Clique Editing to Support Case vs Control Discrimination

Riccardo Dondi[1], Giancarlo Mauri[2], and Italo Zoppis[2]

[1] Dipartimento di Scienze Umane e Sociali, Università degli Studi di Bergamo, Bergamo, Italy
[2] Dipartimento di Informatica, Sistemistica e Comunicazione, Università degli Studi di Milano-Bicocca, Milano - Italy
riccardo.dondi@unibg.it,mauri@disco.unimib.it,zoppis@disco.unimib.it

Abstract. We present a graph-based approach to support case vs control discrimination problems. The goal is to partition a given input graph in two sets, a clique and an independent set, such that there is no edge connecting a vertex of the clique with a vertex of the independent set. Following a parsimonious principle, we consider the problem that aims to modify the input graph into a most similar output graph that consists of a clique and an independent set (with no edge between the two sets). First, we present a theoretical result showing that the problem admits a polynomial-time approximation scheme. Then, motivated by the complexity of such an algorithm, we propose a genetic algorithm and we present an experimental analysis on simulated data.

1 Introduction

Graph modification has been widely applied in different contexts to provide sets of homologus instances from a set of different elements. For example, correlation clustering [3, 15], a well-known problem with applications in machine learning and bioinformatics, asks for the partition of the vertices of a graph in cliques, so that the similarity (expressed as the number of common edges between the input graph and the output graph, plus the number of pairs of vertices not connected in both input and output graphs) is maximized [3].

Here, we focus on a similar problem known as *Sparse Split Graph* problem in the context of case vs control experimental design. In this situation, an important goal is to distinguish healthy subjects (controls) from patients (cases). For instance, consider a set of individuals that could be either affected by some disease or randomly sampled from a healty population in some clinical trial. We can represent the observed sample as the set of vertex in a graph by connecting two vertices whenever two patients share similar symptoms (for example when correlation between their blood pressure is statistically significant). Following this process, a clique might be ideally used to provide a set of homologous patients. Similarly, an independent set of vertices, well separated from that clique,

[3] Different variants of the problem have been investigated, including the weighted and constrained variants [3, 8].

could ideally be used to istantiate healthy subjects. Clearly, the results of the trial may be motivated by many reasons, and not all patients (i.e. cases) affected by the expressed disease share similar symptoms; as a result the graph we obtain is not necessarily partitioned into ill (clique) and healthy subjects (independent set). Nevertheless, optimizing such structure could benefits further classification tasks. In fact, most real domains are best described by structured data where instances of particular types are related to each other in complex ways. In these contexts, relationships (e.g. expressing similarities or differences between instances) can provide suitable source of information to inference mechanism in data mining or machine learning problems. Such an approach has been proved useful for both classification and clustering in many situations, see e.g. [6, 7] for molecular data analysis, [11] for network data, or [12] for a broader view of fundamentals in cluster analysis.

The goal of our approach is then to obtain a suitable partition of the given input graphs in two sets, a clique (cases) and an independent set (controls), such that no edge connecting vertices of the clique with vertices of the independent set are allowed. Importantly, following a parsimonious principle, we expect that such process preserves the maximum number of original relations (edges or missing edges) from the graph observed as an input instance, i.e. preserving the original structure as well as possible. At this point, it is not difficult to imagine how this model can be applied for future discrimination, i.e. when new subjects have to be classified either as cases or controls. For example a simple approach could be to evaluate whether new instances correlate mostly with the ("case") clique or with the independent set of healthy controls. Please notice that, here we do not focus on the classification question. Instead we analyze the combinatorial problem described above, i.e. the Maximum Sparse Split Graph Problem ($\mathcal{MAX} - \mathcal{SSG}$).

