Graph Clustering Via Intra-Cluster Density Maximization

Pierre Miasnikof, Leonidas Pitsoulis, Anthony J. Bonner, Yuri Lawryshyn and Panos M. Pardalos

Abstract Graph clustering, also often referred to as network community detection, is the process of assigning common labels to vertices that are densely connected to each other but sparsely connected to the rest of the graph. There are many different approaches to clustering in the literature. However, in this article, we formulate the clustering problem as a combinatorial optimization problem. Our main contribution is a novel problem formulation that maximizes intra-cluster density, a statistically meaningful quantity. It requires the number of clusters, a softbound on cluster size and a penalty coefficient as parameter inputs. More importantly, it is designed to prevent common degeneracies, like the so-called "mega-clusters". We end with some suggestions on numerical solution techniques and note that an ensemble-like optimization routine seems promising.

1 Introduction

Graph clustering is the process of grouping of vertices into densely connected subsets of vertices that have sparse connections to other subsets of vertices. These subsets are referred to as clusters. The process of assigning cluster labels to vertices, grouping them into clusters, is referred to as graph clustering (or network community detection).

Note on vocabulary: Although there are subtle differences between the concepts of graph clustering and network community detection, in this document we use the two interchangeably.

P. Miasnikof (✉) · A. J. Bonner · Y. Lawryshyn
University of Toronto, Toronto, ON, Canada
e-mail: p.miasnikof@mail.utoronto.ca

L. Pitsoulis
Aristotle University of Thessaloniki, Thessaloniki, Greece

P. M. Pardalos
University of Florida, Gainesville, FL, USA

National Research University HSE, Nizhny Novgorod, Russian Federation

I. Bychkov et al. (eds.), *Network Algorithms, Data Mining, and Applications*, Springer Proceedings in Mathematics & Statistics 315,
https://doi.org/10.1007/978-3-030-37157-9_3

Fig. 1 Good clustering

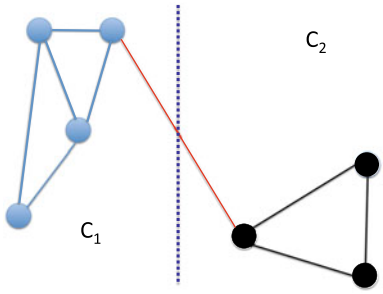
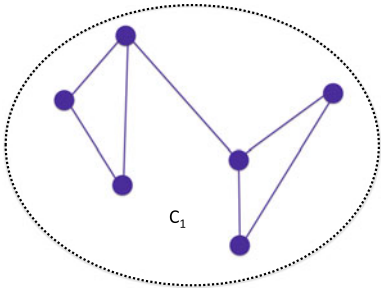


Fig. 2 Bad clustering



The definition of graph clusters (or network communities) remains a matter of debate in the literature (e.g., [10, 25]). However, most authors agree that a cluster can be described as a dense subgraph within a sparse graph (e.g., [8, 22, 23, 28], we quote these authors, but their definition is very common throughout the literature).

In Fig. 1, we see properly labeled (clustered) vertices. On either side of the dotted line, we observe densely connected vertices and very sparse (only one edge) connections between each cluster (C_1 and C_2). On the other hand, in Fig. 2, we observe two cliques connected to each other by only a single edge being labeled as all belonging to the same cluster. Clearly, in this example, labeling each triangle as belonging to a separate cluster would be more reasonable.

It is also important to draw a distinction between clusters and cliques (e.g., "the clique problem" or "maximal clique problem" [27]), since clusters may or may not be cliques. In fact, according to Fortunato and Hric [10], clusters typically are not cliques.

In this article, we formulate the clustering problem as a mean intra-cluster density maximization problem. While most authors who have used optimization-based approaches maximize modularity, we maximize a statistically meaningful and robust quantity. Modularity is known to be fragile and problematic in many ways and has been shown to be less responsive to graph structure than mean intra-cluster density (e.g., [9, 17]). To the best of our knowledge, this formulation is novel and has not been used in the past.

