2\right) \\
= 8L^2(1-m) \\
\left(2L_{\text{in}}^{C_1} + L_{\text{out}}^{C_1}\right) + \left(2L_{\text{in}}^{C_2} + L_{\text{out}}^{C_2}\right) = 2L
\end{cases}
$$

Going back to the notation of $D_{C_i} = 2L_{\text{in}}^{C_i} + L_{\text{out}}^{C_i}$, $i = 1, 2$, and setting $y = L_{\text{out}}^{C_1} = L_{\text{out}}^{C_2}$ we have

$$
\begin{cases}
4L(y+y) + 2(D_{C_1}^2 + D_{C_2}^2) = 8L^2(1-m) \\
D_{C_1} + D_{C_2} = 2L
\end{cases}
$$

With $D_{C_2} = 2L - D_{C_1}$, where we choose $D_{C_1} \geq D_{C_2}$, we obtain

$$
8Ly + 2(D_{C_1}^2 + (2L - D_{C_1})^2) = 8L^2(1-m)
$$

or

$$
(D_{C_1} - L)^2 + 2Ly = L^2 - 2mL^2
$$

From our initial Diophantine to MGE translation we have that $B = 2L$ and $C = L^2 - 2mL^2$, thus the solution to $x^2 + By = C$ is obtained from a solution to the corresponding MGE problem as $x = D_{C_1} - L$, and $y = L_{\text{out}}^{C_1}$, with C_1 the largest community.

□

In our proof, we have relied on quantifying the number of links in and between communities that would lead to a given modularity and we have not relied on a possible graph realization. Although the difference is subtle, since the Diophantine problem depends on numbers, our reliance on link numbers instead of real links in a graph is crucial. Numbers can be stored in binary representation and therefore only grow logarithmically in the size of the input, while real links in a graph cannot be represented in binary notation (and are often represented via an adjacency matrix).

Within a community C_i, several (sub)-graph structures can be devised that obey the required number $L_{\text{in}}^{C_i}$ of links in the solution vector $\vec{L_C} = \left\{L_{\text{in}}^{C_i}, L_{\text{out}}^{C_i}\right\}_{i=1,\ldots,c}$ to the MGE problem. The denser (in terms of the average degree $E[D]$) this community graph is, the better it actually reflects a community, and the less likely it becomes that another partitioning would result in a higher modularity.

4. Changing the modularity via link rewiring

We identify three link rewiring steps, referred to as transformations, to change a graph's modularity.

Transformation 1. The modularity m of a graph G (partitioned into communities C_i) increases by replacing an inter-community link between C_i and C_j with an intra-community link in C_i or C_j (in Figure 1).

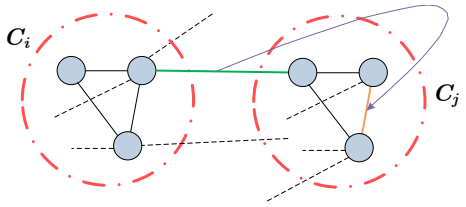


Figure 1: Replacing an *inter-community* link between C_i and C_j with an *intra-community* link in C_j (Transformation 1).

The difference Δm_1 in modularity between G and the resulting graph G' after having rewired is

$$
\Delta m_1(G, D_{C_i}, D_{C_j}) = \frac{2L + D_{C_j} - D_{C_i} - 1}{2L^2}
$$

The derivation of Δm_1 has been placed in the Appendix. Because the sum of all degrees equals twice the number of links, we have $D_{C_i} < 2L$ and $D_{C_j} \geq 1$. Therefore,

$$
\Delta m_1(G, D_{C_i}, D_{C_j}) > \frac{2L + 1 - 2L - 1}{2L^2} = 0
$$

The reverse operation, which decreases the modularity, is also possible: provided that we assure that a rewiring does not disconnect the graph.

Transformation 2. If there are two communities C_i and C_j, such that $D_{C_i} - D_{C_j} > 2$, then the modularity can be increased by moving an intra-community link from C_i to C_j (in Figure 2).

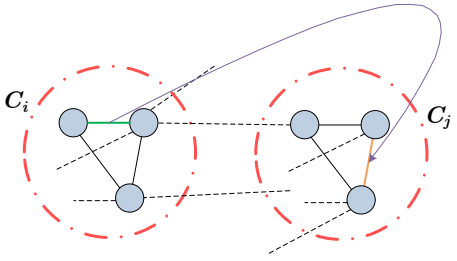


Figure 2: Replacing an *intra-community* link in C_i with an *intra-community* link in C_j (Transformation 2).

In this case, the number of inter-community links remains the same, while D_{C_j} is increased by 2 and D_{C_i} decreased by 2. The difference Δm_2 in modularity, as derived in the Appendix, after this transformation is

$$
\Delta m_2(G, D_{C_i}, D_{C_j}) = \frac{D_{C_i} - D_{C_j} - 2}{L^2} > 0
$$

Transformation 2 demonstrates that the modularity of G increases by making the cumulative degrees D_{C_i} of all the communities as close as possible.

Transformation 3. The modularity of a graph G increases by replacing an inter-community link between C_i and C_j with an intra-community link in a third community C_k, if $2L + D_{C_i} + D_{C_j} > 2D_{C_k} + 3$ (in Figure 3).