$\mathcal{MAX} - \mathcal{SSG}$ has the same input of the Minimum Sparse Split Graph Problem [9], but with a complementary objective function. Given a graph, Minimum Sparse Split Graph and $\mathcal{MAX} - \mathcal{SSG}$ aim to partition the graph in two sets, a clique and an independent set, such that there is no edge connecting a vertex of the clique with a vertex of the independent set; the objective function of Minimum Sparse Split Graph is the minimization of the number of modified relations (edges removed or inserted with respect to the input graph), the objective function of $\mathcal{MAX} - \mathcal{SSG}$ is the maximization of the number of unchanged relations.

Since Minimum Sparse Split Graph is NP-hard [9], the same complexity result holds also for the Maximum Sparse Split Graph problem. Indeed a graph G' which is an optimal solution of Minimum Sparse Split Graph is also an optimal solution of $\mathcal{MAX} - \mathcal{SSG}$: since it modifies the minimum number of inserted or deleted edges, it maximizes the number of relations of the input graph that are not modified. Clearly, also the converse is true: a graph G' which is an optimal solution of $\mathcal{MAX} - \mathcal{SSG}$ is also an optimal solution of Minimum Sparse Split Graph.

We thus consider two possible directions of research, namely approximation algorithms and genetic algorithms. For the first direction, we present in Section 3,

a theoretical result, that is we prove that the Maximum Sparse Split Graph problem admits a Polynomial Time Approximations Scheme (PTAS), which means that the problem can be solved within any factor $1+\varepsilon$ in time $O(n^{1/\varepsilon})$, for any constant ε, where n is the number of vertices of the input graph. While this result shows that solutions close to the optimal can be computed in polynomial time, such a result is only of theoretical interest, due to the time complexity of the proposed PTAS. Hence, we consider genetic algorithms as a possible direction to compute optimal or near optimal solution for the Maximum Sparse Split Graph problem. We give in Section 4 a natural genetic algorithm and we present some experimental results on a simulated data set, that show that the genetic algorithm converges fast, in particular when the input data has a moderate similarity to Sparse Split Graph Problem.

2 Preliminaries

In this section we introduce some basic definitions that will be useful in the rest of the paper and we give the formal definition of the Maximum Sparse Split Graph problem. We recall that a graph is a clique if it is a complete graph, that is a graph where each pair of vertices is connected by an edge. A graph is an independent set if no pair of vertices is connected by an edge.

Given a graph $G = (V, E)$ and a set V' of vertices, with $V' \subseteq V$, we denote by $G[V']$ the graph induced by V', that is $G[V'] = (V', E')$, where $E' = \{\{v_i, v_j\} : v_i, v_j \in V' \wedge \{v_i, v_j\} \in E\}$.

Now, we give the formal definition of *Sparse Split Graph*, since starting from a given graph, we aim to compute a sparse split graph that can be obtain by preserving the maximum number of relations represented by the input graph.

Definition 1. *Given a graph $G = (V, E)$, G is a* Sparse Split Graph *if V can be partitioned in two sets V_1 and V_2 such that $G[V_1]$ is a clique, $G[V_2]$ is an independent set and, for each $v_i \in V_1$ and $v_j \in V_2$, $\{v_i, v_j\} \notin E$.*

Before giving the formal definition of the Maximum Sparse Split Graph Problem, we introduce the definition of agreement between two graphs, as it is useful to describe the objective function we aim to maximize.

Definition 2. *Given a graph $G = (V, E)$ and a graph $G' = (V, E')$, we define the* agreement *of G and G', denoted by $A(G, G')$, as the number of edges that belong to both E and E' plus the number of edges that do not belong to both E and E'.*

Now, we are ready to give the definition of the Maximum Sparse Split Graph Problem ($\mathcal{MAX} - \mathcal{SSG}$).

Problem 1. Maximum Sparse Split Graph Problem ($\mathcal{MAX} - \mathcal{SSG}$)
Input: a graph $G = (V, E)$.
Output: a graph $G' = (V, E')$, such that G' is a sparse split graph.
Objective function: $A(G, G')$ is maximized.

In the remaining part of the paper, we denote by n the number of vertices in V, that is $n = |V|$.

