Clustering data that are connected through a network

Stefano Benati

Dipartimento di Sociologia e Ricerca Sociale, Università di Trento

Justo Puerto

IMUS, Universidad de Sevilla

Antonio M. Rodríguez-Chía

Faculty of Sciences, Universidad de Cadiz

March 26, 2016

1 Introduction

The problem of clustering consists in discovering/detecting how a population is partitioned into two or more subgroups, each specified by distinct features. The typical outcome of a clustering algorithm is the assignment of observations to groups and, most of the times, some estimation of the characteristics of every group. There are many techniques proposed for clustering. Some of them are constructive, in the sense that observations are merged together one at a time until the required partition is found, hierarchical trees being an example. Other techniques have combinatorial structure, for example the k-means, in the sense that clustering is formulated as an optimization problem in which different partitions are evaluated through the objective function. Finally, there are techniques that assume probability distributions on the population features, so that the optimal clustering is calculated maximizing the likelihood function, for example with the EM algorithm. Each technique can be the most appropriate for some application, but all of them works well if hidden groups are well separated. Conversely, if the borders between groups are uncertain, because many observations coming from different groups have similar features, then the precision of the algorithms decreases, to the point that they just output random groups.

In this contribution, it is shown that the capability of detecting true clusters is enhanced if, together with the "'standard"' individual data that are pertinent to one population unit, the researcher can observe relational data too, that is, data describing the connections between units. To **make, built** an example in social network analysis, suppose that a researcher wants to determine groups with homogeneous cultural orientations within a population of individual. Data come from a survey in which people **answer** to questions about their attitudes to religion, politics, family and so on. It is often the case that individuals classified as *"secular"* are not neatly separated from individuals classified as *"religious"*, as they can share same views on social matters different from religion, Inglehart and Baker (2000). Therefore the classification algorithm could not correctly classify individuals, thereby weakening the entire subsequent analysis, for example providing wrong estimations of the groups parameters. A possible remedy to the uncertainty could be to consider an additional data: the relationships, for example friendship, kinship, sympathy, and so on, that exist between individuals. This datum presents an important additional information: Social scientists know the principle of *homophylia*, McPherson et al. (2001), for which people **try to have like-minded friends**. With this datum at hand, the uncertain group attributions of a clustering algorithm can be resolved.

It is discussed in Wasserman and Faust that there is an intrinsic difference between individual and relational data. Individual data are specific attributes that form the compositional dimension of the social actor, and must be distinguished from the structural dimension of the social actor. This structural dimension is represented by the relations between individuals. Following Wasserman and Faust, there is a third dimension of an actor, and it is called affiliation: It is the individual membership to specific groups, as clubs, companies or social and cultural classes. Often this variable is unknown and must be determined through the classification process.

Individual data are given in the form a matrix D of n rows and m columns, in which n is the number of individuals and m is the number of features. Relational data are given in the form of a graph $G = (V, E)$, in which V are the individuals, $|V| = n$, and there is an arc $ij \in E$ if there is a relation between $i, j \in V$. The data structure that combines the graph $G = (V, E)$ with the data matrix D forms the triplet $G = (V, E, D)$. This data structure is called attributed graph, as the matrix D can be viewed as referring to node data, see **?**.

Previous research on attributed graph tried to to apply standard clustering algorithms to a simplified version of the problem. One form of simplification is obtained projecting the matrix D into the graph G, defining weights on arcs (i, j) that depend on the similarity between individual i, j. A weighted graph G' is obtained and analyzed using known graph partition techniques, Neville et al. (2003). The other form of simplification proceeds the other way round, projecting the connections of E into the matrix D, and then applying a clustering algorithm, Combe et al. (2012). A more ingenious way of combining relational and individual data is suggested in Cheng et al. (2012) it is suggested to analyze an augmented graph, formed by two classes of nodes: nodes representing individuals and nodes representing features, and arcs can connect both kind of nodes. Node distances are calculated through random walks, and then the$k$-medoids clustering is applied. To summarize, whether G is projected to D or D to G, the researcher has the advantage that no new technique is to be developed for clustering. But the drawback is that compositional and relational data are treated as homogeneous information, while their nature is intrinsically different.