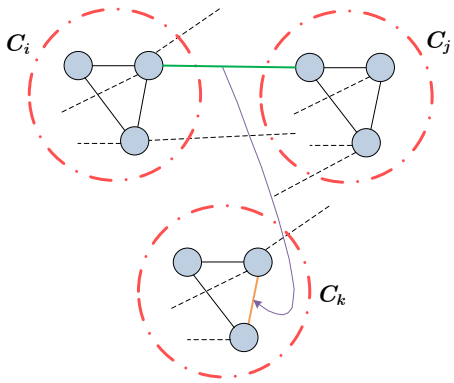


Figure 3: Replacing an *inter-community* link between C_i and C_j with an *intra-community* link in a third community C_k (Transformation 3).

As demonstrated in the Appendix, the difference between the modularity of G and the resulting graph G' is

$$\Delta m_3(G, D_{C_i}, D_{C_j}, D_{C_k}) = \frac{2L + D_{C_i} + D_{C_j} - 2D_{C_k} - 3}{2L^2} > 0$$

Transformation 3 is in fact obtained by consecutively applying Transformations 1 and 2.

In our proposed graph generator TMGG, explained in Section 5, we start with an initial graph and subsequently apply the transformations until we reach the desired modularity. We propose to start with the connected graph (determined in our previous work [21]) of L links and c communities that has maximum modularity

$$m_{\max} = 1 - \frac{1}{c} - \frac{c-1}{L} - \begin{cases} \frac{1}{2L^2}, & r = 0 \\ \frac{r(c-2r)}{2cL^2}, & 1 \le r \le \lfloor \frac{c}{2} \rfloor \\ \frac{(c-r)(2r-c)}{2cL^2}, & \lfloor \frac{c}{2} \rfloor < r \le c-1 \end{cases}$$

where $r = L \bmod c$.

5. Tunable modularity graph generator

Let us denote by *community graph* the abstraction where a node reflects one community and a link connects two nodes from different communities. In this section, we propose the **T**unable **M**odularity **G**raph **G**enerator (TMGG) algorithm that generates graphs with a given modularity m and number c of partitions. Our generator starts by generating a graph of maximum attainable modularity for a given m and c in INITIALIZE. The initial *community graph* is a tree with no more than 1 link between two communities. We subsequently use Transformations 1 and 2 (in REPLACEINTERNALEXTERNAL and SHIFTINTERNAL, respectively) to increase/decrease the modularity towards the desired modularity m.

Algorithm 1: INITIALIZE

input : Number L of links, number c of communities
output: Max modularity $m_{\max} = \max\{m(\mathsf{L}, \mathsf{c})\}$, initial *community graph* C, initial internal link sums $\{L_{\text{in}}^{C_i}\}_{i=1,\ldots,\mathsf{c}}$

```
1  r ← L mod c, k ← ⌊L/c⌋, m_max ← 1 − 1/c − (c−1)/L;
2  L_in^{C_1} ← k, i ← 2;
3  if r == 0 then
4      while i ≤ c do
5          C : create a link (i − 1, i)
6          L_in^{C_i} ← k − 1, i ← i + 1;
7      m_max ← m_max − 1/(2L^2)
8  else if r ≤ ⌊c/2⌋ then
9      while i ≤ c − r do
10         C : create a link (i − 1, i)
11         L_in^{C_i} ← k − 1;
12         if i ≤ r then
13             C : create a link (i, c − i + 1);
14             L_in^{C_{c−i+1}} ← k;
15         i ← i + 1;
16     L_in^{C_i} ← k, m_max ← m_max − r(c−2r)/(2cL^2)
17 else
18     while i ≤ r do
19         C : create a link (i − 1, i)
20         L_in^{C_i} ← k;
21         if i ≤ c − r then
22             C : create a link (i, c − i + 1);
23             L_in^{C_i} ← L_in^{C_i} − 1, L_in^{C_{c−i+1}} ← k;
24         i ← i + 1;
25     m_max ← m_max − (c−r)(2r−c)/(2cL^2);
```

We vary the order of using these transformations, resulting in three generator variants:

- STARTREPLACING
- STARTSHIFTING
- RANDOM

All generator variants use INITIALIZE to construct a community graph of maximum attainable modularity $m_{\max}$ for a given L and c. Variant STARTREPLACING (lines 6-11 in TMGG) starts by applying procedure REPLACEINTERNALEXTERNAL to the community graph to establish a modularity close to the interval $[m-\epsilon, m+\epsilon]$. If the obtained modularity fluctuates twice around the interval $[m - \epsilon, m + \epsilon]$ (explained in the next paragraph of this section), STARTREPLACING continues with the procedure SHIFTINTERNAL (lines 10-11 in TMGG). As soon as the range $[m - \epsilon, m + \epsilon]$ is met, the algorithm stops. Similarly, the variant

Procedure ReplaceInternalExternal (Transformation 1)

input : Number L of links, number c of communities, desired modularity m, the current modularity m_{cur}, the current modularity change Δm_{cur}, the current *state* $\in \{1, 2\}$, internal link sums $\{L_{\text{in}}^{C_i}\}_{i=1,\ldots,\text{c}}$

