

We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

5,800

Open access books available

142,000

International authors and editors

180M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com



Clustering Network Data Using Mixed Integer Linear Programming

*Harun Pirim, Amin Aghalari
and Mohammad Marufuzzaman*

Abstract

Network clustering provides insights into relational data and feeds certain machine learning pipelines. We present five integer or mixed-integer linear programming formulations from literature for a crisp clustering. The first four clustering models employ an undirected, unweighted network; the last one employs a signed network. All models are coded in Python and solved using Gurobi solver. Codes for one of the models are explained. All codes and datasets are made available. The aim of this chapter is to compare some of the integer or mixed-integer programming network clustering models and to provide access to Python codes to replicate the results. Mathematical programming formulations are provided, and experiments are run on two different datasets. Results are reported in terms of computational times and the best number of clusters. The maximum diameter minimization model forms compact clusters including members with a dominant affiliation. The model generates a few clusters with relatively larger size. Additional constraints can be included to force bounds on the cluster size. The NP-hard nature of the problem limits the size of the dataset, and one of the models is terminated after 6 days. The models are not practical for networks with hundreds of nodes and thousands of edges or more. However, the diversity of models suggests different practical applications in social sciences.

Keywords: network clustering, signed networks, integer programming, python application, social networks

1. Introduction

Network and graph terms are used interchangeably, while the latter recalls more of a mathematical phenomenon since the study of networks has roots in graph theory [1]. Complex systems of real life can be modeled as networks with certain features. Watts and Strogatz [2] propose that real-world systems exhibit small world characteristics with a connection topology between regular and random. Zhang and Zhang [3] argue based on simulations that the real networks have longer average shortest paths than expected to allow a substantial increase in the network modularity. That is a trade-off between network efficiency and modularity. Albert et al. [4] claim scale-free characteristic of real-world networks that makes them vulnerable to targeted attacks but robust against random failures. Ravasz and

Barabási [5] show that the scale-free and high degree clustering characteristics of real networks emerge as a result of hierarchical organization. Balkundi and Harrison [6] investigate the effect of network structure on the team viability and performance. Similarly, Balague et al. [7] report quantification of individual player contributions to the team performance using a network approach. Barabási and Oltvai [8] discuss network biology to understand the internal organization and evolution of a cell. Those studies convince that any system of objects with certain relations can be modeled as a network. The objects (i.e., nodes of the network) can represent individuals, animals, routers, documents, etc., where the relations (i.e., edges of the network) quantify a proximity of a relationship or existence of a relationship (i.e., binary networks).

Network analysis focuses on both descriptive and predictive tasks within the broad scope of network science since network structures at node, edge, network, or intermediate levels reveal information about a function or an emergent behavior. Clustering network data is essential to network analysis to support downstream machine learning tasks such as prediction of a node's label or directly describe a group of similar objects. There is an abundance of network clustering algorithms [9–12] mainly categorized under community structure finding title. However, mixed-integer linear programming (MILP) formalism, despite its NP-hard nature, provides a flexible representation of the clustering problem and generates a global optimal solution to the problem [13]. The ease of algebraic representation of the problem including the objective function and the constraints and inclusion of special constraints imposed by the nature of the application on hand, availability of special modeling languages and solvers make MILP clustering suitable for many applications. It is conventional to design heuristic algorithms to support the MILP formalism to solve larger problems with thousands of nodes and millions of edges compromising the global optimality [14].

The aim of this chapter is to compare some of the integer linear (ILP) and mixed-integer linear programming (MILP) network clustering models and to provide access to Python codes to replicate the results. Mathematical programming formulations are provided, and experiments are run on two different datasets. Results are reported in terms of computational times and the best number of clusters.

We summarize three ILP and two MILP clustering models providing Python codes and the datasets to generate the results. The chapter is structured as follows. Section 2 introduces the ILP and MILP models. Section 3 presents the Python application and results, followed by the discussion and the conclusion sections.

