Preprocessing Technique for Cluster Editing via Integer Linear Programming

Luiz Henrique Nogueira Lorena[1(✉)], Marcos Gonçalves Quiles[1], André Carlos Ponce de Leon Ferreira de Carvalho[2], and Luiz Antonio Nogueira Lorena[3]

[1] Federal University of São Paulo - UNIFESP, São José dos Campos, Brazil
luiz-lorena@hotmail.com, quiles@unifesp.br
[2] University of São Paulo - USP, São Carlos, Brazil
andre@icmc.usp.br
[3] National Institute for Space Research - INPE, São José dos Campos, Brazil
luizlorena54@gmail.com

Abstract. This paper addresses the Cluster Editing problem. The objective of this problem is to transform a graph into a disjoint union of cliques using a minimum number of edge modifications. This problem has been considered in the context of bioinformatics, document clustering, image segmentation, consensus clustering, qualitative data clustering among others. Here, we focus on the Integer Linear Programming (ILP) formulation of this problem. The ILP creates models with a large number of constraints. This limits the size of the problems that can be optimally solved. In order to overcome this limitation, this paper proposes a novel preprocessing technique to construct a reduced model that feasibly maintains the optimal solution set. In comparison to the original model, the reduced model preserves the optimal solution and achieves considerable computational time speed-up in the experiments performed on different datasets.

Keywords: Cluster Editing · Preprocessing technique · Unsupervised learning

1 Introduction

The Cluster Editing (CE) problem [5], also referred in the literature as Correlation Clustering [2], ask to transform an undirected graph G by minimizing the number of editions, i.e., insertions or deletions of edges, to create a vertex-disjoint union of cliques. Figure 1 shows a CE instance were three editions are made on the graph resulting on two disjoint cliques represented by the vertices set {A, B, C, D} and {E, F, G}.

This problem has been considered in the context of bioinformatics [4, 6], clustering documents [2], image segmentation [14], consensus clustering [1, 10], and qualitative data clustering [10, 11]. The CE problem is a NP-hard problem [2, 19], thus heuristics, approximations and data reduction methods were proposed in the literature [3, 9, 12, 18].

D.-S. Huang et al. (Eds.): ICIC 2018, LNCS 10954, pp. 287–297, 2018.
https://doi.org/10.1007/978-3-319-95930-6_27

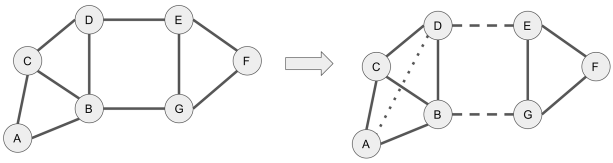

Fig. 1. Cluster Editing instance. Solid lines represent edges that are maintained, dashed lines are removed edges and dotted lines are edges that were inserted.

Grotschel and Wakabayashi [11] introduced an ILP model for the CE. They derived it from a mathematical analysis of the corresponding problem polytope, proposing several partition inequalities for this purpose. As there are an exponential number of these inequalities, they followed a cutting plane approach where the inequalities are added to the Linear Program only if a current fractional solution violates them.

The ILP creates models with a large number of constraints that limits the size of the problems that can be optimally solved. Therefore, here, we propose a preprocessing technique to construct a reduced model. In comparison to the original model, the reduced model preserves the optimal solution and achieves remarkable computational time speed up in the experiments performed on different datasets.

This work is organized as follows. In Sect. 2 the problem is defined and the integer linear programming formulation is presented. Section 3 presents the new preprocessing technique. The computational experiments are provided in Sect. 4. Finally, conclusions are drawn in Sect. 5.

2 Cluster Editing via ILP

The CE problem can be formulated as a maximization or minimization problem [7]. This paper considers a minimization version of graph clustering where the goal is to minimize the number of editions (edges deleted between clusters plus the number of edges inserted inside clusters).

The following notation is introduced to explain the proposed preprocessing technique. Given an undirected graph $G = (V, E)$ where V is the set of vertices and E is the set of edges. The edge weight values, represented by w_{ij}, are 1 if the edge exists in the graph and –1 for missing edges. The number of vertices in the graph is defined as $n = |V|$, while $m = |E|$ represent its number of edges.