1 find i and j, such that $\Delta m_1(G, D_{C_i}, D_{C_j})$ is minimum;
2 **if** $m_{\text{cur}} > \text{m}$ **then** `// in state 1`
3 **if** *state* $== 2$ *and* $\Delta m_1(G, D_{C_i}, D_{C_j}) \geq \Delta m_{\text{cur}}$ **then return** *false*;
4 **if** $L_{in}^{C_j} == 0$ **then** break;
5 C: add 1 link between C_i and C_j;
6 $\Delta m_{\text{cur}} \leftarrow \Delta m_1(G, D_{C_i}, D_{C_j})$, $m_{\text{cur}} \leftarrow m_{\text{cur}} - \Delta m_{\text{cur}}$;
7 $L_{\text{in}}^{C_j} \leftarrow L_{\text{in}}^{C_j} - 1$, *state* $\leftarrow 1$;
8 **else** `// in state 2`
9 **if** *state* $== 1$ *and* $\Delta m_1(G, D_{C_i}, D_{C_j}) \geq \Delta m_{\text{cur}}$ **then return** *false*;
10 $\Delta m_{\text{cur}} \leftarrow \Delta m_1(G, D_{C_i}, D_{C_j})$, $m_{\text{cur}} \leftarrow m_{\text{cur}} + \Delta m_{\text{cur}}$;
11 **if** $\exists!$ *a link between* C_i *and* C_j **then** break;
12 C: remove 1 link between C_i and C_j **if** C is still connected; **otherwise** break;
13 $L_{\text{in}}^{C_j} \leftarrow L_{\text{in}}^{C_j} + 1$, *state* $\leftarrow 2$;
14 **return** *true*

Procedure ShiftInternal (Transformation 2)

input : Number L of links, number c of communities, desired modularity m, the current modularity m_{cur}, the current modularity change Δm_{cur}, the current *state* $\in \{1, 2\}$, internal link sums $\{L_{\text{in}}^{C_i}\}_{i=1,\ldots,\text{c}}$

1 find i and j, such that $\Delta m_2(G, D_{C_i}, D_{C_j})$ is minimum;
2 **if** $m_{\text{cur}} > \text{m}$ **then** `// in state 1`
3 **if** *state* $== 2$ *and* $\Delta m_2(G, D_{C_i}, D_{C_j}) \geq \Delta m_{\text{cur}}$ **then return** *false*;
4 $\Delta m_{\text{cur}} \leftarrow \Delta m_2(G, D_{C_i}, D_{C_j})$, $m_{\text{cur}} \leftarrow m_{\text{cur}} - \Delta m_{\text{cur}}$;
5 $L_{\text{in}}^{C_i} \leftarrow L_{\text{in}}^{C_i} + 1$, $L_{\text{in}}^{C_j} \leftarrow L_{\text{in}}^{C_j} - 1$, *state* $\leftarrow 1$;
6 **else** `// in state 2`
7 **if** *state* $== 1$ *and* $\Delta m_1(G, D_{C_i}, D_{C_j}) \geq \Delta m_{\text{cur}}$ **then return** *false*;
8 $\Delta m_{\text{cur}} \leftarrow \Delta m_2(G, D_{C_i}, D_{C_j})$, $m_{\text{cur}} \leftarrow m_{\text{cur}} + \Delta m_{\text{cur}}$;
9 $L_{\text{in}}^{C_i} \leftarrow L_{\text{in}}^{C_i} - 1$, $L_{\text{in}}^{C_j} \leftarrow L_{\text{in}}^{C_j} + 1$, *state* $\leftarrow 2$;
10 **return** *true*

StartShifting (lines 12-17 in TMGG) tries to obtain a modularity in the interval $[m - \epsilon, m + \epsilon]$, but with a reversed order of the procedures as in StartReplacing. First, the procedure ShiftInternal is preferred over ReplaceInternalExternal. Finally, the last variant Random (lines 18-23 in algorithm TMGG) randomly chooses one of the procedures ReplaceInternalExternal (with a certain probability p) and ShiftInternal (with probability $(1 - p)$) until the value in the interval $[m - \epsilon, m + \epsilon]$ is achieved.

For a very small value of ϵ, a modularity in $[m - \epsilon, m + \epsilon]$ may not be found. The termination condition effectuates when in consecutive (link rewiring) transformations the modularity value alternatively goes below and above the interval $[m-\epsilon, m+\epsilon]$ (lines 3 and 9 in ReplaceInternalExternal; lines 3 and 7 in ShiftInternal; and line 25 in TMGG), without getting closer to that interval. In the algorithm, this is reflected by the current modularity going from *state* 1 (above m) to 2 (below m) or *vice versa* twice in a row. Hence, TMGG either finds a modularity in the interval $[m - \epsilon, m + \epsilon]$ (as it "converges" towards the interval) or it terminates when no further improvements are observed in four consecutive transformations. All three variants StartReplacing, StartShifting and Random return the *community* graph, i.e., a family of graphs or the topology between communities and the number of links within each community. Based on the output, we are able to construct arbitrary graphs with a given number of links for each community. The topological differences of the resulting graphs are studied in Section 6.

5.1. Algorithm complexity and accuracy

The algorithm variants approach the given value m with different speed and accuracy. In the paper, we use the probability $p = 0.5$ in the variant Random, leading to an equal probability in choosing between ReplaceInternalExternal and ShiftInternal. For $p \approx 0$, Random would be closer to the StartReplacing variant, and for $p \approx 1$, Random would be closer to the StartShifting variant. Figure 4 presents the speed in terms of number of iteration steps, at which the three algorithm variants approach the requested modularity m. One iteration step corresponds to a single modularity change in the TMGG variants.

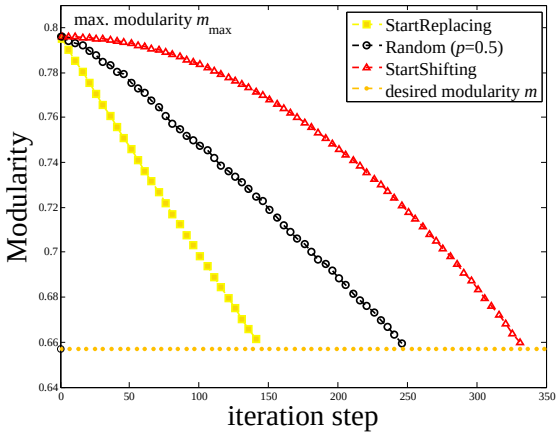


Figure 4: Approaching speed of algorithm variants, with $L = 1000$, $c = 5$, $m = 0.655$, $m_{\text{max}} = 0.796$ and $\epsilon = 5 \cdot 10^{-3}$. One iteration step corresponds to a single modularity change in the TMGG variants.

