ISSN 1855-3966 (printed edn.), ISSN 1855-3974 (electronic edn.)
ARS MATHEMATICA CONTEMPORANEA 18 (2020) 33–49
https://doi.org/10.26493/1855-3974.2079.7b1
(Also available at http://amc-journal.eu)

Clustering via the modified Petford-Welsh algorithm

Barbara Ikica [*]

Faculty of Mathematics and Physics, University of Ljubljana, and
Institute of Mathematics, Physics, and Mechanics, 1000 Ljubljana, Slovenia

Received 9 August 2019, accepted 21 September 2019, published online 8 June 2020

Abstract

Detecting meaningful communities has become crucial to advance our knowledge in diverse research areas that deal with datasets which can be naturally represented as networks. By regarding vertex clustering as the opposite problem of vertex colouring, we were able to leverage the Petford-Welsh colouring algorithm to develop a fast, efficient, and highly-scalable decentralised clustering algorithm. Its greatest potential lies in outperforming conventional methods when the community structure is fairly clear-cut.

Keywords: Graph algorithm, community detection, the Petford-Welsh algorithm, simulated annealing, complex networks.

Math. Subj. Class. (2020): 05C85, 05C15, 91C20

1 Introduction

Over the past few decades, deploying complex networks to represent and analyse vast amounts of data has become prevalent across various disciplines [4, 21, 30]. Being able to uncover a network's community structure has turned out to be an invaluable tool when trying to shed light on its intricate organisation and functioning [11].

However, to date we have not even gained an insight into what constitutes a satisfactory clustering solution, let alone developed an approach that would have proven to be universally applicable and scalable to an ever-increasing abundance of data. As a matter of fact, according to the *"No Free Lunch Theorem"* for community detection, *"there can be no algorithm that is optimal for all possible community detection tasks"* [22]. As a consequence,

[*]The author is grateful for valuable discussions with Janez Povh, Janez Žerovnik, Matjaž Konvalinka, and Matjaž Perc. This research was supported by the Slovenian Research Agency projects N1-0057, N1-0071, and J1-8155.

E-mail address: barbara.ikica@fmf.uni-lj.si (Barbara Ikica)

this has led to a number of clustering algorithms, stemming from different aspects, a variety of objective functions, stand-alone quality metrics, and information recovery metrics [10].

In this paper, we introduce a clustering algorithm motivated by a heuristic approach to vertex colouring proposed by Petford and Welsh [23, 37]. Our method was founded on the observation that finding an appropriate vertex clustering is, in essence, dual to the problem of finding a proper vertex colouring. Indeed, local proliferation of the colour of a vertex should, in principle, lead to densely interconnected monochromatic neighbourhoods. By associating colours with clusters one thus attains a clustering solution that captures the very fundamental intuition behind community detection – a division into tightly-knit groups, loosely connected to one another. Hence, the terms *colour(ing)* and *cluster(ing)* will be used interchangeably.

Due to its heuristic nature, the resulting algorithm typically runs in linear time $\mathcal{O}(|E|)$ in the number of edges, which makes it computationally less demanding compared to other widely used methods, and, consequently, highly scalable. Moreover, as it only leverages local knowledge about the network, it does not require specifying any objective function to be optimised. Nevertheless, as there is a general lack of consensus regarding both validity and quality of clustering solutions, it may be beneficial to define an application-specific quality measure in order to filter the resulting solutions based on desirable properties. These can also be accommodated by varying the only prerequisite parameter of the algorithm, the weight parameter ω that enables tuning the degree of randomness, and thus renders the algorithm more widely applicable.

2 Method

It should be noted that our algorithm can be applied to any kind of graphs – be it directed or undirected, connected or disconnected, weighted or unweighted. Regardless, for simplicity, let us assume that we are only dealing with simple connected undirected graphs with possibly weighted edges.

Thus, let $G = (V, E)$ be a simple connected undirected graph with possibly weighted edges. Furthermore, let $A = [a_{uv}]_{u,v \in V}$ denote its (weighted) adjacency matrix with the entry a_{uv} representing the weight of the edge $uv \in E$ ($a_{uv} = 0$ if $uv \notin E$) and assume that, in the case of an unweighted graph, $a_{uv} = 1$ if and only if $uv \in E$.

For the purpose of the algorithm, we say that a vertex $v \in V$ is *bad* if $c(u) \neq c(v)$ for the colouring $c\colon V \to \{1, 2, \ldots, k\}$ currently constructed by the algorithm and some vertex $u \in V$ such that $uv \in E$. Analogously, an edge $uv \in E$ that satisfies $c(u) \neq c(v)$, i.e., a bichromatic edge, is said to be a *bad edge*. It will also prove convenient to adopt the notation $\mathcal{W}(v, i) = \sum_{u \in V : c(u) = i} a_{uv}$ for the sum of the weights of the edges incident to vertices of colour i adjacent to vertex v. As a side note, in the case of an unweighted graph, $a_{uv} \in \{0, 1\}$ for all $u, v \in V$, and the expression $\mathcal{W}(v, i)$ reduces to the number of neighbours of vertex v of colour i.

The main idea of the algorithm goes as follows. During initialisation, each vertex gets assigned a random colour. Afterwards, in an iterative procedure, a vertex with at least one neighbour of a different colour – dubbed a bad vertex – is chosen uniformly at random and is reassigned colour i chosen proportionally to $e^{\mathcal{W}(v,i)/T}$ where $\mathcal{W}(v, i)$ denotes the sum of the weights of the edges incident to v with an endpoint of colour i, and T denotes the temperature parameter. Hence, from a local point of view, the more abundant a particular colour is, the more likely it will spread out – even more so, when the corresponding vertices form strong bonds with the rest of the graph. This weighting function is motivated by the simulated annealing heuristics [15] – we will be using an equivalent expression, i.e., $\omega^{\mathcal{W}(v,i)}$ for appropriately chosen weight $\omega > 1$.

The recolouring process repeats until there are no more bad vertices or the stopping criterion, a sufficiently low variance `Var` in the number of bad edges calculated over a sliding window of length $l \in \mathbb{N}$, is met. To facilitate the computation and subsequent updating of the sliding-window variance, at each step of the iteration, the algorithm records the number of bad edges, stores it in the list `bad_edges`, and re-evaluates `Var` only by adding and subtracting local contributions.