An approach that keeps separated the two data types is presented in Xu et al. (2014). Here, it is assumed that there is a probabilistic model underlying to relational and compositional data. As commonly assumed in model-based clustering, each unit belongs to a latent class, that determines the probabilities to have some feature and to form some link. Each class is characterized by its distribution function depending on parameters. The parameters are then estimated through the maximization of the likelihood function and the EM algorithm. The approach is elegant, but requires high computational times and the explicit assumptions of the multivariate distributions.

In this paper, we propose a combinatorial problem to cluster attributed graph. More formally, the combinatorial structure of the objective function is borrowed from previous clustering models, but it is imposed that the eligible groups must be connected, in the sense that there must be a path from each pair i, j belonging to the same group. The clustering model to which the connectivity constraints are superimposed is the clique-partitioning, a method similar to the k-means, but with the advantage that k is not to be fixed in advance [1]. This approach differs from the previous ones in two main aspects. First: The maximum likelihood maximization is simplified by a combinatorial objective function that is easier to optimize (as the k-means algorithm is faster than EM optimization). Second: Compositional and relational data are not merged, but kept separate and they play a different role in the

[1]A warning is necessary here: The reader should not be mislead by the term *clique*, that apparently implies networks data. The way in which the term is used in cluster analysis depends on the fact that the individual data D are viewed as embedded on a complete graph.

problem formulation.

From a pure optimization point of view, the model in this paper combines two combinatorial problems, namely clique partition, coming from clustering individual data, and the spanning tree detection, coming from imposing connections on the relational data. The first one is a well-known NP-hard problem. Moreover, although the spanning tree design can be modeled as a continuous linear program, when superimposed on other combinatorial structure (clique partition in this case) becomes also extremely difficult. Thus, it is not surprising that our model is NP-hard and also extremely difficult; as it will be clear in the following sections. In spite of that, we have found several valid integer linear programming formulations, some of them rather promising, based on different rationale. These formulations provide exact solutions for the considered problems for medium size problems (up to 40 units). Beyond this limit, we propose to use heuristic approaches. For this purpose we also present a heuristic algorithm whose performance is compared with exact solutions up to admissible sizes.

The paper is so structured:::::

2 Problem Formulation

At its origin, Grotschel and Wakabayashi (1989); Johnson et al. (1993) the clique partitioning problem has been formulated for clustering data. An equivalence relation is defined when two observations share the same qualitative feature and, as observations are characterized by many features, many equivalence relations are defined. To determine the partition, all equivalences must be aggregated, so that the resulting equivalence classes define the clusters. As equivalence relations can be viewed as arcs of a complete graph, this clustering problem has been termed clique partitioning.

The clique partitioning problem is formulated as follows. Let $V = \{1, \dots, n\}$ be the set of units and let $F_k, k = 1, \dots m$, be the set of binary features measured on V, let data be collected in the matrix $D = [d_{ik}]$. Each binary variable defines an equivalence relation between units, that is unit i is equivalent to j according to features k iff $d_{ik} = d_{jk}$. Let $m_{ij} = \#(\text{equivalence relations between } i \text{ and } j)$, then the cost function is $c_{ij} = m - 2 * m_{ij}$. The values of c_{ij} range from the minimum $-m$, denoting full concordance between units i and j, to the maximum of m, denoting discordance between i and j. The clique partitioning problem is defined as finding the partition $\Pi = \{V_1, \dots, V_p\}$ such that the following objective function is minimized:

$$f(\Pi) = \sum_{k=1}^{p} \sum_{i,j \in V_k} c_{ij} \tag{1}$$

As the problem is in minimization form, units for which c_{ij} is negative tend to be in the same group. Conversely, units for which c_{ij} is positive tend to be in different groups. As the number p of groups is not determined in advance, but it is the output of the algorithm, and since weights are both positive and negative, the problem is different from the p-cut problem, the k-means and the p-median, and other clustering algorithms in which the number of cluster must be defined a-priori.

When relational data are available they are stored in the graph $G = \{V, E\}$, in which nodes $i \in V$ are the units, and edges $(i, j) \in E$ describe the unit links. For $Q \subseteq V$, $G[Q] = (Q, E[Q])$ is the subgraph induced by Q, e.g. the graph with edges $(i, j) \in E[Q]$ iff $i, j \in Q$. The partition $\Pi = \{V_1, \ldots, V_p\}$. is feasible iff $G[V_i], i = 1, \ldots, p$ are all connected subgraph. Then the Connected Clique Partitioning Problem (el nombre se puede cambiar) is to find the partition $\Pi = \{V_1, \ldots, V_p\}$ that is formed by components connected on G and that minimizes the objective function (1).