The following ILP formulation can be used for cluster editing [6, 11]:

$$(CE_{ILP}): \text{ Minimize } \sum_{e \in E} w(e) - \sum_{i<j} w_{ij} x_{ij}$$

$$\text{subject to } \quad x_{ij} + x_{jk} - x_{ik} \leq 1 \quad i<j<k \tag{1}$$

$$x_{ij} - x_{jk} + x_{ik} \leq 1 \quad i<j<k \tag{2}$$

$$-x_{ij} + x_{jk} + x_{ik} \leq 1 \quad i < j < k \tag{3}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in [1..n]$$

where $x_{ij} = 1$ if i and j are part of the solution and 0 otherwise.

The edge editions are considered on (CE_{ILP}) model solution in the following way: an edge is inserted if $x_{ij} = 1$ for $w_{ij} = -1$ and is removed if $x_{ij} = 0$ for $w_{ij} = 1$. Therefore, the objective function returns the minimum number of editions.

Constraints (1–3) are called "transitivity constraints". Constraint 1, for instance, enforces that: if vertex i is in the same cluster as vertex j, and j is in the same cluster as k then i is in the same cluster as k.

3 Preprocessing Technique

The (CE_{ILP}) model has $O(n^2)$ variables and $O(n^3)$ constraints, which creates models with a large number of redundancies. Though not critical to solving the problem, these constraints affect the solver efficiency and might prohibit its usage.

Grotschel and Wakabayashi [11], in 1989, were the first to propose a strategy to deal with such redundancy. A cutting plane algorithm was created to identify violated constraints during the execution of a relaxed version of this problem. It was confirmed experimentally that, for many instances, the cutting plane ends with a small fraction of the transitivity constraints. This fact evidences a great redundancy of transitivity constraints in the original model (CE_{ILP}).

Other techniques, introduced in the context of clique partitioning problem [16] and modularity optimization in complex networks [8], try to identify redundancies in the ILP model by analyzing the graph representation of the clustering instance. Recently, Nguyen et al. [17] generalized the approach of Dinh et al. [8] to some constrained clustering problems. All those techniques were capable of reducing the size complexity of the transitivity constraints from $O(n^3)$ to $O(nm)$.

In the context of Cluster Editing, Bocker et al. [5, 6] introduced techniques that focus on reducing the problem size instead of improve the ILP model. They identify patterns that can be removed from the problem instance without changing the groups obtained in the optimal solution. The reduced problem is solved exactly by using the cutting plane proposed by Grotschel and Wakabayashi [11] and a fixed parameter branching algorithm. The experimental results shows that those techniques were capable of solving large graphs with 1000 vertices and several thousand edge modifications.

Here a new approach is introduced, focusing on the identification of the ILP model redundancy. It tries to identify redundancies by analyzing the graph representation of the clustering instance [8, 16]. Our approach is based on the identification of triangles formed by edges (graph edges and missing edges) that correspond to the transitivity constraints of a model (CE_{ILP}).

Figure 2 presents the edge weight distribution within triangles considered by the transitive constraints of the (CE_{ILP}) model. The analysis of such triangles helps to

identify constraints that do not need to be considered in the model as they do not result in editions.

Transitivity constraints corresponding to triangles T1, T3 and T4 do not need to be considered because of the following reasons:

- T1: the optimization objective of (CE_{ILP}) tries to set all variables $x_{ij} = x_{jk} = x_{ik} = 1$ for $w_{ij} = w_{jk} = w_{ik} = 1$ since all transitivity constraints are satisfied.
- T2: the optimization objective of (CE_{ILP}) tries to set all variables $x_{ij} = x_{jk} = x_{ik} = 0$ for $w_{ij} = w_{jk} = w_{ik} = 0$ since all transitivity constraints are satisfied.
- T3: the optimization objective of (CE_{ILP}) tries to set the value 1 to the variable relative to the positive edge weight and 0 to the remaining variables. This satisfies all the transitivity constraints.

Constraints that deals with triangles of type T2 must be considered as the transitivity constraints are not satisfied when variables corresponding to positive edges are set to 1. To satisfy those constraints variables corresponding to the negative edge weights must be set to 1 which leads to one edge editing.