2. Pure and mixed-integer linear programming models

Given a graph $G(V, E)$ with the set of vertices V and the set of edges E , the clustering models here generate a crisp clustering $C = \{C_i\}_{i \in I}$ with clusters C_i comprising nonoverlapping members [15]. Clustering problem is ill-posed since there is no consensus on what defines a cluster. A general adoption is having clusters with similar members and separate clusters holding different members. Bittner et al. [16] propose combining cluster analysis solution with the information gained from subjective choices.

The first model, referred to as **(M1)**, addresses the well-studied graph coloring problem [17]. The problem inherently requires finding the minimum number of

colors to assign the nodes of the network such that no adjacent nodes have the same color. The corresponding clustering problem can be viewed as finding clusters of diverse members.

$$(M1) : \min \sum_{c=1}^k y_c$$

st

$$\sum_{c=1}^k x_{ic} = 1, \forall i \in V(1) \quad (1)$$

$$x_{ic} + x_{jc} \leq 1, \forall (i, j) \in E(2)$$

$$y_c \geq x_{ic}, \forall i \in V, \forall c \in C(3)$$

$$y_c, x_{ic} \in \{0, 1\}, \forall i \in V, \forall c \in C(4)$$

In (M1), x_{ic} are binary variables taking value 1 if member i is assigned to cluster c , 0 otherwise. y_c are binary variables taking value 1 if color c is used, 0 otherwise. The objective is to minimize the number of colors used. The initial number of colors, k is a user-defined parameter. k can be chosen arbitrarily as long as the feasibility of the model is retained. The first set of constraints (1) implies each member is assigned to one cluster only. The second set of constraints (2) avoids adjacent nodes sharing the same color. The third set of constraints (3) relates two different sets of variables to each other. A color c is used once at least a node is assigned the color. The last set of constraints (4) implies domain of the variables to be binary.

The second model (M2) minimizes the total distance within clusters [18].

$$(M2) : \min \sum_{i=1}^{N-1} \sum_{j=i+1}^N d_{ij} y_{ij}$$

st

$$\sum_{c=1}^k x_{ic} = 1, \forall i \in V(1) \quad (2)$$

$$\sum_{i=1}^N x_{ic} \geq 1, \forall c \in C(2)$$

$$y_{ij} \geq x_{ic} + x_{jc} - 1, i = 1, \dots, N-1; j = i+1, \dots, N; \forall c \in C(3)$$

$$y_{ij} \geq 0, i = 1, \dots, N-1; j = i+1, \dots, N(4)$$

$$x_{ic} \in \{0, 1\}, \forall i \in V, \forall c \in C(5)$$

N is the number of objects to cluster. k is the number of clusters, a user-defined parameter. y_{ij} are binary variables taking value 1 if members i, j are in the same cluster, 0 otherwise. d_{ij} are the distance values between members i, j . The first set of constraints (1) is similar to the first model. The second set of constraints avoids empty clusters (2). The third set of constraints (3) is to relate variables to each other in a linear fashion. The last two sets of constraints (4, 5) restrict the domains of variables.

The third model (**M3**) minimizes the maximum diameter of clusters [19].

$$(M3) : \min D_{max}$$

st

$$D_c \geq d_{ij}(x_{ic} + x_{jc} - 1) \forall i \in V, \forall j \in V, \forall c \in C(1)$$

$$\sum_{c=1}^k x_{ic} = 1 \forall i \in V(2)$$

$$\sum_{i=1}^N x_{ic} \geq 1 \forall c \in C(3)$$

$$D_{max} \geq D_c \forall c \in C(4)$$

$$x_{ic} \in \{0, 1\} \forall i \in V, \forall c \in C(5)$$

$$D_c \geq 0 \forall c \in C(6)$$

In addition to the variable, parameter, and constraint definitions for the second model (**M2**), an additional continuous variable, namely, D_c , is introduced in (**M3**). The first set of constraints (1) implies D_c to be the diameter of cluster c , i.e., the longest shortest path in the cluster. Second and third set of constraints (2, 3) are similar to (1, 2) sets of constraints in (**M2**). The fourth set of constraints (4) implies D_{max} is the maximum of D_c values defining the largest diameter. The last two constraints (5, 6) are domain-related.