To be more specific, let $b_{\texttt{step}} = \texttt{bad\_edges}\,[\texttt{step}]$ be a shorthand notation for the number of bad edges at step `step` of the iteration, and let $\mu_{\texttt{step}}$ and $\mathrm{Var}_{\texttt{step}}$ denote the sample mean and the variance of the number of bad edges at step `step` over the sliding window of length l, respectively, i.e.,

$$\mu_{\texttt{step}} = \frac{1}{l} \sum_{s=\texttt{step}-l+1}^{\texttt{step}} b_s,$$

$$\mathrm{Var}_{\texttt{step}} = \mathrm{Var}\left([b_s]_{s=\texttt{step}-l+1}^{\texttt{step}}\right) = \frac{1}{l-1} \sum_{s=\texttt{step}-l+1}^{\texttt{step}} b_s^2 - \frac{l}{l-1}\mu_{\texttt{step}}^2.$$

The variance at step $\texttt{step}+1$ is then calculated by first updating the sample mean according to

$$\mu_{\texttt{step}+1} = \mu_{\texttt{step}} + \frac{1}{l}\left(b_{\texttt{step}+1} - b_{\texttt{step}-l+1}\right),$$

which is immediately followed by

$$\mathrm{Var}_{\texttt{step}+1} = \mathrm{Var}_{\texttt{step}} + \frac{1}{l-1}\left(b_{\texttt{step}+1} - b_{\texttt{step}-l+1}\right) \cdot \\ \cdot \left(b_{\texttt{step}+1} + b_{\texttt{step}-l+1} - \mu_{\texttt{step}+1} - \mu_{\texttt{step}}\right).$$

All in all, the algorithm thus takes as an input a given graph G, a suitably chosen weight $\omega > 1$, and an initial number of clusters $k \in \mathbb{N}$. Optionally, two additional parameters that serve as a stopping condition may be specified – a tolerance $\texttt{tol} > 0$ on the variance `Var` in the number of edges spanning distinct clusters and a length $l \in \mathbb{N}$ of the sliding window over which this variance is computed. Alternatively – or additionally, a pre-given maximum number of steps can be used as a termination condition. Schematically, the algorithm can be outlined as detailed below (refer to Algorithm 1).

Observe that for the initial number of colours k any upper bound on the number of clusters may be taken. E.g., one may even use the cardinality of the vertex set. Regardless of the choice, the number of colours present in the graph will drop to its natural level as the iterations proceed. Indeed, as soon as a colour $1 \le i \le k$ disappears, $i \notin \{c(v) \mid v \in V\}$, it cannot reappear ever again, since the prerequisite $\mathcal{W}(v,i) \neq 0$ is not met by any vertex $v \in V$ (cf. line 7 in Algorithm 1). Note that permanently discarding absent colours also turned out to enhance numerical stability and consistency of the clustering solutions generated by the algorithm.

More crucially, the weight parameter ω should be chosen more carefully as it may have a profound affect on the algorithm's performance. A sufficiently small ω yields a uniform

Algorithm 1 The modified Petford-Welsh algorithm (mPW)

```
Input: $G, \omega, k, \texttt{tol}, l$
Output: $c$
 1: generate an initial $k$-colouring $c$
 2: $\texttt{bad\_edges}\,[0] \leftarrow \big|\{uv \in E \mid c(u) \neq c(v)\}\big|$
 3: $\texttt{step} \leftarrow 1$
 4: $\texttt{Var} \leftarrow \texttt{tol}$
 5: while $(\texttt{bad\_edges}\,[\texttt{step}-1] > 0)$ and $(\texttt{Var} \geq \texttt{tol})$ do
 6:     choose a bad vertex $v$ uniformly at random
 7:     choose a new colour $1 \leq i \leq k$ for $v$ with probability proportional to $\omega^{\mathcal{W}(v,i)}$ if
        $\mathcal{W}(v,i) \neq 0$
 8:     $c(v) \leftarrow i$
 9:     $\texttt{bad\_edges}\,[\texttt{step}] \leftarrow \big|\{uv \in E \mid c(u) \neq c(v)\}\big|$
10:     if $(\texttt{step} \geq l-1)$ then
11:         $\texttt{Var} \leftarrow \mathrm{Var}\left([\texttt{bad\_edges}\,[s]]_{s=\texttt{step}-l+1}^{\texttt{step}}\right)$
12:     end if
13:     $\texttt{step} \leftarrow \texttt{step}+1$
14: end while
15: return $c$
```

distribution over the set of all possible outcomes that result from recolouring a vertex at each iteration step, leaving the diffusion of colours to chance. At the other end of the spectrum, a large ω puts higher weight on more frequent colours – the more neighbours of a certain colour a vertex has, the higher the probability of it being assigned the same colour. The final decision whether to choose an intermediate value of ω or a value closer to the extremes should be based on the specifics of the application. Moreover, to achieve better results, the model could be extended by implementing a self-tuning mechanism that iteratively optimises ω in a way analogous to [29].

Nevertheless, ω was set to the constant value of 6 for all test runs, the length l of the sliding window was fixed at the number of vertices $|V|$, and the tolerance `tol` was chosen by trial and error but was kept within the range of $[10^{-4}, 10^{-2}]$. A key insight that led to the deployment of `tol` as a stopping criterion in the first place was a typical steady decline in the number of bad edges followed by a flatter region observed across all experiments.

It is worth noting that optimising the initial parameters ω and `tol` through, e.g., dynamic adaptation to the specifics of the problem under consideration should lead to an improvement in overall performance. Undertaking this task lies beyond the scope of this work but nonetheless deserves more attention in an independent study. All data and the source code for the modified Petford-Welsh algorithm are available at [14].

2.1 Fine-tuning

Essentially, the vertex colouring that is returned as the output of the algorithm lends itself to a natural interpretation in terms of clusters – each vertex colour class corresponds to a cluster. However, there are a few hurdles to overcome.