The variant StartReplacing reaches m in the smallest number of iterations, which is expected because its modularity change

Algorithm 2: TMGG

input : Number L of links, number c of communities, desired modularity m, variant algVariant, probability p
output: *community graph C*, internal link sums $\{L_{\text{in}}^{C_i}\}_{i=1,\ldots,\text{c}}$

```
[m_max, C, {L_in^{C_i}}_{i=1,...,c}] ← Initialize (L,c);
m_cur ← m_max;
if m_cur − ε > m then return There is no graph with modularity in [m − ε, m + ε];
Δm_cur ← +∞, state ← 0, approachM ← true;
switch algVariant do
    case StartReplacing // try 1st Transformation 1 then 2
        while |m_cur − m| > ε and approachM == true do
            approachM ← ReplaceInternalExternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
        approachM ← true;
        while |m_cur − m| > ε and approachM == true do
            approachM ← ShiftInternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
    case StartShifting // try 1st Transformation 2 then 1
        while |m_cur − m| > ε and approachM == true do
            approachM ← ShiftInternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
        approachM ← true;
        while |m_cur − m| > ε and approachM == true do
            approachM ← ReplaceInternalExternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
    case Random // choose randomly Transformation 1 or 2
        while |m_cur − m| > ε and approachM == true do
            choose randomly 1) with probability p OR 2) with probability (1 − p):
            1) approachM ← ReplaceInternalExternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
            2) approachM ← ShiftInternal (L,c,m,m_cur,Δm_cur,state,{L_in^{C_i}}_{i=1,...,c});
            if the procedure has changed then state ← 0; approachM ← true;
    otherwise break;
if approachM == false then return There is no graph with modularity in [m − ε, m + ε];
```

$\Delta m_1 = O(1/L)$ is bigger than the modularity change $\Delta m_2 = O(1/L^2)$ in StartShifting. Regarding the time complexity, all three variants start with Initialize, which "costs" $O(c)$. If we denote by m_{start} the initial modularity obtained after Initialize, we obtain the time complexity of StartReplacing as

$$O(\text{StartReplacing}) = \frac{m_{\text{start}} - m - \epsilon}{O(1/2L)} = O((m_{\text{start}} - m - \epsilon)L)$$

Similarly, the time complexity of StartShifting is

$$O(\text{StartShifting}) = \frac{m_{\text{start}} - m - \epsilon}{O(1/2L^2)} = O((m_{\text{start}} - m - \epsilon)L^2)$$

Moreover, because $\Delta m_2 < \Delta m_1$, we have a better accuracy in StartShifting. The variant Random is in between StartShifting and StartReplacing, in terms of the approaching speed, the time-complexity and the accuracy. The modularity of the produced graph, if one is returned, differs from the desired modularity m by at most $\pm\epsilon$ in all three variants. The smaller ϵ, the higher the accuracy. Figure 4 illustrates that both StartReplacing and Random variants attain the modularity m linearly, as opposed to a "non-linear" ($\Delta m_2 = O(1/L^2)$) decrease for the variant StartShifting.

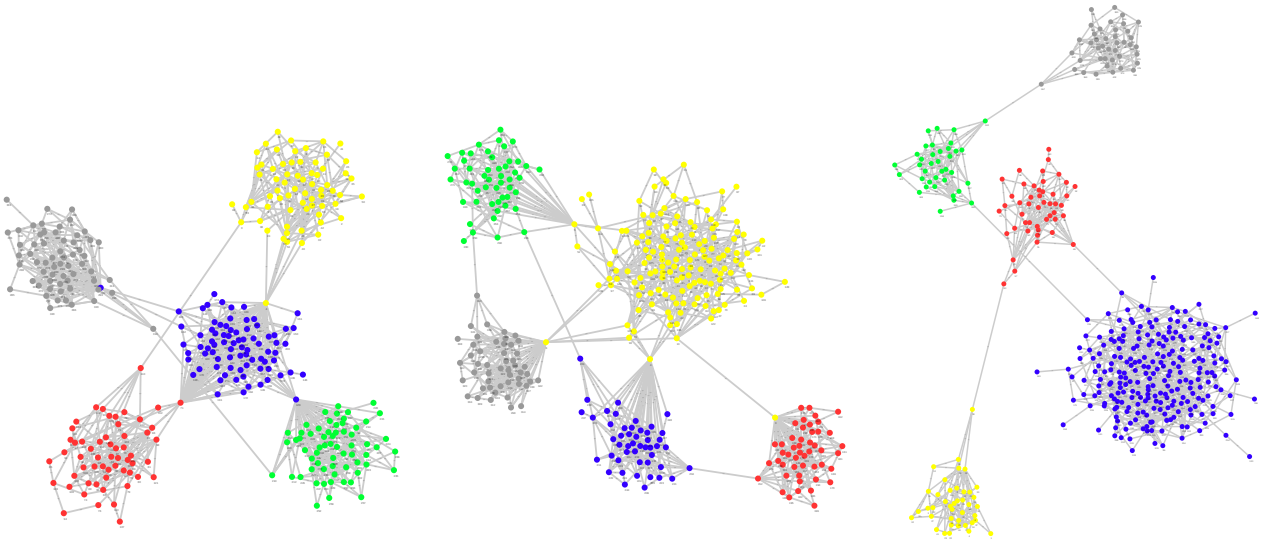
6. Properties of the obtained graphs

The three algorithm variants generate *community graphs* with different topological properties.