3 A PTAS for $\mathcal{MAX} - \mathcal{SSG}$

In this section, we present a PTAS for the $\mathcal{MAX} - \mathcal{SSG}$ problem. The algorithm is based on the smooth polynomial programming technique of [1], a technique that has been previously applied to design PTAS for graph modification problems (for example a variant of Correlation Clustering [5]) and problems in phylogenetics (Maximum Quartet Inconsistecny [10]).

We now briefly present the smooth polynomial programming technique. A *c-smooth polynomial integer program* over variables $x_1, \ldots, x_m$ is a problem having the following form:

$$\begin{aligned} &\text{maximize } p(x_1, \ldots, x_m) \\ &\text{subject to } z_j \leq q_j(x_1, \ldots, x_m) \leq u_j \\ &\qquad x_i \in \{0, 1\} \text{ for } 1 \leq i \leq m \end{aligned} \tag{1}$$

where each $q_j(x_1, \ldots, x_m)$ is a polynomial on variables $x_1, \ldots, x_m$ that has maximum degree d, and the coefficients of each degree-ℓ monomial are in the interval $[-cm^{d-\ell}, cm^{d-\ell}]$. Denote by OPT the optimal value of a c-smooth polynomial integer program. In [1] it is shown that, for each constant $\delta > 0$, there exists an approximation algorithm that, in time $O(m^{\delta^{-2}})$, computes a 0/1 assignment $\langle a_1, \ldots, a_m \rangle$ to the variables $\langle x_1, \ldots, x_m \rangle$ of a c-smooth polynomial integer program, such that, by assigning $x_1 = a_1, \ldots, x_m = a_m$, $p(x_1, \ldots, x_m)$, has value at least $OPT - \delta m^d$.

Before presenting our formulation of $\mathcal{MAX} - \mathcal{SSG}$ as a c-smooth polynomial integer program, we prove a property of the $\mathcal{MAX} - \mathcal{SSG}$ problem, namely we give a lower bound on the value of an optimal solution of $\mathcal{MAX} - \mathcal{SSG}$ on any instance G.

Lemma 1. *Given an instance $G = (V, E)$ of $\mathcal{MAX} - \mathcal{SSG}$, an optimal solution of $\mathcal{MAX} - \mathcal{SSG}$ on instance G has value at least $\frac{n(n-1)}{4}$.*

Proof. Given a graph $G = (V, E)$, consider the following solutions of $\mathcal{MAX} - \mathcal{SSG}$ on instance $G = (V, E)$:

- $G' = (V, \emptyset)$, that is G' contains no edge;
- $G'' = (V, V \times V)$, that is G'' is a clique.

Consider an edge $\{v_i, v_j\} \in E$, then by construction $\{v_i, v_j\}$ is an edge of G''. Consider two vertices v_i, v_j of V such that $\{v_i, v_j\} \notin E$; it follows by construction that $\{v_i, v_j\}$ are not connected by an edge in G'. Hence denote by $A(G', G)$ ($A(G'', G)$, respectively) the number of agreement between G' and G (between G'' and G, respectively). Then

$$A(G', G) + A(G'', G) = \frac{n(n-1)}{2}$$

hence one of $A(G', G)$ or $A(G'', G)$ is at least $\frac{n(n-1)}{4}$. Since an optimal solution has agreement value greater or equal than $A(G', G)$ and $A(G'', G)$, the lemma follows. □

From Lemma 1, it follows that an approximation algorithm with additive error δn^2 is a $(1+\frac{\delta}{\alpha})$ approximation algorithm, for some fixed constant α, which proves the existence of a PTAS for the problem.