The fourth model (**M4**) maximizes the modularity metric [15]. Modularity quantifies the compactness of clusters compared with a random model. The clusterings resulting in modularity values greater than 0.3 and closer to one are perceived to have high modularity.

$$(M4) : \max \frac{1}{2m} \sum_{i,j} m_{i,j} (1 - x_{i,j})$$

st

$$x_{i,l} \leq x_{i,j} + x_{j,l} \forall i, j, l, i \neq j \neq l(1)$$

$$x_{i,j} \in \{0, 1\} \forall i, j(2)$$

x_{ij} variables take 1 if members i, j are in different clusters, 0 otherwise. $m_{i,j}$ are the elements of the modularity matrix defined as follows.

$$m_{i,j} := a_{i,j} - \frac{d_i d_j}{2m} \quad (5)$$

a_{ij} are elements of adjacency matrix being 1 if i, j are connected, 0 otherwise. d_i is the degree of node i . m is the number of edges for the graph. The first set of constraints (1) define transitivity or clique. If nodes i, j and j, l are in the same cluster then nodes i, l must be in the same cluster either. The second set of constraints (2) restricts variables to binary.

The fifth model (**M5**) is designed to cluster signed networks [20]. Signed networks assume negative or positive edges between objects. The ideal clustering minimizes the frustration edges, i.e., positive edges between clusters or negative edges within clusters.

$$(M5) : \min \sum_{(i,j) \in E} f_{ij}$$

st

$$\sum_{c \in C} x_{ic} = 1 \forall i \in V(1)$$

$$f_{ij} \geq x_{ic} - x_{jc} \forall (i,j) \in E^+, \forall c \in C(2) \quad (6)$$

$$f_{ij} \geq x_{ic} + x_{jc} - 1 \forall (i,j) \in E^-, \forall c \in C(3)$$

$$x_{ic} \in \{0, 1\} \forall i \in V, \forall c \in C(4)$$

$$f_{ij} \in \{0, 1\} \forall (i,j) \in E(5)$$

f_{ij} are binary frustration variables. The first set of constraints (1) are similar to (1) in (M2). The second set of constraints (2) implies that frustration exists between nodes i, j if they belong to different clusters although they share a positive edge. The third set of constraints (3) implies that frustration exists between nodes i, j if they belong to the same cluster sharing a negative edge. The last two sets of constraints (4, 5) are domain-related ones.

3. Python application and results

The Python language is used to code the algebraic models, and the Gurobi solver [21] is used to solve them optimally. Python/GUROBI is one of the most widely used package in solving optimization problems. In our demonstration, we use model 3 (M3) for its simplicity to introduce to the readers. However, the link to the repository of the source codes for all other models (i.e., M1–M5) is provided for the interested readers. Due to the availability of open-source packages and simplicity in maintaining the coding paradigm, the Python programming language is utilized in this study as compared with the other available languages.

We explain the code for the third model only to save space and since the algebraic expressions are quite similar. All codes and data are available through: <https://github.com/harunpirim/networkclustering>. UKfaculty data [22] is about the friendship network of 81 faculty of a UK university. The data is retrieved from the R package `igraphdata`. The original data represents a directed network with 817 edges. We collapsed the edges to transform the network into an undirected one. There would be an edge between authors if there was at least one directed edge between them. The undirected network has 577 edges. The shortest path distances are obtained from the original directed network ignoring the weights. A signed network is retrieved from the `signet` R package [23] to be employed in model 5. The network represents 91 countries and 521 positive or negative relationships. The inclusion of the datasets is decided based on the scalability and existence of edge signs. The transformation of directed edges is to fit the data to the selected network clustering models.