Initial random colouring may inadvertently lead to disconnected clusters, which invalidates one of the main premises underlying clustering in general. This is resolved in the final part of the algorithm by extracting subgraphs corresponding to different colour classes and assigning new colours to their connected components, thus obtaining a subdivision of the original clustering solution. Besides, an additional optional tweak is implemented to enable a further refinement of the clustering solution by discarding singleton clusters that are potentially left over after the iterations terminate. This is achieved by recolouring every such cluster with the most frequently represented colour in its neighbourhood.

Although our algorithm already typically runs in linear time $\mathcal{O}(|E|)$, there is still room for improvement. In the era of ever-expanding data, developing fast and efficient tools is of utmost importance, as the sheer amount and complexity of data can make many existing algorithms computationally unfeasible. A further significant reduction in runtime could be achieved through parallelism, as the very design of our method makes it easily implementable. In fact, as vertex updates are computed locally and independently of one another (that is, neglecting the unlikely event of simultaneously updating both endpoints of an edge), they can be performed synchronously. This approach has already proved efficient in practice for the Petford-Welsh algorithm [38].

3 Numerical experiments

In this section we compare the performance of our algorithm against several state-of-the-art clustering algorithms. In particular, we restrict our analysis to algorithms that, similar to ours, do not require specifying any additional parameters beforehand. In order to make experiments more comparable, we use the implementations of these algorithms that are readily available in the `python-igraph` (version 0.7.1) software package that comprises of the `igraph` high performance `C` library for network analysis and a `Python` interface [8]. These are *Edge betweenness* relying on iterative removal of edges with high betweenness scores, *Fastgreedy*, *Multilevel*, and *Leading eigenvector* optimising modularity in different ways, *Infomap* analysing the information flow of random walks, *Label propagation* based on an iterative process in which each vertex adopts a label that the majority of its neighbours has, *Spinglass* utilising a spin-glass model and simulated annealing, and *Walktrap* performing random walks. For a thorough review of the methods, their computational complexity and performance, we refer to survey [35] and the references therein.

As far as our algorithm is concerned, it is implemented in `Python`. However, in order to make it comparable to `igraph`'s implementations in `C`, the part of the code that corresponds to the iterative procedure is compiled using `Cython` [5] that converts it into `C` code. The initialisation and the fine-tuning processes are compiled in `Python`. The experiments were performed on a computer equipped with an Intel Core i7-8550U 1.80 GHz processor with 16.0 GB of physical memory (RAM).

3.1 Datasets

We investigated the performance of our proposed method on a class of artificially generated networks as well as on several real-world networks, some of which were given a ground-truth division into clusters. However, it should be noted that it can be misleading to compare the ground-truth clusters, possibly derived from latent network information, with those detected by clustering algorithms operating solely with structural information about the network.

All synthetic networks with embedded community structure were generated by means of the Lancichinetti-Fortunato-Radicchi (LFR) benchmark [17], a commonly used benchmark for testing clustering algorithms. It tries to mimic observations found in many real-world networks by producing networks whose vertex degrees and cluster sizes follow power-law distributions with exponents $2 \leq \gamma \leq 3$ and $1 \leq \beta \leq 2$, respectively. Moreover, its tunable mixing parameter $\mu \in [0, 1]$, the fraction of a vertex's neighbours that belong to different clusters, enables varying how well-defined the clusters are. For $\mu > 0.5$, clusters tend to blend in and become increasingly indistinguishable; as noted in [35], clusters in the strong sense disappear.

Further analysis was carried out on diverse real-world networks ranging from social to infrastructural networks. With the exception of one, all of them have an observationally or experimentally pre-defined community structure. Their basic properties are summarised in Table 1.

Table 1: Real-world networks used in our experiments.

Graph	Vertices	Edges	Ground truth
Zachary's karate club	34	78	Yes
Dolphins	62	159	Yes
UK faculty	79	552	Yes
Political books	105	441	Yes
American college football	115	613	Yes
Political blogs	1222	16714	Yes
Cora citation network	23166	89157	Yes
International E-road network	1040	1305	No

For simplicity, all networks are treated as unweighted and undirected, and we only consider their largest connected components.

3.2 Quality metrics

Most commonly, clustering results are evaluated using internal and external quantitative approaches [34]. The former are derived on the basis of distinct intrinsic statistical characteristics of the clustering solution obtained, whereas the latter compare it to a given ground-truth clustering, provided such a clustering exists.

With regard to internal quality measures, we used the now-standard *modularity* Q [21], *conductance* ϕ [10], and *coverage* γ [10], defined as

$$Q(\mathcal{C}) = \frac{1}{2|E|} \sum_{v,w \in V} \left(a_{vw} - \frac{k_v k_w}{2|E|} \right) \delta(c_v, c_w),$$

$$\phi(\mathcal{C}) = 1 - \frac{1}{|\mathcal{C}|} \sum_{C_i \in \mathcal{C}} \phi(C_i) \quad \text{with}$$

$$\phi(C_i) = \frac{\sum_{v \in C_i, w \notin C_i} a_{vw}}{\min\left\{ \sum_{v \in C_i, w \in V} a_{vw}, \sum_{v \notin C_i, w \in V} a_{vw} \right\}},$$

$$\gamma(\mathcal{C}) = \frac{\sum_{v,w \in V} a_{vw} \delta(c_v, c_w)}{\sum_{v,w \in V} a_{vw}}.$$

Here, $\mathcal{C} = \{C_i\}_i$ denotes a partition into clusters, $A = [a_{vw}]_{v,w \in V}$ the network's adjacency matrix, k_v the degree of vertex v and c_v the community it has been assigned to with respect to $\mathcal{C}$.

One should bear in mind that despite the general prevalence of these indices, they come with a number of caveats. For example, modularity suffers from the resolution limit that displays a bias towards detecting larger clusters, as its value increases by merging clusters smaller than the characteristic size [12]. In spite of there being no clear winner in terms of general applicability, the study published in [10] suggests that conductance correlates best with information recovery metrics, which tend to be more reliable than internal indices when ground truth is supplied.