2 Previous Work

A complete review of the very rich graph clustering literature is beyond the scope of this article. However, we note many authors have approached graph clustering using various techniques. There exist many competing formulations and solution techniques in the literature. The main ones are

- Spectral (e.g., [16]),
- Markov (e.g., [5]), and
- Optimization.
  - Modularity maximization (e.g., [1, 10, 18]),
  - Other objective functions (e.g., [6, 7, 15]).

While there are many competing approaches to clustering, Fortunato and Hric claim that determining which is the best clustering technique under all circumstances is not a clear-cut case [10] and that most algorithms cannot adapt to every dataset and consistently provide superior clusterings.

It is also important to note that spectral methods are very costly and do not scale well at all. Spectral clustering methods require eigendecomposition of the graph's Laplacian, a projection into a metric space and the application of a k-means algorithm. This difficulty with scaling has also been noted by Schaeffer, in her review [25]. While there are techniques that allow us to take advantage of sparsity and the partial computation of eigenvalues, the combined costs of such an approach make it prohibitive for large graphs. Even in more recent work, where authors claim their algorithms are "faster and more accurate" than legacy techniques in this area, they still require onerous computations (e.g., [13]). Finally, it should also be noted that Fortunato and Hric describe spectral methods as inaccurate in the case of sparse graphs [10] and that clusterable graphs are typically sparse.

As for Markov-based techniques, they require simulating a random walk over the graph (i.e., matrix multiplications), as well as multiple element-wise and row operations. While Markov clustering does not require the number of clusters as input, it relies on costly operations. Also, as highlighted by Fortunato and Hric, algorithms that do not require the number of clusters as an input parameter have been found to be less accurate than those that do require it [10].

On the other hand, optimization-based approaches lend themselves very well to approximate solution techniques, which carry a lower computational cost. Indeed, because of the NP-hardness of the problem [8, 25], solving these and other types of combinatorial optimization problems is often successfully done via (meta-)heuristic solution techniques (e.g., [21]), which explore subsets of the solution space. In the specific case of graph clustering, many authors have made use of meta-heuristic optimization techniques (e.g., [1, 14, 18, 26]), in order to find approximate solutions and overcome the NP-hard nature of the problem. Additionally, meta-heuristic optimization techniques are easily parallelizable and well suited to implementation on high-performance computing platforms.

3 Problem Formulation

Using the almost universally accepted definition that a cluster is a dense subgraph within a sparser graph, we formulate an optimization problem that assigns cluster labels to nodes in a manner that yields a high mean intra-cluster density. Our goal is to assign cluster labels so that the mean intra-cluster density is higher than the graph's global density.

Recall, (sub)graph density is defined as the ratio of the total number of edges or sum of edge weights ($|E_i|$) over the maximum possible number of edges in a given (sub)graph, given the number of vertices (n_i):

$$\kappa_i = \frac{|E_i|}{0.5 \times n_i(n_i - 1)}$$

In the case of an unweighted graph, the quantity κ_i represents the empirical estimate of the probability that two vertices are connected. In the case of a weighted graph, it represents the mean edge weight. This definition of density and its interpretations apply to graphs without multiple edges or self-loops.

We assert that a graph whose vertices have been well clustered (labeled) will display a mean intra-cluster density, a quantity we call $\bar{K}_{\text{intra}}$ [17], that is higher than the graph's global density. This assertion was initially presented and detailed in Miasnikof et al. [17]. Mathematically, we assert that, for a properly clustered graph with $|C|$ clusters, the following inequality should hold:

$$\bar{K}_{\text{intra}} = \frac{1}{|C|} \sum_{i=1}^{|C|} \kappa_i > K = \frac{|E|}{0.5 \times N(N-1)}$$

where

$$N = \sum_{i=1}^{|C|} n_i$$

$$\kappa_i = \frac{|E_{ii}|}{0.5 \times n_i(n_i - 1)}$$

In the equations above, $|E|$ is the total number (weight) of edges on the graph and N is the total number of vertices. Similarly, we compute κ_i for each cluster, where $|E_{ii}|$ is the total number (weight) of edges connecting two vertices in cluster "i" and n_i is the number of vertices in that same cluster.