Now we present a formulation of $\mathcal{MAX}-\mathcal{SSG}$ as a c-smooth polynomial integer program. Consider the variables $x_{i,j}$, with $1 \leq i \leq n$ and $1 \leq j \leq 2$; $x_{i,1}$ has value is 1 if and only if the v_i is assigned to the clique of a solution, otherwise is 0; $x_{i,2}$ has value is 1 if and only if the v_i is assigned to the independent set of a solution, otherwise is 0. Denote by $e_{i,j}$ a constant equal to 1 if $\{v_i, v_j\} \in E$ (else $e_{i,j}$ is equal to 0), and denote by $n_{i,j}$ a constant equal to 1 if $\{v_i, v_j\} \notin E$ (else $n_{i,j}$ is equal to 0). We can given a formulation of $\mathcal{MAX}-\mathcal{SSG}$ as follows:

$$\begin{aligned} \text{maximize} & \sum_{i,j} e_{i,j}x_{i,1}x_{j,1} + n_{i,j}(x_{i,2}x_{j,2} + x_{i,1}x_{j,2} + x_{i,2}x_{j,1}) \\ \text{subject to } & x_{i,1} + x_{i,2} = 1 \qquad \text{for } 1 \leq i \leq n \\ & x_{i,t} \in \{0,1\} \end{aligned} \tag{2}$$

The formulation given for $\mathcal{MAX}-\mathcal{SSG}$ is a c-smooth polynomial integer program with degree $d = 2$. Following the approach in [10], we design a PTAS by first computing in polynomial time a fractional solution for the c-smooth polynomial integer and then rounding this solution as follows: the variable among $x_{i,1}$ and $x_{i,2}$ having the maximum value is rounded to 1, the other variable is rounded to 0. Such a rounding procedure has an additive error δn^2 (for some constant $\delta > 0$, see the proof of Theorem 1) with respect to an optimal solution of the c-smooth polynomial integer program.

Now, we can conclude this section by proving the that we have indeed designed a PTAS.

Theorem 1. *$\mathcal{MAX}-\mathcal{SSG}$ admits a PTAS.*

Proof. Consider the value OPT of an optimal solution of $\mathcal{MAX}-\mathcal{SSG}$ on instance G and let OPT_c be the value of an optimal solution to the corresponding c-smooth polynomial integer program. It follows that $OPT \leq OPT_c$. Let A_c be a solution of the rounding procedure of the c-smooth polynomial integer program of $\mathcal{MAX}-\mathcal{SSG}$. By [1, 10], A_c can be computed in time $O(n^{\delta^{-2}})$, for each constant $\delta > 0$, so that $A_c \geq OPT_c - \delta n^2 \geq OPT - \delta n^2$.

Since, by Lemma 1, $OPT \geq \alpha n^2$, for some constant $\alpha > 0$, it follows that

$$A_c \geq \alpha n^2 - \delta n^2 = \alpha n^2(1 - \frac{\delta}{\alpha}) = OPT(1-\varepsilon)$$

where $\varepsilon = \frac{\delta}{\alpha}$. □

4 A Genetic Algorithm for $\mathcal{MAX} - \mathcal{SSG}$

The algorithm presented in Section 3 is mainly of theoretical interests due to its computational complexity. Therefore to provide an efficient heuristic for $\mathcal{MAX} - \mathcal{SSG}$, we consider an approach based on genetic algorithms [13].

Given an input graph $G = (V, E)$, the genetic algorithm represents a solution (that is a bipartition (V_1, V_2) of the vertex-set V, where V_1 is a clique and V_2 an independent set), as a chromosome that represents the vertex membership properties. Given a chromosome c, we have $c[i] = 1$ if vertex i is part of a clique; 0 otherwise. As usual for genetic algorithms, we consider standard operators such as *mutations* and *crossover*. More specifically, we adapt the mutation operator as follows. The mutation process applied here aims to modify the set that a vertex belongs to, that is if a vertex belong to the independent set is moved to the clique, whereas a vertex that belongs to the clique is moved to the independent set. Selection of elements in the population for modification follows a fitness proportionate scheme (see e.g. [13]).

Next, we give more details about the mutation and crossover operators.

