Louvain and Leiden:

Advances in Novel Modularity-Based Community Detection

Lewis D.L. Hart

Linacre College

University of Oxford

A dissertation submitted for the degree of

MSc in Mathematics and Foundations of Computer Science

Trinity 2022

Acknowledgements

Thank you to Professor Renaud Lambiotte for your guidance and your patience. Thank you to my friends for your support and kindness. Especially, thank you to Ameya, Jono and Sofia. Thank you most of all to my parents for everything else, and more.

Abstract

Community detection is a challenging and active research domain of network science. Community detection involves breaking down complex networks into densely connected groups of nodes, called communities. However, detecting communities algorithmically remains an abstruse challenge. Every known community detection method is vulnerable to some limitation [4], and computational complexities vastly vary [111]. One much-discussed [35, 36, 39, 19] approach to solve the community detection problem is via the optimisation of a quality function known as Newman-Girvan modularity [79]. Modularity optimisation, however, is an extremely difficult problem (specifically, an **NP**-complete problem [17]). Therefore, we resort to heuristic methods. There are two chief aims in designing a heuristic: the accuracy and the speed of the algorithm. Many such heuristics have been proposed [27, 93, 108, 79]. In 2008, the Louvain method [15] was first propounded and it has since become one of the most-cited works in all of network science. As recently as 2016 [111] the Louvain method was still reported to be among the fastest and best performing known community detection algorithms. In 2019 a new algorithm, called the Leiden method, was proposed by Traag et al in [103]. Leiden was derived from Louvain. In [103] it is claimed that Leiden surpasses Louvain in terms of both speed and accuracy, and Traag et al conclude that "the Leiden algorithm is strongly preferable to the Louvain algorithm". In this study we conduct our own numerical simulations to test Louvain and Leiden using 10 real-world complex networks, 6 of which have ground-truth communities meta-data. We use the Adjusted Rand Index (ARI) [51], which quantifies the symmetry between the output partition and the ground-truth, as our external evaluation measure. In the proceedings, we define and explain modularity, the Louvain method, and the Leiden method. Then, we present our findings. Our experiments represent, what we believe to be, the first ground-truth comparative analysis of Louvain and Leiden.

Contents

Chapter 1

Introduction to Community Detection in Complex Networks

Community [1] detection[2] within complex networks is a crucial research objective of network science. The growing recognition in recent years of the importance of community detection has led to extensive discussions in contemporary scientific literature [27, 15, 103, 37, 26, 47]. The focus of this dissertation is two preeminent community detection algorithms: the Louvain method (2008) [15] and the Leiden method (2019) [103]. The Leiden method is derived from the Louvain method and is the result of combining many previous developments to Louvain (*e.g.* [106, 100, 83, 9]) together with a new *partition refinement* phase to make one complete algorithm.

Louvain and Leiden both optimise a *quality function* called Girvan-Newman modularity [80] (abbrev. *modularity*). Let $G = (V, E)$ denote a graph[3] and let $\mathcal{P}$ denote a partition of V into groups of nodes $C \in \mathcal{P}$ (we call each subset C a *community*). Modularity, denoted Q, is a function which quantifies the quality of a partition by assigning a value $Q(\mathcal{P})$ to the partition between the range of -0.5 and 1. In general, we say the closer $Q(\mathcal{P})$ is to 1, the better[4] the partition $\mathcal{P}$.

There are two key objectives when it comes to designing modularity-based community detection methods: the speed of the underlying algorithm (*i.e.* its computational complexity), and the modularity value of the output partition. Both the Louvain and Leiden methods were built with scalability in mind: they both function exceptionally

[1] The term *module*, or *cluster*, is sometimes adopted elsewhere in the literature [5] and connotes the exact same attributes that we use to define a community in this work.

[2] We refer the reader to [35] for a comprehensive introduction to community detection, and to [37] for a review of the wider literature.

[3] Throughout this work, a graph refers to a simple, weighted, connected graph unless stated otherwise.

[4] However, as we explain in Section 2.3, this is not always the case.

fast, but they also return partitions of high modularity [15, 103]. This is essential in the context of analysing real-world complex networks, which are of an increasing (and technically illimitable) size.

The initial findings of Traag et al in [103] suggest the Leiden method significantly outperforms its predecessor, the Louvain method, in terms of both speed and modularity of the output partition. The authors compared the computation time and modularity of Louvain and Leiden in 6 large empirical networks, and reported that Leiden produced partitions with higher modularity in every case. In these networks, Leiden was at least twice as fast as Louvain, in some cases up to 20 times faster. Traag et al concluded that "the Leiden algorithm is strongly preferable to the Louvain algorithm". In this dissertation we conduct, what we believe to be, the first ground-truth comparative analysis[5] of the Louvain method with the Leiden method, and we use modularity and the Adjusted Rand Index (ARI) [51] as our comparative measures (Section 4.2).

Both Louvain and Leiden aim to solve the community detection problem. In the remainder of this first chapter we introduce the concept of complex networks and present the community detection problem. In the second chapter we derive modularity under the configuration model, explain why good partitions into communities should have high modularity, and briefly discuss the resolution limit [36]. In the third chapter we describe and explain the mathematics behind Louvain and Leiden. In the fourth and final chapter we reveal the results of our original experiments, where we used used ten real-world complex networks to test the speed and quality of Louvain versus Leiden.

1.1 Complex Networks

A complex network is a discrete mathematical object: its *raison d'être* is to capture the nature of a *complex system* which exists in the real-world by representing the structure of this complex system as a graphical object. In the natural and human-made world complex systems abound. Complex systems are constituted of many, much smaller components which interact with each other. To varying extents, connected components are inter-dependent on each other. The structure of the complex system evolves autonomously by interacting with its environment, and its internal mechanisms are often too complicated to control or to predict [12].

[5] Our methods are based on those seen in [81] and [89, 57, 40, 30].

Figure 1.1: A complex network which represents the social network of hundreds of League of Nations personnel, constructed by aggregating the metadata of thousands of archive documents. Each node represents a person, and edges represent that two people correspond frequently with each other. Node size increases in scale with node degree. **Credit:** Martin in "*La connaissance est un réseau*" [44].

Complex systems permeate natural science, social science, the economy, and technology. By modelling data as a complex network, the behaviour of many systems in the real-world can be sufficiently, or entirely, captured. To give just a few examples, complex networks have been used to model the acquaintances of persons (a social network) in [20, 98], the pages of the internet (an information network) in [42, 110], and cellular functions of molecules (a biological network) in [49, 91].

A challenge is that a complex system is not itself a pure mathematical object. The total quantity of all physical information transferred back and forth between components in a complex system can be enormous. A complex network extracts the fundamental structural features of a complex system and distils this information in a computationally practicable way. By observing and measuring a complex system, we can generate statistics which represent its key structure. Then, using this data we can generate a representation of the complex system as a graph $G = (V, E)$ which captures the connections between each pair of interdependent data-points. Data-points correspond to *nodes* (also termed *vertices*), and connections correspond to *edges* (also termed *links*). We use V to denote the set of all nodes, and E to denote the set of all edges. The type of graph that we use (*e.g.* weighted, directed, etc) is entirely dependent on the context of the complex system involved.

As Aristotle observed some 2,500 years ago, "The whole is more than the sum

of its parts.". If we vary the resolution at which we observe the components of a complex network, then the structure and dynamics of this network may appear to change, but the observed structural adjacency data does not. This is perhaps the most striking feature of complex systems. It can be difficult to understand the role of any single node when its behaviour is considered in the wider context of the system.

1.2 Communities and Community Detection

Let $G = (V, E)$ denote a graph and let $\mathcal{P} = \{C_1, \ldots, C_N\}$ denote a partition of the nodes V into N subsets (with $1 \leq N \leq |V|$). We call each subset $C \in \mathcal{P}$ a *community*. Community detection algorithms are designed to satisfy the community detection problem: given a graph $G = (V, E)$ as input, determine the optimal communities.

It is commonly agreed [66, 87] that communities within a network are locally edge-dense subgraphs. However, we take no account of edge-density in this definition of a community above. It is not mathematically defined what makes one community stronger than another, nor what makes one partition better than another. This is because there is no universally accepted [12, 80, 37] measure of edge-density.

At first glance, the community detection problem may seem the same as the graph partitioning problem [22], however this is superficial. The community detection problem is in many ways more difficult: in the graph partitioning problem the number of subsets N is specified as a parameter, whereas in the community detection problem the algorithm must determine the number (and size) of the subsets by its own mechanism.

Consider a network of acquaintances where people are represented by nodes, *e.g.* Figure 1.1. We know from our own experience that such networks have communities in them: subsets of nodes within which internal connections are dense, but between which external connections are less dense [41]. However, whilst it is relatively simple to gather the required adjacency data of who knows whom (*i.e.* the edges) in a social network, it is much harder to identify and measure which groups are those in which everyone knows each other (*i.e.* the communities).

Community detection algorithms identify obscure community structures and allow network scientists to uncover communities with methods which rely only on the adjacency data observed. By recognising community structures, we can isolate the unique functional components in a real-world network which are in contrast to the ubiquitous features. This could have practical uses, for example to target marketing

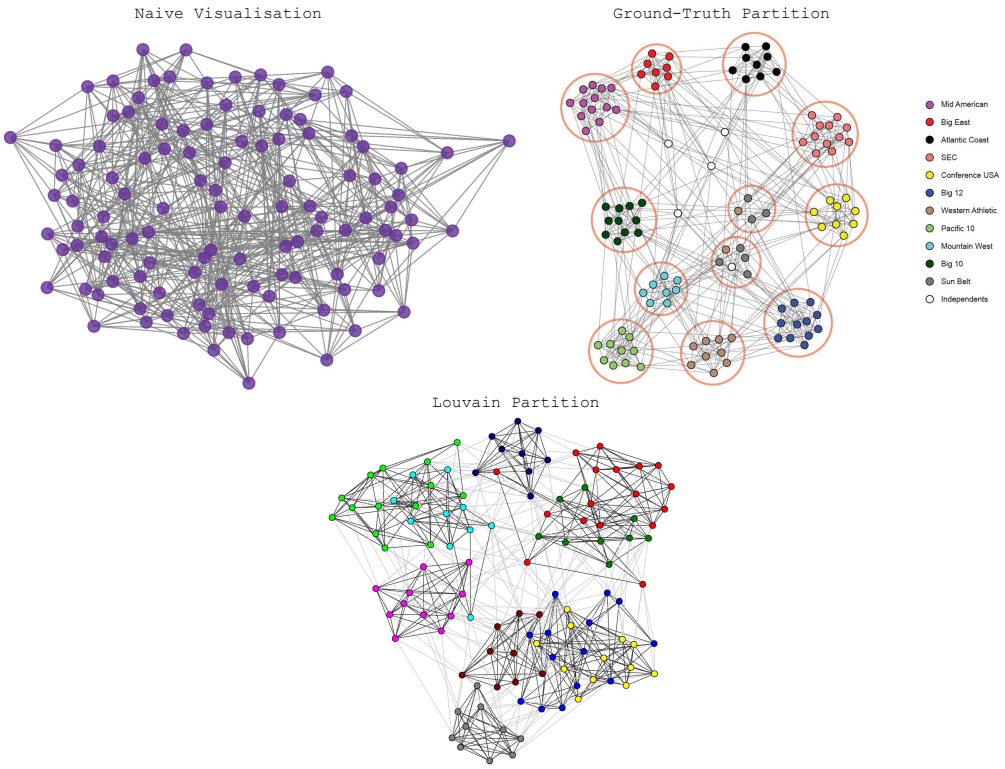

Figure 1.2: Three different visualisations of a complex network which represents American College football teams at a conference. This is one of the complex networks used in our experiments in section 4.3. **Credit:** (Right): Network data published by Girvan in "*Community structure in social and biological networks*" [41] and figure by SNAP Stanford [63]. (Left and Bottom): Author of this dissertation.

towards similar customers in co-purchasing networks [32], or to stunt contagion by isolating closely connected people in disease-modelling networks [69].

In very small graphs, and in graphs belonging to certain families of graphs, *e.g.* a connected caveman graph (Figure 2.1) or a ring of cliques (Figure 2.2), we can identify the best partition of V into communities manually. There is a guiding collective intuition: every community $C \in \mathcal{P}$ should have a high density of intra-community edges, *i.e.* the subgraph induced by C, $G[C]$, should be densely internally connected. As the size of communities and their number increases, the problem of identifying these groups becomes increasingly difficult. For example this can be seen by comparing Figure 1.2 with Figure 2.1. As the number of nodes $n = |V|$ increases, the size of the multi-dimensional partition space explodes (see Example 3.0.1), and we must resort to community detection algorithms.

Community structures have been widely observed [104, 35], but to algorithmically

obtain the best partition into communities remains an abstruse challenge. Many community detection methods have been developed [80, 96, 29, 15], and have evolved from very different mathematical foundations. There is significant discourse [59, 111, 35, 103, 81] on the subject of which community detection method is best. However, every known community detection method has limitations [4], computational complexities vary [111], and the search continues for better and faster algorithms.

Community detection algorithms usually work by defining a desirable quantity that can be measured in the network, and then manipulating the network over several iterations (*e.g.* removing edges, removing nodes, or changing edge-weights) with the aim of optimising this quantity. Notable categories of community detection algorithms include, but are not limited to, randomised algorithms (*e.g.* optimising random walks [96]), optimising centrality (*e.g.* optimising betweenness centrality [41]), and optimising modularity (*e.g.* Louvain [15] and Leiden [103]). In the next section we define and derive modularity [79], then discuss the behaviour of this quality function in some depth.

Chapter 2

Modularity: A Quality Function

In 2004 in [80], Newman and Girvan proposed modularity as a quality function to measure the total strength of the community structure within a given partition $\mathcal{P}$. In the same paper, Newman and Girvan provided the first modularity-optimisation algorithm [80]. This algorithm takes an input graph G, and the output is a high modularity partition of the nodes $\mathcal{P}$. High modularity partitions consist of many edge-dense communities. Therefore, Newman and Girvan had founded an entirely new approach to the community detection problem, which became of immediate scientific interest and led to several follow-up papers [36, 39, 77, 114, 19]. In these papers, adjustments to modularity to make the function compatible across a range of different graphs were considered, and also wider applications beyond community detection[1] were discussed.

So what is modularity? Modularity is a function which takes two arguments: a graph G, and a partition $\mathcal{P}$. The output is a real number $Q(\mathcal{P})$ in the range $[-1/2, 1)$ (Lemma 1). Modularity is formulated via the incorporation of a *null model*. A null model is a random statistical object which generates one or more random graphs G' according to a set of structural constraints prescribed by the empirical network G.

Modularity is a statistical measure of the non-uniformity of the edge-distribution in a given network G, where this uniformity is measured with respect to the chosen null model P. This allows us to differentiate between which structural features in G are unique, and which are ubiquitous. Modularity quantifies the deviation in edge-density over all communities in the partition by subtracting the expected edge-density (in a random graph generated by a null model) from the observed edge-density inside each community $C_i \in \mathcal{P}$.

[1] There exist numerous wider practical uses of modularity. For example, these include evaluating partition stability [71], controlling node dispersion in graph visualisation [82], and scale-reduction of larger graphs into smaller graphs such that the community structure is inertial [7].

Random graphs generated by the null model should not have clear community structures, since their edges are randomly wired in a way which is designed to be unbiased. When random graphs do have strong community structures, this should be purely chance [12]. Below, we provide the most general formulation of modularity, which is defined for any choice of null model.

Definition 2.0.1 (General Formulation of Modularity, Q)**.** Let $G = (V, E)$ denote a simple, unweighted graph. Let $m = |E|$ denote the number of edges in G, and let $\mathcal{P} = \{C_1, \ldots, C_C\}$ denote a partition of V into N communities $C_1, \ldots, C_N$, wherein each node $v_i \in V$ is assigned to some community $C_i \in \mathcal{P}$. Then

$$Q(\mathcal{P}) := \frac{1}{2m} \sum_{i,j} (A_{ij} - P_{ij}) \delta(C_i, C_j) \tag{2.1}$$

is the modularity of the partition $\mathcal{P}$.

Above, we have used A to denote the adjacency matrix of G (and so A_{ij} is equal to the number of edges which connect node i to node j in G), we have used P to denote the null model (and so P_{ij} is equal to the expected number of edges which connect node i to node j in a random graph prescribed by the null model), and we have used δ to denote the Kronecker-Delta function, which is defined

$$\delta(C_i, C_j) := \begin{cases} 1 & \text{if } C_i = C_j \\ 0 & \text{otherwise.} \end{cases} \tag{2.2}$$

Note that the pre-factor of $1/2m$ is for normalisation, and ensures $|Q| < 1$.

The null model takes a given graph $G = (V, E)$ as input, and generates a random graph $G' = (V', E')$ as output. The node-structure in the random graph must closely resemble the original graph G, but the edge-structure should be sufficiently random. Therefore, it is commonly constrained in most null models [58] that the number of nodes in the random graph G' is exactly the same[2] as in the input graph G. On applying the null model to a given empirical given network G, a selection of the unique structural features of G (*e.g.* the number of nodes, the number of edges, the degree distribution, *etc.*) are seized, and these features are absorbed by the null model as parameters which generate an ensemble.

An *ensemble* is a set $\mathcal{G}$ of random graphs generated by the null model. The structure of each graph $G' \in \mathcal{G}$ is entirely random, except for the constraints imposed by the parameters.

[2]This also means that we label each node $v \in V$ identically in V', since $|V| = |V'|$.

To use the modularity formula above (equation (2.1)), a particular null model P must be specified. For this null model, we must derive an approximation for the average number of edges that exist from any node $i \in V'$ to another node $j \in V'$. This approximation must hold for any pair of nodes $(i, j) \in V' \times V'$. This is done by constructing an $n \times n$ matrix of random variables, which we denote with A'. Each entry A'_{ij} of the matrix A' is itself a discrete random variable: specifically, A'_{ij} is the distribution of the number of edges ij in E'.

To calculate A'_{ij} directly, we could count the total number of edges between two nodes $i, j \in V'$ for all $G' \in \mathcal{G}$, then divide by the total number of realisations of G'. However, this approach is too computationally demanding. In fact, even generating a large sample of the ensemble to find such an estimate would be too demanding, since the total information contained in $\mathcal{G}$ quickly becomes too large. Therefore we resort to estimating A'_{ij} instead. This is done by approaching the problem probabilistically, and we approximate P with

$$P_{ij} = \mathbb{E}[A'_{ij}] \tag{2.3}$$

where $\mathbb{E}[A'_{ij}]$ denotes the expected value of A'_{ij} [58]. We will find an estimate for $\mathbb{E}[A'_{ij}]$ by averaging an edge-count over all possible pairs of nodes $i, j \in V'$, and this is explained in detail in the next section.

The choice of null model is up to the user. However, modularity is most well-understood when the *configuration model* is used as the null model. The original formulation of modularity by Newman and Girvan in [80] uses the configuration model. This is the default choice of null model for modularity, and the configuration model itself has been the subject of investigation in the scientific literature since the early 1990s [75, 68].

Real-world complex networks exhibit interesting and non-trivial structural connectivity patterns. These patterns are not found [58] in simple giant graphs, such as lattices, nor in random graphs generated by naive models. Whilst the degree distribution does not uniquely define a graph, the degree distribution does impose significant constraints on the structure and function of a complex network in real-world contexts.