For example, in Fig. 3, we show an example of a well-labeled graph. The graph's global density is $K = 0.43$, while the mean intra-cluster density (assuming only two clusters) is $\bar{K}_{\text{intra}} = \frac{1}{2}(\kappa_1 + \kappa_2) = \frac{1}{2}(0.83 + 1) = 0.915$. Here, our earlier assertion regarding the inequality $K < \bar{K}_{\text{intra}}$ clearly holds and can also be easily visualized in this small example.

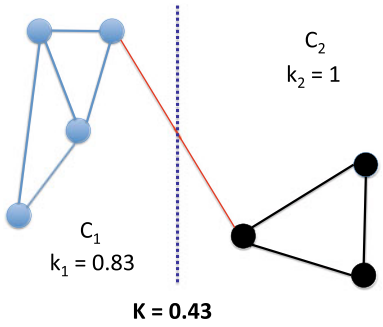


Fig. 3 Global and intra-cluster densities

3.1 Optimization Problem: Mean Intra-Cluster Density Maximization

We opt for an objective function inspired by the quadratic maximization formulation presented by Fan and Pardalos [6], which we modify slightly. These authors maximize the following objective function:

$$\underset{x_{i,k},x_{j,k}}{\text{maximize}} \left\{ z = \sum_{i,j,k} w_{i,j} x_{i,k} x_{j,k} \right\} \tag{1}$$
$$(x_{i,k} \in \{0,1\},\ w_{i,j} \in \mathbb{R}_+,\ \forall i,\ j \in V,\ \forall k \in C)$$

In this function, $x_{i,k}$ is an indicator variable that is equal to 1 if vertex i is assigned to cluster "k" and $w_{i,j}$ is the weight of the edge connecting vertices i and j. The summation iterates over every vertex(i)-vertex(j)-cluster(k) triplet, given a vertex set V and a set of clusters C.

In our modified formulation, we maximize $\bar{K}_{\text{intra}}$ and also add a penalty function, $P(M)$, that penalizes very large or very small clusters and prevents degenerate solutions:

$$\underset{x_{i,k},x_{j,k}}{\text{maximize}} \left\{ \sum_{k=1}^{|C|} \frac{|E_{k,k}|}{0.5 \times n_k(n_k - 1)} - \lambda P_k(M) \right\} \tag{2}$$

where

$$n_k = \sum_j x_{j,k} \quad \text{(num vertices in cluster '}k\text{')} \tag{3}$$

$$|E_{k,k}| = \sum_{i,j} w_{i,j} x_{i,k} x_{j,k} \quad \text{(sum edges in cluster '}k\text{')} \tag{4}$$

$$P_k(M) = \max\{0, n_k - M\} \tag{5}$$

OR,

$$P_k(M) = (n_k - M)^2 \tag{6}$$

M is a parameter input specifying a softbound for reasonable cluster sizes. It is determined by judgement and domain expertise. Similarly, λ is a penalty coefficient which is also determined through domain expertise. Our penalty function is a type of ($L1$ or $L2$) regularization [11]. It favors clusters with fewer than M vertices in the first case (Eq. 5) and clusters with roughly M vertices in the second case (Eq. 6), but keeps larger and smaller sized clusters within the feasible set.