- Mutation. Each individual from current population at time i is modified with probability 0.1 as follows:
  - A random sample of vertices with low degree in the original graph takes value 0 over the chromosome (that is it is moved to the independent set);
  - A random sample of vertices from cliques of size three of the original graph takes value 1 over the chromosome (that is the three vertices belong to the clique).
- Crossover is applied with probability 0.8.

We consider cliques of size three of the original graph, as they can be computed efficiently (in time $O(n^3)$), and they may represent subsets of larger cliques.

Genetic algorithm uses a fitness function to evaluate a solution. The fitness function for our genetic algorithm is defined as the the negation of the Hamming distance between the adjacency matrix of the solution graph and the adjacency matrix of the input graph (we omit here the constant value $\binom{n}{2}$ used in the objective function of $\mathcal{MAX} - \mathcal{SSG}$). Finally we also consider elitist selection (or elitism) when constructing a new population from an existing one, to allow the best individuals to be part of the next generation. It is known that such an approach can be used to guarantee that the solution quality does not decrease from one generation to the next [2].

The genetic algorithm was coded in R using the Genetic Algorithm package [14] downloadable at https://cran.r-project.org/web/packages/GA/index.html.

4.1 Experimental Results

We apply the genetic algorithm to syntetic data. We sample input graphs from two classical random models (see e.g. [4]), Erdos-Renyi and Barabasi[4]. The first

[4] We use Random Graphs (RGs) as generative models to simulate observations.

Table 1. Models and parameters. ER: Erdos-Renyi model; BA: Preferential attachment (*Barabasi*). GA parameters: Population size, Generation, Elitism, Crossover probability and Mutation probability.

Type	Pop. Size	Generat.	Elitism	Cross Pr	Mut Pr
$ER(10, 1/2)$	100	17.7	10	0.8	0.1
$ER(50, 1/2)$	500	64.3	50	0.8	0.1
$ER(100, 1/2)$	1000	106.6	100	0.8	0.1
$ER(200, 1/2)$	2000	176.2	200	0.8	0.1
$BA(10)$	100	15.1	10	0.8	0.1
$BA(50)$	500	41.1	50	0.8	0.1
$BA(100)$	1000	53.9	100	0.8	0.1
$BA(200)$	2000	46.4	200	0.8	0.1

model, *Erdos-Renyi* $ER(n, p)$ is used to generate graphs with n vertices (every possible edge is created with the same constant probability p).

Table 2. Models and results. ER: Erdos-Renyi model; BA: Preferential attachment (*Barabasi*). Results: Fitness value, CPU time: user,system, elapsed (in seconds) and Relative mutation rate, computed as the percentage of the number of mutations (edge addition and edge deletions) in a graph among all the potential mutations in the original graph.

	Final	CPU			Relative
Type	Fitness Value	User	Sys	Elaps.	Mut. rate
$ER(10, 1/2)$	-13.0	1.777	0.00	1.778	28.9
$ER(50, 1/2)$	-508.0	29.918	0.470	30.442	41.6
$ER(100, 1/2)$	-2196.1	153.196	0.360	153.727	44.7
$ER(200, 1/2)$	-9182.5	1365.656	62.717	1428.373	46.4
$BA(10)$	-8.0	1.512	0.000	1.517	17.8
$BA(50)$	-54.0	25.207	0.040	26.378	4.1
$BA(100)$	-130.1	73.594	0.112	272.099	2.5
$BA(200)$	-372.0	231.403	6.769	239.079	2.2

The second model, *Barabasi* $BA(n)$, uses a simple random mechanism to generate graphs with n vertices through *preferential attachment*. A vertex is added for each step to the current growing graph, starting with a single vertex and no edges. Then, in the next steps, a vertex v is added to the graph by defining edges that connects the new vertex with vertices already part of the graph. The probability that a vertex i already in the graph is chosen is given by $Pr(i) \sim k_i^\alpha + a$, where k_i is the degree of vertex i at the current time and a and α are parameters. In our simulation, we use $\alpha, a = 1$.