After reading the input files and populating the parameters (e.g., d_{ij}), a null GUROBI model (**model3**) is created as follows:

```
model3 = Model()
```

Next, the model variables, namely x_{ic} , D_c , and D_{max} , are constructed and added to the **model3** model as follows:


```

X_ic, D_c = {}, {}
model3 = Model()
for c in C:
    for i in V:
        X_ic[i,c] = model3.addVar(vtype=GRB.BINARY, lb=0, ub=1, name="X_ic[%s, %s]" % (i,c))

for c in C:
    D_c[c] = model3.addVar(vtype=GRB.CONTINUOUS, lb=0, name="D_c[%s]" % (c))

D_max = model3.addVar(vtype=GRB.CONTINUOUS, lb=0, name="D_max")

```

Constraint set (1) of **M3** can be added to GUROBI's **model3** model as follows:

```

for i in V:
    for j in V:
        for c in C:
            model3.addConstr(D_c[c] >= d_ij[i,j] *(X_ic[i,c] +X_ic[j,c] -1), name='const1')

```

Likewise, constraint sets (2)–(4) of **M3** can be added to GUROBI's **model3** model as follows:

```

for i in V:
    model3.addConstr(quicksum(X_ic[i,c] for c in C) == 1, name= 'const2')

for c in C:
    model3.addConstr(quicksum(X_ic[i,c] for i in V) >= 1, name= 'const3')

for c in C:
    model3.addConstr(D_max >= D_c[c], name='const4')

```

The objective function of **M3** can be added to GUROBI's **model3** model as follows:

```

model3.setObjective(D_max, GRB.MINIMIZE)

```

where, GRB.MINIMIZE represents minimizing the objective function of **M3**. The objective function of **M3** minimizes the maximum diameter of clusters. As such, GRB.MINIMIZE is used. Following creating the model and adding all the variables and constraints, the **model3** model is optimized as follows:

```

model3.optimize()

```

Cluster membership vector is stored in the **val_map** (i.e., the first line in the code below) dictionary. The membership information is used in visualization.

```

val_map = {}
for c in C:
    for i in V:
        if X_ic[i,c].x > 0:
            val_map[i] = c

```

The network is visualized in **Figure 1**. The distinct node colors represent the clusters. Here the number of clusters is 8 that is the optimal cluster size within the range of 2–10. The sensitivity of the objective function value when the number of clusters changes is plotted.

The largest cluster (cluster 6 in **Figure 2**) has 20 members, while the smallest one (cluster 7 in **Figure 2**) has 5 members. Remaining cluster sizes are 7, 7, 8, 9, 11, 14. The dataset provides group information with respect to faculty affiliated to three distinct universities. **Figure 2** illustrates the faculty (numbers on x axis) affiliations (numbers 1, 2, 3 on y axis) in each of eight clusters (numbers 1–8 on y axis). Broken green lines corresponding to faculties 50 and 70 imply unknown affiliations. Clusters include at least two affiliations. Most of the clusters have a dominant affiliation. When the number of clusters increases to 9, the optimal solution produces two clusters of sizes 6 and 10 with single affiliated members.

The best number of clusters between 2 and 10 and solution times are listed in **Table 1**. Models **M1**, **M3**, **M4** optimal cluster numbers are close to each other. They