6.1. Topological properties

The variant StartShifting ends up with a community graph, with a very small number of inter-community links. In most of the cases, the community graph is a tree or very close to a tree. On the other hand, there are just a few (usually only one) communities with a very high number of links and all the other communities have a similar number of links. Unlike StartShifting, the StartReplacing variant generates graphs with higher number of inter-community links, but all the communities have a similar number of intra-community links (communities with similar size). These properties are exhibited in Figure 5. When comparing the number of inter-community links, the variant Random ($p = 0.5$) is somewhere in between StartShifting and StartReplacing.

Table **??** shows the difference in topological metrics for the three graphs produced by the three variants for given values of L, c, m and ϵ. The variant Random ($p = 0.5$) has topological metrics' values that lie in between the corresponding val-



(a) StartReplacing (b) Random ($p = 0.5$) (c) StartShifting

Figure 5: Graphs returned by the three algorithm variants ($L = 1000$, $c = 5$, $m = 0.655$ and $\epsilon = 5 \cdot 10^{-3}$).

Table 1: Topological metrics of the three returned graphs ($L = 1000$, $c = 5$, $m = 0.655$ and $\epsilon = 5 \cdot 10^{-3}$).

Algorithm variant	$E[D]$	C	$E[H]$	ρ_D	μ_{N-1}	λ_1	K
StartReplacing	5.88	0.355	3.70	-0.06	0.041	9.87	84%
Random ($p = 0.5$)	5.67	0.167	4.00	-0.04	0.036	8.84	86%
StartShifting	4.93	0.151	5.26	-0.01	≈ 0	6.52	95%

ues for StartReplacing and StartShifting. In general, the variant StartReplacing (StartShifting) produces graphs with the highest (lowest) average degree $E[D]$; the highest (lowest) average clustering coefficient C; the lowest (highest) average hop-count $E[H]$; the highest (lowest) algebraic connectivity μ_{N-1}; the highest (lowest) spectral radius λ_1; and the smallest (largest) assortativity ρ_D.

We define the *modularity quality coefficient* $K = \frac{m}{m_{\max}}$ as a ratio between the desired modularity m and the maximum modularity $m_{\max}$ of the obtained graph (using Newman's algorithm [16], because as stated before, finding the $m_{\max}$ is also an NP-complete problem [2]). Because $m_{\max}$ is the maximum of a given graph with an unknown number c of communities, we have $K \in [0, 1]$. The higher K, the more likely the original number c of communities is preserved. Table **??** (the last column) shows that the StartShifting variant has produced the graph with the largest K due to the small number of inter-community links and "higher link density" within the communities, followed by Random ($p = 0.5$) and StartReplacing.

In Figure 6, we display the relation between the average clustering coefficient and the desired modularity. The average clustering coefficient reflects to what extent nodes tend to cluster together and depends on the number of triangles in a graph. Figure 6 shows a linear relation between the modularity and the average clustering coefficient, where StartReplacing produces the graphs with highest average clustering coefficient. The

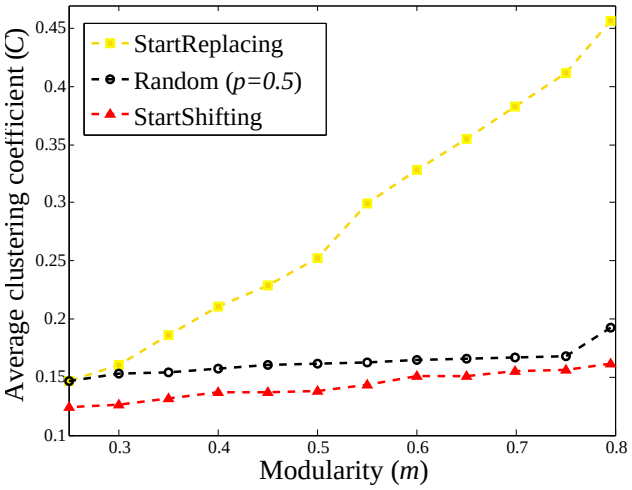
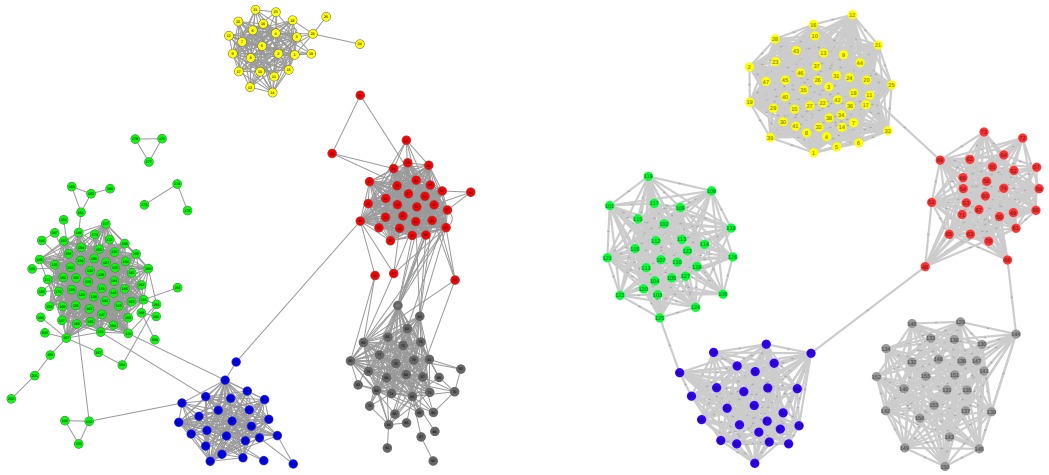


Figure 6: Clustering coefficient C as a function of the desired modularity value m for the algorithm variants with $L = 1000$, $c = 5$ and $\epsilon = 5 \cdot 10^{-3}$. Internally, the communities are constructed as random graphs.