There is another possibility, removing an edge of the graph. Those circumstances are considered in Fig. 3. There are three possible variants of such triangle depending on vertex order:

- T2A: considering the optimization objective of (CE_{ILP}), the best possibility of editing (1 edition) that satisfy all transitivity constraints are the following:
  - $x_{ij} = x_{jk} = x_{ik} = 1$ (1 edge inserted)
  - $x_{ij} = 1, x_{jk} = x_{ik} = 0$ (1 edge removed)
  - $x_{jk} = 1, x_{ij} = x_{ik} = 0$ (1 edge removed)
- T2B: considering the optimization objective of (CE_{ILP}), the best possibility of editing (1 edition) that satisfy all transitivity constraints are the following:

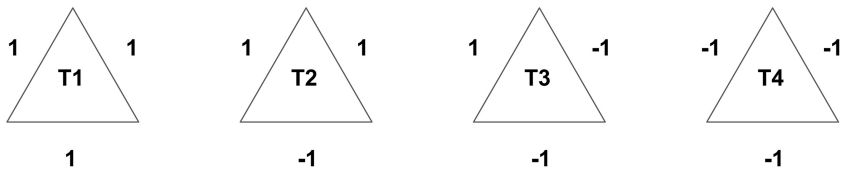

Fig. 2. Possible edge weights distribution.

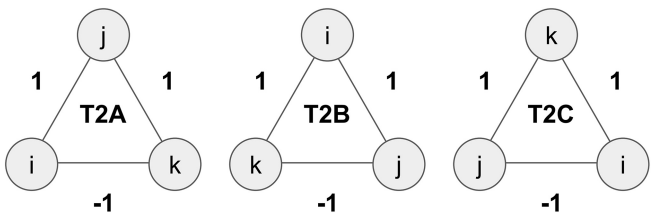

Fig. 3. Possible edge label distribution for triangles of type T2.

- $x_{ij} = x_{jk} = x_{ik} = 1$ (1 edge inserted)
- $x_{ij} = 1, x_{jk} = x_{ik} = 0$ (1 edge removed)
- $x_{ik} = 1, x_{ij} = x_{jk} = 0$ (1 edge removed)

- T2C: considering the optimization objective of (CE_{ILP}), the best possibility of editing (1 edition) that satisfy all transitivity constraints are the following:
  - $x_{ij} = x_{jk} = x_{ik} = 1$ (1 edge inserted)
  - $x_{ik} = 1, x_{ij} = x_{jk} = 0$ (1 edge removed)
  - $x_{jk} = 1, x_{ij} = x_{ik} = 0$ (1 edge removed)

T2 triangles are also known as conflict triples in literature and are the roots to some data reduction methods [5, 6]. In this paper, the information of the edge weight distribution within triangles is used directly in the model (CE_{ILP}) to identify what transitivity constraints should be considered while constructing the ILP model. The data is not reduced or modified, and only transitivity constraints corresponding to T2 triangles need to be taken into account. Hence, the following reduced model is proposed:

$$(CER_{ILP}): \text{ Minimize } \sum_{e \in E} w(e) - \sum_{i<j} w_{ij} x_{ij}$$

$$\text{subject to} \quad x_{ij} + x_{jk} - x_{ik} \leq 1 \quad i, j, k \in S1 \tag{4}$$

$$x_{ij} - x_{jk} + x_{ik} \leq 1 \quad i, j, k \in S2 \tag{5}$$

$$-x_{ij} + x_{jk} + x_{ik} \leq 1 \quad i, j, k \in S3 \tag{6}$$

$$x_{ij} \in \{0, 1\} \quad i, j \in [1..n]$$

where

$$S1 = \{i<j<k | w_{ij} = +1 \wedge w_{jk} = +1 \wedge w_{ik} = -1\}$$

$$S2 = \{i<j<k | w_{ij} = +1 \wedge w_{jk} = -1 \wedge w_{ik} = +1\}$$

$$S3 = \{i<j<k | w_{ij} = -1 \wedge w_{jk} = +1 \wedge w_{ik} = +1\}$$

The sets S1, S2, and S3 enforce that only constraints corresponding to T2 triangles must be considered while creating the model. This can be considered as a preprocessing technique that produces a model (CER_{ILP}) with a small number of constraints in comparison to the original model (CE_{ILP}). For instance, considering the example depicted in Fig. 1, the model (CE_{ILP}) has 105 constraints in contrast to the model (CER_{ILP}), which has only 11 constraints. Both models find the best number of editions, but the reduced model has 1.89 of speedup in computational times.