2.1 Flow based formulation with two indices variables

Let $G = (V, E)$ be an undirected network with node set $V = \{v_1, ..., v_n\}$ and edge set E, such that, $e_{ij}(= e_{ji}) \in E$ represents the edge joining the nodes v_i and v_j. For any $v_i, v_j \in V$ with $i < j$, let C_{ij} be the interlink cost (benefit) between v_i and v_j, i.e., the cost (benefit) generated by including in the same block these two nodes.

In the following, we present a valid formulation for the problem of finding the cheapest partition of the nodes of a graph with respect to the full pairwise interlink cost with the requirement that all nodes within a block must be connected. In order to guarantee the connection of the nodes in the same block, we use the flow based formulation of the Spanning Tree given by **?** for the capacitated minimal directed tree problem. However, since in our case we only need the connection among all the nodes in the same block, it is not mandatory to avoid cycles in the resulting graph connecting these nodes, i.e., it is not necessary to have a tree, so we can use a relaxed version of that formulation.

In the formulation of our problem we will use continuous flow variables to guarantee the connection among the nodes of the same block, defined on the arcs of the directed network $D = (V, A)$, where A consists of the set of arcs (i, j) and (j, i) such that the edge $e_{ij}(= e_{ji}) \in E$. We will assume that we have the same number of single source nodes as the number of blocks, which will be the nodes with the highest index in each block, with outflow the cardinality of the corresponding block minus one and zero inflow. All other nodes have a demand of one unit.

Therefore, in order to give a formulation of this problem based on the above ideas, we define the following families of variables. For any $i, k = 1, \dots, n$ such that $i \leq k$, the variable z_{ik} is defined as:

$$z_{ik} \quad = \quad \begin{cases} 1, & \text{if node } v_i \text{ is assigned to block } k, \\ 0, & \text{otherwise.} \end{cases}$$

For any $i, j = 1, \dots, n$ such that $i < j$, the variable x_{ij} is defined as:

$$x_{ij} \quad = \quad \begin{cases} 1, & \text{if nodes } v_i \text{ and } v_j \text{ are in the same block,} \\ 0, & \text{otherwise.} \end{cases}$$

And, for any $(i, j) \in A$, the variable f_{ij} is defined as:

$$f_{ij} \quad = \quad \text{amount of flow sent from node } i \text{ to node } j.$$

Therefore the flow-based formulation is as follows:

$$(F_{flow}) \quad \min \quad \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \tag{2}$$

$$s.t. \quad x_{ij} \leq z_{ik} + \sum_{\ell=j,\ell\neq k}^{n} z_{j\ell}, \quad \forall i, j, k = 1, \dots, n,\ i < j, i \leq k. \tag{3}$$

$$x_{ij} \geq z_{ik} + z_{jk} - 1, \quad \forall i, j, k = 1, \dots, n,\ i < j \leq k, \tag{4}$$

$$z_{ik} \leq z_{kk}, \quad \forall i, k = 1, \dots, n,\ i \leq k, \tag{5}$$

$$\sum_{k=i}^{n} z_{ik} = 1, \quad \forall i = 1, \dots, n, \tag{6}$$

$$\sum_{i=1:\,(k,i)\in A}^{n} f_{ki} - \sum_{i=1:\,(i,k)\in A}^{n} f_{ik} = \sum_{i=1}^{k} z_{ik} - 1, \quad \forall k = 1, \dots, n, \tag{7}$$