From each random model we sample 10 graphs for each choice of n (number of vertices of the graph) in $\{10, 50, 100, 200\}$. We increase the size of the population accordingly (see Table 1). The results are given in Table 2. We stop the execution

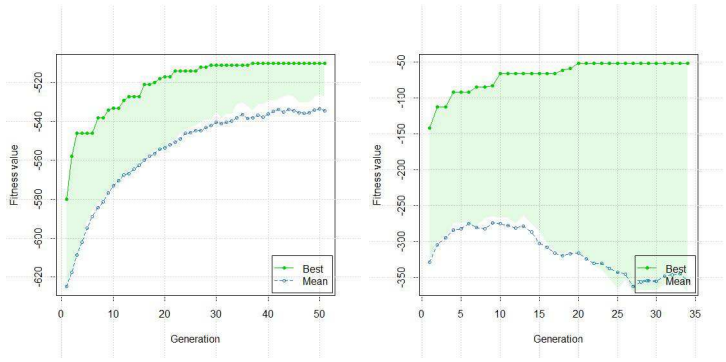

Fig. 1. Experimental Results of the GA for $ER(50, 1/2)$ (left diagram) and $BA(50)$ (right diagram).

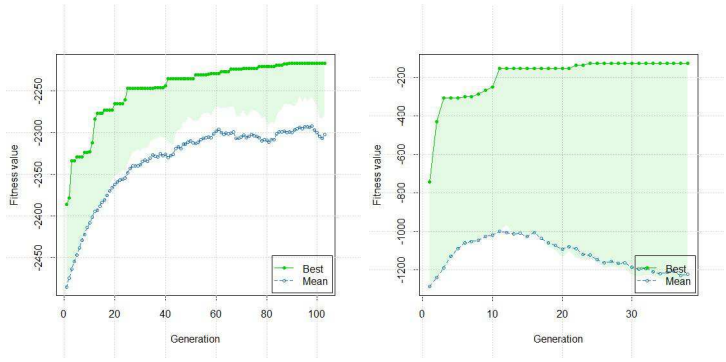

Fig. 2. Experimental Results of the GA for $ER(100, 1/2)$ (left diagram) and $BA(100)$ (right diagram).

of the genetic algorithm when the fitness value is not increased in more than 15 generations. As shown in the same table, the final fitness value (as expected), decreases as the number of vertices increases. Moreover, we observe that for graphs from ER this value is smaller than for graphs from BA. Accordingly, the required CPU time for convergence is greater for graphs sampled from ER than for graphs from BA.

Our results [5] show that the genetic algorithm has good performances on simulated data, even for graph with 200 of vertices. In this case the algorithm was able to converge in a reasonable time (approximately 1428 seconds for samples from ER and approximately 239 seconds for samples from BA).

In the analysis we consider also the *relative mutation rate*, that is the percentage of the number of modifications (edge additions and edge deletions) in a graph among all the potential mutations in the original graph. Interestingly, this

[5] The tests were performed on a machine with 8 GB RAM, Intel Core i7 2.30 GHz.

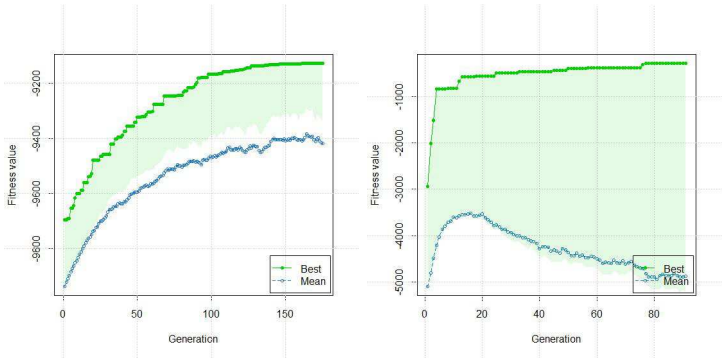

Fig. 3. Experimental Results of the GA for $ER(200, 1/2)$ (left diagram) and $BA(200)$ (right diagram).

value is significantly lower in graphs from BA than in graphs from ER. Moreover, in graphs from ER this value increases as the number of vertices increases, while in graphs from BA we observe the opposite behavior. This fact suggests that graphs from BA have a structure that is closer to that of a Sparse Split Graph with respect to the graphs from ER; furthermore, this property becomes stronger as the number of vertices increases.