We should aim to ensure that the degree distribution of random graphs $G' \in \mathcal{G}$ is sufficiently similar to the degree distribution of the complex network G. For this reason, it is not appropriate, for example, to use the Erdös-Renyi random graph model as our null model, since the Erdös-Renyi model generates networks with a Poissonian

degree distribution, and such complex networks are atypical in the real-world [21, 58]. The idea behind the configuration model is to generate random graphs G' which have nodes with exactly the same degrees as nodes in G.

In other words, the configuration model obeys the same degree distribution as a given network, but distributes edges across the network uniformly at random. Thus, every random graph $G' = (V', E') \in \mathcal{G}$ generated by the configuration model has exactly the same total number of edges, and exactly the same total number of vertices as does G. We describe the graph-generating process of the configuration model via defining an algorithm which generates an instance of $G' \in \mathcal{G}$. Let $G = (V, E)$ with $|E| = m$ be the input network, then:

1. Initially, each node $i \in V$ is assigned k_i half-edges (thus, the total number of these half-edges is exactly twice the total number of edges.).
2. The set of all $2m$ half-edges is partitioned into pairs: these pairings are drawn uniformly at random from the set of all possible pairings. This set of possible pairings is naturally reduced in size as each pairing is drawn. Finally, when $2m$ pairs have been drawn, each pair of half-edges is merged, and becomes a single edge.

We obtain different $G' \in \mathcal{G}$ depending on the pairings which were randomly chosen in step 2. Note that the random wiring procedure in the algorithm above does not forbid pairings of two half-edges connected to the same node (which generates a self-loop), nor does it forbid two or more pairings of half-edges each connecting the same two nodes (which generates multi-edges).

In the following section, we derive an expression for P under the configuration model, which is necessary in order to make use of the general formulation of modularity (2.1). If we were to adjust the configuration model algorithm so that it disallowed the possibility of self-loops or multi-edges, then this would significantly complicate the analytical calculation of P. In practice, there is no need to exclude either of these possibilities, as we shall explain.

In the second section, we explain why, if a partition $\mathcal{P}$ has high modularity, then this indicates $\mathcal{P}$ is a good partition of V into communities. We do this by reformulating modularity in terms of the numbers of intra-community and inter-community edges in each community. In the final section we briefly discuss an important limitation of modularity, known as the resolution limit [36].

2.1 Deriving Modularity Under the Configuration Model

Let $G = (V, E)$ denote a graph. Let $\mathcal{G}$ denote an ensemble generated by the configuration model with input graph G, and let $G' = (V', E') \in \mathcal{G}$ denote a random graph in this ensemble. In order to derive an equation for the modularity as formulated under the configuration model, we must first derive an expression for P, as per equation (2.1). The entries of P are given by $P_{ij} = \mathbb{E}[A'_{ij}]$, as per equation 2.3. Therefore, we require an approximation for $\mathbb{E}[A'_{ij}]$, the expected number of edges between nodes $i, j \in V' \times V'$.

We now show that this problem reduces to finding the probability, p_{ij}, that nodes $i, j \in V'$ are connected by an edge $ij \in E'$. Note that, since G is undirected, we have $p_{ij} = p_{ji}$. Let $i, j \in V' \times V'$ be two nodes in the random graph G'. Then, the expected number of edges between i and j is

$$\begin{aligned}
\mathbb{E}\left[A'_{ij}\right] =& \sum_{a=0} a\mathbb{P}\left(A'_{ij} = a\right) \\
=& 0.\mathbb{P}\left(A'_{ij} = 0\right) + 1.\mathbb{P}\left(A'_{ij} = 1\right) + \sum_{a\geq 2} a\mathbb{P}\left(A'_{ij} = a\right) \\
=& \mathbb{P}\left(A'_{ij} = 1\right) \\
=& p_{ij}
\end{aligned} \tag{2.4}$$

Above, the first line follows from applying the definition of the expected value of a discrete random variable, and the third line follows from the fact that the ratio of the expected number of multi-edges in relation to the total number of edges in E tends to zero in the limit of large networks[3] (equation (2.14)).

In the calculation above we have assumed nodes i and j are distinct (*i.e.* $i \neq j$). In the case nodes i and j are not distinct, then $\mathbb{E}[A_{ii}]$ is the expected number of self-loops connected to node $i = j$. According to the configuration model algorithm, a self-loop is generated with equal probability to any other edge. However, as the number of nodes n increases, the proportion of self-loops with respect to all edges rapidly decreases, since the total number of entries A_{ij} increases with n^2, and the total number of entries A_{ii} increases with n. Thus, our assumption that nodes i and j are distinct is reasonable in the limit of large networks.

We now prove a formula for p_{ij} which satisfies equation (2.4). Let $i \in V'$ denote a node which has been assigned its half-edges in step 1 of the algorithm above. There

[3] We postpone this proof until the end of this section.

exist $2m-1$ half-edges with which i can possibly be paired to form an edge in step 2. Of these $2m-1$ half-edges, node j has exactly k_j many. Thus, the probability that any one half-edge is connected to node j is $\frac{k_j}{2m-1}$.

Moreover, any half-edge is equally likely to be paired with any other, and node i has k_i half-edges. Thus, the expected probability that node i is connected to node j is

$$\frac{k_i k_j}{2m-1} \tag{2.5}$$

Since we have technically summed the probabilities for all pairings of half-edges at i with j in the above, it is important to note that equation (2.5) specifically denotes the expected number of edges between i and j, and not the total probability i is connected to j [78]. As the total number of edges in G becomes very large, we have $\frac{k_i k_j}{2m} << 1$. Therefore, it is sufficient to relax the denominator, and use $\frac{k_i k_j}{2m}$ as an approximation instead. Thus,

$$p_{ij} = \frac{k_i k_j}{2m} \tag{2.6}$$

is the expected number of edges between i and j in the random graph $G' \in \mathcal{G}$.

Combining equations (2.3),(2.4), (2.6), and (2.5) completes the derivation of the formula for the matrix P, where the entries P_{ij} are prescribed by the configuration model. In particular, the entries P_{ij}, are given by

$$P_{ij} = \mathbb{E}[A'_{ij}] = p_{ij} = \frac{k_i k_j}{2m} \tag{2.7}$$

Looking to eqn. (2.7), we see the probability for an edge ij to exist, P_{ij}, is clearly not uniform - this contrasts with the Erdös-Renyi model, for example. With equation (2.7), we have thus derived a formula for modularity under the configuration model, as was the aim of this section. The general formula becomes

$$Q(\mathcal{P}) := \frac{1}{2m}\sum_{i,j}(A_{ij} - \frac{k_i k_j}{2m})\delta(C_i, C_j) \tag{2.8}$$

and this is the formula for modularity we will use going forward.

To be rigorous in our derivation above, it only remains to prove that the (expected) number of multi-edges is negligible in the limit of large networks[4]. We now show that

[4]In the case of self-loops the proof is similar, and we refer the reader to [78] for details. Our derivation of the density of multi-edges also closely follows [78].

the density of multi-edges tends to zero as m increases, *i.e.*

$$\sum_{i\neq j, i,j\in V'\times V', a\geq 2} a\mathbb{P}(A'_{ij}=a)/m \to 0 \text{ as } m\to\infty \tag{2.9}$$

To do this, it is helpful to define the first and second moments of the degree distribution.

Definition 2.1.1 (First and Second Moments of Degree Distribution)**.** Let $\{k\}$ denote the degree distribution of a graph $G=(V,E)$. The first moment of the degree distribution is

$$\langle k\rangle := \frac{1}{n}\sum_{i\in V} k_i \tag{2.10}$$

and the second moment of the degree distribution is

$$\langle k\rangle^2 := \frac{1}{2}\sum_{i\in V} k_i^2. \tag{2.11}$$

where k_i denotes the (weighted) degree of vertex $i\in V$.

The first moment is the expected degree of a vertex in G. On applying the Handshaking Lemma to the definition of the first moment (2.10), we have $\langle k\rangle = \frac{2m}{n}$. The probability that nodes $i,j\in V'$ have at least 2 multi-edges is

$$\mathbb{P}\left(A'_{ij}\geq 2\right) = \frac{k_i k_j}{2m}.\frac{(k_i-1)(k_j-1)}{2m} = \frac{1}{(2m)^2}k_i\left(k_i-1\right)k_j\left(k_j-1\right) \tag{2.12}$$

where the second term in (2.12) follows from observing the following: since a multi-edge pairing is equally likely as any other half-edge pairing, after a first edge ij is generated by a pairing with probability p_{ij}, the probability of a second edge being paired between i and j (a multi-edge) is found by adjusting the formula for p_{ij} so that the degrees of nodes i and j reduced by 1.

Next, we sum this probability over all nodes $i,j\in V'$, and apply some simple algebra:

$$
\begin{aligned}
\frac{1}{2}\sum_{i\neq j,\ i,j\in V'}\mathbb{P}\left(A'_{ij}\geq 2\right) =& \frac{1}{2}\frac{1}{(2m)^2}\sum_{i\neq j,\ i,j\in V} k_i\left(k_i-1\right)k_j\left(k_j-1\right)\\
=& \frac{1}{2}\frac{1}{n^2\langle k\rangle^2}\sum_{i\in V} k_i\left(k_i-1\right)\sum_{j\in V} k_j\left(k_j-1\right)\\
=& \frac{1}{2}\frac{1}{\langle k\rangle^2}\sum_{i\in V}\left(\frac{k_i^2}{n}-\frac{k_i}{n}\right)\sum_{j\in V}\left(\frac{k_j^2}{n}-\frac{k_j}{n}\right)\\
=& \frac{1}{2}\frac{1}{\langle k\rangle^2}\left(\langle k\rangle^2-\langle k\rangle\right)^2\\
=& \frac{1}{2}\left(\frac{\langle k^2\rangle-\langle k\rangle}{\langle k\rangle}\right)^2\\
=& \frac{1}{2}\left(\frac{n}{2m}\langle k^2\rangle-1\right)^2
\end{aligned}
\tag{2.13}
$$

where the pre-factor of $1/2$ is due to double counting the edges in the sum, and where we use $2m = \langle k\rangle n$. By dividing this equation by n, we calculate an upper bound for the expected density of multi-edges between a pair of distinct nodes $(i,j)\in V'\times V'$

$$
\sum_{i\neq j,i,j\in V'\times V',a\geq 2} a\mathbb{P}(A'_{ij}=a)/m \leq \frac{1}{2}(\frac{1}{2m}\langle k^2\rangle-1)^2 \tag{2.14}
$$

Therefore, if $\langle k^2\rangle$ is constant and finite, then the expected number of multi-edges is zero as m increases.

The size of $\langle k^2\rangle$ (relative to $2m$) is dependent on the context of the empirical network. Therefore, when choosing the null model the context of the empirical network should always be taken into account, since contextual forces may exist which drive the formation of edges. In [10], Barabasi and Albert discovered that many real-world networks (*e.g.* the internet, citation networks, airport networks) have a power law degree distribution[5] [3, 16]. They termed these *scale-free* networks. Figure 1.1 is a visualisation of a scale-free social network.

However, in [21] Broido and Clauset published a wide array of statistical evidence which demonstrated scale-free networks are empirically rare[6]. Their study was constituted of 927 social, biological, technological, transportation, and information networks. This work of Broido and Clauset was not well-received by Barabassi [11].

[5] *i.e.* the proportion of nodes with degree k is distributed $Ck^{-\alpha}$ where C is a normalisation constant, and where $\alpha > 1$ is a scalar. Usually $2 < \alpha < 3$ [21].

[6] In 88% of the networks tested in [21], log-normal distributions were as good or better fits for the degrees than power law distributions, and only 4% met the strongest criterion for evidence of scale-free structure.

Whether or not scale-free networks are routine in the real-world still levies contemporary discourse [8], and Holme offers a useful introduction [50] to the wider literature. For our purposes, it is important to know that in the limit of large networks (*i.e.* as $m \to \infty$), scale-free networks have a finite first moment $\langle k \rangle$, but a divergent second moment $\langle k^2 \rangle$. This has direct implications to the bound derived above. If a network is scale-free, then the default configuration of modularity is a less reliable measure. This is also one motivation for choosing a context-dependent null model instead of the configuration model, an idea which is the subject of [31].

2.2 Intra-community and Inter-community Edges

Let $G = (V, E)$ denote a graph, let $\mathcal{P}$ denote a partition of V, and let $C \in \mathcal{P}$ denote a community in this partition. In this section we take an analytical approach and investigate the formula for the default configuration of modularity Q, equation (2.8). Our objective is to explain why partitions with high modularity values correspond to community structures which are intra-community edge-dense and inter-community edge-sparse. This naturally agrees with our common understanding of what a strong community structure is. If a partition $\mathcal{P}$ has high modularity, then we expect each community $C \in \mathcal{P}$ to comprise of few nodes with high degree, and of many more nodes with low degree[7]. We explain why.

If the subgraph $G[C]$ induced by a community $C \in \mathcal{P}$ is a connected subgraph, then we say C is an *internally connected* community. If a community C is not internally connected, then we say C is an *internally disconnected* community. In the following section we prove[8] two lemmas. The first is Lemma 1, which states the range of modularity, $-0.5 \leq Q(\mathcal{P}) < 1$. The second is Lemma 2, which states that for any graph G, if $\mathcal{P}$ is the partition of maximum-modularity, then every community $C \in \mathcal{P}$ is internally connected.

We now introduce some new notation. The set of *intra-community* edges in C is the subset of edges in E which have both end-nodes in C. The set of *inter-community* edges in C is the subset of edges in E which have exactly one end-node in C. We use $E(C, C') := \{\{v, w\} \in E : v \in C,\ w \in C'\}$ to denote the set of edges which connect nodes in community C to nodes in some other community $C' \in \mathcal{P}$. We use $E(C) := E(C, C) = \{\{v, w\} \in E : v, w \in C_i\}$ to denote the set of all edges

[7] These types of community structures are often called *assortative* in the literature [58].
[8] The proofs follow the arguments given by Brandes et al in [18].

which connect nodes in community C to other nodes in C. Thus, the set of all intra-community edges in C is $E(C)$, and the set of all intra-community edges in $\mathcal{P}$ is $E(\mathcal{P}) := \bigcup_{i=1}^{k} E(C_i)$. Each community C can be uniquely identified in G by the set of its intra-community edges, $E(C)$, *i.e.* by the edges within the subgraph induced by C, $G[C] := (C, E(C))$.

We use m_C^{in} to denote the total weighted edge-count of all intra-community edges in community C, and we use m_C^{out} to denote the total weighted edge-count of all inter-community edges in community C. Note that in the case G is simple, we have $m_C^{\text{in}} := |E(C)|$ intra-community edges, and $m_C^{\text{out}} := \sum_{C' \in \mathcal{P} \setminus C} |E(C, C')|$ inter-community edges.

Definition 2.2.1 (Trivial Partitions, $\mathcal{P}_v, \mathcal{P}_V$)**.** There are two *trivial partitions* of V. The trivial partition of each node into its own community (the *singleton partition*), denoted $\mathcal{P}_v := \{\{v\} : v \in V\}$. The trivial partition of all nodes into one community, denoted $\mathcal{P}_V := \{V\}$.

We now reformulate Q in terms of only the two variables m_C^{in} and m_C^{out}. The purpose of the Kronecker-delta function is to ensure that if nodes i and j are in different communities then their contributing summand to Q is zero (*i.e.* only non-zero when nodes i and j are in the same community). Therefore, we can replace δ with an outer sum over all communities $C \in \mathcal{P}$ instead:

$$Q(\mathcal{P}) = \sum_{C \in \mathcal{P}} Q(C) \tag{2.15}$$

where

$$Q(C) := \frac{1}{2m} \sum_{i,j \in C} A_{ij} - \frac{1}{(2m)^2} \left(\sum_{i \in C} k_i \right)^2 \tag{2.16}$$

denotes the contribution of C to Q. We call this property of a quality function *additivity* [35]. Note that the sum of degrees term follows from the indices, *i.e.*

$$\sum_{i,j \in C} \frac{k_i k_j}{2m} = \frac{1}{2m} \sum_{i \in C} k_i \sum_{j \in C} k_j = \frac{1}{2m} \left(\sum_{i \in C} k_i \right)^2 \tag{2.17}$$

Observe that, within each of the sums in the contribution $Q(C)$ above, the summands are always positive (*i.e.* $A_{ij} \geq 0$ and $k_i > 0$). Thus, owing to the parity of their pre-factors, we call the first sum in equation (2.16) the *positive contribution*[9] of C to Q, and the second sum in equation (2.16) the *negative contribution* of C to Q.

[9] In the literature, the positive contribution is also known as the *coverage* of community C [35].

Modularity imposes a penalty for two nodes $i, j \in C$ whenever i and j are disconnected, *i.e.* for every intra-community edge in C which is missing from the set of all possible intra-community edges in C. For example in the case of a simple, unweighted, connected graph, if two nodes $i, j \in C$ are connected by an edge, then the contribution of i, j to Q is positive, since $A_{ij} = 1$ and $\frac{k_i k_j}{2m} < 1$. On the other hand, if two nodes $i, j \in C$ are not connected by an edge in this graph, then their contribution to Q is be negative, since $A_{ij} = 0$.

In the positive contribution, only intra-community edges within C are counted. This sum counts each intra-community edge in C exactly twice, since G is undirected (and thus for all nodes i, j, we have $A_{ij} = A_{ji}$), thus

$$\sum_{i,j \in C} A_{ij} = 2|E(C)| = 2m_C^{\text{in}} \tag{2.18}$$

Therefore, if the positive contribution is large, then this implies that community $C \in \mathcal{P}$ has many intra-community edges. The positive contribution is maximised exactly when the number of intra-community edges in C is also maximised, *i.e.* when community C is edge-dense. Note that the positive contribution can always be increased by absorbing any neighbouring vertex into community C. However, in doing so this will always increase the negative contribution. In this way, owing to the quadratic term, each additional inter-community edge is penalised more than the previous.

The maximum possible negative contribution a pair of nodes $i, j \in C$ can make occurs when i and j both have very high degrees, but are not connected by an edge. Thus, the configuration model penalises edges which are absent from pairings of high degree nodes in C more than with pairings of low degree nodes [58]. The negative contribution is maximised exactly when the total degree of nodes in C is also maximised. This follows from observing that

$$\left(\sum_{i \in C_1} k_i\right)^2 + \left(\sum_{i \in C_2} k_i\right)^2 \leq \left(\sum_{i \in C_1 \cup C_2} k_i\right)^2 \tag{2.19}$$

since $a^2 + b^2 \leq (a + b)^2$ for all non-negative numbers a, b. Therefore, the larger a community $C \in \mathcal{P}$ is, the larger the negative contribution within $Q(C)$ is, *i.e.* partitions with too many large communities are not optimal. We demonstrate this in Example 2.2.1.