To validate against externally given ground truth, we made use of two well-established information-theoretic measures, *normalised mutual information* NMI [9] and the *adjusted Rand index* ARI [34]. Given two partitions $\mathcal{C}_1$ and $\mathcal{C}_2$, they can be calculated as follows

$$\mathrm{NMI}(\mathcal{C}_1,\mathcal{C}_2) = \frac{\mathrm{MI}(\mathcal{C}_1,\mathcal{C}_2)}{\sqrt{\mathrm{H}(\mathcal{C}_1)\mathrm{H}(\mathcal{C}_2)}} \quad \text{with} \quad \mathrm{MI}(\mathcal{C}_1,\mathcal{C}_2) = \mathrm{H}(\mathcal{C}_1) + \mathrm{H}(\mathcal{C}_2) - \mathrm{H}(\mathcal{C}_1,\mathcal{C}_2),$$

$$\begin{aligned}\mathrm{ARI}(\mathcal{C}_1,\mathcal{C}_2) &= \frac{\mathrm{RI}(\mathcal{C}_1,\mathcal{C}_2) - \mathrm{E}[\mathrm{RI}(\mathcal{C}_1,\mathcal{C}_2)]}{\max(\mathrm{RI}(\mathcal{C}_1,\mathcal{C}_2)) - \mathrm{E}[\mathrm{RI}(\mathcal{C}_1,\mathcal{C}_2)]} \\ &= \frac{2(n_{00}n_{11} - n_{01}n_{10})}{(n_{00}+n_{01})(n_{01}+n_{11}) + (n_{00}+n_{10})(n_{10}+n_{11})}.\end{aligned}$$

In the formulas above, H denotes the Shannon entropy, i.e., $\mathrm{H}(\mathcal{C}_i) = -\sum_{C\in\mathcal{C}_i} \frac{|C|}{|V|} \log \frac{|C|}{|V|}$ and $\mathrm{H}(\mathcal{C}_1,\mathcal{C}_2) = -\sum_{C_i^1\in\mathcal{C}_1, C_j^2\in\mathcal{C}_2} \frac{|C_i^1\cap C_j^2|}{|V|} \log \frac{|C_i^1\cap C_j^2|}{|V|}$, and n_{ij} denotes the number of pairs of vertices that fall in the same cluster (different clusters) of $\mathcal{C}_1$ if $i=1$ ($i=0$) and in the same cluster (different clusters) of $\mathcal{C}_2$ if $j=1$ ($j=0$). Both NMI and ARI were computed using `Python` library `Scikit-learn`.

As a final remark, all metrics under consideration assume values in the range of 0-1, and combined higher values across all quality measures indicate better performance.

3.3 Results

Adequately equipped with all the necessary tools, we are now able to present our experimental results. First of all, let us see how well our algorithm performs on the set of real-world networks which was briefly described in Table 1.

As evident from Tables 2 to 8 provided in Section A.1 of the Supplementary materials, the modified Petford-Welsh algorithm, if ran a sufficiently large number of times, either recovered the underlying ground-truth division into clusters or constructed a clustering closest to the ground truth in terms of both NMI and ARI. Notice that modularity may indeed not be the best indicator of the quality of clustering solutions. On the other hand, conductance tends to achieve highest values for the mPW, which is consistent with findings in [10].

What is more, although the International E-road network does not come with a predetermined community structure, solutions generated by the mPW – besides achieving the highest conductance (see Table 9 in Section A.1 of the Supplementary materials) – also seem to be in line with a natural partition of Europe and Central Asia into countries, as shown on Figure 1 below. On top of that, relaxing the tolerance `tol` on the variance in the number of bichromatic edges leads to a coarser partition into geopolitical regions within both continents.

In all experiments, the mPW turned out to be the fastest-performing method, although it should be pointed out that the time elapsed during the initialisation phase and the fine-tuning procedures was not measured in order to be able to benchmark the code against `python-igraph` C implementations.

On a different note, results on the LFR benchmark (here, we refer to Section A.2 in the Supplementary materials and the figures within) indicate that the mPW is among the best-performing methods in terms of NMI and ARI when the community structure is well-defined, that is, in the range $\mu \in [0.1, 0.4]$. As the clusters become more and more intertwined, its performance drops, but achieves best results in the region of $\mu \gtrapprox 0.7$, where all methods, as expected, struggle the most. Here, the difference between the mPW and Label propagation becomes the most apparent – although both methods use only local information, are non-deterministic (and thus need to be executed repeatedly in order to achieve best results), with almost linear complexity in the number of edges, Label propagation, as opposed to the mPW, can either easily get trapped into local minima or inevitably lead to a single giant cluster.

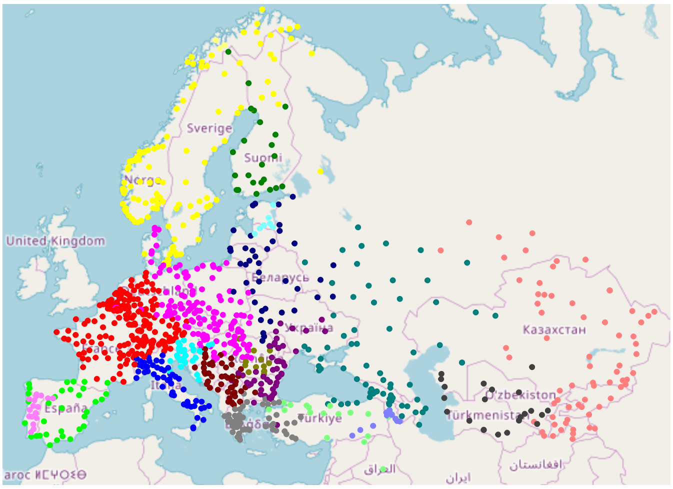

Figure 1: A clustering solution generated by the modified Petford-Welsh algorithm. It consists of 20 clusters; the corresponding quality metrics are equal to $Q = 0.830$, $\phi = 0.847$, and $\gamma = 0.933$.

It should be taken into account that optimising the choice of the initial parameters ω and `tol` lies beyond the scope of this study; implementing it should lead to better results. Indeed, dynamically adapting `tol` to account for the mixing parameter μ already yielded a significant improvement.

All in all, predominantly, the mPW is a fast algorithm that scales efficiently to large networks and produces high-quality clustering solutions when the community structure is reasonably well-defined. Due to its non-deterministic nature, it can be best used in order to find a division into clusters that optimises an application-dependant pre-given quality measure.