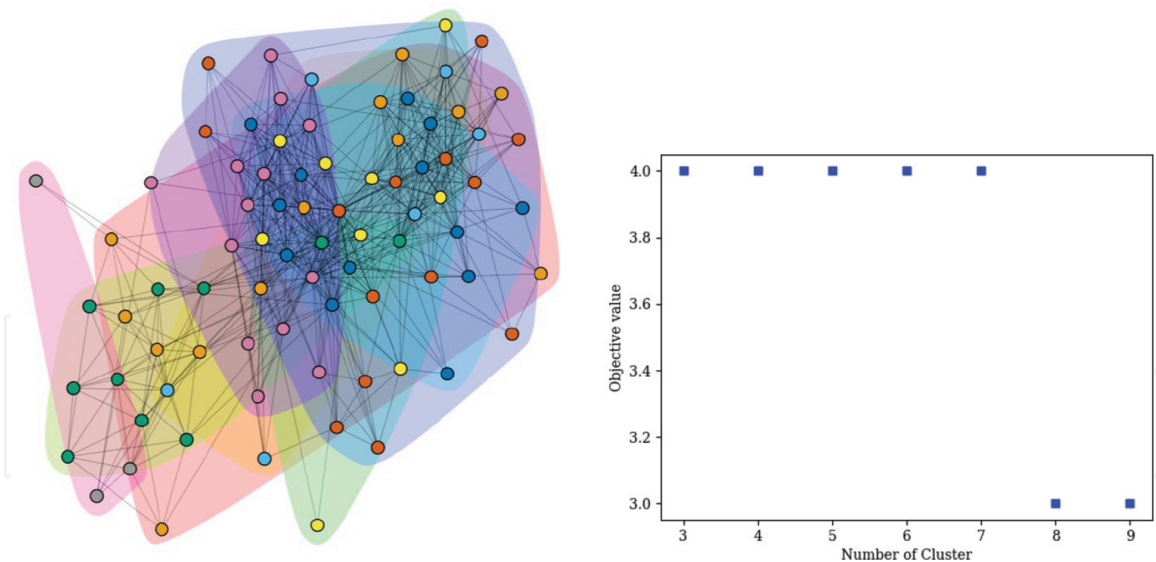


Figure 1.
 Clusters of the UKfaculty network with the number of clusters eight.

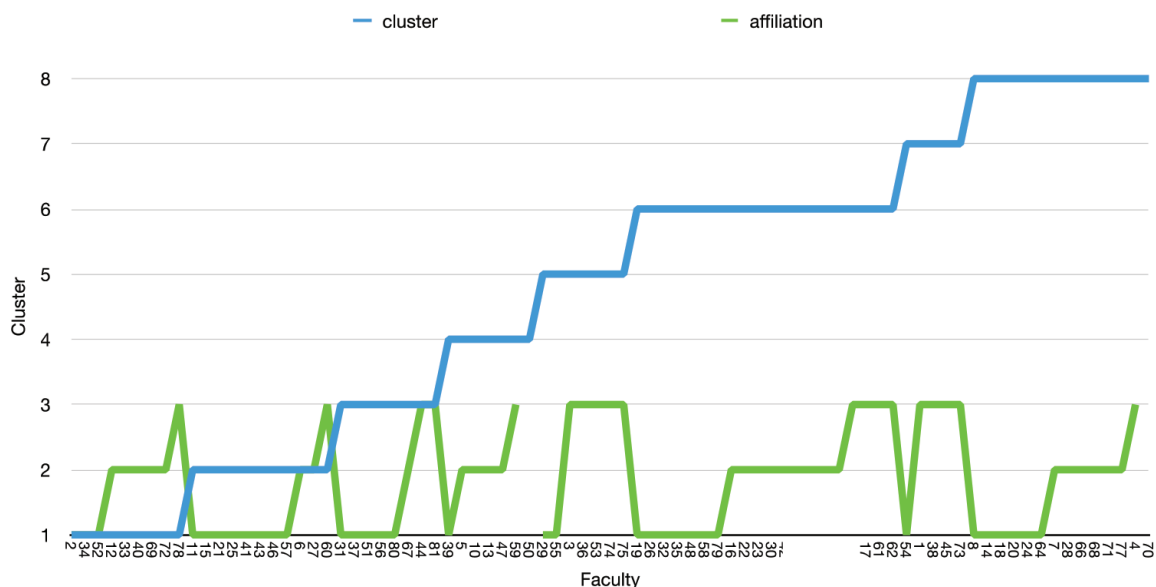


Figure 2.
 Faculty affiliations in clusters.

used the same dataset to generate clusters. M4 has seven clusters without user input of number of clusters, while M1 and M3 are provided the number of clusters between 2 and 10 to decide on the best number of clusters. M1 finds the optimal solution fastest among all. M2 is computationally the most expensive since it could not find the optimal solution in 6 days. M5 used a signed network dataset. It has the least number of clusters to find the optimal solution once provided number of clusters between 2 and 10. It has the second best solution time among all models with a slightly different network structure with more nodes and edges compared with other models.