Putting it all together,

$$\underset{x_{i,k},x_{j,k}}{\text{maximize}} \left\{ \sum_{k=1}^{|C|} \left[\sum_{i,j} \left(\frac{w_{i,j} x_{i,k} x_{j,k}}{0.5 \times n_k(n_k - 1)} - \lambda P_k(M) \right) \right] \right\} \tag{7}$$

Our model maximizes the mean probability of connection/edge weight within clusters (mean density). In the case of unweighted graphs, it maximizes the similarity between our clusters and cliques, on average (and cliques are arguably very strong clusters). It also avoids mega-clusters, the tendency displayed by many clustering techniques to create extremely large uninformative clusters and degenerately small clusters as well. The numerator in the fractions corresponds to the quantity "z" in Eq. 1. Dividing our numerator by the denominator yields a measure of connection density within each cluster. The denominator also acts as a natural form of cardinality constraint, limiting the number of vertices assigned to each cluster. The penalty function also enforces a soft upper bound on cluster cardinality, in the case of Eq. 5 or a soft cardinality upper and lower bound in the case of Eq. 6. Naturally, we also add constraints forcing each vertex to belong to exactly one cluster.

As mentioned previously, our formulation does not require cardinality constraints for the number of vertices in each cluster, as in the Fan and Pardalos formulation [6]. Together the denominators and penalty function in the inner summation ensure our cluster sizes remain reasonable and faithful to the graph's structure. We do, however, impose a set of constraints that ensure all vertices are assigned to exactly one cluster:

$$\sum_{k=1}^{|C|} x_{i,k} = 1 \quad \forall i \in V \text{ (V is the set of all vertices)} \tag{8}$$

3.2 Overcoming Common Degeneracies

Our formulation is designed to avoid two common degeneracies observed in graph clustering, the appearance of uninformative mega-clusters and "garbage collector clusters". These degeneracies are especially frequent and exacerbated when clustering is conducted using optimization-based approaches that maximize a sum of non-negative terms.

3.2.1 Mega-Clusters

When attempting to cluster vertices, it is not uncommon for an algorithm to group all nodes together into only one or a few very large clusters, leaving the vast majority of clusters very sparsely populated. This is particularly true in the case of an optimization-based approach that maximizes modularity, a widely used clustering quality measure originally introduced by Newman and Girvan in 2004 [19] (Modularity maximization is by far the most common optimization-based approach to clustering.)

This grouping of vertices into a few very large clusters is due to a well-known and documented degeneracy of modularity, known as resolution limit. Fortunato and Bathélemy [9] describe how any clustering quality function that is defined as a sum of quality measures (scores) of individual clusters, as is the case with modularity, suffers from this limit. The authors describe how terms from smaller clusters are dominated by terms from larger clusters. Because the smaller clusters' contribution to the sum is dominated by the larger clusters, the final result is also dominated, which leads to the resolution limit.

Our formulation has two features that prevent this domination of large clusters. First, the $\bar{K}_{\text{intra}}$ portion of our objective function contains a denominator which is proportional to the number of vertices. The resulting ratio ensures independence between the score of each cluster and its number of vertices. Second, our formulation imposes a penalty to the scores of very small or very large clusters, which may even assign a negative score to such clusters.

3.2.2 Garbage Collector Clusters

The garbage collector cluster is a special case of a mega-cluster. When maximizing a sum of unweighted cluster-level quality measurements (nonnegative numbers), an algorithm can return a solution which contains a few very small but very strong very dense clusters with very-high-quality scores and include all remaining vertices in one or a few very large cluster with a very low score. The risk of this pitfall is exacerbated in the case of $\bar{K}_{\text{intra}}$, since all cluster scores are weighted equally, regardless of size. For example, in a graph with three clusters, a clustering algorithm could return two clusters each containing only two connected vertices and lump all remaining vertices in one large poorly connected cluster. In such a case, the aggregate score would be high, because two of the three clusters would have a very high score.

To prevent this situation, many formulations impose cardinality constraints on the clusters (e.g., [6]). We impose a penalty which can potentially assign a negative score to an unreasonably small or large cluster with low density.

4 Numerical Experiments

Numerical implementation is still work in progress and is beyond the scope of this article. We are still in the process of implementing the methods discussed here. We have, however, very early-stage experimental results and find our experiences with an ensemble-like routine to be of interest.