ORCID iDs

Barbara Ikica https://orcid.org/0000-0002-2367-8610

References

[1] Cora citation network dataset – KONECT, April 2017, `http://konect.uni-koblenz.de/networks/subelj_cora`.

[2] Euroroad network dataset – KONECT, April 2017, `http://konect.uni-koblenz.de/networks/subelj_euroroad`.

[3] L. A. Adamic and N. S. Glance, The political blogosphere and the 2004 U.S. election: divided they blog, in: J. Adibi, M. Grobelnik, D. Mladenić and P. Pantel (eds.), *Proceedings of the 3rd International Workshop on Link Discovery (LinkKDD 2005)*, ACM, 2005 pp. 36–43, doi:10.1145/1134271.1134277, held in Chicago, Illinois, USA, August 21 – 25, 2005.

[4] A.-L. Barabási and M. Pósfai, *Network Science*, Cambridge University Press, Cambridge, UK, 2016.

[5] S. Behnel, R. Bradshaw, C. Citro, L. Dalcín, D. S. Seljebotn and K. Smith, Cython: the best of both worlds, *Comput. Sci. Eng.* **13** (2011), 31–39, doi:10.1109/mcse.2010.118.

[6] V. D. Blondel, J.-L. Guillaume, R. Lambiotte and E. Lefebvre, Fast unfolding of communities in large networks, *J. Stat. Mech. Theory Exp.* **2008** (2008), P10008, doi:10.1088/1742-5468/2008/10/p10008.

[7] A. Clauset, M. E. J. Newman and C. Moore, Finding community structure in very large networks, *Phys. Rev. E* **70** (2004), 066111, doi:10.1103/physreve.70.066111.

[8] G. Csárdi and T. Nepusz, The igraph software package for complex network research, *InterJournal Complex Syst.* (2006), Manuscript 1695 (9 pages), `https://www.interjournal.org/manuscript_abstract.php?361100992`.

[9] L. Danon, A. Díaz-Guilera, J. Duch and A. Arenas, Comparing community structure identification, *J. Stat. Mech. Theory Exp.* **2005** (2005), P09008, doi:10.1088/1742-5468/2005/09/P09008.

[10] S. Emmons, S. G. Kobourov, M. Gallant and K. Börner, Analysis of network clustering algorithms and cluster quality metrics at scale, *PLoS ONE* **11** (2016), e0159161, doi:10.1371/journal.pone.0159161.

[11] S. Fortunato, Community detection in graphs, *Phys. Rep.* **486** (2010), 75–174, doi:10.1016/j.physrep.2009.11.002.

[12] S. Fortunato and M. Barthelemy, Resolution limit in community detection, *Proc. Natl. Acad. Sci. U.S.A.* **104** (2007), 36–41, doi:10.1073/pnas.0605965104.

[13] M. Girvan and M. E. J. Newman, Community structure in social and biological networks, *Proc. Natl. Acad. Sci. U.S.A.* **99** (2002), 7821–7826, doi:10.1073/pnas.122653799.

[14] B. Ikica, A clustering algorithm based on a modification of the Petford-Welsh algorithm, `https://github.com/ikicab/mPW`.

[15] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, Optimization by simulated annealing, *Science* **220** (1983), 671–680, doi:10.1126/science.220.4598.671.

[16] V. Krebs, Amazon's sales data of political books, `http://www.orgnet.com/divided.html`.

[17] A. Lancichinetti, S. Fortunato and F. Radicchi, Benchmark graphs for testing community detection algorithms, *Phys. Rev. E* **78** (2008), 046110, doi:10.1103/physreve.78.046110.

[18] D. Lusseau, K. Schneider, O. J. Boisseau, P. Haase, E. Slooten and S. M. Dawson, The bottlenose dolphin community of Doubtful Sound features a large proportion of long-lasting associations, *Behav. Ecol. Sociobiol.* **54** (2003), 396–405, doi:10.1007/s00265-003-0651-y.

[19] T. Nepusz, A. Petróczi, L. Négyessy and F. Bazsó, Fuzzy communities and the concept of bridgeness in complex networks, *Phys. Rev. E* **77** (2008), 016107, doi:10.1103/physreve.77.016107.

[20] M. E. J. Newman, Finding community structure in networks using the eigenvectors of matrices, *Phys. Rev. E* **74** (2006), 036104, doi:10.1103/physreve.74.036104.

[21] M. E. J. Newman, *Networks: An Introduction*, Oxford University Press, New York, NY, USA, 2010, doi:10.1093/acprof:oso/9780199206650.001.0001.

[22] L. Peel, D. B. Larremore and A. Clauset, The ground truth about metadata and community detection in networks, *Sci. Adv.* **3** (2017), e1602548, doi:10.1126/sciadv.1602548.

[23] A. D. Petford and D. J. A. Welsh, A randomised 3-colouring algorithm, *Discrete Math.* **74** (1989), 253–261, doi:10.1016/0012-365x(89)90214-8.

[24] P. Pons and M. Latapy, Computing communities in large networks using random walks, *J. Graph Algorithms Appl.* **10** (2006), 191–218, doi:10.7155/jgaa.00124.

[25] U. N. Raghavan, R. Albert and S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, *Phys. Rev. E* **76** (2007), 036106, doi:10.1103/physreve.76.036106.

[26] J. Reichardt and S. Bornholdt, Statistical mechanics of community detection, *Phys. Rev. E* **74** (2006), 016110, doi:10.1103/physreve.74.016110.

[27] M. Rosvall, D. Axelsson and C. T. Bergstrom, The map equation, *Eur. Phys. J. Spec. Top.* **178** (2009), 13–23, doi:10.1140/epjst/e2010-01179-1.

[28] M. Rosvall and C. T. Bergstrom, Maps of random walks on complex networks reveal community structure, *Proc. Natl. Acad. Sci. U.S.A.* **105** (2008), 1118–1123, doi:10.1073/pnas.0706851105.

[29] J. Shawe-Taylor and J. Žerovnik, Adapting temperature for some randomized local search algorithms, in: N. Mastorakis, V. M. Mladenov and B. Suter (eds.), *Advances in Scientific Computing, Computational Intelligence and Applications*, WSES Press, Danvers, MA, USA, Mathematics and Computers in Science and Engineering, pp. 82–87, 2001.