We now reconstruct the sum of total degrees as an edge-count to proceed as follows. Only edges which have end-nodes in community C are counted. Each of these edges is strictly either an inter-community edge or an intra-community edge. In the sum, inter-community edges are counted exactly once, since these edges have only one end-node in C, and intra-community edges are counted exactly twice, since these edges have two end-nodes in C. Thus,

$$\sum_{i \in C} k_i = 2m_C^{\text{in}} + m_C^{\text{out}} \tag{2.20}$$

We then have

$$Q\left(\mathcal{P}\right) = \sum_{C \in \mathcal{P}} \left(\frac{1}{m} m_C^{\text{in}} - \frac{1}{\left(2m\right)^2} \left(2m_C^{\text{in}} + m_C^{\text{out}}\right)^2 \right) \tag{2.21}$$

where Q is now expressed in terms of only the two variables m_C^{in} and m_C^{out}.

Example 2.2.1. Looking to the caveman graph in Figure 2.1 we can clearly see that the most natural partition into communities is the maximum modularity partition. Suppose we were to increase the size of the communities by merging community C_2 with community C_3. Then, $Q(C_1) + Q(C_2 \cup C_3) = \frac{17}{90} + \frac{17}{90} < \frac{17}{30} = Q(\{C_1, C_2, C_3\})$. Therefore, this partition with fewer communities is not optimal, because the communities are too large, and so there are too few contributions. If we were to try to increase the total number of communities instead then we would see the partition is not optimal either: even though the number of contributions will increase, the average size of each contribution will decrease.

Lemma 1 (Range of Modularity)**.** *Let $G = (V, E)$ denote a graph with $|E| = m \geq 1$ edges. Let $\mathcal{P}$ be a partition of V. Then, $-1/2 \leq Q(\mathcal{P}) < 1$.*

Proof. Looking to the modularity equation for a partition $Q(\mathcal{P})$, equation (2.8), we see that the maximum value of the positive contribution (*i.e.* the sum of intra-community edges) is attained if and only if $\mathcal{P} = \mathcal{P}_V$ (*i.e.* the trivial partition of all nodes into community). In this case, $\sum_{i,j \in \mathcal{P}} A_{ij} = 2m$ by the handshaking lemma. Since this bound is independent of the negative contribution, this implies that $Q(\mathcal{P}) \leq 1$ is an upper bound for all partitions $\mathcal{P}$. We now show that this inequality is strict. If $\mathcal{P} = \mathcal{P}_V$, then $Q(\mathcal{P}_V) = 0$ since

$$Q(\mathcal{P}_V) = \frac{1}{2m} \sum_{i,j \in V} A_{ij} - \frac{1}{(2m)^2} (\sum_{i \in V} k_i)^2 = \frac{2m}{2m} - \frac{(2m)^2}{(2m)^2} = 0 \tag{2.22}$$

which follows from the Handshaking Lemma, and this proves the stated global upper bound for Q.

We now prove the global lower bound for Q by considering each community $C \in \mathcal{P}$ separately. The contribution of C to Q is given by

$$Q(C) = \frac{1}{m} m_C^{\text{in}} - \frac{1}{(2m)^2} \left(2m_C^{\text{in}} + m_C^{\text{out}}\right)^2 \tag{2.23}$$

which follows from using the additivity property with equation (2.21) above. Observe that $Q(C)$ varies in m_C^{in}, but is strictly (quadratically) decreasing in m_C^{out}. The partial derivative of $Q(C)$ with respect to m_C^{in} is then given by

$$\frac{\partial Q\,(C)}{\partial m_C^{\text{in}}} = \frac{1}{m} - \frac{1}{m^2} \left(2m_C^{\text{in}} + m_C^{\text{out}}\right) \tag{2.24}$$

Therefore there is only one maximum when varying m_C^{in}, namely $m_C^{\text{in}} = \frac{1}{2}(m - m_C^{\text{out}})$. The negative contribution $Q(C)$ is maximised when m_C^{out} is as large as possible, and therefore a partition $\mathcal{P}$ which satisfies the global lower bound of Q occurs only when $m_C^{\text{in}} = 0$ for all communities $C \in \mathcal{P}$. In the case $\mathcal{P} = \mathcal{P}_V$, then $m_C^{\text{in}} \geq 1$. Thus, we require a partition with at least two communities. The negative contribution is maximised exactly when the total degree of nodes in C is also maximised, as per equation (2.19). Therefore, each community $C \in \mathcal{P}$ must be as large as possible, and consist of as many inter-community edges as possible, which requires $\mathcal{P} = \{C_1, C_2\}$ with $|C_1| = n/2 = |C_2|$ and $m_{C_1}^{\text{out}} = n/2 = m_{C_2}^{\text{out}}$.

Thus, Q achieves its lower bound only in the specific case of the bipartite graph $G = K_{a,b}$ on $a + b$ vertices, and when using the canonic clustering $\mathcal{P} = \{C_a, C_b\}$ with $a = n/2 = b$ and n even. In this specific case, we have

$$Q(\mathcal{P}) = Q(C_a) + Q(C_b) = 2(-\frac{1}{4}) = -\frac{1}{2}$$

which proves the global lower bound, $-1/2 \leq Q$.

□

Lemma 2. *Let $G = (V, E)$ denote a graph. If $\mathcal{P}$ is a partition of V with maximum modularity, then all communities in $\mathcal{P}$ are internally connected.*

Proof. Let $\mathcal{P}$ be a partition of V with maximum modularity (*i.e.* $Q(\mathcal{P}) \geq Q(\mathcal{P}')$ holds for all partitions $\mathcal{P}'$). Suppose for a contradiction, that there exists a community

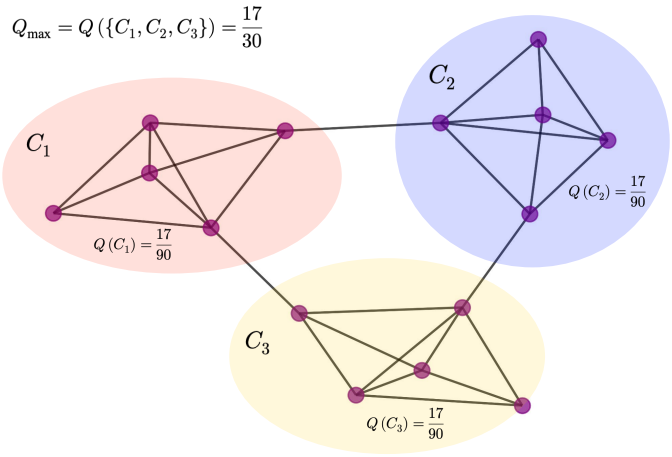

Figure 2.1: A demonstration of the maximum modularity partition $\{C_1, C_2, C_3\}$ for an instance of the connected caveman graph generated from 3 cliques, each with 5 nodes. Caveman graphs often arise in social network theory and in examples of the small-world phenomenon (*e.g.* [10, 107, 94, 34]). **Credit:** Author of this dissertation.

$C \in \mathcal{P}$ which is internally disconnected[10] (the subgraph induced by C, $G[C]$, consists of two or more components).

Let $v \in C$ be a node in this community. We construct a new partition $\mathcal{P}'$ by refining community C into two separate communities, $C_v, C'_v \in \mathcal{P}'$. We keep all other communities the same. The community C_v is the set of all nodes in C which belong to the same component as v in $G[C]$. The community, C'_v, is the set of all nodes in C which belong to a different component than v in $G[C]$. Using equation (2.21) and the additivity property we have

$$\begin{aligned}
&Q(\mathcal{P}) - Q(\mathcal{P}') = Q(C) - (Q(C_v) + Q(C'_v)) \\
&= \frac{1}{m}\left(m_C^{\text{in}} - (m_{C_v}^{\text{in}} + m_{C'_v}^{\text{in}})\right) - \frac{1}{(2m)^2}\left(\left(2m_C^{\text{in}} + m_C^{\text{out}}\right)^2 - \left(\left(2m_{C_v}^{\text{in}} + m_{C_v}^{\text{out}}\right)^2 + \left(2m_{C'_v}^{\text{in}} + m_{C'_v}^{\text{out}}\right)^2\right)\right) \\
&= \frac{1}{(2m)^2}\left(\left(\left(2m_{C_v}^{\text{in}} + m_{C_v}^{\text{out}}\right)^2 + \left(2m_{C'_v}^{\text{in}} + m_{C'_v}^{\text{out}}\right)^2\right) - \left(2m_C^{\text{in}} + m_C^{\text{out}}\right)^2\right) \\
&< 0
\end{aligned}$$

which is a contradiction. The third line follows from the fact that $m_C^{\text{in}} = m_{C_v}^{\text{in}} + m_{C'_v}^{\text{in}}$, since all intra-community edges of C are distributed between C_v and C'_v. The last line follows from the inequality $a^2 + b^2 \leq (a+b)^2$ for all non-negative numbers a, b with $a = 2m_{C_v}^{\text{in}} + m_{C_v}^{\text{out}}$ and $b = 2m_{C'_v}^{\text{in}} + m_{C'_v}^{\text{out}}$. This inequality is strict when $a, b \neq 0$. □

[10] Note that this does not imply there exist vertices in C which are disconnected from G itself. If this were the case, then G would be a disconnected graph.

2.3 The Resolution Limit

Any community detection algorithm has shortcomings, and the imperfections owned of each method are often unique [4]. One of the most important and much-discussed [56, 25, 13, 60, 46, 24] limitations of modularity is the resolution limit, which was discovered by Fortunato and Barthélemy in [36]. There are other important limitations, for example *non-locality* [18, 19] and the *rugged topology* [43] of the modularity-partition space, however we do not discuss these here.

The resolution limit causes incorrect community detection in large networks. In large networks, because of the resolution limit, groups of smaller communities are sometimes detected as single communities, *i.e.* merged together. We demonstrate this with an example.

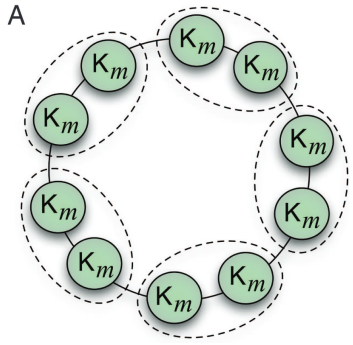

Figure 2.2: A sketch of a ring of cliques network. **Credit:** Fortunato and Barthélemy in "*Resolution limit in community detection*" [36].

Example 2.3.1. Consider Figure 2.2, which is a sketch of a *ring of cliques*. A ring of cliques consists of n identical cliques (each clique has m nodes,) and L total edges. Each clique is minimally connected to exactly two others. Since each clique is locally dense, we can see that the most natural partition is the one where each clique is a community. However, in [36], for a ring of cliques it was proven that if $n > \sqrt{L}$, with n even, then modularity optimisation leads to incorrect community detection. Specifically, partitions where two or more cliques are detected as single communities have higher modularity than the natural partition into n cliques. This is indicated by the dashed lines in the figure. What we can conclude from this example is that in large networks (*i.e.* with large m), the maximum-modularity partition may not be the most natural partition into communities.

The resolution limit arises from the dependency of the null model term on $\frac{1}{2m}$. One approach to mitigate this affect is to incorporate a *resolution parameter* $\gamma > 0$

into the general formulation of modularity. The resolution parameter was proposed by Kumpula et al in [56], and enables the tradeoff between intra-community edges and inter-community edges to be manually arbitrated.

$$Q(\mathcal{P}) := \frac{1}{2m}\sum_{i,j}(A_{ij} - \gamma P_{ij})\delta(C_i, C_j) \tag{2.25}$$

If $\gamma = 1$ then the default formulation of modularity (2.21) is recovered. If $\gamma > 1$ then Q favours smaller[11] communities and, if $\gamma < 1$ then Q favours larger communities.

There is no adaptation of modularity-based community detection which avoids the resolution limit entirely, but one approach is to use *multi-resolution* community detection methods which incorporate equation (2.25), *e.g.* [67, 92, 25, 46, 24]. These methods are extremely computationally expensive (relative to Louvain and Leiden) since they involve many rounds of modularity optimisation, possibly reusing the same partition many times but with different values of γ.

Varying γ adds extra dimensions to the basal partition space which, as we have seen in Example 3.0.1, is already enormous. However, multi-resolution methods illuminate significantly more complex structural patterns than do Louvain and Leiden. Even so, many implicitly depend on the Louvain Method for modularity optimisation - it may be that they would be even more effective operating with the Leiden Method.

[11] The terms larger and smaller here are relative to the optimal community-size when $\gamma = 1$.

Chapter 3

Fast Modularity Optimisation

In this chapter we describe both Louvain and Leiden in detail. In 2011, when first proposed [15], Louvain was recognised as the fastest community detection method available. Since then, this view has been widely upheld [59, 111, 111]. Leiden was proposed in 2019 by Traag et al in [103] and is derived from Louvain. Traag et al claim that Leiden is faster and more accurate than Louvain.

Louvain and Leiden are both modularity optimisation heuristics. To optimise modularity is to find a partition $\mathcal{P}$ for which Q attains its maximum value. With optimisation problems either exact algorithms or heuristic algorithms can be used. Exact algorithms return optimal solutions, and their solution is proven to be always optimal. On the other hand, a heuristic provides a solution which is close to optimal, and usually in good time. Sometimes a heuristic provides an optimal solution, but without proof of its optimality [5].

In 2006 in [17] Brandes et al proved that the optimisation of modularity is an NP-hard problem. This result was confirmed after much discourse apropos of the complexity of modularity optimisation, which for a long time had been speculated [79] to be NP-hard, owing to similarities with the MAX-CUT problem. In practical terms, this means that modularity-based community detection must be done with heuristics, since exact algorithms are computationally infeasible. Modularity optimisation is, furthermore, an NP-complete problem [17]. Therefore it is not possible for an algorithm to decide whether a partition is optimal, which adds some difficulty to designing a heuristic. Many novel heuristics have been proposed, for example greedy agglomeration[1] [27], simulated annealing [93], and spectral methods [108, 79].

[1] Both Louvain [15] and Leiden [103] are greedy agglomeration heuristics.

Before moving on to Louvain and Leiden, here is one example which elucidates why modularity optimisation is a computationally expensive problem: the size of the partition space quickly explodes and therefore can never be totally explored.

Example 3.0.1 (Total Number of Partitions)**.** Let $G = (V, E)$ denote a graph with $|V| = n$ vertices. The number of possible partitions of V into k sets (communities) is known as the *Stirling Number of the Second Kind* [54] and is denoted $S(n, k)$. The total number of all possible partitions is known as the n-th *Bell number* [35] and is denoted $B_n = \sum_{k=0}^{n} S(n, k)$. In the limit of large n, in [65] Lovász proved that B_n has the asymptotic form

$$B_n \sim \frac{1}{\sqrt{n}} \left(\lambda\left(n\right)\right)^{n+1/2} e^{\lambda(n)-n-1} \tag{3.1}$$

where $\lambda(n) = e^{W(n)} = \frac{n}{W(n)}$, and where $W(n)$ is the *Lambert W function*[2] [86].

The Bell number B_n grows at such a rapid rate with n that it can be difficult to comprehend. For example the Karate Club network, which can be seen in Figure 4.6, contains 34 nodes. This means that there are $B_{34} \sim 2.12 \times 10^{28}$ different partitions of this small club into different communities. This is more than the total number of seconds which have passed since the universe existed (approximately 4.42×10^{17} [2]). Therefore, even for small networks, we see it is hopeless to try and evaluate (or even enumerate) all partitions of V.

[2]The Lambert W function is defined by the relation $n = W(n)e^{W(n)}$. Thus, W is the inverse of the function $f(n) = ne^n$. Note, the inverse of f cannot be expressed in terms of elementary functions.

3.1 The Louvain Method

In 2008, a number of researchers were working on community detection algorithms in the Belgian town of Louvain. The results of their work were revealed in their novel paper "*Fast unfolding of communities in large networks*" [15]. The method they proposed became known as the *Louvain Method*. The Louvain method is a heuristic algorithm which rapidly and autonomously optimises certain[3] quality functions. In practice, the Louvain method is most widely-used as a community detection method via choosing modularity as the objective function to be optimised.

The Louvain Method is arguably the most popular and widely-used community detection method yet conceived, owing much of its popularity to the speed of its underlying algorithm. On its introduction in 2008, the Louvain method became widely regarded as the fastest modularity-based optimisation algorithm known to network science (in comparison to all other alternative contemporary methods). The following ten years saw a wide-range of evidence [59, 111, 103] which supports this consensus.

The original paper [15] is one of the most-cited works in all network science literature on the subject of the community detection problem. In this paper, Louvain was experimentally shown to be faster than all other prominent modularity-based community detection methods existing at the time, including an earlier agglomeration heuristic by Clauset et al [27], which might be considered a predecessor of Louvain. The claims by Blondel et al in [15] were later verified, agreed, and furthered by Lancichinetti and Fortunato in 2009 in [59], wherein the authors reported that the Louvain method was the fastest out of a wide range and variety of community detection methods, some of which were not modularity-based. Since its inception the Louvain method has remained one of the fastest known modularity-based optimisation algorithms in all network science literature, and as recently as 2016, in [111] the Louvain method was still reported to be among the fastest and best performing known community detection algorithms.

A process which naturally and often features in community detection algorithms is the transfer of nodes to and from different communities. This regularly happens many times throughout the computation. We now introduce new notation[4] to concisely

[3] The original proposal by Blondel et al in [15] exhibited Louvain as a heuristic which optimises modularity. Later, other authors in this space used Louvain to optimise objective functions other than modularity, *e.g.* the map equation [70].

[4] Adopted from [103].

describe these processes. Let $G = (V, E)$ denote a graph, and let $\mathcal{P}\{C, \ldots, C'\}$ denote a partition of V. Let $v \in C$ be a node in community C. We use $\mathcal{P}(v \mapsto C')$ to denote the partition that is obtained when node v is removed from community C and moved to community C' instead, *i.e.*

$$\mathcal{P}(v \mapsto C') := \{C \setminus v, \ldots, C' \cup v\}. \tag{3.2}$$

Suppose the partition $\mathcal{P}$ is replaced with partition $\mathcal{P}(v \mapsto C')$ instead. The resulting change in modularity is denoted[5] with $\Delta Q_{\mathcal{P}}(v \mapsto C')$, *i.e.*

$$\Delta Q_{\mathcal{P}}(v \mapsto C) := Q(\mathcal{P}(v \mapsto C)) - Q(\mathcal{P}) \tag{3.3}$$

Suppose now, that instead of just one node $v \in C$ moving community, an entire set of nodes $S \in C$ is moved. The resulting change in modularity is denoted with $\Delta Q_{\mathcal{P}}(S \mapsto C')$. An empty community is denoted with $\emptyset$. Therefore, $\Delta Q_{\mathcal{P}}(S \mapsto \emptyset)$ is the change in modularity which results from moving a set of nodes S to an empty community (*i.e.* to make S itself a new community in $\mathcal{P}$).

The algorithm which understands the Louvain method is an *agglomerative greedy heuristic*. As the algorithm progresses, the size of the communities increases. Nodes are initially assigned to small communities, and the mechanism of the algorithm proceeds by iteratively grouping nodes into larger communities. Louvain initialises with the singleton partition $\mathcal{P}_v$, then 2 phases are repeated until no further increase in modularity is possible. Note that for any graph G, we have $Q(\mathcal{P}_v) \leq 0$. Since in this case, the positive contribution to Q is always zero for every community in $\mathcal{P}_v$, and the negative contribution is non-zero (since G is connected).