4 Experimental Results

The experiments and algorithms were coded in C++14 and executed on a computer with the following configuration: Intel Core i7-6770HQ (3,5 GHz) with 32 GB RAM running Windows 10 64-Bit. The commercial solver IBM ILOG CPLEX [13] 12.7.1 was used to solve the ILP models.

The following datasets were used to compare the performance and the quality of the solution of the proposed model (CER_{ILP}) against the original model (CE_{ILP}):

- *LFR benchmark networks*. Networks created with the benchmark developed by Lancichinetti-Fortunato-Radicchi (LFR) [15]. The following parameters were used: number of vertices $n = \{50,\ 100,\ 200\}$, the average degree was set to 5 and the maximum degree was set to 10. The default values were used for the remaining parameters. Networks with increasing values for the mixing parameter (μ) were used to blur the community distinction.
- *Random unweighted graphs*. Proposed by [6] in the following manner: given a number of nodes n and parameter k, uniformly selected an integer $i \in [1, n]$, which defines a cluster with i vertices. The process continues with the remaining $n \leftarrow n - 1$ vertices until $n \leq 5$ holds. In this case, all remaining vertices are assigned to the last cluster. Finally, an estimated value $k' \leq k$ is used to perform uniformly editions (add/remove edges). This dataset can be found online[1]. Datasets with sizes $n = \{100,\ 200,\ 300,\ 1000,\ 1500,\ 2000\}$ were selected for the experiments.

4.1 Experiments with LFR Networks

LFR benchmark networks [15] were created to evaluate the performance of the proposed preprocessing technique. Given a number of vertices n, one network with predefined clusters was created for each mixing parameter (μ). The amount of edges between clusters increase proportionally to μ. As a consequence, the clusters become more interconnected and the clustering problems more difficult.

Table 1 presents the results obtained for the (CE_{ILP}) and (CER_{ILP}) models on the LFR benchmark networks. Column n represents the number of vertices of the graph; μ is the mixing parameter; columns *Obj*, $\#C$ and *Time* are defined for both models and represent, respectively, the objective value, the number of constraints, and the computational time in seconds. Finally, column $\%C$ represents the percentage of constraints removed from the original model and S corresponds to the computational time speedup obtained by (CER_{ILP}).

It can be observed, based on Table 1, that (CER_{ILP}) achieves a better performance than (CE_{ILP}) while preserving the optimal number of editions for all the considered instances. This higher performance is due to a large number of redundant transitivity constraints disregarded by the preprocessing technique (above 99%). Consequently, the computational times are drastically improved providing speedups from 15 to 13755. All instances are solved by (CER_{ILP}) in less than 2 s.

[1] https://bio.informatik.uni-jena.de/data/.

Table 1. Results obtained by (CE_{ILP}) and (CER_{ILP}) on LFR benchmark networks.

		CE_{ILP}			CER_{ILP}				
n	μ	Obj	#C	Time	Obj	#C	Time	%C	S
50	0.1	50	58800	0.63	50	250	0.04	99.57	15.18
	0.2	65	58800	1.01	65	375	0.07	99.36	15.27
	0.3	71	58800	1.21	71	411	0.07	99.30	17.28
	0.4	80	58800	1.79	80	486	0.10	99.17	18.77
	0.5	78	58800	2.70	78	423	0.11	99.28	24.81
	0.6	74	58800	1.68	74	381	0.07	99.35	24.96
	0.7	88	58800	2.73	88	533	0.14	99.09	19.90
	0.8	87	58800	5.37	87	522	0.21	99.11	25.02
	0.9	89	58800	5.29	89	528	0.27	99.10	19.74
100	0.1	86	485100	6.23	86	428	0.05	99.91	118.44
	0.2	108	485100	6.52	108	627	0.05	99.87	126.95
	0.3	147	485100	11.73	147	893	0.15	99.82	75.86
	0.4	149	485100	16.01	149	890	0.11	99.82	143.08
	0.5	173	485100	46.76	173	1078	0.23	99.78	201.69
	0.6	183	485100	82.62	183	1153	0.63	99.76	131.68
	0.7	182	485100	74.43	182	1127	0.36	99.77	208.65
	0.8	193	485100	72.91	193	1260	0.27	99.74	267.89
	0.9	172	485100	49.43	172	1004	0.19	99.79	257.55
200	0.1	223	3940200	157.55	223	1118	0.18	99.97	865.92
	0.2	223	3940200	74.58	223	1270	0.11	99.97	651.88
	0.3	257	3940200	96.49	257	1563	0.11	99.96	911.65
	0.4	296	3940200	221.62	296	1688	0.17	99.96	1310.29
	0.5	331	3940200	558.39	331	2004	0.17	99.95	3460.32
	0.6	333	3940200	956.43	333	1925	0.20	99.95	4697.06
	0.7	342	3940200	5074.88	342	1908	0.37	99.95	13755.17
	0.8	376	3940200	10547.42	376	2333	1.73	99.94	6084.59
	0.9	481	3940200	5855.49	481	3388	2.03	99.91	2877.47