As mentioned previously, the clustering problem is NP-hard. Solving it for a graph of even just moderate size is impossible to do exactly. For this reason, we employ meta-heuristic optimization techniques. When it is hard or impossible to obtain analytical solutions to an optimization problem, there exists a vast array of global optimization search techniques to approximate a globally optimal solution. While each of these techniques has its own specificities, they all explore the feasible set of a problem in some systematic manner.

In our experiments, we use a greedy algorithm and simulated annealing, both separately and in combination, to optimize the $L1$ regularized version of our objective function. We experiment with a greedy algorithm alone, simulated annealing alone, and an ensemble-like routine which uses the best solution yielded by the greedy algorithm as a starting point for simulated annealing.

We deliberately keep graph sizes small to obtain a proof of concept, but do explore sensitivity to graph size (number of edges and vertices). We are still pursuing our experiments on larger graphs, refining our algorithms, and implementing them on high-performance parallel computing platforms, but find the results yielded by our ensemble-like routine worthy of mention.

4.1 Meta-Heuristic Techniques

As mentioned earlier, we use a greedy algorithm, simulated annealing and an ensemble-like routine which combines both techniques. Details about each technique are provided in the following sections.

4.1.1 Simulated Annealing

Simulated annealing [2, 3, 24] is a well-known meta-heuristic optimization technique. It does not systematically apply a greedy logic to determine a move from the current solution to the next one. In simulated annealing, we search the feasible set by moving from a current solution to a new one according to the following set of search rules. If the new solution in our search improves the objective function, then the move is automatically accepted. In the case where the new solution does not improve the objective function, it is accepted if a random draw is lower than an evolving probability of acceptance. This acceptance procedure allows simulated annealing to break out of local optima.

In our implementation, we begin with a simulated annealing algorithm with a random starting point, as is common practice. We also use the best solution returned by our greedy algorithm as a starting point. Finally, using both starting points, we experiment with 1 million and 5 million runs, to examine sensitivity to the number of search iterations.

4.1.2 Greedy Algorithm

We apply a greedy algorithm to vertices sorted in depth-first order. Our decision to sort vertices in a depth-first order is motivated by the work of Creusefond et al. [4] who used a lexicographic depth-first ordering (LexDFS) to detect clusters. While we did not use LexDFS but used a plain depth-first traversal to order vertices instead, we did notice better results than when we greedily assigned vertices in random order. The steps in our algorithm are detailed below:

- **INPUT:**
  - Graph: "G",
  - Number of clusters: "k",
  - Percent of nodes to be randomly assigned: "$pRand$", and
  - Number of repeated runs: "R".
- **OUTPUT:**
  - Vertex-cluster assignments
  - **Steps:**
    - Randomly assign $\lfloor pRand \times$ num vertices$\rfloor$ uniformly to each cluster;
    - Sort remaining vertices using depth-first ordering with a random starting point;
    - While remaining vertices list is not empty:
  - Assign vertex to cluster where objective function improves the most;
  - **Steps:**
    - Record objective function value;
    - Repeat 'R' times and select best run;
    - **Return** best vertex-cluster assignments.

4.1.3 Ensemble-Like Learning Routine

Here, we are guided by the work of Ovelgönne and Geyer-Schulz [20], who combined weak clustering algorithms to obtain better results. Our routine consists of using the best solution obtained by a greedy algorithm that was run for 50,000 iterations as a starting point for our simulated annealing algorithm. We use this seeded starting point, instead of using a random starting point.

4.2 Preliminary Results

We conduct numerical experiments on two different synthetic graphs of varying sizes. These graphs were generated using the stochastic block model [12]. Each graph's characteristics are reported in Table 1. We record the best objective function values (best solutions) returned by each algorithm in (Table 2). We then compare them among themselves, to the synthetic graphs' known features, and the graphs' global density.

While our initial results are still a distance away from the synthetic graph's figures, the clusters identified by the combined greedy-simulated annealing routine fit our definition of a good cluster, since the mean intra-cluster density is greater than the graph's global density, in both experiments ($0.46 > 0.38$ and $0.59 > 0.34$). Our initial results do indicate that the ensemble-like optimization routine which combines a greedy step and simulated annealing yields better results than simulated annealing alone. We find that beginning with a seeded starting point consisting of the solution obtained with just 50,000 iterations of a greedy algorithm significantly improves the results returned by simulated annealing alone.