The first phase is the *Local Moving* phase, and the second is the *Node Aggregation* phase. Each completion of both phases is called a *pass*. After each pass is completed, the output is a partition of equal or greater modularity, and it is guaranteed [15] that modularity cannot be increased by merging any two communities together in the partition, *i.e.* for all $C_1, C_2 \in \mathcal{P}$, one has $\Delta Q_{\mathcal{P}}(C_1 \mapsto C_2) < 0$. A pass which results in no increase in modularity is called a *stable iteration*. After a stable iteration, the Louvain algorithm terminates and the guarantee is stronger: modularity cannot be increased by moving any node to a different community in the partition, *i.e.* $\Delta Q_{\mathcal{P}}(v \mapsto D) \leq 0$ for all $v \in C$ and for all $D \in \mathcal{P} \cup \emptyset$.

[5] Note, when the partition $\mathcal{P}$ is implicit we will usually write $\Delta Q(v \mapsto C')$ instead.

Given an input graph, the Local Moving Phase uses a greedy sorting algorithm to reassign nodes into different communities. The output is a new partition which is more optimal apropos of equal or greater modularity. Given an input partition, the Node Aggregation phase involves structurally reducing the network topology. The output is a new (less complex) graph, which is called the *aggregate graph*. The node-structure and edge-structure of this aggregate graph is described below. The first round of the Local Moving phase is the slowest step in the entire process of Louvain [103]. However, subsequent rounds of the Local Moving phase (using the aggregate graph as input) are completed much more quickly, since the aggregate graph consists of significantly less adjacency data than its parent graph.

Since the ordering of nodes in phase 1 is random, Louvain is technically a stochastic algorithm. This means that the greedy sorting heuristic may lead to output a different partition, depending on the order in which nodes are considered. The ordering usually has an insubstantial effect on the quality of the partition [106]. However, the ordering is known to influence the total computation time [15, 106]. It is widely believed [14] that the Louvain method runs in $O(nlogn)$ time[6], which belongs to the complexity class P[7]. We now describe both phases of the Louvain algorithm in detail.

3.1.1 The Louvain Algorithm

Phase 1: Local Moving

First, all nodes are assigned a random ordering and indexed $V = \{v_1, \ldots, v_n\}$. At the beginning of each pass the partition is set to the singleton partition $\mathcal{P} := \mathcal{P}_v$, *i.e.* each node is assigned to its own community. Then, the greedy heuristic sorts all nodes into different communities via an iterative process. This begins with the first node $v_1 \in V$. The sorting is determined by a calculation: the total change in modularity that results when node v_1 is moved to the community of a neighbour instead (equation (3.4)). Since the sorting is greedy, node v_1 is moved to whichever community results in the largest increase in modularity, or not moved if all moves result in a decrease in modularity.

For all communities $C \in \mathcal{P}$ such that $\Gamma(v_1) \cap C \neq \emptyset$, the heuristic calculates $\Delta Q_{\mathcal{P}}(v_1 \mapsto C)$. If $\Delta Q_{\mathcal{P}}(v_1 \mapsto C) > 0$ for some neighbouring community C, then v_1 is moved to whichever community yields this greatest positive change in total

[6] As of the time of writing however, the proof of this complexity bound remains open.

[7] Therefore, since modularity optimisation is NP-Hard and Louvain is so fast, it can be said that Louvain must misidentify the optimal partition at least some of the time (assuming P $\neq$ NP).

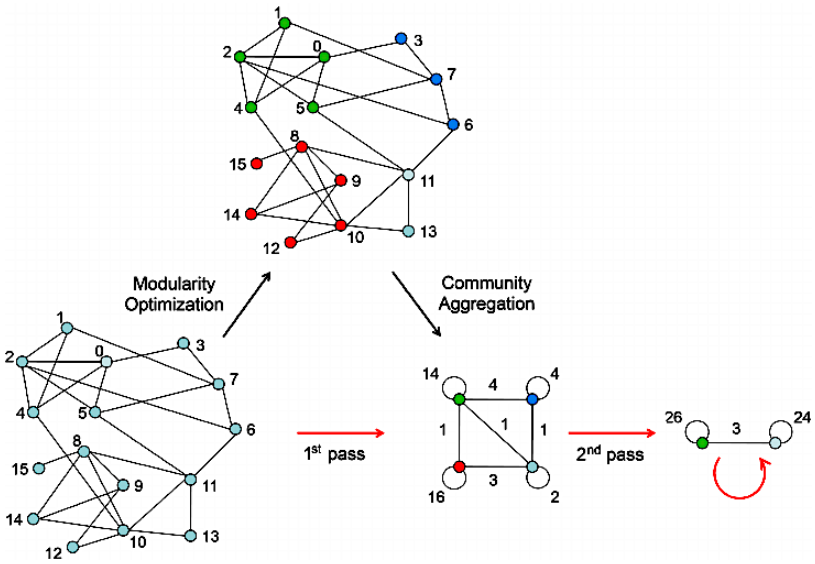

Figure 3.1: A demonstration of the Louvain method on a network. **(Bottom Left):** The input network. On initialisation of the first pass each node is assigned a random label and to a singleton community. **(Top):** The partition obtained after the first round of the Local Moving Phase has been completed. **(Bottom Right):** The aggregate networks obtained after the completions of the first and second passes. Observe that the self loops (which correspond to intra-community edges) significantly outweigh the multi-edges (which correspond to inter-community edges). This means that the modularity score of the partition will be high. **Credit:** Blondel et al in '*Fast Unfolding of Communities in Large Networks*' [15]

modularity. If $\Delta Q_{\mathcal{P}}(v_1 \mapsto C) < 0$ for every neighbouring community C, then the community of v_1 does not change. If, in any case, there is a tie for the greatest positive change in $\Delta Q_{\mathcal{P}}(v_1 \mapsto C)$, then this is resolved arbitrarily.

In [15], Blondel et al showed that the change in modularity which results from moving node v_i to community C (*i.e.* $\Delta Q_{\mathcal{P}}(v_i \mapsto C)$) can be calculated very quickly in two simple steps. The first step is to calculate the change in modularity which results from moving node v_i to its own singleton community (*i.e.* $\Delta Q_{\mathcal{P}}(v_i \mapsto \emptyset)$). Let $\mathcal{P}' := \mathcal{P}(v \mapsto \emptyset)$ denote this partition. The second step is to calculate the change in modularity which results from moving node v_i from its singleton community to community C in $\mathcal{P}'$. Since modularity is additive, we have

$$\Delta Q_{\mathcal{P}}(v_i \mapsto C) = \Delta Q_{\mathcal{P}}(v_i \mapsto \emptyset) + \Delta Q_{\mathcal{P}'}((v_i \mapsto C)) \tag{3.4}$$

where the explicit formulas for these modularity changes are omitted for brevity, and we refer to [15] for details. A sketch which demonstrates this equation is provided in Figure 3.2.

After v_1 has been sorted, the partition $\mathcal{P}$ is updated. Next, using the updated partition, the greedy sorting heuristic above is repeated for the next node in the indexing. On sorting the last node $v_n \in V$, the greedy heuristic terminates. If at least one node was moved to a different community following a complete execution of the greedy heuristic, then all nodes are sorted by the greedy heuristic again. If no nodes were moved following a complete execution of the greedy heuristic, then the local moving phase terminates, and the algorithm proceeds to execute phase 2.

Phase 2: Node Aggregation

In this phase an aggregated graph is constructed using the partition $\mathcal{P}$ obtained in phase 1. We call the network that was used in phase 1 the parent network. For each community $C \in \mathcal{P}$, exactly one node is inserted into the aggregate graph. The edges are assigned in the following way. Each intra-community edge in the parent network corresponds to a self-loop in the aggregate graph. Each inter-community edge in the parent network corresponds to a multi-edge in the aggregate graph. Edge-weights in the aggregate graph are determined by summing the corresponding edge-weights in the parent graph. An example of an aggregate graph can be seen in Figure 3.1.

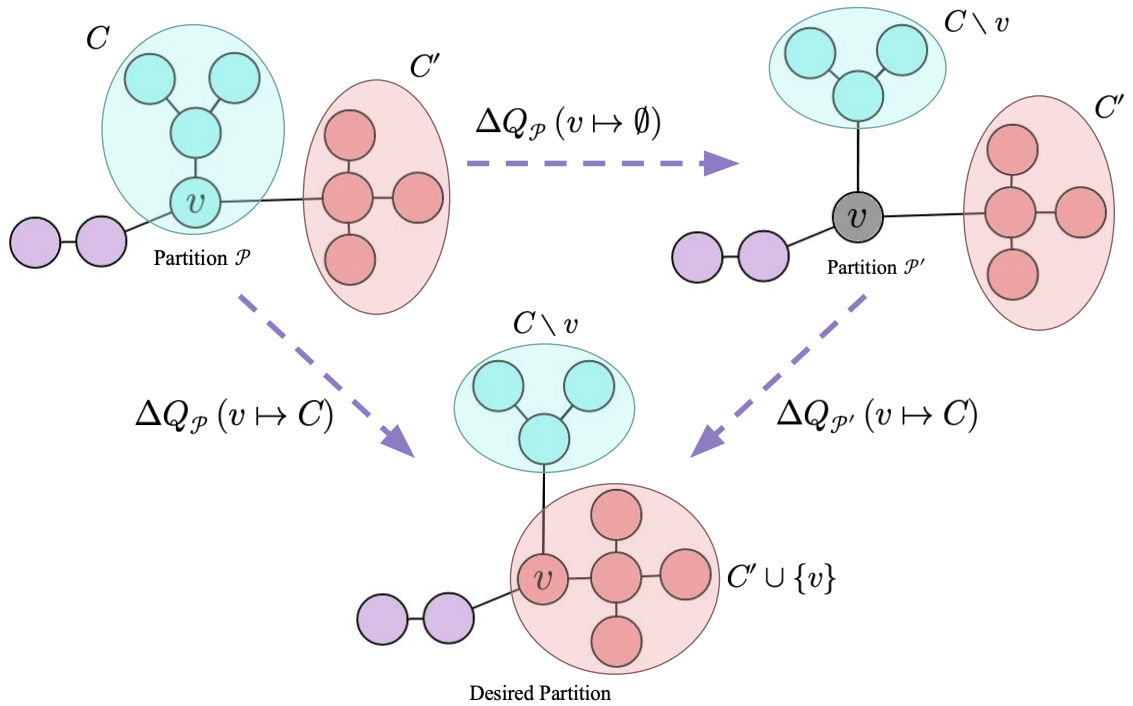

Figure 3.2: A demonstration of equation (3.4), incorporated within the Louvain greedy heuristic. This 2 step process enables rapid computation of the modularity changes which are required in each step of the Local Moving phase. **Credit:** Author of this dissertation.

3.2 The Leiden Method

Leiden was introduced by Traag et al in "*From Louvain to Leiden: guaranteeing well-connected communities*" [103]. In a similar fashion to Louvain, the Leiden Method takes its name from the city where it was developed, in this case the Dutch city of Leiden, some 200km from Louvain. The Leiden method is not a completely new algorithm per se, but rather a development of the Louvain Method. Whereas the Louvain algorithm consists of two phases, the Leiden method consists of three: *Fast Local Moving*, *Partition Refinement*, and *Node Aggregation*.

The Fast Local Moving and Partition Refinement phases of Leiden were largely created from four prior developments[8] of Louvain, combined together into one whole algorithm. Because of this, the mechanisms of Leiden are significantly more complicated than those of the Louvain. These developments are *Smart Local Move* propounded by Waltman et al. in [106] (further improved by Ozaki et al. in [83]), as well as *Fast Local Move* introduced[9] by Bae et al. in [9], and *Random Neighbour Move* proposed by Traag in [100]. The authors of Leiden also acknowledge [103] that their Fast Local Moving method is similar to another recent idea in this space called '*pruning*', which was proposed in 2016 by Ozaki et al in [83].

The Node aggregation phase largely remains the same, except for one key difference: the parent network used is the refined partition $\mathcal{P}_{\text{ref}}$ obtained in phase 2. The Fast Local Moving and Partition Refinement phases are more complicated, and we explain all phases in detail[10] in the next subsection.

When Traag et al investigated Louvain in [103], they discovered[11] that in the 6 real-world networks they tested up to 16% of Louvain-generated communities were internally disconnected. According to lemma 2.2, within a maximum modularity partition every community is internally connected. Therefore, theoretically, we can see that this is an important limitation of Louvain.

In [103] the authors of Leiden write "The Leiden algorithm has been specifically

[8] Each of the methods in the papers cited in this paragraph can themselves be considered as standalone improvements to the Louvain method.

[9] Note that in this paper their method is termed '*prioritisation*', and takes a slightly different form from its incorporation within Leiden.

[10] For more details, we refer to the supplementary information in [103], which also contains pseudo-code for both Louvain and Leiden.

[11] The disconnected communities problem has been observed previously [69, 109, 88], but Traag et al were the first to study this problem in the context of Louvain.

designed to address the problem of badly connected[12] communities.". In this regard, Traag et al were successful. After a stable iteration, it is guaranteed [103] that a Leiden-generated partition contains no disconnected communities. This is ensured by incorporating definitions 3.2.1 and 3.2.2 into the sorting procedure of the Partition Refinement phase. In the following definitions γ is the resolution parameter from equation (2.25). Let $\|S'\| := \sum_{s \in S'} \|s\|$ denote the recursive size of set S'.

Definition 3.2.1 (Well-connected node)**.** Let $G = (V, E)$ be a graph, let $\mathcal{P}$ be a partition of V, and let $C \in \mathcal{P}$ be a community in $\mathcal{P}$. Let $S \subseteq C$ be a subset of community C, and let $v \in S$ be a node which is contained within S. If

$$E(v, S - v) \geq \gamma \|v\|.(\|S\| - \|v\|) \tag{3.5}$$

then we say v is a node which is *well-connected* to S. In the case v is well-connected to C itself (*i.e.* $S = C$), we say v is well-connected to its community C.

Definition 3.2.2 (Well-connected community)**.** Let $G = (V, E)$ be a graph, let $\mathcal{P}$ be a partition of V, and let $C \in \mathcal{P}$ be a community in $\mathcal{P}$. Let $C \subseteq S \subseteq V$ be a subset of V which contains community C If

$$E(C, S - C) \geq \gamma \|C\|.(\|S\| - \|C\|) \tag{3.6}$$

for all such sets S, then we say C is a community which is *well-connected* to $\mathcal{P}$.

After each pass, and after each stable iteration, Leiden-generated partitions also have the same guarantees as do Louvain-generated partitions. After each pass it is guaranteed that modularity cannot be increased by merging any two communities together in the partition, *i.e.* for all $C_1, C_2 \in \mathcal{P}$, one has $\Delta Q_{\mathcal{P}}(C_1 \mapsto C_2) < 0$. After a stable iteration, modularity cannot be increased by moving any node to a different community in the partition, *i.e.* $\Delta Q_{\mathcal{P}}(v \mapsto D) \leq 0$ for all $v \in C$ and for all $D \in \mathcal{P} \cup \emptyset$. There are a handful of novel connectivity notions which were invented by Traag et al and are also guaranteed by Leiden at various stages of the iterative process. We refer to [103] for details.

[12] The term *badly connected* here is a generalisation of the notion of internally disconnected communities.

3.2.1 The Leiden Algorithm

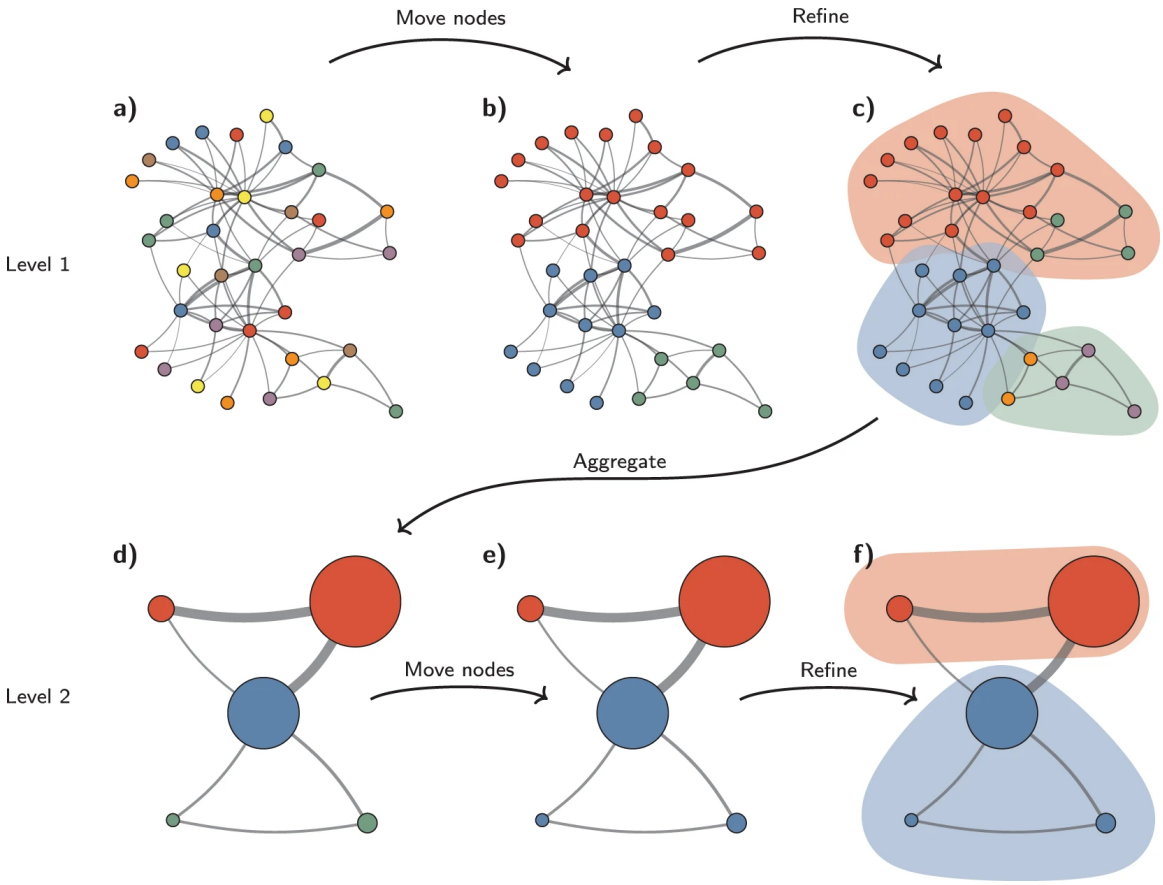

Figure 3.3: A schematic of Leiden. Each level is a demonstration of a pass. **a):** The singleton partition $\mathcal{P}_v$. Note, some colours overlap. This is the partition on initialisation of Leiden. **b):** The non-refined partition $\mathcal{P}$ obtained from the Local Moving Phase. There are 3 communities in $\mathcal{P}$: red, green and blue. **c):** The refined partition $\mathcal{P}_{\text{ref}}$ obtained from the Partition Refinement phase. Communities in $\mathcal{P}$ are represented with background colours. The red and green communities have both been refined into 2 new communities. **d):** The aggregate network obtained from the Node Aggregation phase. The aggregate network is built using $\mathcal{P}_{\text{ref}}$ as the parent network, but $\mathcal{P}$ (red, green and blue) is used as the initial partition for the subsequent Fast Local Moving Phase. **e):** The new partition obtained from the Fast Local Moving phase on the aggregate network. **f):** The new refined partition. No communities are refined. **Credit:** Traag et al in '*From Louvain to Leiden: guaranteeing well-connected communities*' [103].