$$f_{ij} \leq (n-1) x_{ij}, \quad \forall (i, j) \in A,\ i < j, \tag{8}$$

$$f_{ji} \leq (n-1) x_{ij}, \quad \forall (j, i) \in A,\ i < j, \tag{9}$$

$$x_{ij} \in \{0, 1\}, \quad \forall i, j = 1, \dots, n,\ i < j, \tag{10}$$

$$z_{ik} \in \{0, 1\}, \quad \forall i, k = 1, \dots, n,\ i \leq k. \tag{11}$$

The objective function (2) accounts for the total interlinks cost of the nodes within the same block. The family of constraints (3) ensure that if i and j go in the same block then there is at least one representative of a block where both nodes can be assigned. With (4) we guarantee that the variable x_{ij} takes the value 1 if and only if nodes i and j are in the same block. Both together ensure the complete interconnection among the nodes of the same block. The family of constraints (5) ensures that each node is assigned to a block represented by a node v_k if v_k is assigned to this block. In addition, the family (6) guarantees that each node belong to just one block and this is represented by the node with the greatest index. Constraints (7) are the equations of balance of flow for the nodes of the graph. In particular, the node representing each block has an outgoing flow equal to the number of nodes in this block minus one and the remaining nodes of the block has demand 1. The families of constraints (8) and (9) avoid the flow between nodes of different blocks. Finally, (10) and (11) give us the binary condition over the variables x and z respectively. Observe that using (3) and (4), the integrality condition over the x variables can be relaxed and then, family of constraints (10) can be removed.

2.1.1 Valid inequalities

In the following, we propose some families of valid inequalities for the above formulation:

1. The complete interconnection among the nodes of the same block is reinforced as follows, see **??**:

$$x_{ij} + x_{j\ell} - x_{i\ell} \leq 1, \quad \forall i, j, l = 1, \ldots, n, \; i < j < \ell, \tag{12}$$

$$x_{i\ell} + x_{j\ell} - x_{ij} \leq 1, \quad \forall i, j, \ell = 1, \ldots, n, \; i < j < \ell, \tag{13}$$

$$x_{ij} + x_{i\ell} - x_{j\ell} \leq 1, \quad \forall i, j, \ell = 1, \ldots, n, \; i < j < \ell. \tag{14}$$

2. The idea that each block is represented by the node with the largest index is reinforced as follows:

$$z_{kk} + \sum_{j=k+1}^{n} x_{kj} \geq 1, \quad \forall k = 1, \ldots, n, \tag{15}$$

$$z_{kk} + x_{kj} \leq 1, \quad \forall j, k = 1, \ldots, n, \; j > k. \tag{16}$$

3. The relationship between variable x and z is strengthened as follows:

$$z_{ij} \leq x_{ij}, \quad \forall i, j = 1, \ldots, n, \; i < j. \tag{17}$$

Taking into account these valid inequalities, we present the following strengthening of F_{flow},

$$\begin{aligned} (\overline{F_{flow}}) \quad \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \\ s.t. \quad & (3) - (17). \end{aligned}$$

2.2 MTZ based formulation with two indices variables

In this section we propose a second formulation for our problem using the Miller-Tucker-Zemlin (MTZ) inequalities. MTZ inequalities guarantee the connectivity of the solutions and prevent cycles, these constraints were initially proposed by **?** in the context of the Traveling Salesman Problem. They have been adapted to other problems and reinforced by different authors, see, e.g. **????**. The MTZ formulation for the STP builds an arborescence rooted at a specified node $r \in V$, in which arcs follow the direction from the root to the leaves. It uses binary variables to represent the arcs of the arborescence. Each edge $e_{ij} \in E$, is associated with a pair of binary variables, f_{ij} and f_{ji}, which take the value 1 if and only if arcs (i,j) and $(j,i) \in A$ belong to the arborescence, respectively. In addition, it uses continuous variables ℓ_i, denoting the position that node v_i occupies in the arborescence with respect to r. In our case, we build as many trees as the number of blocks, and the roots of these trees will be the nodes with the largest index in each block. Since we will use this family of inequalities to ensure the connection among the nodes in the same block (not to build an arborescence), some of the constraints of this family can be removed.

Hence, in order to give a formulation for our problem based of the above ideas, we use the two families of binary variables that we have used in the previous formulation, i.e. x and z, and now, as mentioned, the variables f_{ij} are also binary and defined as

$$f_{ij} = \begin{cases} 1, & \text{if the arc } (i,j) \in A \text{ is chosen,} \\ 0, & \text{otherwise.} \end{cases}$$