StartReplacing produced graphs have many inter-community links, which means there is a higher probability of also having triangles spanning different communities than with StartShifting, which generates few inter-community links.



(a) User-centric friendship network of the person X in Facebook. (b) TMGG modeled network.

Figure 7: Real Facebook friendship and TMGG constructed networks.

6.2. *Online social network modeling*

To demonstrate that TMGG can indeed generate realistic community-structured networks we will make a comparison with a real *user-centric friendship network* of a single person X in Facebook[4], as displayed in Figure 7a. The nodes are Facebook friends of X and a link exists between two nodes if the corresponding two friends of X are also friends of each other. The visualization shows a clear community structure. Using TMGG (variant StartShifting), we have generated a network, in Figure 7b, that has the same modularity ($m = 0.7$), number of communities ($c = 5$) and number of links ($L = 1773$) as the Facebook network of X. The two networks have similar properties, such as similar average nodal degree ($E[D] = 20$) and clustering coefficient ($C = 0.68$), which supports our claim that TMGG can generate realistic networks.

7. Conclusions

We have considered the problem of constructing graphs with a given modularity and have proved that deciding whether such a graph exists is NP-complete. Subsequently, we have proposed a heuristic algorithm TMGG that generates graphs with a given modularity and number of links. TMGG has three variants and all start from a graph with maximum modularity [21] that is altered via rewiring. Furthermore, we have analyzed the difference in speed and accuracy of the three variations, and we have studied the topological properties of the graphs generated by them. All three TMGG variants produce community graphs, i.e. a family of graphs consisting of the topology between communities and the number of links within each community. The community graph presents ample flexibility to generate and fine-tune the final graph towards other desired topological properties, such as nodal degree distribution, without affecting the modularity.

Acknowledgments

We are grateful to Norbert Blenn for the useful discussions. This research has been supported by the EU FP7 Network of Excellence in Internet Science EINS (project no. 288021) and by the GigaPort3 project led by SURFnet.

Appendix A. The derivations for the modularity changes

We consider the difference Δm in modularity between the graph G and the graph G', obtained from G after a change in communities C_i and C_j. Using the modularity definition (2), the difference is reflected in $\left(D_{C_p} - D_{C_k}\right)^2 - \left(D'_{C_p} - D'_{C_k}\right)^2 \neq 0$, with $p \in \{i, j\}$. Hence, Δm boils down to

$$
\begin{aligned}
\Delta m &= \frac{L_{\text{inter}} - L'_{\text{inter}}}{L} - \frac{1}{8cL^2} \sum_{p=1}^{c} \sum_{k=1}^{c} \left[\left(D_{C_p} - D_{C_k}\right)^2 - \left(D'_{C_p} - D'_{C_k}\right)^2 \right] \\
&= \frac{L_{\text{inter}} - L'_{\text{inter}}}{L} - \frac{2}{8cL^2} \sum_{k=1}^{c} \left[\left(D'_{C_i} - D'_{C_k}\right)^2 - \left(D_{C_i} - D_{C_k}\right)^2 \right. \\
&\quad \left. + \left(D'_{C_j} - D_{C_k}\right)^2 - \left(D_{C_j} - D_{C_k}\right)^2 \right] \\
&= \frac{L_{\text{inter}} - L'_{\text{inter}}}{L} - \frac{1}{4cL^2} \sum_{\substack{k=1 \\ k \neq i,j}}^{c} \left[\left(D_{C_i} + D'_{C_i} - 2D_{C_k}\right)\left(D'_{C_i} - D_{C_i}\right) \right. \\
&\quad \left. + \left(D_{C_j} + D'_{C_j} - 2D_{C_k}\right)\left(D'_{C_j} - D_{C_j}\right) \right] \\
&\quad - \frac{1}{4cL^2} \left(D'_{C_i} - D_{C_i} - (D'_{C_j} - D_{C_j})\right)\left(D'_{C_i} + D_{C_i} - (D'_{C_j} + D_{C_j})\right)
\end{aligned}
\tag{A.1}
$$

[4] http://www.facebook.com

Appendix A.1. Transformation 1

Here, $L'_{\text{inter}} = L_{\text{inter}} - 1$, $D'_{C_i} = D_{C_i} - 1$ and $D'_{C_j} = D_{C_j} + 1$ as has been discussed in Transformation 1. The expression (A.1) becomes