Phase 1: Fast Local Moving

Given an initial partition $\mathcal{P}$, the nodes of this partition are locally sorted by the Fast Local Moving heuristic in order to obtain a new partition. This new partition is more optimal apropos of equal or greater modularity. In the first pass of phase 1 the

initial partition is the singleton partition, *i.e.* $\mathcal{P} := \mathcal{P}_v$. In every subsequent pass the initial partition is the partition $\mathcal{P}$ obtained in phase 1 from the previous pass. This is demonstrated in Figure 3.3. On initialisation of phase 1 a queue $\mathcal{Q} = \{v_1, \ldots, v_n\}$ is generated which contains all nodes in V. Nodes are assigned their position in $\mathcal{Q}$ via an arbitrarily random decision.

At each step of the heuristic the node at the front of the queue is sorted into a new community (or not), and then removed from the queue. When a node is sorted, some new nodes may be added to the back of the queue. Each node $v_i \in Q$ in the queue is sorted into the community which results in the greatest gain in modularity (*i.e.* the community $\arg\max_{C \in \mathcal{P} \cup \emptyset} \Delta Q_{\mathcal{P}}(v_i \mapsto C)$), just as was the case in Louvain[13].

If the community of node v_i is unchanged, then node v_i is removed from the front of $\mathcal{Q}$, and the heuristic proceeds to the next node in the queue. If the community of node v_i is changed to a different community, then each neighbour of node v_i which is not already in $\mathcal{Q}$ is appended to the back of $\mathcal{Q}$ (in an arbitrary order), and then node v_i is removed from the front of the queue. This continues until the queue is empty, at which point the non-refined partition $\mathcal{P}$ is saved, and phase 1 terminates.

Phase 2: Partition Refinement

In this phase a new partition, the refined partition $\mathcal{P}_{\text{ref}}$, is generated from $\mathcal{P}$ via a heuristic algorithm called *Random Neighbor Move* [100]. On termination of Random Neighbour Move, each community $C \in \mathcal{P}$ will either be unchanged in $\mathcal{P}_{\text{ref}}$, or will have been split into two or more (refined) communities in $\mathcal{P}_{\text{ref}}$. Within $\mathcal{P}_{\text{ref}}$ many communities $C \in \mathcal{P}$ will be refined into smaller communities, but this is not always the case [103].

Random Neighbour Move merges each singleton community $\{v\}$ with a neighbouring community via a stochastic sorting procedure. In this sorting procedure node v can be merged with any community which increases modularity (*i.e.* any community C for which $\Delta Q_{\mathcal{P}}(v \mapsto C)$ is positive), but node merges which decrease modularity are excluded[14], since this makes the refinement phase more efficient. In [103] it was proven that even when these node-merges are excluded, it is still possible for Leiden to return the maximum modularity partition[15].

[13] Note that this means the first $n = |V|$ nodes in phase 1 are processed in exactly the same way in both Louvain and Leiden.

[14] This is in contrast to simulated annealing algorithms [92, 45], for example, which do allow node-merges which decrease modularity.

[15] Louvain is limited in this regard. In the same paper [103], Traag et al proved that in some circumstances it is impossible for Louvain to return the maximum modularity partition, since in Louvain nodes are always greedily-merged.

Each node v is sorted by the heuristic in only one instance, and after all nodes have been sorted exactly once, the heuristic terminates. The heuristic functions in either one of two ways: by merging singleton communities together, or by merging singleton communities with larger communities.

It is precedent to make two pre-programming notes with regards to the sorting criteria. Firstly, nodes are only sorted locally, *i.e.* two nodes v_1, v_2 can be merged into the same community in $\mathcal{P}_{\text{ref}}$ if and only if v_1, v_2 belong to the same community $C \in \mathcal{P}$. This ensures that $\mathcal{P}_{\text{ref}}$ is indeed a refinement of $\mathcal{P}$ in respect of the formal combinatorial meaning of the term. Secondly, a node $v \in C \in \mathcal{P}$ is merged with a community $C' \in \mathcal{P}_{\text{ref}}$ if and only if both v is well-connected to C (Definition 3.2.1), and $C' \subseteq C$ is well-connected to C (Definition 3.2.2).

The refined partition $\mathcal{P}_{\text{ref}}$ is then obtained as follows. Initially, $\mathcal{P}_{\text{ref}}$ is set equal to the singleton partition $\mathcal{P}_v$. Then, v is randomly merged with community C. The degree of randomness in the selection of a community is determined by a parameter $\theta > 0$, and the probability a community C' is selected is

$$\mathbb{P}\left(C' = C\right) \begin{cases} \exp\left(\frac{1}{\theta}\Delta Q_{\mathcal{P}}\left(v \mapsto C\right)\right) & \text{if } \Delta Q_{\mathcal{P}}\left(v \mapsto C\right) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

The larger the increase in modularity, the more likely a community is to be selected. Note that this means the greedy decision (*i.e.* the community C for which $\Delta Q_{\mathcal{P}}(v \mapsto C)$ is most positive) is always the most likely outcome.

Phase 3: Node Aggregation

In this phase an aggregate graph is built in the same way as in the Node Aggregation phase of Louvain. However, the input partition which generates the aggregate graph is $\mathcal{P}_{\text{ref}}$ instead of $\mathcal{P}$.

Since $\mathcal{P}_{\text{ref}}$ is a refinement of $\mathcal{P}$, we expect the refined partition to contain more communities than $\mathcal{P}$. Therefore, the network aggregated by Leiden should contain more information (*i.e.* more nodes and more specific edge structures) than the network aggregated by Louvain. This means that in the exploration of the aggregate network during the Fast Local Moving phase of a subsequent pass, Leiden has greater maneuverability with which to merge communities, and should have more opportunities to identify an optimal partition.

Chapter 4

Louvain and Leiden: A Ground-Truth Comparative Analysis

In this chapter we describe the original numerical simulations which were conducted for this dissertation. Our experiments involved testing Louvain and Leiden on ten real-world complex networks. Six of these networks had ground-truth communities meta-data. For each network, we measured the quality of the partitions which were generated by each method. This was done for the partition generated at the end of each pass and until termination. Then, the modularity of each partition was calculated. For the networks with ground-truth the Adjusted Rand Index (ARI) [51] was also calculated, and was used as an external measure of accuracy. The ARI has been used to compare the efficacy of different community detection methods in this way before (*e.g.* [81, 89, 57, 40, 30]). We describe the ARI in more detail in the methods section.

Our simulations were processed on a 2019 Macbook Pro with 8 GB of 2133 MHz memory and one 1.4 GHz Quad-Core Intel Core i5 processor. Python 3 was used with the `LeidenAlg`[1] [103] package. This package exposes Louvain and Leiden to Python via the `igraph` [28] package. The data analysis was also conducted in Python, using the `scikit-learn` [84] package.

4.1 Materials

Here, we describe the real-world context of each complex network that was used in our experiments. In Table 4.1 we have summarised some of their defining empirical

[1] Available at https://github.com/vtraag/leidenalg.

quantities. There is a limited availability of complex networks with complete ground-truth communities meta-data, and it was difficult to find such networks which were of an appropriate size to process using our computer. Therefore, for some of our experiments, we also included 4 empirical networks which did not have ground-truth metadata. Note that the 4 networks without ground-truth contain more nodes. All networks are unweighted and undirected[2].

- *Karate club network.* This data set is derived from a university sports club in the 1970s and was studied by Zachary [113]. Nodes represent members of the club, and edges represent a social connection between two members. Communities represent the club's unexpected and unplanned split into two groups.

- *Political books network.* This data set was derived from a study by V. Krebbs [55] using data on books published around the time of the 2004 presidential election. Nodes represent books about US politics sold by Amazon, and edges represent two buyers purchasing the same books. Communities represent groups of books on similar topics.

- *American college football network.* This data set was derived from an American college football league by Girvan [41], and represents the schedule of Division I games during the season Fall 2000. Nodes represent football teams, and edges represent scheduled games. Communities represent 12 conferences. Teams play more games with rivals in the same conference than with other teams. See Figure 1.2 for an illustration of the ground-truth partition of this network.

- *Ego-network.* This data set consists of Facebook friends information collected by McAuley and Leskovec [62]. Nodes represent Facebook users, and edges represent Facebook friendships. Communities[3] represent hand-labelled 'friend circles' of a single Facebook user.

- *Email-EU-core network.* This data set was collected by Leskovec et al in [64], and consists of email contacts between members of a large European research institution. Nodes represent people, and edges represent if they have had contact via email. Communities represent the 42 departments at the research institute.

[2]The original data [64] for the Email EU-core network is for a directed network. We have transformed the network into an undirected network for our experiments.

[3]Note, overlapping friend circles have been amalgamated into single communities in this data set.

- *Political blogs network.* This data set was collected by Adamic and Glance in [1] after the 2004 United States election. Nodes represent blog posts, and edges represent a post citing another. Communities represent bloggers supporting either Democrats or Republicans.

- *Fly Optical Lobe.* This data set is a brain network of the optic lobe (*drosophila medulla*) of a fly adn was generated from an MRI brain scan by Amunts et al in [6]. Nodes represent physical regions of the brain, and edges represent fiber tracts between regions.

- *Oxford Birds.* This is a social network of wild birds in their habitat of Wytham Woods, Oxfordshire, and was collected by Firth et al in [33]. Nodes represent birds, and edges represent that two birds used the same nest-box.

- *General Relativity* and *High Energy Theory.* These data sets were collected by Leskovec et al in [64], and are the Arxiv GR-QC (General Relativity and Quantum Cosmology) collaboration network, and the Arxiv HEP-TH (High Energy Physics Theory) collaboration network. Nodes represent authors, and edges represent that two authors collaborated on a paper.

Complex Network	$\|V\|$	$\|E\|$	*Comm.*	$\langle k \rangle$	$\langle k^2 \rangle$	*Clus.*	*Diam.*
Karate club	34	78	2	4.5882	35.647	0.5706	5
Political books	105	441	3	8.4000	2223.0	0.4875	7
College Football	115	613	12	10.6608	114.43	0.4032	4
Facebook Ego	792	14025	24	35.417	2071.1	0.483	10
Email EU-core	1005	16064	42	31.968	2386.6	0.450	7
Political blogs	1222	16714	2	27.3552	35772.0	0.3203	8
Fly Optical Lobe	1781	9016	N/A	10.125	809.94	0.26285	6
Oxford Birds	4047	15864	N/A	7.8399	1008.4	0.2937	6
General Relativity	5242	14496	N/A	5.5307	93.29	0.52964	17
High Energy Theory	9877	25998	N/A	5.2644	66.045	0.4714	17

Table 4.1: Structural data for each of the 10 real-world complex networks used in our experiments. Arranged in order of number of nodes. **Key:** *Comm.* = number of communities in the ground-truth partition, *Clus.* = average clustering coefficient, and *Diam.* = diameter. See [81] for source of networks with ground-truth. See [6, 64, 33, 95] for source of networks without ground-truth.

4.2 Methods

Each network above was processed with Louvain and with Leiden. Every partition at the end of each pass was recorded. Then, for every network above, the modularity of each Louvain-generated and Leiden-generated partition (at the end of each pass) was calculated and compared. For both algorithms we used a resolution parameter of $\gamma = 1$, *i.e.* the default configuration of modularity (equation (2.8)). For the Leiden algorithm, we set the refinement randomness parameter to $\theta = 0.01$, as per the recommendations in [103].

We ran Louvain and Leiden 100 times on each of the 6 networks with ground-truth in Table 4.1. We recorded the average modularity of the final partition, the average ARI of the final partition, the average computation time, the average number of communities in the final partition, and the average number of passes. We also recorded the standard deviations of each of these measures. In the case of the 4 networks without ground-truth and with more nodes, we ran Louvain and Leiden 50 times on each network. Since there was no ground-truth, we could not calculate the ARI. Otherwise, our methods were the same.

The ARI is a measure of pairwise agreement measured for each node, then summed over all nodes. The ARI quantifies the extent of the symmetry between the output partition and the ground-truth partition. The range of the ARI is $[-1, 1]$. An ARI score of 0 indicates an entirely random partition. ARI scores closer to 1 indicate good partitions, and ARI scores closer to -1 indicate poor partitions. The ARI is a statistic obtained by calculating the concordance of each node $v \in V$ over $\mathcal{P}$ and $\mathcal{P}'$ in a way which categorises pairings into one of four distinct regimes, then tallying over all nodes in V. The number of nodes in each regime is denoted with a, b, c and d, and we use $N = a + b + c + d = \binom{n}{2} = \frac{1}{2}n(n-1)$ to denote the total number of pairs of nodes $(v_i, v_j) \in V^2$.

Let $G = (V, E)$ denote a graph with $n = |V|$ vertices. Let $\mathcal{P} = \{C_1, \ldots, C_r\}$ and $\mathcal{P}' = \{C'_1, \ldots, C'_s\}$ be two partitions of V (into r and s communities respectively). The information which records the concordance of each pair of communities in $(C_i, C'_j) \in \mathcal{P} \times \mathcal{P}'$ is registered in a contingency table (Table 4.2). The (i, j)-th element of the contingency table is $n_{ij} := |C_i \cap C'_j|$, *i.e.* the number of nodes which are in both community $C_i \in \mathcal{P}$ and $C'_j \in \mathcal{P}'$. The entries n_{ij} of the contingency table form an $r \times s$ matrix, which we denote with $\mathbf{N}$. The matrix $\mathbf{N}$ is termed the *confusion matrix* of the partitions $\mathcal{P}$ and $\mathcal{P}'$ in the literature [23].

$\mathcal{P}$ / $\mathcal{P}'$	C_1	C_2	$\dots$	C_r	Marginals
C'_1	n_{11}	n_{12}	$\dots$	n_{1r}	$n_{\to 1}$
C'_2	n_{21}	n_{22}	$\dots$	n_{2r}	$n_{\to 2}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
C'_s	n_{s1}	n_{s2}	$\dots$	n_{sr}	$n_{\to s}$
Marginals	$n_{\downarrow 1}$	$n_{\downarrow 2}$	$\dots$	$n_{\downarrow r}$	$\sum_{ij} n_{ij} = N$

Table 4.2: The Contingency table of two partitions $\mathcal{P}$ and $\mathcal{P}'$.

The sequence of the row-wise totals of the confusion matrix is denoted with $(n_{\to 1}, \dots, n_{\to r})$, where each entry is given by

$$n_{\to i} := \sum_{j=1}^{s} n_{ij} \tag{4.1}$$

The sequence of the column-wise totals of the confusion matrix is denoted with $(n_{\downarrow 1}, \dots, n_{\downarrow r})$, where each entry is given by

$$n_{\downarrow i} := \sum_{i=1}^{r} n_{ij}. \tag{4.2}$$

These two sequences are called the *marginals* of the confusion matrix. Each row-marginal $n_{\to i}$ records the size of community $C_i \in \mathcal{P}$, and each column-marginal $n_{\downarrow i}$ records the size of community $C'_i \in \mathcal{P}'$. We now define the four regimes a, b, c and d:

$$a := \frac{1}{2}\left(\sum_{i=1}^{r}\sum_{j=1}^{s} n_{ij}^2 - n\right) \tag{4.3}$$

is the total number of pairs (v_i, v_j) such that nodes v_i and v_j belong to the same community in both partitions (true positives)[4],

$$b := \frac{1}{2}\left(\sum_{i=1}^{r} n_{\to i}^2 - \sum_{i=1}^{r}\sum_{j=1}^{s} n_{ij}^2\right) \tag{4.4}$$

is the total number of pairs (v_i, v_j) such that nodes v_i and v_j belong to the same community in $\mathcal{P}$, but to two different communities in $\mathcal{P}'$ (false negatives),

[4]These information theoretic terms are in respect of $\mathcal{P}$ being the ground-truth partition.

$$c := \frac{1}{2}(\sum_{j=1}^{s} n_{\downarrow j}^2 - \sum_{i=1}^{r}\sum_{j=1}^{s} n_{ij}^2) \tag{4.5}$$

is the total number of pairs (v_i, v_j) such that nodes v_i and v_j belong to two different communities in $\mathcal{P}$, but to the same community in $\mathcal{P}'$ (false positives),

$$d := \frac{1}{2}(n^2 + \sum_{i=1}^{r}\sum_{j=1}^{s} n_{ij}^2 - \sum_{i=1}^{r} n_{\rightarrow i}^2 - \sum_{j=1}^{s} n_{\downarrow j}^2) \tag{4.6}$$

and is the total number of pairs (v_i, v_j) such that nodes v_i and v_j belong to different communities in both partitions (true negatives).

The Rand Index (RI) itself [90] is defined as $\text{RI} := \frac{a+d}{N}$. A limitation of the Rand Index [90] is that there exists no mechanism to mitigate the possibility that two partitions are highly symmetrical by chance alone [76]. Therefore, the Adjusted Rand Index incorporates the expected value of the Rand Index, which is denoted with $\mathbb{E}[\text{RI}]$. This is the average value of the Rand Index of a partition which is randomly generated, but is constrained so that all marginals are equal to the original partition[5]. Therefore the ARI is well-described as a corrected-for-chance version of the Rand index:

Definition 4.2.1 (Adjusted Rand Index)**.** The Adjusted Rand Index (ARI) is owed to Hubert and Arabie [51], and is defined

$$\text{ARI} := \frac{\text{RI} - \mathbb{E}[\text{RI}]}{1 - \mathbb{E}[\text{RI}]} \tag{4.7}$$

$$= \frac{N(a+d) - [(a+b)(a+c) - (c+d)(b+d)]}{N^2 - [(a+b)(a+c) + (c+d)(b+d)]} \tag{4.8}$$

where the denominator in the first line follows from the fact that the range of the Rand Index is $[0, 1]$, and where the concise formulation in the second line is owed to Steinley [99].

The ARI is one of the most commonly used similarity measures to compare two clusterings of a given set of objects [23], and there are many examples [81, 40, 52, 53] of the ARI being used as a measure of the clustering accuracy in the context of complex networks and community detection. The ARI is also the recommended external clustering evaluation measure of choice in [74]. We refer the reader to [51] and [23, 112, 97] for further details in regards to the subject of the ARI measure itself.

[5]There are some conceptual symmetries here between the ARI and the configuration model (section 2).

4.3 Results

For each network with ground-truth, the results of the partitions generated by 1 instance of Louvain and Leiden are displayed in Figure 4.3, and the average results of 100 instances of Louvain and Leiden are displayed in Figures 4.1 and 4.4. For each of the networks without ground-truth, the average results of 50 instances of Louvain and Leiden are displayed in Figure 4.4. The computation times and average number of passes for all 10 networks are displayed in Figure 4.2.