Problems with $n > 300$ vertices were not tested because the (CE_{ILP}) fails to solve them due to lack of memory. It is worth noting that the (CER_{ILP}) can solve problems with a higher number of vertices, based on the results presented in Table 1.

4.2 Experiments with Random Unweighted Graphs

Proposed by Bocker et al. [6], these datasets were generated by disturbing an ideal cluster graph using random edge insertions and deletions. Given a number of vertices n, 10 networks with predefined clusters were created for each corresponding k. The values of k were selected according to the following rule $k = c * n$ with $c = \{0.25,\ 0.5,\ 1,\ 1.25,\ 1.5,\ 1.75,\ 2\}$.

Datasets with $n = \{100,\ 200,\ 300\}$ were considered initially for the experiments because (CE_{ILP}) failed to solve instances with $n > 300$ due to lack of memory. Table 2 presents the results obtained for the (CE_{ILP}) and (CER_{ILP}) models on such networks. Each row represents the average result for the 10 datasets considering each pair (n, k). Column n represents the number of vertices of the graph; k is the upper limit for the number of editions; columns *Obj*, $\#C$ and *Time* are defined for both models and represent, respectively, the average objective value, average number of constraints, and computational times in seconds. Finally, column $\%C$ represents the average percentage of constraints removed from the original model and S corresponds to the average computational time speedup obtained by (CER_{ILP}).

Next, we run a set of experiments with larger datasets $n = \{1000,\ 1500,\ 2000\}$ to test the scalability of the (CER_{ILP}). The objective value cannot be compared to (CE_{ILP}), but the percentage of constraint elimination can be estimated.

Table 2. Results obtained by (CE_{ILP}) and (CER_{ILP}) on random unweighted graphs.

		CE_{ILP}			CER_{ILP}				
n	k	Obj	#C	Time	Obj	#C	Time	%C	S
100	25	25.0	485100	10.85	25.0	1745.8	0.03	99.64	346.66
	50	49.0	485100	7.03	49.0	3001.4	0.04	99.38	164.36
	75	74.2	485100	5.35	74.2	4409.1	0.07	99.09	84.59
	100	97.8	485100	5.14	97.8	6547.8	0.11	98.65	49.88
	125	121.4	485100	4.79	121.4	7726.2	0.09	98.41	52.59
	150	145.6	485100	4.92	145.6	8369.7	0.10	98.27	52.64
	175	168.2	485100	4.61	168.2	10108.1	0.10	97.92	47.19
	200	192.4	485100	4.62	192.4	11542.6	0.11	97.62	43.14
200	50	50.0	3940200	242.98	50.0	5889.6	0.08	99.85	3266.11
	100	99.6	3940200	141.69	99.6	13819.6	0.22	99.65	728.56
	150	149.6	3940200	64.90	149.6	17697.5	0.23	99.55	299.32
	200	198.0	3940200	51.93	198.0	24434.4	0.30	99.38	188.05
	250	248.0	3940200	41.75	248.0	32242.4	0.36	99.18	121.56
	300	295.2	3940200	41.19	295.2	36497.3	0.40	99.07	105.71
	350	343.0	3940200	41.09	343.0	37213.1	0.41	99.06	103.98
	400	391.4	3940200	41.41	391.4	46879.4	0.50	98.81	85.44
300	75	74.8	13365300	1621.70	74.8	12447.7	0.15	99.91	11386.81
	150	149.4	13365300	875.95	149.4	29056.6	0.42	99.78	2265.54
	225	224.0	13365300	430.22	224.0	45402.3	0.70	99.66	683.91
	300	298.2	13365300	262.33	298.2	57886.7	0.79	99.57	340.42
	375	371.5	13365300	203.77	371.5	78641.0	0.98	99.41	215.10
	450	445.2	13365300	188.92	445.2	80117.3	0.98	99.40	206.10
	525	518.8	13365300	186.77	518.8	103056.6	1.19	99.23	165.03
	600	593.2	13365300	195.56	593.2	126468.7	1.46	99.05	140.61