$$\begin{aligned}
&\Delta m_1(G, D_{C_i}, D_{C_j}) = \\
&= \frac{1}{L} - \frac{1}{4cL^2} \sum_{\substack{k=1 \\ k\neq i,j}}^{c} \Big[\Big(2D_{C_i} - 2D_{C_k} + 1\Big) - \Big(2D_{C_j} - 2D_{C_k} - 1\Big)\Big] \\
&\quad - \frac{1}{4cL^2} \Big(D_{C_i} - D_{C_i} - (D_{C_j} - D_{C_j}) + 2\Big)\Big(D_{C_i} + D_{C_i} - (D_{C_j} + D_{C_j}) + 2\Big) \\
&= \frac{1}{L} - \frac{2}{4cL^2} \sum_{\substack{k=1 \\ k\neq i,j}}^{c} \Big(D_{C_i} - D_{C_j} + 1\Big) - \frac{2\Big(2D_{C_i} - 2D_{C_j} + 2\Big)}{4cL^2} \\
&= \frac{1}{L} - \frac{c-2}{2cL^2} \Big(D_{C_i} - D_{C_j} + 1\Big) - \frac{D_{C_i} - D_{C_j} + 1}{cL^2} \\
&= \frac{1}{L} - \frac{c-2+2}{2c \cdot L^2} \Big(D_{C_i} - D_{C_j} + 1\Big) = \frac{1}{L} - \frac{1}{2L^2} \Big(D_{C_i} - D_{C_j} + 1\Big) \\
&= \frac{2L - 1 - D_{C_i} + D_{C_j}}{2L^2}
\end{aligned}$$

Appendix A.2. Transformation 2

Here, $L'_{\text{inter}} = L_{\text{inter}}$, $D'_{C_i} = D_{C_i} - 2$ and $D'_{C_j} = D_{C_j} + 2$ as has been discussed in Transformation 2. The expression (A.1) becomes

$$\begin{aligned}
\Delta m_2(G, D_{C_i}, D_{C_j}) &= -\frac{1}{4cL^2} \sum_{\substack{k=1 \\ k\neq i,j}}^{c} \Big[\Big(D_{C_i} + D'_{C_i} - 2D_{C_k}\Big)\Big(D'_{C_i} - D_{C_i}\Big) \\
&\quad + \Big(D_{C_j} + D'_{C_j} - 2D_{C_k}\Big)\Big(D'_{C_j} - D_{C_j}\Big)\Big] \\
&\quad - \frac{1}{4cL^2} \Big(D'_{C_i} - D_{C_i} - (D'_{C_j} - D_{C_j})\Big)\Big(D'_{C_i} + D_{C_i} - (D'_{C_j} + D_{C_j})\Big) \\
&= \frac{4}{4cL^2} \sum_{\substack{k=1 \\ k\neq i,j}}^{c} \Big[\Big(D_{C_i} - D_{C_k} - 1\Big) - \Big(D_{C_j} - D_{C_k} + 1\Big)\Big] \\
&\quad + \frac{4}{4cL^2} \Big(2D_{C_i} - 2 - 2D_{C_j} - 2\Big) \\
&= \frac{1}{cL^2} \sum_{\substack{k=1 \\ k\neq i,j}}^{c} \Big(D_{C_i} - D_{C_j} - 2\Big) + \frac{2}{cL^2} \Big(D_{C_i} - D_{C_j} - 2\Big) \\
&= \frac{c-2+2}{cL^2} \Big(D_{C_i} - D_{C_j} - 2\Big) = \frac{1}{L^2} \Big(D_{C_i} - D_{C_j} - 2\Big)
\end{aligned}$$

Appendix A.3. Transformation 3

The difference Δm in modularity between the graph G and the graph G', obtained from G after a change in communities C_i, C_j and C_k (Transformation 3) is

$$\begin{aligned}
\Delta m_3(G, D_{C_i}, D_{C_j}, D_{C_k}) &= \frac{1}{L} - \frac{1}{4cL^2} \sum_{\substack{p=1 \\ p\neq i,j,k}}^{c} [\Big(D_{C_i} + D'_{C_i} - 2D_{C_p}\Big)\Big(D'_{C_i} - D_{C_i}\Big) \\
&\quad + \Big(D_{C_j} + D'_{C_j} - 2D_{C_p}\Big)\Big(D'_{C_j} - D_{C_j}\Big) + \Big(D_{C_k} + D'_{C_k} - 2D_{C_p}\Big)\Big(D'_{C_k} - D_{C_k}\Big)] \\
&\quad - \frac{1}{4cL^2} \Big(D'_{C_i} - D_{C_i} - (D'_{C_j} - D_{C_j})\Big)\Big(D'_{C_i} + D_{C_i} - (D'_{C_j} + D_{C_j})\Big) \\
&\quad - \frac{1}{4cL^2} \Big(D'_{C_i} - D_{C_i} - (D'_{C_k} - D_{C_k})\Big)\Big(D'_{C_i} + D_{C_i} - (D'_{C_k} + D_{C_k})\Big) \\
&\quad - \frac{1}{4cL^2} \Big(D'_{C_j} - D_{C_j} - (D'_{C_k} - D_{C_k})\Big)\Big(D'_{C_j} + D_{C_j} - (D'_{C_k} + D_{C_k})\Big) \\
&= \frac{1}{L} + \frac{1}{4cL^2} \sum_{\substack{p=1 \\ p\neq i,j,k}}^{c} [\Big(2D_{C_i} - 2D_{C_p} - 1\Big) + \Big(2D_{C_j} - 2D_{C_p} - 1\Big) \\
&\quad - 2\Big(2D_{C_k} - 2D_{C_p} + 2\Big)] - \frac{1}{4cL^2} (-1-2)\Big(2D_{C_i} - 1 - (2D_{C_k} + 2)\Big) \\
&\quad - \frac{1}{4cL^2} (-1-2)\Big(2D_{C_j} - 1 - (2D_{C_k} + 2)\Big) \\
&= \frac{1}{L} + \frac{1}{2cL^2} \sum_{\substack{p=1 \\ p\neq i,j,k}}^{c} \Big(D_{C_i} + D_{C_j} - 2D_{C_k} - 3\Big) \\
&\quad + \frac{3}{4cL^2} \Big(2D_{C_i} + 2D_{C_j} - 4D_{C_k} - 6\Big) \\
&= \frac{1}{L} + \frac{2c - 6 + 6}{4cL^2} \Big[D_{C_i} + D_{C_j} - 2D_{C_k} - 3\Big] \\
&= \frac{2L + D_{C_i} + D_{C_j} - 2D_{C_k} - 3}{2L^2}
\end{aligned}$$

References

[1] M. E. J. Newman, M. Girvan, Finding and evaluating community structure in networks, Phys. Rev. E 69 (2004) 026113.