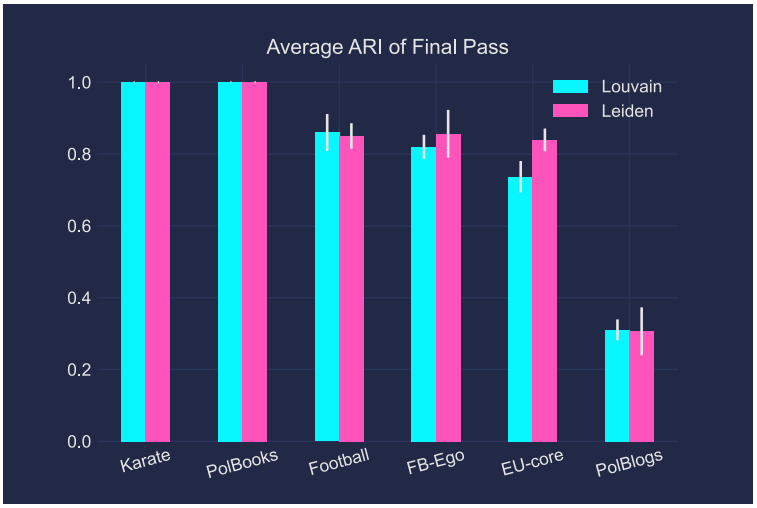

Figure 4.1: Average Adjusted Rand Index for the 6 networks with ground-truth in Table 4.1. Averages are calculated on the final partitions obtained after a stable iteration from 100 instances of Louvain and Leiden. Corresponding average modularity values in Figure 4.4. **Credit:** Author of this dissertation.

In the six ground-truth networks, both algorithms reached a stable iteration within a similar number of passes. In the four other networks Louvain reached a stable iteration in fewer passes than Leiden. In all cases the standard deviation in the number of passes was greater for Leiden. In some cases the average number of communities in the final partition was the same for Louvain and Leiden. However, in the Email EU-Core network and the two collaboration networks we found that Leiden had a tendency to generate more communities. The number of communities in the final partition generated by either method was rarely equal to the number of communities in the ground-truth partition.

The computation times contrasted more starkly. Louvain was faster than Leiden in all networks we tested. The computation time varied much more for Leiden than for Louvain. In the 4 networks without ground-truth and with more nodes, Louvain was

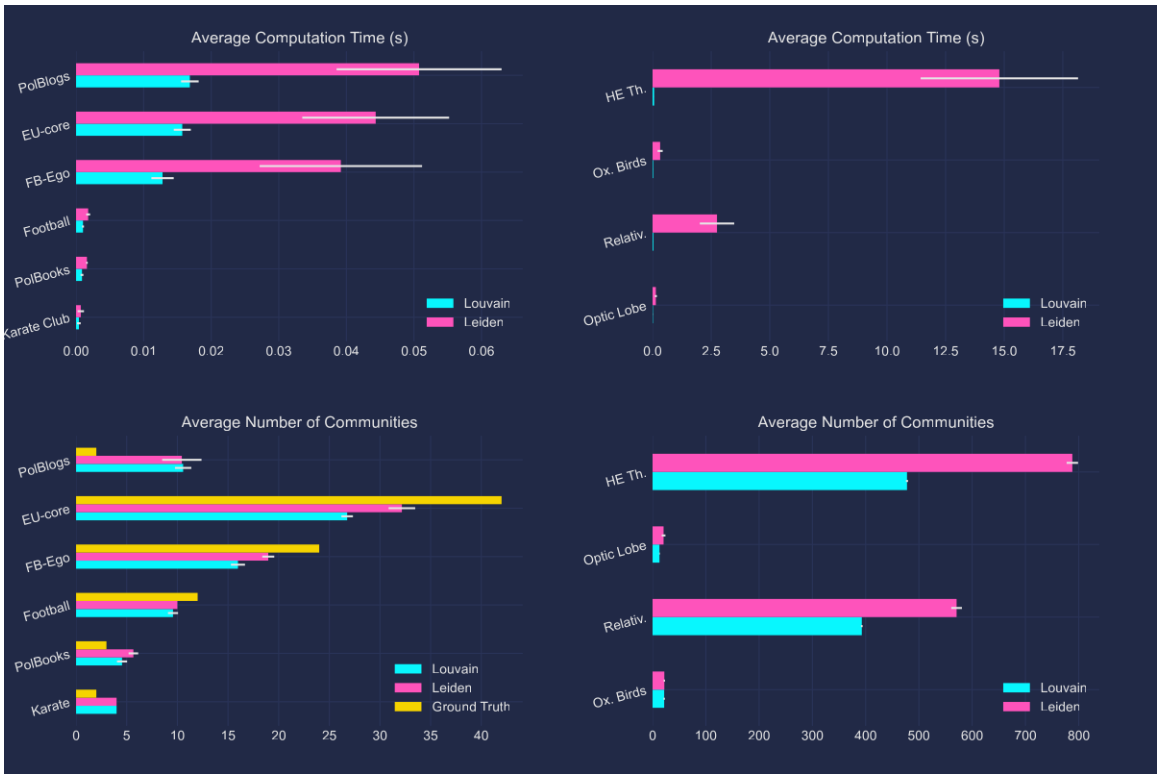

Figure 4.2: **(Left):** The 6 networks with ground-truth. Averages are taken from 100 instances and correspond with partitions in Figure 4.4. **(Right):** The 4 networks without ground-truth, which have more nodes. Averages are taken from 50 instances, and correspond with partitions in Figure 4.5. **(Top):** Average computation times to reach a stable iteration. **(Bottom):** Average number of communities in the final partition. **Credit:** Author of this dissertation.

significantly faster. For example in the High Energy Physics Theory Collaboration network Louvain took on average 0.07 seconds to reach a stable iteration, whereas Leiden took 16.54 seconds.

In most of the networks we found the modularity values of the final partition obtained by both Louvain and Leiden were almost identical. However, note that in terms of modularity, in the General Relativity network Louvain improved on Leiden by 4%, and in the High Energy Physics Theory Collaboration network by 5%.

In most of the networks with ground-truth, the ARI values were mixed, but fairly close for both methods. In half of the networks with ground-truth that we tested, the ARI values of the final partitions were equal. In the Karate Club network and the Political Books network, both Louvain and Leiden returned partitions with perfect ARI (*i.e.* equal to 1). In the Email EU-Core network Leiden improved on Louvain by 14% in terms of ARI.

In general, the ARI was more variable than the modularity, as can be seen in Figure 4.1. The spread of ARI values for the final Leiden-generated partitions was slightly greater than for the final Louvain-generated partitions. Across instances we also widely observed that the ARI was more variable for Leiden than Louvain. For example this can be seen in the E-mail EU-Core network in Figure 4.3. Leiden continues to reassign nodes in order to reach a stable iteration after the first pass and until termination, however in doing so the ARI values vary non-monotonically, even though there is only a marginal increase in modularity across these passes.

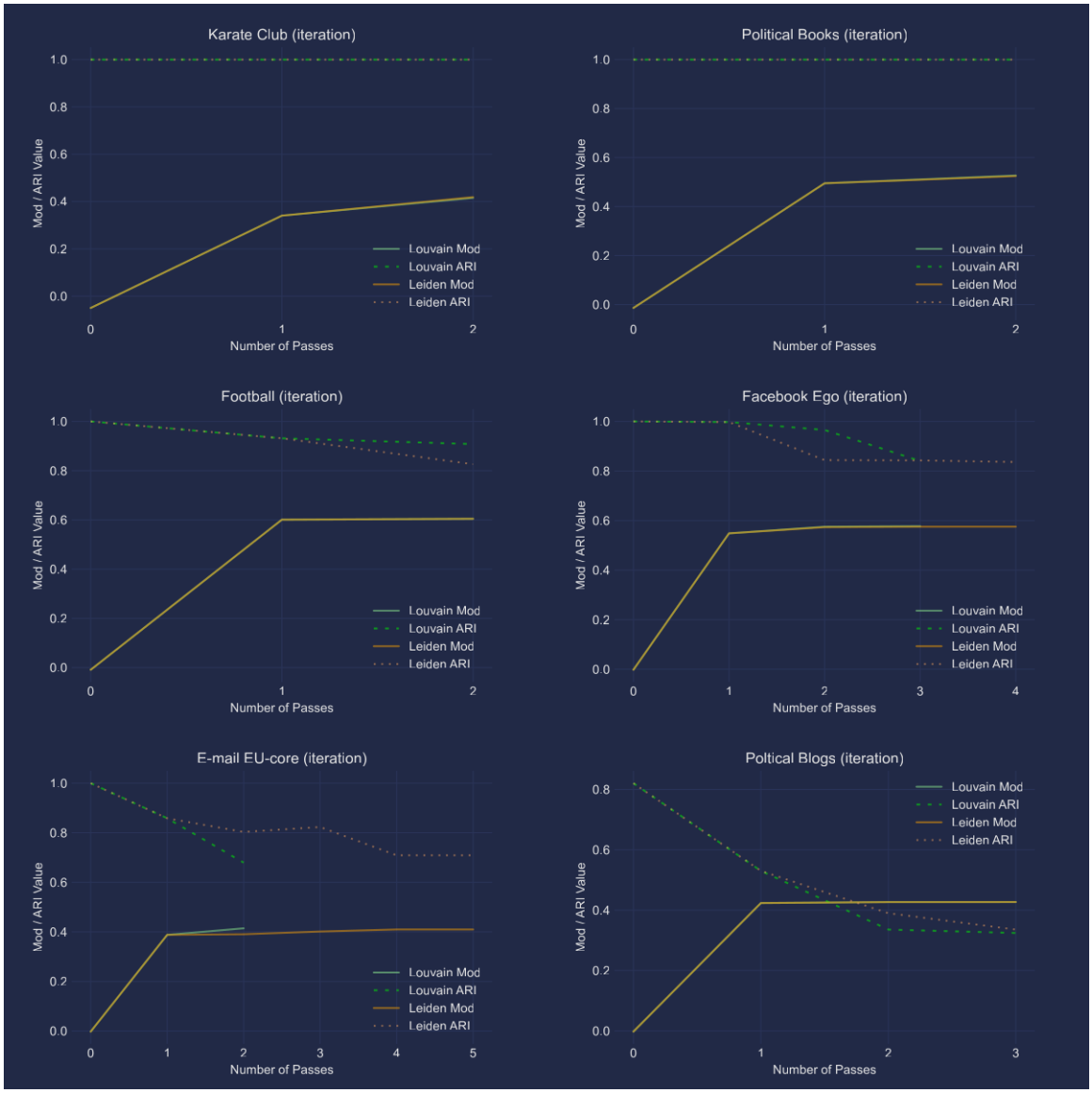

Figure 4.3: Examples which demonstrate interesting behaviour of modularity and ARI in separate instances of Louvain and Leiden using the 6 networks with ground-truth in Table 4.1. Each algorithm was run until a stable iteration was reached. These partitions correspond with those in Figure 5.1. **Credit:** Author of this dissertation.

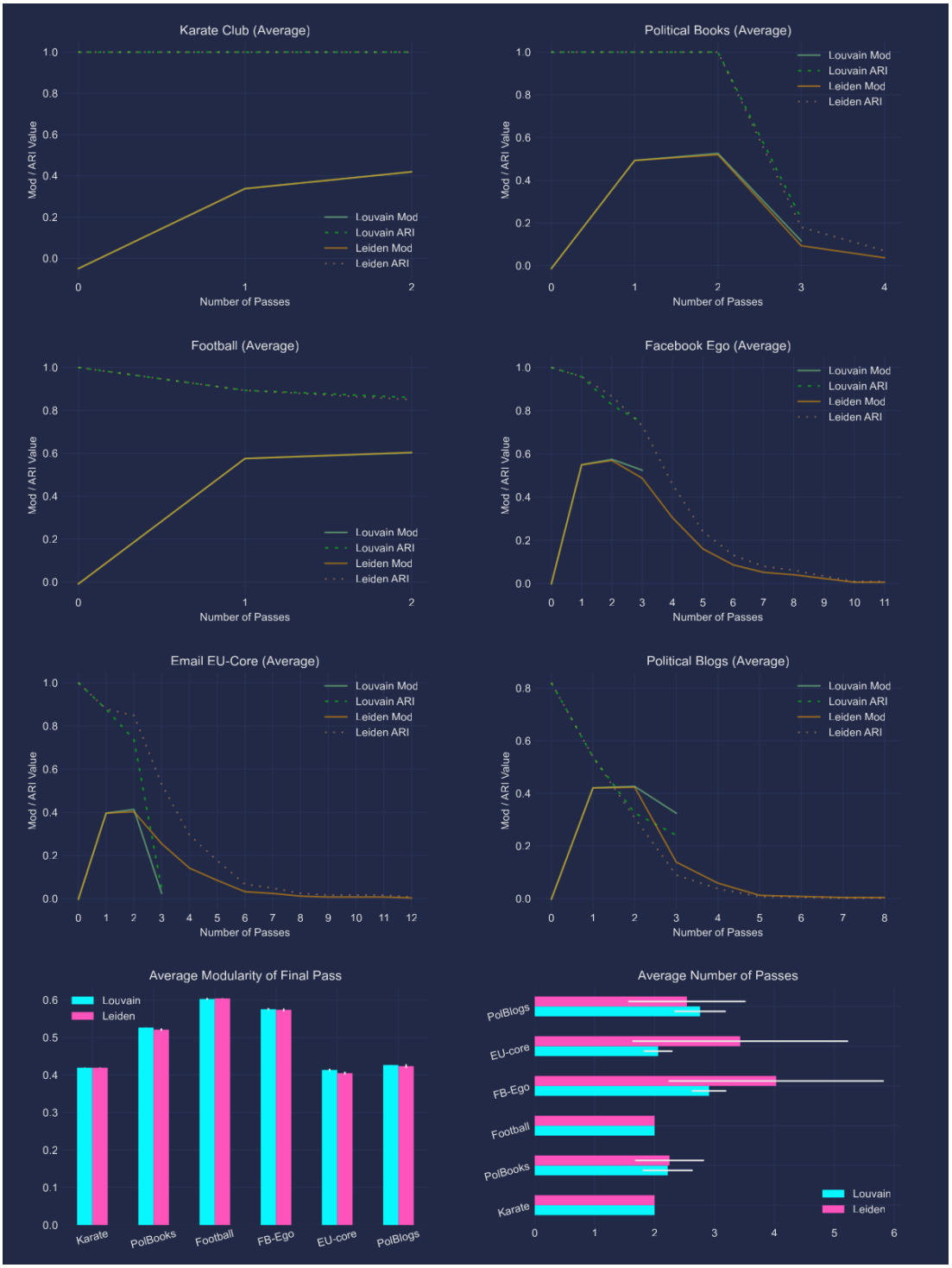

Figure 4.4: One hundred instances of Louvain and Leiden using the 6 networks with ground-truth in Table 4.1. Each algorithm was run until a stable iteration was reached. Computation times in Figure 4.2. **(Top):** Modularity and ARI values are plotted against partitions generated at each pass, and averages are taken at each pass. **(Bottom Left):** The average modularity values of the final 100 partitions generated by each algorithm. The corresponding ARI values are Figure 4.1. **(Bottom Right):** The average number of passes until termination. **Credit:** Author of this dissertation.

Figure 4.5: Fifty instances of Louvain and Leiden using the 4 networks without ground-truth in Table 4.1. Each algorithm was run until a stable iteration was reached. Computation times in Figure 4.2. **(Top):** Modularity values are plotted against partitions generated at each pass, and averages are taken at each pass. **(Bottom Left):** The average modularity values of the final 50 partitions generated by each algorithm. **(Bottom Right):** The average number of passes until termination. **Credit:** Author of this dissertation.

4.4 Discussion

We now comment on our results and exposit ideas for extensions to our experiments. Due to time constraints of this work, these could not be carried out, and may be considered as potential avenues of further study. When looking at our findings it is worth taking into account the practical limitations of our experiments. We note that we were restricted by our available computing power[6], since both Louvain and Leiden amass all requisite storage within the computer memory (*i.e.* RAM).

Before beginning this dissertation, as part of our research, the seminal literature on the subject suggested [103] that Leiden would outperform Louvain in both speed[7] and quality. Therefore, when we began our experiments we expected to find similar results. In terms of speed, we did not find this. Indeed we found the opposite; that Louvain was faster than Leiden in every network we tested.

We observed that this difference in speed became more pronounced in larger networks, as can be seen in Figures 4.4 and 4.5. In the largest network that we tested, the High Energy Physics Theory Collaboration network with 25998 edges, we found that Louvain was far faster than Leiden. In reality this is seen as a fairly small network. To give some perspective, the smallest network tested in [103] was the DBLP Network, which has more than a million edges. In this network, Traag et al found Leiden was faster.

At some point between the size of the networks we tested and the networks tested in [103], it would seem that Leiden overtakes Louvain in terms of speed. We recommend further research to establish where exactly this tipping point is, to what extent it is determined by the number of edges, and whether or not this is consistent across a range of networks in various contexts.

It should be noted that the canonical implementations of Louvain and Leiden are both written in Java. Both algorithms run faster in Java than in Python, owing to complexities in the base languages. Our experiments were conducted using the `LeidenAlg` package [103], written by Traag et al, which exposes Louvain and Leiden to Python. The precise impact on the speed of Leiden when translated from Java to Python is unknown. Therefore, it would be useful to rerun these experiments in

[6] For example in [103], the Web of Science network required over 750 iterations (on average) of Leiden to reach a stable iteration.

[7] For example, in some of the (artificial) benchmark networks tested in [103], Louvain required approximately 2.5 days to terminate, whereas Leiden took fewer than 10 minutes.

Java to see if the difference in computation times between Louvain and Leiden is less marked.

We now discuss partition quality. Leiden-generated partitions vary more than Louvain-generated partitions, since there are more stochastic processes in Leiden. This can be seen by looking at the modularity and ARI values in Figure 4.4. Since Leiden has more opportunities to explore the underlying partition space, we expected Leiden to out-perform Louvain in terms of modularity, Furthermore, as observed by Traag et al [103], Louvain has a tendency to generate partitions which have internally disconnected communities, and Leiden corrects this. In the ground-truth partitions of all the complex networks we tested, none of the communities were disconnected. Therefore, we also expected that Leiden-generated partitions would have higher ARI. However, neither of these expectations were borne out in our experiments, with the exception of the Email EU-Core network.

There was no clear difference between Louvain and Leiden in terms of modularity. In most of the networks, the average modularity of the final partitions for both methods was almost identical, as can be seen in Figure 4.4. The greatest difference in modularity between the two methods was seen in the two collaboration networks in Figure 4.5. In these collaboration networks, Louvain attained higher average modularity than Leiden, and these were also the two networks where Louvain out-performed Leiden in terms of computation time by the largest margin.

Therefore, in regards to modularity, our results contrast with the conclusion of Traag et al in [103], which states "Leiden is strongly preferable to the Louvain algorithm". In fact, looking more closely at this paper [103], it can be seen that that the majority of experiments involved artificial networks, which were invented[8] by Traag et al for the study. In the 6 real-world networks tested in [103], which on average contain millions of nodes, the modularity increase of Leiden over Louvain was marginal (approximately 0.015% on average).

It has been shown [85] that the maximum modularity partition is rarely the same as the ground-truth partition. This has led to ground-truth methods being questioned, and discourse regarding whether or not we should treat ground-truth as a reliable standard [85]. Looking to our own experiments, for example, it can be seen in Figure 4.6 that the ground-truth partition of the Karate Club network has modularity $Q = 0.37$. Louvain and Leiden both return (non-identical) partitions with the

[8] These artificial benchmark networks bear resemblance to the Lancichinetti–Fortunato–Radicchi (LFR) benchmark model [61].

same modularity, $Q = 0.42$. Therefore, Louvain and Leiden identify partitions which are more edge-dense than the ground-truth partition itself.