Using these variables, the formulation of our problem is:

$$\begin{aligned} (F_{MTZ}) \quad \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \\ s.t. \quad & (3) - (6),\ (10), (11), \\ & \ell_i + 1 \le \ell_j + n(1 - f_{ij}), \quad \forall i, j = 1, \dots, n,\ (i,j) \in A, && (18) \\ & \sum_{i=1,\,(i,j)\in E}^{n} f_{ij} = 1 - z_{jj}, \quad \forall j = 1, \dots, n, && (19) \end{aligned}$$

$$f_{ij} + f_{ji} \leq x_{ij}, \quad \forall (i,j) \in A,\, i < j, \tag{20}$$

$$f_{ij} \in \{0,1\}, \quad \forall (i,j) \in E. \tag{21}$$

The family of constraints (18) guarantee the label assigned to node j is at least the label assigned to node i if the arc $(i,j) \in A$ is chosen. Actually, this family avoids the tours. In addition, (19) ensure there is, at least, an incident arc at each node different from the one representing its block. The family (20) guarantees that arcs joining nodes of different blocks cannot be chosen. Finally, constraints (21) give the binary condition of the f variables. As in F_{flow} formulation, we can relax the integrality condition over the x variables; and then, family of constraints (10) can be removed.

2.2.1 Valid inequalities

Valid inequalities (12)-(17) derived for the flow formulation are still valid for this formulation. Moreover, we have some additional ones.

Through the following two families of inequalities, we guarantee that the label assigned to the nodes representing the blocks is 1;

$$\ell_i \geq z_{ii}, \quad \forall i = 1, \ldots, n, \tag{22}$$

$$\ell_i \leq 1 + n(1 - z_{ii}), \quad \forall i = 1, \ldots, n. \tag{23}$$

The following inequalities ensure that the label associated with a node which is not representing its block will be at least 2,

$$2(1 - z_{ii}) \leq \ell_i, \quad \forall i = 1, \ldots, n. \tag{24}$$

Taking into account these valid inequalities, we present the following strengthening of F_{MTZ},

$$(\overline{F_{MTZ}}) \quad \min \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij}$$
$$s.t. \quad (3) - (6),\ (10), (11), (18) - (24).$$

2.2.2 Alternative formulation

Based on the previous formulation we can provide an alternative formulation where the family of variables z has been substituted by variables y_k $\forall k = 1, \ldots, n$, defined as:

$$y_k = \begin{cases} 1, & \text{if node } k \text{ is the node with the highest index in its partition,} \\ 0, & \text{otherwise.} \end{cases}$$

The formulation of the problem using this new family of variables is:

$$\begin{aligned} (F2_{MTZ}) \quad \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \\ s.t. \quad & (10),\ (12) - (14),\ (18),\ (20), (21), \\ & y_k + \sum_{j=k+1}^{n} x_{kj} \geq 1, \quad \forall k = 1, \ldots, n, && (25) \\ & y_k + x_{kj} \leq 1, \quad \forall k < j = 1, \ldots, n, && (26) \\ & \sum_{i=1,\, (i,j) \in A}^{n} f_{ij} = 1 - y_j, \quad \forall j = 1, \ldots, n, && (27) \\ & y_k \in \{0, 1\}, \quad \forall k = 1, \ldots, n. && (28) \end{aligned}$$

The families of constraints (25)-(26) guarantee that each node is assigned to just one block and this is represented by the node with the greatest index. Actually, these two families of constraints is a rewriting constraints (15) and (16) using variables y. In the same sense, constraints (27) are an adaptation of constraints (19) using variables y. Again, the integrality condition on the y variables can be relaxed by constraints (27).

The families of valid inequalities (22)-(24) are still valid for this formulation.

Taking into account these valid inequalities, we present the following strengthening of $F2_{MTZ}$,

$$\begin{aligned} (\overline{F2_{MTZ}}) \quad \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \\ s.t. \quad & (10),\ (12) - (14),\ (18),\ (20) - (28). \end{aligned}$$

2.3 Branch & Cut Procedure

For the three previous formulations, we have developed a B&C procedure, where we have introduced a family of cuts in each node of the branch and bound tree. In order to describe this procedure, for a given node of the branch and bound tree, let $\bar{x}$, $\bar{z}$ be the optimal values of variables x and z, respectively, in that node. For any $\varepsilon > 0$, we consider the following steps:

STEP 0 Consider that a node v_ℓ belongs to a block k with $\ell \leq k$ if and only if $\bar{z}_{\ell k} > \varepsilon$.