Our ensemble approach yields better results than running simulated annealing for millions of additional iterations. Increasing the number of iterations of the simulated annealing algorithm does not improve the outcome, while using a seeded starting point does. These results suggest the choice of starting point is critical and that our objective function seems to have multiple local optima. This lack of improvement indicates the presence of a local optimum and outlines the need for an algorithm that will explore the solution set more widely.

Table 1 Test graph characteristics

	Graph 1	Graph 2
Number of vertices ($\|V\|$)	38	104
Number of edges ($\|E\|$)	236	2,036
K (global graph wide density)	0.34	0.38
$\bar{K}_{\text{intra}}$	0.87	0.88
Number of clusters ($\|C\|$)	4	4
Size of solution set	$\sim 10^{22}$	$\sim 10^{62}$

Table 2 Best objective function values

	Graph 1	Graph 2
Simulated annealing (10^6 runs)	0.28	0.30
Simulated annealing (5×10^6 runs)	0.28	0.30
Greedy (50,000 runs)	0.52	0.37
Simulated annealing w/ Greedy (10^6 runs)	0.59	0.46
Simulated annealing w/ Greedy (5×10^6 runs)	0.59	0.46

5 Conclusion and Future Work

We have formulated the graph clustering problem as a combinatorial optimization problem, using a novel formulation that maximizes mean intra-cluster density. Our objective function penalizes unreasonably sized clusters, which eliminates the need for cluster-size constraints used in other formulations in the literature. While numerical implementation remains work in progress, we do note that an ensemble-like approach which combines a greedy first pass with simulated annealing yields far better results than simulated annealing alone, regardless of the number of iterations.

Future work will involve testing on a wider array of graphs, exploring the use of other meta-heuristic solution techniques and implementation on parallelized high-performance computing platforms. Specific focus will be placed on starting point and escaping local optima.

References

1. Aloise, D., Caporossi, G., Hansen, P., Liberti, L., Perron, S., Ruiz, M.: Modularity maximization in networks by variable neighborhood search. In: Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D. (eds.) Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, 13–14 Feb 2012, pp. 113–128 (2012). http://www.ams.org/books/conm/588/11705
2. Bertsimas, D., Tsitsiklis, J.: Simulated annealing. Stat. Sci. **8**(1), 10–15 (1993)
3. Brownlee, J.: Clever Algorithms: Nature-Inspired Programming Recipes, 1st edn. Lulu.com (2011)
4. Creusefond, J., Largillier, T., Peyronnet, S.: Finding compact communities in large graphs. In: Proceedings of the 2015 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining 2015, ASONAM '15, pp. 1457–1464. ACM, New York, NY, USA (2015). https://doi.org/10.1145/2808797.2808868
5. van Dongen, S.: Graph clustering by flow simulation. Ph.D. thesis, Faculteit Wiskunde en Informatica, Universiteit Utrecht (2000)
6. Fan, N., Pardalos, P.M.: Linear and quadratic programming approaches for the general graph partitioning problem. J. Global Optim. **48**(1), 57–71 (2010). https://doi.org/10.1007/s10898-009-9520-1
7. Fan, N., Pardalos, P.M.: Robust optimization of graph partitioning and critical node detection in analyzing networks. In: Proceedings of the 4th International Conference on Combinatorial Optimization and Applications—Volume Part I, COCOA'10, pp. 170–183. Springer, Berlin, Heidelberg (2010). http://dl.acm.org/citation.cfm?id=1940390.1940405
8. Fortunato, S.: Community detection in graphs. Phys. Rep. **486**, 75–174 (2010). https://doi.org/10.1016/j.physrep.2009.11.002
9. Fortunato, S., Barthélemy, M.: Resolution limit in community detection. Proc. Natl. Acad. Sci. **104**(1), 36–41 (2007). http://www.pnas.org/content/104/1/36.abstract
10. Fortunato, S., Hric, D.: Community detection in networks: a user guide. ArXiv e-prints (2016)
11. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, Second Edition: Data Mining, Inference, and Prediction, 2nd ed. Springer Series in Statistics. Springer (2009)
12. Holland, P.W., Laskey, K.B., Leinhardt, S.: Stochastic blockmodels: First steps. Soc. Netw. **5**(2), 109–137 (1983). https://doi.org/10.1016/0378-8733(83)90021-7. http://www.sciencedirect.com/science/article/pii/0378873383900217
13. Jin, J.: Fast community detection by score. Ann. Stat. **43** (2015)