Table 3 presents the results obtained for the (CER_{ILP}). Each row represents the average result for the 10 datasets and each pair (n, k). Column n represents the number of vertices of the graph; k is the upper limit for the number of editions; for (CE_{ILP}) only the average number of constraints is presented ($\#C$). Columns Obj, $\#C$ and $Time$ represent the average objective value, average number of constraints and computational times in seconds for the (CER_{ILP}), respectively. Finally, column $\%C$ highlights the average percentage of constraints removed from the original model.

Table 3. Results obtained by (CE_{ILP}) and (CER_{ILP}) on large random unweighted graphs.

		CE_{ILP}		CER_{ILP}		
n	k	#C	Obj	#C	Time	%C
1000	250	498501000	249.8	162941.7	9.29	99.97
	500	498501000	499.8	338462.5	14.56	99.93
	750	498501000	749.0	475923.0	17.91	99.90
	1000	498501000	998.0	652948.9	21.47	99.87
	1250	498501000	1247.5	900571.3	25.61	99.82
	1500	498501000	1494.6	1040558.8	24.14	99.79
	1750	498501000	1743.2	1235538.8	32.12	99.75
	2000	498501000	1990.0	1183563.7	41.57	99.76
1500	375	1684126500	374.8	363423.8	15.95	99.98
	750	1684126500	749.6	717436.6	39.40	99.96
	1125	1684126500	1123.6	1167055.5	73.22	99.93
	1500	1684126500	1494.6	1040558.4	24.14	99.79
	1875	1684126500	1872.8	1817375.0	89.43	99.89
	2250	1684126500	2245.2	2125008.0	150.52	99.87
	2625	1684126500	2618.6	2276352.3	227.35	99.86
	3000	1684126500	2991.2	3048967.0	386.80	99.82
2000	500	3994002000	500.0	739933.5	76.19	99.98
	1000	3994002000	998.8	1149414.0	77.29	99.97
	1500	3994002000	1498.4	2138914.5	220.12	99.95
	2000	3994002000	1998.0	2708588.2	340.46	99.93
	2500	3994002000	2497.2	3210726.6	403.91	99.92
	3000	3994002000	2994.4	4150590.4	346.08	99.90
	3500	3994002000	3495.8	4417540.9	742.18	99.89
	4000	3994002000	3993.0	5290777.8	1389.50	99.87

From Table 3, it can be verified that a large number of redundant transitivity constraints are disregarded by the preprocessing technique (above 99%). All instances are solved by (CER_{ILP}) in less than 23 min. This result shows the scalability of the proposed technique on the datasets proposed by Bocker et al. [6].

5 Conclusions

A novel preprocessing technique for the Clique Edition problem was proposed in this work. The experimental results showed that the reduced model (CER_{ILP}) provided a considerable reduction of transitivity constraints, preserved the optimal solution set, and improved the computational speedup compared to model (CE_{ILP}) for all considered instances.

This technique has the advantage of working as complementary to other techniques, like the cutting plane proposed by Grotschel and Wakabayashi [11]. The (CER_{ILP}) can provide means to increase the size of future instances that the Integer Linear Programming approach can execute.

Our preprocessing technique might also be combined with other approaches, such as the method for reducing the input graph proposed in [6]. Thus, the preprocessed reduced graph may speed up the solution of the ILP problem even further.

Finally, we expect that the results obtained in the ILP context can be used to guide the construction of better heuristic techniques for the unweighted Cluster Editing problem.