[30] S. H. Strogatz, Exploring complex networks, *Nature* **410** (2001), 268–276, doi:10.1038/35065725.

[31] L. Šubelj and M. Bajec, Robust network community detection using balanced propagation, *Eur. Phys. J. B* **81** (2011), 353–362, doi:10.1140/epjb/e2011-10979-2.

[32] L. Šubelj and M. Bajec, Model of complex networks based on citation dynamics, in: L. Carr, A. H. F. Laender, B. Farias Lóscio, I. King, M. Fontoura, D. Vrandecic, L. Aroyo, J. P. M. de Oliveira, F. Lima and E. Wilde (eds.), *Proceedings of the 22nd International Conference on World Wide Web (WWW '13)*, ACM, New York, NY, USA, 2013 pp. 527–530, doi:10.1145/2487788.2487987, held in Rio de Janeiro, Brazil, May 13 – 17, 2013.

[33] V. A. Traag and J. Bruggeman, Community detection in networks with positive and negative links, *Phys. Rev. E* **80** (2009), 036115, doi:10.1103/physreve.80.036115.

[34] C. Wiwie, J. Baumbach and R. Röttger, Comparing the performance of biomedical clustering methods, *Nat. Methods* **12** (2015), 1033–1038, doi:10.1038/nmeth.3583.

[35] Z. Yang, R. Algesheimer and C. J. Tessone, A comparative analysis of community detection algorithms on artificial networks, *Sci. Rep.* **6** (2016), 30750, doi:10.1038/srep30750.

[36] W. W. Zachary, An information flow model for conflict and fission in small groups, *J. Anthropol. Res.* **33** (1977), 452–473, doi:10.1086/jar.33.4.3629752.

[37] J. Žerovnik, A randomized algorithm for k-colorability, *Discrete Math.* **131** (1994), 379–393, doi:10.1016/0012-365x(94)90402-2.

[38] J. Žerovnik and M. Kaufman, A parallel variant of a heuristical algorithm for graph coloring — Corrigendum, *Parallel Comput.* **18** (1992), 897–900, doi:10.1016/0167-8191(92)90035-6.

Appendix A Supplementary materials

A.1 Real-world networks

Our analysis on real-world networks is supported by the numerical results presented in the tables below. In columns NMI, ARI, ϕ, γ, and Q, maximum values attained over r runs (as specified below) are reported. Column *Clusters* shows the median number of clusters returned and column $t[s]$ the average running time in seconds over the same set of runs. For the modified Petford–Welsh algorithm (mPW), only running times of the code compiled to C are measured (excluding the initialisation phase and the fine-tuning procedures). In all cases except for the Spinglass method applied to the *Political blogs* and the *International E-road network*, the number of runs r was set to 100. Due to a relatively high complexity of Spinglass, we needed to resort to a smaller value of $r = 10$ when running it on networks of order greater than ≈ 1000 (although this was still too much of a hurdle for the *Cora citation network*). Edge betweenness was not even run on these networks noting its prohibitively high computational complexity, namely $\mathcal{O}(|E|^2|V|)$ [35]. Bold font indicates maximum values attained column-wise (except for the column corresponding to $t[s]$, where the minimum values are indicated).

Table 2: Zachary's karate club [36].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness [13]	0.517	0.392	0.424	0.692	0.401	5	2.083e−03
Fastgreedy [7]	0.576	0.568	0.574	0.756	0.381	3	1.630e−04
Multilevel [6]	0.516	0.392	0.558	0.731	0.419	4	1.512e−04
Leading eigenvector [20]	0.612	0.435	0.487	0.667	0.393	4	3.191e−03
Infomap [28, 27]	0.578	0.591	0.668	0.821	0.402	3	5.579e−03
Label propagation [25]	0.865	0.882	**0.773**	**0.949**	0.415	3	7.300e−05
Spinglass [26, 33]	0.627	0.509	0.563	0.756	**0.420**	4	2.922e−01
Walktrap [24]	0.531	0.321	0.434	0.590	0.353	5	1.538e−04
mPW	**1.000**	**1.000**	**0.773**	**0.949**	0.403	2	**3.080e−07**

Table 3: Dolphins [18].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	0.600	0.395	0.570	0.799	0.519	5	1.792e−02
Fastgreedy	0.602	0.451	0.575	0.824	0.495	4	8.213e−04
Infomap	0.649	0.390	0.571	0.774	0.528	5	8.299e−02
Label propagation	**1.000**	**1.000**	**0.880**	**0.962**	0.526	4	3.263e−04
Leading eigenvector	0.497	0.283	0.544	0.711	0.491	5	7.615e−03
Multilevel	0.564	0.327	0.585	0.755	0.519	5	1.217e−03
Spinglass	0.690	0.452	0.654	0.805	**0.529**	5	4.357e−01
Walktrap	0.565	0.417	0.613	0.824	0.489	4	2.257e−03
mPW	**1.000**	**1.000**	**0.880**	**0.962**	0.528	3	**4.210e−07**

Table 4: UK faculty [19].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	0.875	0.898	0.540	0.841	0.441	4	1.826e−01
Fastgreedy	0.882	0.854	0.564	0.783	0.457	4	5.425e−04
Infomap	0.838	0.817	0.583	0.763	**0.461**	4	5.510e−03
Label propagation	0.898	0.920	0.722	**0.953**	0.443	3	1.763e−04
Leading eigenvector	0.898	0.913	0.495	0.772	0.408	4	4.711e−03
Multilevel	0.948	0.957	0.683	0.826	0.446	3	6.230e−04
Spinglass	0.894	0.842	0.583	0.764	**0.461**	4	5.776e−01
Walktrap	0.838	0.817	0.583	0.763	**0.461**	4	1.544e−03
mPW	**0.951**	**0.968**	**0.743**	**0.953**	0.443	3	**4.250e−07**