It has been observed that in real-world networks, there exist many partitions which attain modularities close to the optimum. This is known as the rugged topology of modularity [43, 85], and the coincidence of partitions which are structurally different, but have equal modularity here, is an example of this phenomenon. To our knowledge, no modularity optimisation method has found a partition of the Karate Club which has modularity greater than $Q = 0.42$. However, since modularity optimisation is **NP**-complete [18] it is not possible to decide whether a partition is optimal. Here are two examples [72, 72] of using different modularity optimisation methods which also return modularity $Q = 0.42$.

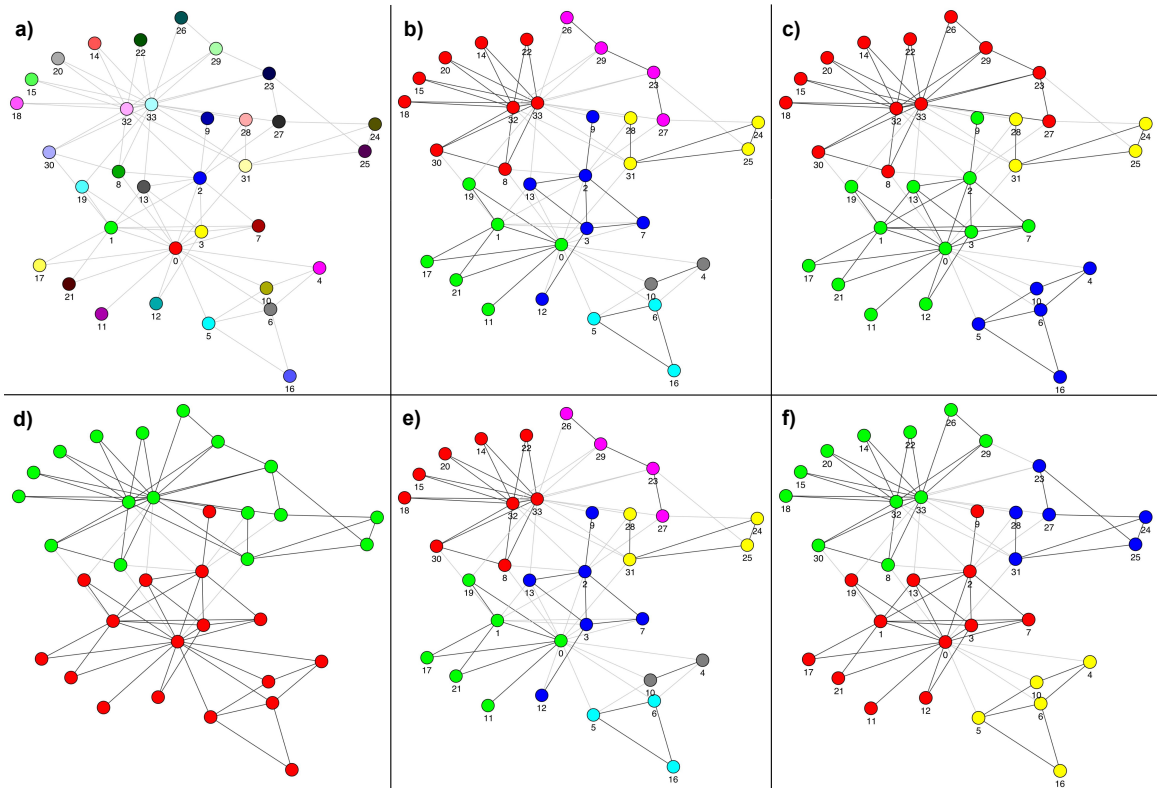

Figure 4.6: Different partitions of the Karate Club Network. Louvain and Leiden were run on this network, and both algorithms converged after two passes. Intra-community edges are black and inter-community edges are grey. Note, these partitions are the same as those generated in Figures 4.4 and 5.1. **a):** $Q = -0.05$. The partition into singleton communities $\mathcal{P}_v$. This is the partition on initialisation of both Louvain and Leiden. **b):** $Q = 0.34$. Louvain, pass 1. **c):** $Q = 0.42$. Louvain, pass 2. **d):** $Q = 0.37$. The ground-truth community partition. **e):** $Q = 0.34$. Leiden, pass 1. **f):** $Q = 0.42$. Leiden, pass 2. **Credit:** Author of this dissertation.

After some preliminary experiments, we also report that when the Constant Potts Model [101] was used with a resolution parameter of $\gamma = 0.05$, Leiden was able to reliably recover the ground-truth partition of the Karate Club network. Therefore, it would be worthwhile to carry out further experiments which investigate the effects of varying the resolution parameter, or where alternate formulations of modularity (*e.g.* [101, 102]) are used as the quality function instead to see if the accuracy of either Louvain or Leiden can be improved (with respect to ground-truth).

Despite the fact that Louvain and Leiden returned partitions with similar modularity values, upon closer inspection, we discovered that these partitions are structurally quite different. For example, this can be seen in Figure 4.6. Using either method, the final partition contained 4 communities, and the ARI of each of these partitions is equal to 1. However, there are only 2 communities in the ground-truth partition. Therefore, we see that the ARI values are somewhat inflated because there is no inherent mechanism to penalise nested communities. This has been observed as a wider trend with modularity-based optimisation methods. In networks which have hierarchical community structures there is a tendency to aggregate communities nearer the top or bottom of this hierarchy [48]. We also observed this in the case of the Political Books network, where on average Louvain-generated partitions contained 4.5 communities and Leiden-generated partitions contained 5.5, even though the ground-truth partition contains only 3 communities.

In addition to the ARI, we decided it would be useful to include other external evaluation measures of partition quality. Meila recently conducted a survey of different measures in [73]. Therefore, in the 6 networks with ground-truth we also calculated the Fowlkes-Mallows index (FMI) [38] and the Normalised Mutual Information (NMI) [105] for each partition generated by Louvain and Leiden. These results are displayed in Figure 5.1 and Figure 5.2. It can be seen in Figure 5.2 that for both Louvain and Leiden the FMI and NMI were usually coincident[9] in each network. We note that the FMI was more variable than the NMI. We found in almost all cases both indices indicated that the final Louvain-generated and Leiden-generated partitions were of similar quality. The trends in FMI, NMI and ARI were all in relative agreement.

[9] However this was not always the case, as can be seen in Figure 5.1.

4.5 Conclusion

Our results challenge the status quo [103] that Leiden is a superior community detection method to Louvain. One way to approach the community detection problem is by using modularity optimisation [80], and both Louvain [15] and Leiden [103] are heuristics which optimise modularity.

In the theoretical component of this dissertation we explained the community detection problem, we defined modularity, and we explained why high modularity values correspond to edge-dense community structures. We then examined the algorithms behind Louvain and Leiden in detail and specified their operations. Speed and quality are the two chief aims when designing a community detection algorithm. In the applied component of this dissertation we conducted our own original numerical simulations to compare the speed and quality of Louvain with Leiden.

We ran Louvain and Leiden on 10 real-world complex networks (Table 4.1) from complex systems belonging to a variety of different contexts (section 4.1). Our experiments focused on smaller complex networks than those used in [103]. Of our 10 complex networks, 6 had ground-truth communities metadata. The other 4 networks had no ground-truth, but had more nodes. In our experiments we ran 100 instances of Louvain and Leiden on the 6 networks with ground-truth, and 50 instances of Louvain and Leiden on the 4 networks with no ground-truth. Our results can be seen in Figures 4.1, 4.2, 4.3, 4.4, 4.5 and 4.6. Our research was the first, to our knowledge, which evaluated Leiden using ground-truth communities meta-data.

Our most surprising findings were in regards to computation time. Our results do not support the assertion of Traag et al in [103] that Leiden is always an improvement over Louvain in terms of speed. We found that Louvain was consistently faster than Leiden (Figure 4.2). In the networks with more nodes, we saw this difference between Louvain and Leiden become even more marked. This pattern could be partially accounted for by the fact that the 10 real-world networks used in our experiments were significantly smaller than the 6 real-world networks used by Traag et al in [103]. We also acknowledge that our experiments were conducted in Python[10]. Both algorithms were originally implemented in Java, which is a faster language than Python.

[10] In our experiments, we used versions of Louvain and Leiden which are exposed to Python via the `LeidenAlg` package, available at https://github.com/vtraag/leidenalg. This package was written by Traag et al, and was published at the same time as Leiden [103]. We used version 0.9.0, which was released on Oct 3, 2022.

To assess the quality of each partition we used modularity [79] and the Adjusted Rand Index (ARI) [51] as the comparative measures. In most cases identical modularity values were returned by both algorithms. The ARI was more variable than modularity, and whilst the ARI scores were generally high for both methods, the ARI scores did not always agree: sometimes Leiden performed better, and sometimes Louvain. The purview of our experiments is restricted to a limited number of small networks. Therefore we recommend further research using a larger sample consisting of networks with ground-truth and of varying size. This would establish which method returns partitions of higher quality, and at what point Leiden overtakes Louvain in terms of speed[11], if that is indeed the case.

[11] As Traag et al claim in [103].

Additional Figures

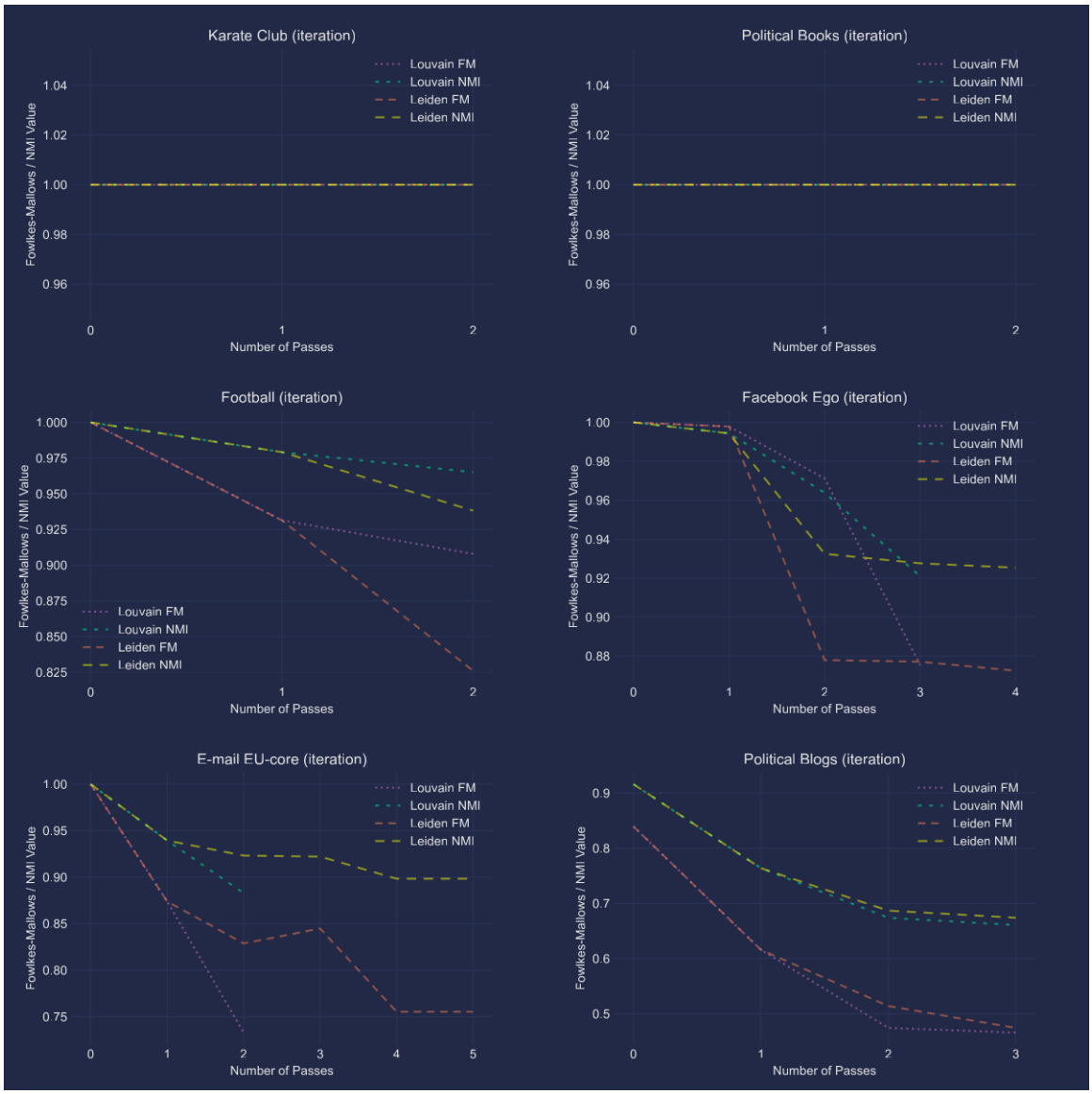

Figure 5.1: Examples which demonstrate interesting behaviour of the Fowlkes-Mallows index [38] and Normalised Mutual Information [105] in separate instances of Louvain and Leiden using the 6 networks with ground-truth in Table 4.1. Each algorithm was run until a stable iteration was reached. These partitions correspond with those in Figure 4.3. **Credit:** Author of this dissertation.

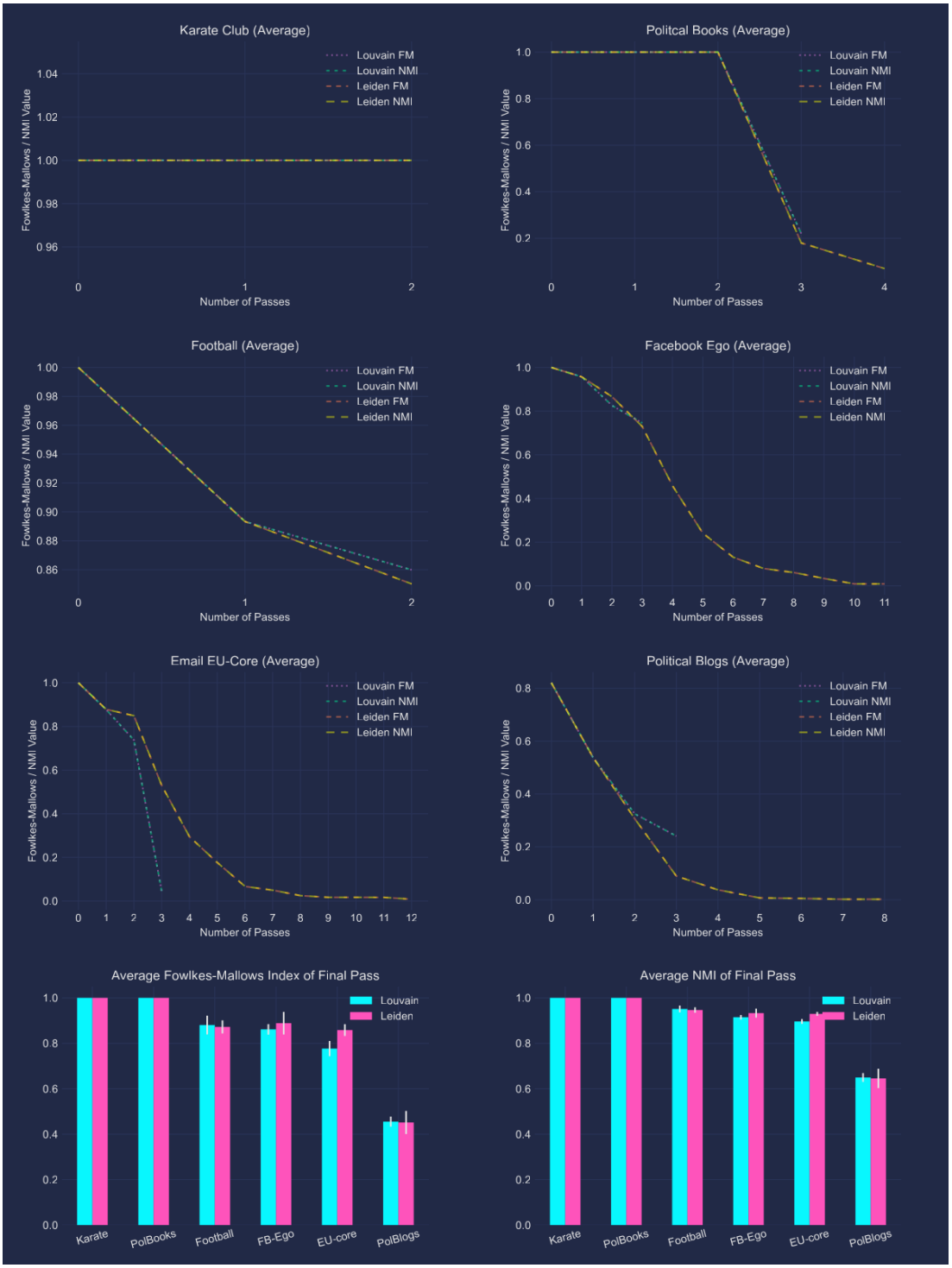

Figure 5.2: One hundred instances of Louvain and Leiden using the 6 networks with ground-truth in Table 4.1. Each algorithm was run until a stable iteration was reached. **(Top):** Fowlkes-Mallows [38] and Normalised Mutual Information [105] values are plotted against partitions generated at each pass, and averages are taken at each pass. Corresponding average modularity and ARI values in Figure 4.4. **(Bottom):** The average Fowlkes-Mallows and Normalised Mutual Information of the final 100 partitions generated by each algorithm. Corresponding final pass ARI values in Figure 4.1. **Credit:** Author of this dissertation.

Bibliography

[1] L. A. Adamic and N. Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[2] N. Aghanim, Y. Akrami, M. Ashdown, J. Aumont, C. Baccigalupi, M. Ballardini, A. Banday, R. Barreiro, N. Bartolo, S. Basak, et al. Planck 2018 results-vi. cosmological parameters. *Astronomy & Astrophysics*, 641:A6, 2020.

[3] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(1):47, 1 2002. ISSN 00346861. doi: 10.1103/RevModPhys.74.47. URL https://journals.aps.org/rmp/abstract/10.1103/RevModPhys.74.47.

[4] R. Aldecoa and I. Marín. Exploring the limits of community detection strategies in complex networks. *Scientific Reports*, 3, 2013. ISSN 20452322. doi: 10.1038/SREP02216. URL https://www.nature.com/articles/srep02216.

[5] D. Aloise, G. Caporossi, P. Hansen, L. Liberti, S. Perron, and M. Ruiz. Modularity Maximization in Networks by Variable Neighborhood Search.

[6] K. Amunts, C. Lepage, L. Borgeat, H. Mohlberg, T. Dickscheid, M.-É. Rousseau, S. Bludau, P.-L. Bazin, L. B. Lewis, A.-M. Oros-Peusquens, et al. Bigbrain: an ultrahigh-resolution 3d human brain model. *Science*, 340(6139): 1472–1475, 2013.

[7] A. Arenas, J. Duch, A. Fernández, and S. Gómez. Size reduction of complex networks preserving modularity. *New Journal of Physics*, 9(6):176, 2007.

[8] I. Artico, I. Smolyarenko, V. Vinciotti, and E. Wit. How rare are power-law networks really? *Proceedings of the Royal Society A*, 476(2241):20190742, 2020.