4. Discussion

The model minimizing the maximum of cluster diameters can be expected to generate clusters of similar sizes. However, the application of the third model (M3) on the UKfaculty data formed three clusters of size greater than 10. The result

Model	Best number of clusters	Solution time (CPU s)	Data size (V, E)
M1	9	0.1	(81, 577)
M2	—	Stopped after 6 days	(81, 577)
M3	8	2.50	(81, 517)
M4	7	6.19	(81, 517)
M5	4	0.48	(91, 521)

Table 1.
Model results.

reveals that the cluster sizes are dependent on the density of the relationships existing in the data. An alternative optimal solution with the cluster size increased from eight to nine formed more uniform-sized clusters. Two cluster sizes were greater than 10. This observation suggests defining extra constraints to force bounds on cluster sizes based on the subject matter expert opinion. Such constraints can improve the results compromised with the extra computational time.

The first model is more suitable for generating clusters with diverse members since the adjacent nodes are assigned to different clusters while optimizing the number of clusters. Second and third models generate compact clusters minimizing the distances inside the clusters. Both models can be used where individual cluster members are required to be close to each other. Third one is more relaxed compared with the second one minimizing the diameter of clusters instead of minimizing the total distance between members of clusters. The second model is computationally the most expensive among all. The fourth model maximizes the modularity metric to generate compact clusters compared with a random model. The modularity metric is widely used to design heuristic algorithms to cluster large scale datasets. The advantage of the modularity optimization is that it does not require the number of clusters as a user input. The last model is designed for clustering signed networks. Signed networks have many applications in social science.

The maximum diameter minimization model (M3) can form dense clusters as in community structure finding approaches [14, 15]. However, mathematical programming approaches to both modularity maximization and modularity density maximization are computationally more costly compared to M3. **Table 1** reports computational efficiency of M3 compared with a modularity maximization model (M4). Many heuristic algorithms are developed [11] for modularity maximization despite the reported deficiencies such as resolution and degeneracy problems. Designing heuristics to optimize M3 can contribute to heuristic approaches in network clustering as such.

5. Conclusion

In this chapter, we provided pure and mixed-integer linear programming formulations with diverse objective functions for crisp clustering problems. The first formulation finds the minimum number of clusters to assign objects that do not share a common edge. The second formulation minimizes the total distance inside clusters. This model is computationally more expensive than others, unable to find the optimal solution in 6 days even for two clusters. The third formulation minimizes the longest shortest path inside clusters. The fourth formulation maximizes the modularity metric to find optimal clustering. The last formulation is designed for signed networks to minimize the number of frustration edges. All models are

coded in Python and solved optimally except the second one. Optimal clustering results are reported with respect to solution times and the number of clusters.

Application of the third model on UKfaculty dataset generated clusters with at least two affiliations. However, most of the clusters have a dominating affiliation. The model produces compact clusters in this sense. When the number of clusters is increased from eight to nine, dominating affiliations are distinguished better. There exist two clusters with a single affiliation. Three clusters out of eight have sizes greater than 10. Increasing the number of clusters while the solution stays optimal or imposing bounds on cluster sizes can form clusters with similar sizes. MILP formulations allow flexibility of defining extra constraints to improve results. The third model can be employed to social network datasets to reveal close relationships as such. The models are not practical for networks with hundreds of nodes and thousands of edges or more. However, heuristics can be developed to solve the models for larger datasets. Defining extra constraints based on expert opinions and designing heuristics are future research directions. The code for the third model is explained in detail. All codes and data are publicly available to make a reproduction of the results possible.