We present the performance results for graphs with 50, 100, 200 vertices in Fig. 1, Fig. 2, Fig. 3, respectively. As the results shown, the GA algorithm converges before 54 generations for graphs from BA even with 200 vertices. The convergence is much slower for graphs from ER. For example, for a graph with 200 of vertices, the GA algorithm requires more than 170 generation to converge.

5 Conclusion

In this work we have presented a methodology to support case vs control discrimination studies based on a graph-theoretical approach. We have presented a PTAS for the $\mathcal{MAX} - \mathcal{SSG}$ problem and we have presented a genetic algorithm for the solution of the problem on synthetic data. In the future, we aim to validate the approach by testing our genetic algorithm on a real case study and on real data. Furthermore, we propose to investigate more flexible variants of our approach, for example allowing a more general structure of the output graph than that of Sparse Split Graph or including weights on the edges to represent confidence values.

References

1. Arora, S., Frieze, A.M., Kaplan, H.: A new rounding procedure for the assignment problem with applications to dense graph arrangement problems. In: 37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996. pp. 21–30 (1996)

2. Baluja, S., Caruana, R.: Removing the genetics from the standard genetic algorithm. In: Prieditis, A., Russell, S.J. (eds.) Machine Learning, Proceedings of the Twelfth International Conference on Machine Learning, Tahoe City, California, USA, July 9-12, 1995. pp. 38–46. Morgan Kaufmann (1995)
3. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1-3), 89–113 (2004)
4. Bollobas, B.: Random Graphs. Cambridge University Press (2001)
5. Bonizzoni, P., Della Vedova, G., Dondi, R., Jiang, T.: On the approximation of correlation clustering and consensus clustering. J. Comput. Syst. Sci. 74(5), 671–696 (2008)
6. Cava, C., Zoppis, I., Gariboldi, M., Castiglioni, I., Mauri, G., Antoniotti, M.: Copy–number alterations for tumor progression inference. Artificial Intelligence in Medicine pp. 104–109 (2013)
7. Cava, C., Zoppis, I., Gariboldi, M., Castiglioni, I., Mauri, G., Antoniotti, M.: Combined analysis of chromosomal instabilities and gene expression for colon cancer progression inference. J. Clinical Bioinformatics 4, 2 (2014)
8. Giotis, I., Guruswami, V.: Correlation clustering with a fixed number of clusters. Theory of Computing 2(13), 249–266 (2006), http://dx.doi.org/10.4086/toc.2006.v002a013
9. Hüffner, F., Komusiewicz, C., Nichterlein, A.: Editing graphs into few cliques: Complexity, approximation, and kernelization schemes. In: Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings. pp. 410–421 (2015)
10. Jiang, T., Kearney, P.E., Li, M.: A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its application. SIAM J. Comput. 30(6), 1942–1961 (2000)
11. Kolaczyk, E.D.: Statistical Analysis of Network Data: Methods and Models. Springer Publishing Company, Incorporated (2009)
12. Long, B., Zhang, Z., Yu, P.S.: Relational Data Clustering: Models, Algorithms, and Applications. Chapman & Hall/CRC (2010)
13. Mitchell, M.: An introduction to genetic algorithms. Complex adaptive systems, MIT press, Cambridge (Mass.) (1996)
14. Scrucca, L.: GA: A package for genetic algorithms in R. Journal of Statistical Software 53(4), 1–37 (2013)
15. Wirth, A.: Correlation clustering. In: Sammut, C., Webb, G.I. (eds.) Encyclopedia of Machine Learning, pp. 227–231. Springer (2010)