[9] S. H. Bae, D. Halperin, J. D. West, M. Rosvall, and B. Howe. Scalable and efficient flow-based community detection for large-scale graph analysis. *ACM Transactions on Knowledge Discovery from Data*, 11(3):32, 3 2017. ISSN 1556472X. doi: 10.1145/2992785. URL http://dx.doi.org/10.1145/2992785.

[10] A.-L. Barabási and R. Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[11] A.-L. Barabási. Love is all you need: Clauset's fruitless search for scale-free networks, Mar 2018. URL goo.gl/bPQv3N.

[12] A.-L. Barabási and M. Pósfai. *Network science*. Cambridge University Press, Cambridge, 2016. ISBN 9781107076266 1107076269. URL http://barabasi.com/networksciencebook/.

[13] J. W. Berry, B. Hendrickson, R. A. Laviolette, and C. A. Phillips. Tolerating the community detection resolution limit with edge weighting. *Physical Review E*, 83(5):056119, 5 2011. ISSN 15393755. doi: 10.1103/PhysRevE.83.056119. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.83.056119.

[14] V. Blondel. Louvain method for community detection. URL https://perso.uclouvain.be/vincent.blondel/research/louvain.html.

[15] V. D. 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), 3 2008. ISSN 17425468. doi: 10.1088/1742-5468/2008/10/p10008. URL https://iopscience.iop.org/article/10.1088/1742-5468/2008/10/P10008.

[16] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2 2006. ISSN 0370-1573. doi: 10.1016/J.PHYSREP.2005.10.009.

[17] U. Brandes, D. Delling, M. Gaertler, R. Goerke, M. Hoefer, Z. Nikoloski, and D. Wagner. Maximizing Modularity is hard. 8 2006. doi: 10.48550/arxiv.physics/0608255. URL https://arxiv.org/abs/physics/0608255v2.

[18] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity-np-completeness and beyond. *ITI Wagner, Faculty of Informatics, Universität Karlsruhe (TH), Tech. Rep*, 19:2006, 2006.

[19] U. Brandes, D. Delling, M. Gaertler, R. Görke, M. Hoefer, Z. Nikoloski, and D. Wagner. On finding graph clusterings with maximum modularity. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 121–132. Springer, 2007.

[20] A. C. M. Brito, F. N. Silva, and D. R. Amancio. A complex network approach to political analysis: application to the Brazilian Chamber of Deputies. 9 2019. doi: 10.1371/journal.pone.0229928. URL http://arxiv.org/abs/1909.02346http://dx.doi.org/10.1371/journal.pone.0229928.

[21] A. D. Broido and A. Clauset. Scale-free networks are rare. *Nature Communications 2019 10:1*, 10(1):1–10, 3 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-08746-5. URL https://www.nature.com/articles/s41467-019-08746-5.

[22] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. *Algorithm engineering*, pages 117–158, 2016.

[23] J. E. Chacón and A. I. Rastrojo. Minimum adjusted rand index for two clusterings of a given size. *Advances in Data Analysis and Classification*, pages 1–9, 2022.

[24] S. Chen, Z. Z. Wang, L. Tang, Y. N. Tang, Y. Y. Gao, H. J. Li, J. Xiang, and Y. Zhang. Global vs local modularity for network community detection. *PloS one*, 13(10):e0205284, 10 2018. ISSN 1932-6203. URL https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0205284.

[25] T. Chen, P. Singh, and K. E. Bassler. Network community detection using modularity density measures. *Journal of Statistical Mechanics: Theory and Experiment*, 2018(5):053406, 2018.

[26] H. Cherifi, G. Palla, B. K. Szymanski, and X. Lu. On community structure in complex networks: challenges and opportunities. *Applied Network Science*, 4 (1), 8 2019. doi: 10.1007/s41109-019-0238-9. URL http://arxiv.org/abs/1908.04901http://dx.doi.org/10.1007/s41109-019-0238-9.

[27] A. Clauset, M. E. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 70(6):6, 12 2004. ISSN 1063651X. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.70.066111.

[28] G. Csardi and T. Nepusz. The igraph software package for complex network research. *InterJournal*, Complex Systems:1695, 2006. URL https://igraph.org.

[29] L. Donetti and M. A. Muñoz. Detecting Network Communities: a new systematic and efficient algorithm. *Journal of Statistical Mechanics: Theory and Experiment*, (10), 4 2004. ISSN 17425468. doi: 10.1088/1742-5468/2004/10/p10012. URL https://arxiv.org/abs/cond-mat/0404652v2.

[30] S. Emmons, S. Kobourov, M. Gallant, and K. Börner. Analysis of network clustering algorithms and cluster quality metrics at scale. *PloS one*, 11(7): e0159161, 2016.

[31] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19):7663–7668, 2011. URL https://www.pnas.org/doi/full/10.1073/pnas.1018962108.

[32] S. Faridizadeh, N. Abdolvand, and S. Rajaee Harandi. *Market Basket Analysis Using Community Detection Approach: A Real Case*, pages 177–198. Springer International Publishing, Cham, 2018. ISBN 978-3-319-95810-1. doi: 10.1007/978-3-319-95810-1_13. URL https://doi.org/10.1007/978-3-319-95810-1_13.

[33] J. A. Firth and B. C. Sheldon. Experimental manipulation of avian social structure reveals segregation is carried over across contexts. *Proceedings of the Royal Society B: Biological Sciences*, 282(1802):20142350, 2015.

[34] A. Flache and M. W. Macy. Small worlds and cultural polarization. *The Journal of Mathematical Sociology*, 35(1-3):146–176, 2011.

[35] S. Fortunato. Community detection in graphs. 6 2009. doi: 10.1016/j.physrep.2009.11.002. URL http://arxiv.org/abs/0906.0612http://dx.doi.org/10.1016/j.physrep.2009.11.002.

[36] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences of the United States of America*, 104(1):36–41, 1 2007. ISSN 00278424. URL https://www.pnas.org/doi/10.1073/pnas.0605965104.

[37] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 7 2016. doi: 10.1016/j.physrep.2016.09.002. URL http://arxiv.org/abs/1608.00163http://dx.doi.org/10.1016/j.physrep.2016.09.002.

[38] E. B. Fowlkes and C. L. Mallows. A method for comparing two hierarchical clusterings. *Journal of the American statistical association*, 78(383):553–569, 1983.

[39] M. Gaertler, R. Görke, and D. Wagner. Significance-driven graph clustering. In *International Conference on Algorithmic Applications in Management*, pages 11–26. Springer, 2007.

[40] R. George, K. Shujaee, M. Kerwat, Z. Felfli, D. Gelenbe, and K. Ukuwu. A comparative evaluation of community detection algorithms in social networks. *Procedia Computer Science*, 171:1157–1165, 2020.

[41] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, 6 2002. ISSN 00278424.

[42] F. B. Gonzaga, V. C. Barbosa, and G. B. Xexéo. The network structure of mathematical knowledge according to the Wikipedia, MathWorld, and DLMF online libraries. *Network Science*, 2(3):367–386, 12 2012. doi: 10.1017/nws.2014.20. URL http://arxiv.org/abs/1212.3536http://dx.doi.org/10.1017/nws.2014.20.

[43] B. H. Good, Y.-A. De Montjoye, and A. Clauset. Performance of modularity maximization in practical contexts. *Physical review E*, 81(4):046106, 2010.

[44] M. Grandjean. La connaissance est un réseau. Perspective sur l'organisation archivistique et encyclopédique. *Les cahiers du numérique*, 10(3):37–54, 9 2014. ISSN 14693380. doi: 10.3166/LCN.10.3.37-54.

[45] R. Guimera and L. A. Nunes Amaral. Functional cartography of complex metabolic networks. *nature*, 433(7028):895–900, 2005.

[46] J. Guo, P. Singh, and K. E. Bassler. Resolution limit revisited: community detection using generalized modularity density. *arXiv*, 12 2020. doi: 10.48550/arxiv.2012.14543. URL https://arxiv.org/abs/2012.14543v1.

[47] W. Guo, R. Chen, Y.-C. Chen, and A. G. Banerjee. Efficient Community Detection in Large-Scale Dynamic Networks Using Topological Data Analysis. 4 2022. doi: 10.48550/arxiv.2204.03191. URL https://arxiv.org/abs/2204.03191v1.

[48] J. Han, W. Li, and W. Deng. Multi-resolution community detection in massive networks. *Scientific reports*, 6(1):1–12, 2016.

[49] L. H. Hartwell, J. J. Hopfield, S. Leibler, and A. W. Murray. From molecular to modular cell biology. *Nature 1999 402:6761*, 402(6761):C47–C52, 12 1999. ISSN 1476-4687. doi: 10.1038/35011540. URL https://www.nature.com/articles/35011540.

[50] P. Holme. Rare and everywhere: Perspectives on scale-free networks. *Nature communications*, 10(1):1–3, 2019.

[51] L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1): 193–218, 1985.

[52] M. Jebabli, H. Cherifi, C. Cherifi, and A. Hamouda. Overlapping community detection versus ground-truth in amazon co-purchasing network. In *2015 11th international conference on signal-image technology & internet-based systems (SITIS)*, pages 328–336. IEEE, 2015.

[53] M. Jebabli, H. Cherifi, C. Cherifi, and A. Hamouda. Community detection algorithm evaluation with ground-truth data. *Physica a: Statistical mechanics and its applications*, 492:651–706, 2018.

[54] H. Koch. The theory of partitions (encyclopedia of mathematics and its applications). london-amsterdam-don mills-sydney-tokyo, addison-wesley publ. company 1976. *Zeitschrift Angewandte Mathematik und Mechanik*, 59(6):285–285, 1979.

[55] V. Krebs. A network of books about recent us politics sold by the online bookseller amazon.com. *Unpublished*, 2008. URL http://www.orgnet.com.

[56] J. M. Kumpula, J. Saramäki, K. Kaski, and J. Kertész. Limited resolution in complex network community detection with potts model approach. *The European Physical Journal B*, 56(1):41–45, 2007.

[57] V. Labatut. Generalised measures for the evaluation of community detection methods. *International Journal of Social Network Mining*, 2(1):44–63, 2015.

[58] R. Lambiotte and M. T. Schaub. *Modularity and Dynamics on Complex Networks*. Elements in Structure and Dynamics of Complex Networks. Cambridge University Press, 2022. doi: 10.1017/9781108774116. URL https://doi.org/10.1017/9781108774116.

[59] A. Lancichinetti and S. Fortunato. Community detection algorithms: A comparative analysis. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 80(5), 11 2009. ISSN 15393755. doi: 10.1103/PhysRevE.80.056117.

[60] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 84(6):066122, 12 2011. ISSN 15393755. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.84.066122.

[61] A. Lancichinetti, S. Fortunato, and F. Radicchi. Benchmark graphs for testing community detection algorithms. *Physical review E*, 78(4):046110, 2008.

[62] J. Leskovec and J. Mcauley. Learning to discover social circles in ego networks. *Advances in neural information processing systems*, 25, 2012.

[63] J. Leskovec and R. Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016.

[64] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1(1):2–es, 2007.

[65] L. Lovász. *Combinatorial Problems and Exercises, Second Edition*, volume 361. American Mathematical Soc., 1993. ISBN 9780444815040.

[66] H. Lu, A. Nayak, and S. Member. A General Definition of Network Communities and the Corresponding Detection Algorithm.

[67] X. Lu, B. Cross, and B. K. Szymanski. Asymptotic resolution bounds of generalized modularity and multi-scale community detection. *Information Sciences*, 525:54–66, 7 2020. ISSN 0020-0255. doi: 10.1016/J.INS.2020.03.082.

[68] T. Luczak. Proceedings of the Symposium on Random Graphs,. *Poznań 1989*, pages 165–182, 1992.

[69] M. Luecken. *Application of multi-resolution partitioning of interaction networks to the study of complex disease*. PhD thesis, University of Oxford, 2016.

[70] R. Martin, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal-Special Topics*, 178(1):13–23, 2009.

[71] C. P. Massen and J. P. Doye. Thermodynamics of community structure. *arXiv preprint cond-mat/0610077*, 2006.

[72] A. Medus, G. Acuna, and C. O. Dorso. Detection of community structures in networks via global optimization. *Physica A: Statistical Mechanics and its Applications*, 358(2-4):593–604, 2005.

[73] M. Meila. Criteria for comparing clusterings. *Handbook of cluster analysis*, pages 619–635, 2016.

[74] G. W. Milligan and M. C. Cooper. A study of the comparability of external criteria for hierarchical cluster analysis. *Multivariate behavioral research*, 21(4): 441–458, 1986.

[75] M. Molloy and B. Reed. A critical point for random graphs with a given degree sequence. *Random structures & algorithms*, 6(2-3):161–180, 1995.

[76] L. C. Morey and A. Agresti. The measurement of classification agreement: An adjustment to the rand statistic for chance agreement. *Educational and Psychological Measurement*, 44(1):33–37, 1984.

[77] S. Muff, F. Rao, and A. Caflisch. Local modularity measure for network clusterizations. *Physical Review E*, 72(5):056107, 2005.

[78] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010. ISBN 9780199206650.

[79] M. E. Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

[80] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2 2004. ISSN 1063651X. doi: 10.1103/PhysRevE.69.026113. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.69.026113.

[81] C. C. Ni, Y. Y. Lin, F. Luo, and J. Gao. Community Detection on Networks with Ricci Flow. *Scientific Reports*, 9(1), 7 2019. ISSN 20452322. doi: 10.48550/arxiv.1907.03993. URL https://arxiv.org/abs/1907.03993v1.

[82] A. Noack. Modularity clustering is force-directed layout. *Physical Review E*, 79 (2):026102, 2009.

[83] N. Ozaki, H. Tezuka, and M. Inaba. A Simple Acceleration Method for the Louvain Algorithm. *International Journal of Computer and Electrical Engineering*, 8(3):207–218, 2016. ISSN 17938163. doi: 10.17706/IJCEE.2016.8.3.207-218.

[84] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[85] L. Peel, D. B. Larremore, and A. Clauset. The ground truth about metadata and community detection in networks. *Science advances*, 3(5):e1602548, 2017.

[86] G. Pólya and G. Szegö. Problems and Theorems in Analysis I. *Problems and Theorems in Analysis I*, 1972. doi: 10.1007/978-3-642-61983-0.

[87] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Paris. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–2663, 3 2004. ISSN 00278424.

[88] U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3): 036106, 2007.

[89] S. Rahiminejad, M. R. Maurya, and S. Subramaniam. Topological and functional comparison of community detection algorithms in biological networks. *BMC bioinformatics*, 20(1):1–25, 2019.

[90] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[91] E. Ravasz, A. L. Somera, D. A. Mongru, Z. N. Oltvai, and A. L. Barabási. Hierarchical organization of modularity in metabolic networks. *Science (New York, N.Y.)*, 297(5586):1551–1555, 8 2002. ISSN 1095-9203. doi: 10.1126/SCIENCE.1073374. URL https://pubmed.ncbi.nlm.nih.gov/12202830/.

[92] J. Reichardt and S. Bornholdt. When are networks truly modular? *Physica D: Nonlinear Phenomena*, 224(1-2):20–26, 12 2006. ISSN 0167-2789. doi: 10.1016/J.PHYSD.2006.09.009.

[93] J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical review E*, 74(1):016110, 2006.

[94] G. Robins, P. Pattison, and J. Woolcock. Small and other worlds: Global network structures from local processes. *American Journal of Sociology*, 110 (4):894–936, 2005.

[95] R. A. Rossi and N. K. Ahmed. The network data repository with interactive graph analytics and visualization. In *AAAI*, 2015. URL https://networkrepository.com.

[96] M. Rosvall and C. T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the national academy of sciences*, 105(4):1118–1123, 2008.

[97] J. M. Santos and M. Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pages 175–184. Springer, 2009.

[98] C. Stadtfeld, K. Takács, and A. Vörös. The Emergence and Stability of Groups in Social Networks. *Social Networks*, 60:129–145, 1 2020. ISSN 0378-8733. doi: 10.1016/J.SOCNET.2019.10.008.

[99] D. Steinley. Properties of the hubert-arable adjusted rand index. *Psychological methods*, 9(3):386, 2004.

[100] V. A. Traag. Faster unfolding of communities: Speeding up the Louvain algorithm. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics*, 92

(3):032801, 9 2015. ISSN 15502376. URL https://journals.aps.org/pre/abstract/10.1103/PhysRevE.92.032801.

[101] V. A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

[102] V. A. Traag, R. Aldecoa, and J.-C. Delvenne. Detecting communities using asymptotical surprise. *Physical Review E*, 92(2):022816, 2015.

[103] V. A. Traag, L. Waltman, and N. J. van Eck. From Louvain to Leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1), 12 2019. ISSN 20452322. doi: 10.1038/s41598-019-41695-z. URL https://www.nature.com/articles/s41598-019-41695-z.

[104] A. L. Traud, C. Frost, P. J. Mucha, and M. A. Porter. Visualization of communities in networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 19(4):041104, 2009.

[105] N. X. Vinh, J. Epps, and J. Bailey. Information theoretic measures for clusterings comparison: is a correction for chance necessary? In *Proceedings of the 26th annual international conference on machine learning*, pages 1073–1080, 2009.

[106] L. Waltman and N. J. Van Eck. A smart local moving algorithm for large-scale modularity-based community detection. *The European Physical Journal B 2013 86:11*, 86(11):1–14, 11 2013. ISSN 1434-6036. doi: 10.1140/EPJB/E2013-40829-0. URL https://link.springer.com/article/10.1140/epjb/e2013-40829-0.

[107] D. J. Watts. Networks, dynamics, and the small-world phenomenon. *American Journal of sociology*, 105(2):493–527, 1999.

[108] S. White and P. Smyth. A spectral clustering approach to finding communities in graphs. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 274–285. SIAM, 2005.

[109] F. A. Wolf, F. K. Hamey, M. Plass, J. Solana, J. S. Dahlin, B. Göttgens, N. Rajewsky, L. Simon, and F. J. Theis. Paga: graph abstraction reconciles clustering with trajectory inference through a topology preserving map of single cells. *Genome biology*, 20(1):1–9, 2019.

[110] Z. Wu, T. Li, and R. Roman. Analyzing Wikipedia Membership Dataset and PredictingUnconnected Nodes in the Signed Networks. 10 2021. doi: 10.48550/arxiv.2110.09111. URL https://arxiv.org/abs/2110.09111v1.

[111] Z. Yang, R. Algesheimer, and C. J. Tessone. A Comparative Analysis of Community Detection Algorithms on Artificial Networks. *Scientific Reports 2016 6:1*, 6(1):1–18, 8 2016. ISSN 2045-2322. doi: 10.1038/srep30750. URL https://www.nature.com/articles/srep30750.

[112] K. Y. Yeung and W. L. Ruzzo. Details of the adjusted rand index and clustering algorithms, supplement to the paper an empirical study on principal component analysis for clustering gene expression data. *Bioinformatics*, 17(9):763–774, 2001.

[113] W. W. Zachary. An Information Flow Model for Conflict and Fission in Small Groups. *Source: Journal of Anthropological Research*, 33(4):452–473, 1977.

[114] E. Ziv, M. Middendorf, and C. H. Wiggins. Information-theoretic approach to network modularity. *Physical Review E*, 71(4):046117, 2005.