Table 5: Political books [16].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	0.562	0.682	0.626	0.905	0.517	5	1.153e−01
Fastgreedy	0.531	0.638	0.648	0.918	0.502	4	4.967e−04
Infomap	0.503	0.536	0.584	0.855	0.523	6	1.142e−02
Label propagation	0.607	0.702	**0.917**	**0.957**	0.523	3	2.345e−04
Leading eigenvector	0.525	0.547	0.555	0.778	0.467	4	7.717e−03
Multilevel	0.516	0.558	0.675	0.853	0.520	4	4.436e−04
Spinglass	0.566	0.657	0.644	0.889	**0.527**	6	8.279e−01
Walktrap	0.544	0.653	0.687	0.914	0.507	4	9.237e−04
mPW	**0.645**	**0.727**	**0.917**	**0.957**	0.511	3	**5.230e−07**

Table 6: American college football [13].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	0.880	0.778	0.533	0.710	0.600	10	2.486e−01
Fastgreedy	0.708	0.474	0.567	0.731	0.550	6	8.467e−04
Multilevel	0.891	0.807	0.547	0.708	**0.605**	10	4.085e−04
Leading eigenvector	0.703	0.464	0.456	0.641	0.493	8	9.658e−03
Infomap	0.924	0.897	0.505	0.690	0.601	12	9.109e−03
Label propagation	0.927	0.889	0.568	0.741	**0.605**	11	2.301e−04
Spinglass	0.929	**0.900**	0.563	0.728	**0.605**	11	3.580e−01
Walktrap	0.888	0.815	0.547	0.705	0.603	10	1.383e−03
mPW	**0.936**	**0.900**	**0.600**	**0.780**	0.603	9	**6.200e−07**

Table 7: Political blogs [3].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	–	–	–	–	–	–	–
Fastgreedy	0.659	0.785	0.451	0.923	**0.427**	10	6.923e−01
Infomap	0.523	0.651	0.250	0.899	0.423	41	3.500e+00
Label propagation	0.723	0.813	**0.857**	**1.000**	0.426	3	6.523e−03
Leading eigenvector	0.693	0.781	0.854	0.926	0.424	2	8.246e−02
Multilevel	0.651	0.774	0.476	0.920	**0.427**	9	3.876e−02
Spinglass	0.649	0.783	0.315	0.922	**0.427**	15	5.976e+01
Walktrap	0.646	0.760	0.484	0.925	0.425	11	4.427e−01
mPW	**0.732**	**0.820**	**0.857**	0.927	0.426	4	**2.000e−06**

Table 8: Cora citation network [1, 32].

Method	NMI	ARI	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	–	–	–	–	–	–	–
Fastgreedy	0.373	0.085	0.701	**0.906**	0.693	159	7.465e+00
Infomap	**0.574**	0.122	0.505	0.678	0.670	1162	1.450e+02
Label propagation	0.546	0.170	0.580	0.793	0.721	722	5.967e−01
Leading eigenvector	0.157	0.008	0.252	0.853	0.311	11	6.480e+00
Multilevel	0.450	**0.190**	**0.792**	0.860	**0.790**	42	2.600e−01
Walktrap	0.522	0.127	0.523	0.797	0.710	1204	3.347e+01
Spinglass	–	–	–	–	–	–	–
mPW	0.537	0.184	0.602	0.771	0.729	517	**7.100e−05**

Table 9: International E-road network [2, 31].

Method	ϕ	γ	Q	Clusters	$t[s]$
Edge betweenness	–	–	–	–	–
Fastgreedy	0.860	0.917	0.861	24	3.741e−03
Infomap	0.663	0.787	0.777	126	4.615e−01
Label propagation	0.731	0.856	0.828	82	7.130e−03
Leading eigenvector	0.794	0.887	0.835	26	3.492e−01
Multilevel	0.873	0.921	0.867	24	4.546e−03
Spinglass	0.866	0.924	**0.872**	25	1.210e+01
Walktrap	0.757	0.886	0.828	67	8.510e−03
mPW	**0.945**	**0.979**	0.845	17	**2.000e−06**

A.2 LFR benchmark

Experiments were also conducted on networks generated by the LFR benchmark model [17]. To this end, we constructed three qualitatively different families of networks on 1000 vertices and, for each of the regimes separately, studied how varying the mixing parameter μ in the range of 0.1 to 0.9 affects the overall performance.

First, the power-law exponents were set to $\gamma = 2$ and $\beta = 1$, the average and the maximum degree to 15 and 100, respectively, and the sizes of the embedded ground-truth clusters were restricted to the interval $[50, 100]$. Further, the second set of trials was carried out on networks with the average degree and the maximum degree equal to 25 and 150, respectively, whereas the power-law exponents were kept the same, i.e., $\gamma = 2$ and $\beta = 1$, and no constraints were imposed on the sizes of the clusters. Lastly, the networks in the third regime followed power-law distributions at the other extreme, with $\gamma = 3$ and $\beta = 2$. For the average degree and the maximum degree we chose 15 and 50, respectively. Again, we left out the optional parameters controlling the permitted cluster sizes.

The plots below display how maximum values of NMI, ARI, and Q attained over a series of runs vary as a function of the mixing parameter μ. For each particular value of μ, 10 networks were generated and each of the methods was run 100 times on top of them. Edge betweenness and Spinglass were omitted from the analysis due to their rather slow execution speed. Moreover, in order for Leading eigenvector to converge, new networks had to be generated occasionally.