IntechOpen

Author details

Harun Pirim*, Amin Aghalari and Mohammad Marufuzzaman
Mississippi State University, Starkville, MS, USA

*Address all correspondence to: pirim@ise.msstate.edu

IntechOpen

© 2022 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

References

- [1] Barabasi AL, Posfai M. Network Science. United Kingdom: Cambridge University Press; 2016
- [2] Watts DJ, Strogatz SH. Collective dynamics of 'small-world' networks. *Nature*. 1998;**393**(6684):440-442
- [3] Zhang Z, Zhang J. A big world inside small-world networks. *PLoS One*. 2009; **4**(5):e5686
- [4] Réka A, Jeong H, Barabási A-L. Error and attack tolerance of complex networks. *Nature*. 2000;**406**(6794):378-382
- [5] Ravasz E, Barabási A-L. Hierarchical organization in complex networks. *Physical Review E*. 2003;**67**(2):026112
- [6] Balkundi P, Harrison DA. Ties, leaders, and time in teams: Strong inference about network structure's effects on team viability and performance. *Academy of Management Journal*. 2006;**49**(1):49-68
- [7] Balague N et al. Overview of complex systems in sport. *Journal of Systems Science and Complexity*. 2013;**26**(1): 4-13
- [8] Barabasi A-L, Oltvai ZN. Network biology: Understanding the cell's functional organization. *Nature Reviews Genetics*. 2004;**5**(2):101-113
- [9] Chakraborty T et al. Metrics for community analysis: A survey. *ACM Computing Surveys*. 2017;**50**(4):1-37
- [10] Rossetti G, Cazabet R. Community discovery in dynamic networks: A survey. *ACM Computing Surveys*. 2018; **51**(2):1-37
- [11] Javed MA et al. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*. 2018;**108**:87-111
- [12] Khomami MMD et al. CFIN: A community-based algorithm for finding influential nodes in complex social networks. *The Journal of Supercomputing*. 2021;**77**:2207-2236
- [13] Rysz M, Pajouh FM, Pasilliao EL. Finding clique clusters with the highest betweenness centrality. *European Journal of Operational Research*. 2018; **271**(1):155-164
- [14] de Santiago R, Lamb LC. A ground truth contest between modularity maximization and modularity density maximization. *Artificial Intelligence Review*. 2020; **53**(6):4575-4599
- [15] Agarwal G, Kempe D. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*. 2008; **66**(3):409-418
- [16] Bittner P, Eckes C, Wolff KE. Conceptual Meaning of Clusters. *Classification in the Information Age*. Berlin, Heidelberg: Springer; 1999. pp. 279-286
- [17] Shirokikh O, Stozhkov V, Boginski V. Combinatorial optimization techniques for network-based data mining. In: *Handbook of Combinatorial Optimization*. Springer; 2013. pp. 631-672
- [18] Nascimento MCV, Toledo FMB, de Carvalho ACPLF. Investigation of a new GRASP-based clustering algorithm applied to biological data. *Computers & Operations Research*. 2010;**37**(8): 1381-1388
- [19] Sağlam B et al. A mixed-integer programming approach to the clustering problem with an application in customer segmentation. *European Journal of Operational Research*. 2006; **173**(3):866-879

[20] Aref S, Neal ZP. Identifying hidden coalitions in the US House of Representatives by optimally partitioning signed networks based on generalized balance. *Scientific Reports*. 2021;**11**(1):1-9

[21] Gurobi Optimization LLC. Gurobi Optimizer Reference Manual. 2021. Available from: <https://www.gurobi.com>

[22] Nepusz T et al. Fuzzy communities and the concept of bridgeness in complex networks. *Physical Review E*. 2008;**77**(1):016107

[23] Schoch D. signnet: An R package to analyze signed networks. 2020. Available from: <https://github.com/schochastics/signnet>