Acknowledgements. The authors thanks FAPESP (Grant No. 2011/18496-7), CNPq (Grant No. 310908/2015-9 and 301836/2014-0), CAPES and IBM for support.

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information. J. ACM **55**, 1–27 (2008). https://doi.org/10.1145/1411509.1411513
2. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Mach. Learn. **56**, 89–113 (2004). https://doi.org/10.1023/B:MACH.0000033116.57574.95
3. Bastos, L., Ochi, L.S., Protti, F., Subramanian, A., Martins, I.C., Pinheiro, R.G.S.: Efficient algorithms for cluster editing. J. Comb. Optim. **31**(1), 347–371 (2016). https://doi.org/10.1007/s10878-014-9756-7
4. Ben-Dor, A., Shamir, R., Yakhini, Z.: Clustering gene expression patterns. J. Comput. Biol. **6**(3–4), 281–297 (1999). https://doi.org/10.1089/106652799318274
5. Böcker, S., Baumbach, J.: Cluster editing. In: Bonizzoni, P., Brattka, V., Löwe, B. (eds.) CiE 2013. LNCS, vol. 7921, pp. 33–44. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-39053-1_5
6. Böcker, S., Briesemeister, S., Klau, G.W.: Exact algorithms for cluster editing: evaluation and experiments. Algorithmica **60**, 316–334 (2011). https://doi.org/10.1007/s00453-009-9339-7
7. Charikar, M., Guruswami, V., Wirth, A.: Clustering with qualitative information. J. Comput. Syst. Sci. **71**, 360–383 (2005). https://doi.org/10.1016/j.jcss.2004.10.012
8. Dinh, T.N., Thai, M.T.: Toward optimal community detection: from trees to general weighted networks. Internet Math. **11**, 181–200 (2014). https://doi.org/10.1080/15427951.2014.950875
9. Fellows, M., Langston, M., Rosamond, F., Shaw, P.: Efficient parameterized preprocessing for cluster editing. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) FCT 2007. LNCS, vol. 4639, pp. 312–321. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74240-1_27

10. Gionis, A., Mannila, H., Tsaparas, P.: Clustering aggregation. ACM Trans. Knowl. Discov. Data (TKDD) **1**(1), 4 (2007). https://doi.org/10.1145/1217299.1217303
11. Grötschel, M., Wakabayashi, Y.: A cutting plane algorithm for a clustering problem. Math. Program. **45**, 59–96 (1989). https://doi.org/10.1007/bf01589097
12. Guo, J.: A more effective linear kernelization for cluster editing. Theor. Comput. Sci. **410**, 718–726 (2009). https://doi.org/10.1016/j.tcs.2008.10.021
13. IBM: IBM ILOG CPLEX 12.7.1 (1987–2017)
14. Kim, S., Yoo, C.D., Nowozin, S., Kohli, P.: Image segmentation using higher-order correlation clustering. IEEE Trans. Pattern Anal. Mach. Intell. **36**, 1761–1774 (2014). https://doi.org/10.1109/tpami.2014.2303095
15. Lancichinetti, A., Fortunato, S., Radicchi, F.: Benchmark graphs for testing community detection algorithms. Phys. Rev. E Stat. Nonlin. Soft Matter Phys. **78**, 046110 (2008). https://doi.org/10.1103/physreve.78.046110
16. Miyauchi, A., Sukegawa, N.: Redundant constraints in the standard formulation for the clique partitioning problem. Optim. Lett. **9**, 199–207 (2014). https://doi.org/10.1007/s11590-014-0754-6
17. Nguyen, D.P., Minoux, M., Nguyen, V.H., Nguyen, T.H., Sirdey, R.: Improved compact formulations for a wide class of graph partitioning problems in sparse graphs. Discrete Optim. **25**, 175–188 (2017). https://doi.org/10.1016/j.disopt.2016.05.003
18. Protti, F., da Silva, M.D., Szwarcfiter, J.L.: Applying modular decomposition to parameterized cluster editing problems. Theory Comput. Syst. **44**, 91–104 (2007). https://doi.org/10.1007/s00224-007-9032-7
19. Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. In: Goos, G., Hartmanis, J., van Leeuwen, J., Kučera, L. (eds.) WG 2002. LNCS, vol. 2573, pp. 379–390. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-36379-3_33