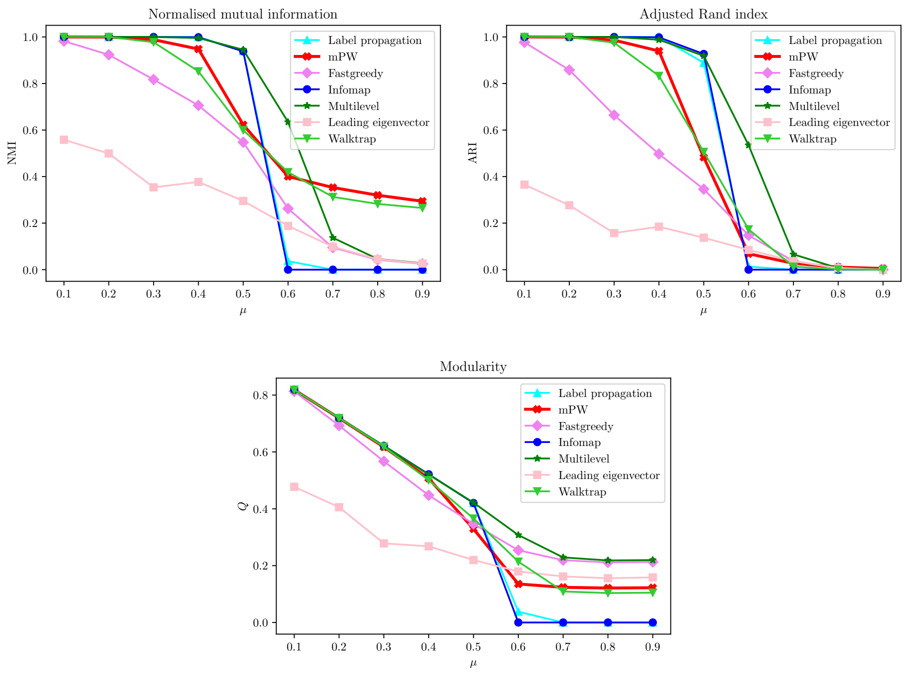

Figure 2: NMI, ARI, and Q plotted as functions of $\mu \in [0.1, 0.9]$, evaluated on networks generated by the LFR benchmark model using $|V| = 1000, \gamma = 2, \beta = 1,$ `k_avg` $=$ $15,$ `k_max` $= 100,$ `c_min` $= 50$ and `c_max` $= 100$.

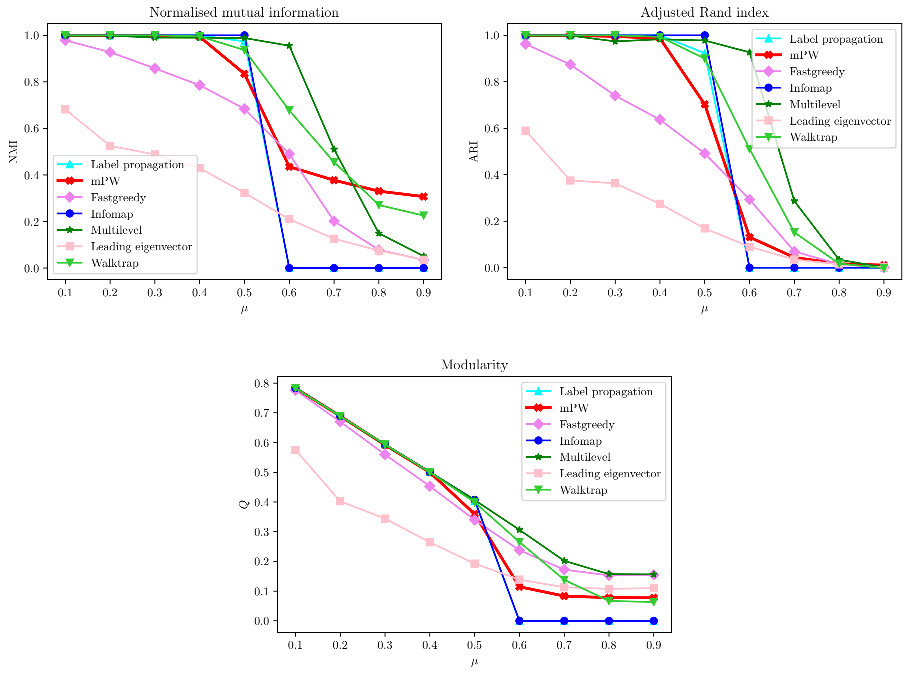

Figure 3: NMI, ARI, and Q plotted as functions of $\mu \in [0.1, 0.9]$, evaluated on networks generated by the LFR benchmark model using $|V| = 1000, \gamma = 2, \beta = 1, \texttt{k\_avg} = 25$ and $\texttt{k\_max} = 150$.

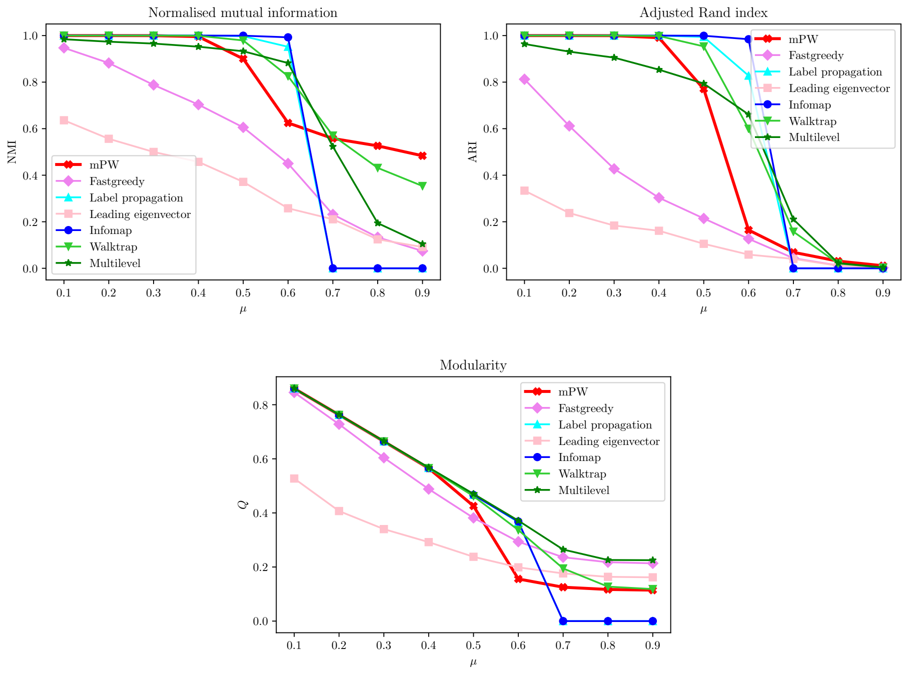

Figure 4: NMI, ARI, and Q plotted as functions of $\mu \in [0.1, 0.9]$, evaluated on networks generated by the LFR benchmark model using $|V| = 1000, \gamma = 3, \beta = 2$, `k_avg` $= 15$ and `k_max` $= 50$.