[2] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, D. Wagner, On Finding Graph Clusterings with Maximum Modularity, in: Graph-Theoretic Concepts in Computer Science, volume 4769 of *Lecture Notes in Computer Science*, Springer Berlin/Heidelberg, 2007, pp. 121–132.

[3] R. Guimerà, L. A. N. Amaral, Functional cartography of complex metabolic networks, Nature 433 (2005) 895–900.

[4] J. Duch, A. Arenas, Community detection in complex networks using extremal optimization, Phys. Rev. E 72 (2005) 027104.

[5] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proceedings of the National Academy of Sciences 104 (2007) 36–41.

[6] A. Lancichinetti, S. Fortunato, Limits of modularity maximization in community detection, Phys. Rev. E 84 (2011) 066122.

[7] R. Guimerà, M. Sales-Pardo, L. A. N. Amaral, Modularity from fluctuations in random graphs and complex networks, Phys. Rev. E 70 (2004) 025101.

[8] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, E. Lefebvre, Fast unfolding of communities in large networks, Journal of Statistical Mechanics: Theory and Experiment (2008) P10008.

[9] P. Erdős, A. Rényi, On the evolution of random graphs, Publications of the Mathematical Institute of the Hungarian Academy of Sciences 5 (1960) 17–61.

[10] E. Gilbert, Random graphs, Annals of Mathematical Statistics 30 (1959) 1141.

[11] R. Albert, A.-L. Barabási, Statistical Mechanics of Complex Networks, Review of Modern Physics 74 (2002) 47–97.

[12] R. Albert, A.-L. Barabási, Topology of evolving networks: Local events and universality, Phys. Rev. Lett. 85 (2000) 5234–5237.

[13] T. Bu, D. Towsley, On distinguishing between internet power law topology generators, in: INFOCOM 2002. Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies., volume 2, pp. 638 – 647.

[14] M. Faloutsos, P. Faloutsos, C. Faloutsos, On power-law relationships of the Internet topology, in: Proceedings of SIGCOMM '99, ACM, New York, NY, USA, 1999, pp. 251–262.

[15] D. J. Watts, S. H. Strogatz, Collective dynamics of small world networks, Nature (1998) 440–442.

[16] M. E. J. Newman, Detecting community structure in networks, Eur. Phys. J. B 38 (2004) 321–330.

[17] A. Clauset, M. E. J. Newman, C. Moore, Finding community structure in very large networks, Phys. Rev. E 70 (2004) 066111.

[18] P. Schumm, C. Scoglio, Bloom: A stochastic growth-based fast method of community detection in networks, Journal of Computational Science 3 (2012) 356 – 366.

[19] S. Fortunato, Community detection in graphs, Physics Reports 486 (2010) 75 – 174.
[20] G. Agarwal, D. Kempe, Modularity-maximizing graph communities via mathematical programming, Eur. Phys. J. B 66 (2008) 409–418.
[21] S. Trajanovski, H. Wang, P. Van Mieghem, Maximum modular graphs, Eur. Phys. J. B 85 (2012) 1–14.
[22] N. Blenn, C. Doerr, S. van Kester, P. Van Mieghem, Crawling and detecting community structure in online social networks using local information, in: IFIP Networking, Prague, Czech Republic, 2012.
[23] M. E. J. Newman, Mixing patterns in networks, Phys. Rev. E 67 (2003) 026126.
[24] P. Van Mieghem, H. Wang, X. Ge, S. Tang, F. A. Kuipers, Influence of assortativity and degree-preserving rewiring on the spectra of networks, Eur. Phys. J. B 76 (2010) 643–652.
[25] P. Van Mieghem, X. Ge, P. Schumm, S. Trajanovski, H. Wang, Spectral graph analysis of modularity and assortativity, Phys. Rev. E 82 (2010) 056113.
[26] G. K. Orman, V. Labatut, H. Cherifi, Qualitative comparison of community detection algorithms., in: DICTAP (2), volume 167 of *Communications in Computer and Information Science*, Springer, 2011, pp. 265–279.
[27] M. Girvan, M. E. J. Newman, Community structure in social and biological networks, Proceedings of the National Academy of Sciences 99 (2002) 7821–7826.
[28] A. Lancichinetti, S. Fortunato, F. Radicchi, Benchmark graphs for testing community detection algorithms, Phys. Rev. E 78 (2008) 046110.
[29] J. P. Bagrow, Evaluating local community methods in networks, Journal of Statistical Mechanics: Theory and Experiment (2008) P05001.
[30] A. Lancichinetti, S. Fortunato, Community detection algorithms: A comparative analysis, Phys. Rev. E 80 (2009) 056117.
[31] P. Van Mieghem, Graph Spectra for Complex Networks, Cambridge University Press, Cambridge, UK, 2011.
[32] K. Manders, L. Adleman, NP-complete decision problems for quadratic polynomials, in: Proceedings of the eighth annual ACM symposium on theory of computing, STOC '76, ACM, New York, NY, USA, 1976, pp. 23–29.
[33] M. R. Garey, D. S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1990.