14. Kazakovtsev, L., Antamoshkin, A.: Genetic algorithm with fast greedy heuristic for clustering and location problems. Informatica (Slovenia) **38**(3) (2014). http://www.informatica.si/index.php/informatica/article/view/704
15. Lancichinetti, A., Radicchi, F., Ramasco, J.J., Fortunato, S.: Finding statistically significant communities in networks. PLoS ONE **6**, e18,961 (2011). https://doi.org/10.1371/journal.pone.0018961
16. von Luxburg, U.: A Tutorial on Spectral Clustering. CoRR **abs/0711.0189** (2007). http://arxiv.org/abs/0711.0189
17. Miasnikof, P., Shestopaloff, A., Bonner, A., Lawryshyn, Y.: A statistical performance analysis of graph clustering algorithms. In: Lecture Notes in Computer Science. Springer (2018)
18. Nascimento, M., Pitsoulis, L.: Community detection by modularity maximization using GRASP with path relinking. Comput. Oper. Res. **40**, 3121–3131 (2013)
19. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Phys. Rev. E, Stat. Nonlinear, Soft Matter Phys. **69**, 026,113 (2004)
20. Ovelgönne, M., Geyer-Schulz, A.: An ensemble learning strategy for graph clustering. In: Bader, D.A., Meyerhenke, H., Sanders, P., Wagner, D. (eds.) Graph Partitioning and Graph Clustering, 10th DIMACS Implementation Challenge Workshop, Georgia Institute of Technology, Atlanta, GA, USA, 13–14 Feb 2012, pp. 113–128 (2012). http://www.ams.org/books/conm/588/11705
21. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover Books on Computer Science. Dover Publications (1998). https://books.google.ca/books?id=u1RmDoJqkF4C
22. Prokhorenkova, L.O., Prałat, P., Raigorodskii, A.: Modularity of complex networks models. In: Bonato, A., Graham, F., Prałat, P. (eds.) Algorithms and Models for the Web Graph, pp. 115–126. Springer International Publishing, Cham (2016)
23. Prokhorenkova, L.O., Prałat, P., Raigorodskii, A.: Modularity in several random graph models. In: The European Conference on Combinatorics, Graph Theory and Applications (EUROCOMB'17), Electronic Notes in Discrete Mathematics **61**, 947–953 (2017). https://doi.org/10.1016/j.endm.2017.07.058. http://www.sciencedirect.com/science/article/pii/S1571065317302238
24. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, pp. 92–115. Prentice-Hall Inc. (1995)
25. Schaeffer, S.E.: Survey: graph clustering. Comput. Sci. Rev. **1**(1), 27–64 (2007). https://doi.org/10.1016/j.cosrev.2007.05.001
26. Tasgin, M., Herdagdelen, A., Bingol, H.: Community Detection in Complex Networks Using Genetic Algorithms. ArXiv e-prints (2007)
27. Weisstein, E.: Clique. MathWorld–A Wolfram Web Resource (2018). http://mathworld.wolfram.com/Clique.html
28. Yang, J., Leskovec, J.: Defining and Evaluating Network Communities Based on Ground-truth. CoRR **abs/1205.6233** (2012). http://arxiv.org/abs/1205.6233