STEP 1 For each block, we identify the pairs of nodes v_i and v_j with $i < j$ that belong to the same block but they are not connected through the nodes of their block. In order to determine if two nodes in the same block are connected through the edges of their block, we have computed the shortest path between each pair of nodes of a block using the algorithm of Floyd-Warshall, taking into account that the length of the edge between two nodes, v_i and v_j, is

$$length(e_{ij}) := \begin{cases} 0, & \text{if } e_{ij} \in E,\ \bar{z}_{ik} > \varepsilon \text{ and } \bar{z}_{jk} > \varepsilon, \\ 1, & \text{otherwise.} \end{cases}$$

Hence, if the length of the shortest path between two nodes is 0, these nodes are connected.

STEP 2 Introduce the following cut:

$$\sum_{\ell=i+1\,:\,\bar{z}_{\ell k}\leq\varepsilon}^{k-1} x_{i\ell} + \sum_{\ell=k+1}^{n} x_{i\ell} + \sum_{\ell=1\,:\,\bar{z}_{\ell k}\leq\varepsilon}^{i-1} x_{\ell i} - x_{ij} \geq 0. \tag{29}$$

The rationale behind inequality (29) is the following. Assume that we are given $v_k \in V$, $\varepsilon > 0$ and a set of nodes, $\mathcal{K} := \{v_\ell \ :\ \ell \leq k \text{ and } \bar{z}_{\ell k} > \varepsilon\}$. The set $\mathcal{K}$ would become a block if the graph induced by its nodes were connected. On the contrary, if $v_i, v_j \in \mathcal{K}$ were not connected in the graph induced by $\mathcal{K}$; then any block containing v_i a v_j should include at least another node, $v_\ell \notin \mathcal{K}$, to ensure the connection of the resulting induced graph. This condition is encoded in the inequality (29) assuming that the inequality $x_{i\ell} \geq x_{ij}$ is fulfilled for some $v_\ell \notin \mathcal{K}$, i.e., if v_i and v_j belong to the same block, which forces $x_{ij} = 1$, then $v_\ell \notin \mathcal{K}$ also belongs to the same block, $x_{i\ell} = 1$.

For the alternative formulation, $F2_{MTZ}$, where the z variables are not used anymore, we consider the following cuts based on the same idea above:

$$\sum_{\ell=i+1\,:\,\bar{x}_{\ell k}<\varepsilon}^{k-1} x_{i\ell} + \sum_{\ell=k+1}^{n} x_{i\ell} + \sum_{\ell=1\,:\,\bar{x}_{\ell k}<\varepsilon}^{i-1} x_{\ell i} - x_{ij} \geq 0 \tag{30}$$

The results obtained applying this Branch & Cut procedure to the formulations $\overline{F_{flow}}$, $\overline{F_{MTZ}}$ and $\overline{F2_{MTZ}}$ will be refered to B&C_$\overline{F_{flow}}$, B&C_$\overline{F_{MTZ}}$, and B&C_$\overline{F2_{MTZ}}$, respectively.

2.4 Incomplete formulations

In order to obtain better computational results in the solution times of the previous formulations and to increase the size of the instances that we are able to solve, we propose to use incomplete formulations. That is, we propose to use formulations which are not valid for our problem and introduce cuts progressively in the nodes of the branch and bound tree to obtain feasible solutions. Based on this idea, we follow three strategies to generate incomplete formulations. Observe that previous formulations have two main groups of constraints, the first one to guarantee the interlink and the second one, to ensure the connection; both among the nodes in the same block.

The first strategy consists of removing the group of constraints that ensures the connection among the nodes of the same block; the second one will consider the same idea but the interlink is modeled in a different way (using exclusively variables x like in Formulation $F2_{MTZ}$). And, the third strategy 1) removes the group of constraints ensuring the connection among the nodes of the same block and 2) partially removes the group of constraints to ensure the interlink among the nodes of the same block. In the following, we show the three incomplete formulations that have provided the best computational results out of an number of tests among different alternatives.

2.4.1 First incomplete formulation

In order to generate the first incomplete formulation, we consider formulation $\overline{F_{MTZ}}$ without constraints (18) and (22)-(24). Therefore, the resulting formulation ensures that all the x variables between pair of nodes in the same block take the value 1, but the connection between each pair of nodes of the same block is not guaranteed. Hence, in the solution procedure we will proceed as in the B&C scheme previously described, i.e., the non-connected solutions obtained in each node of the branch-and-bound tree are cut with the following constraints, where ε is a fixed positive amount:

$$\sum_{\ell=i+1\,:\,\bar{z}_{\ell k}\leq\varepsilon}^{k-1} \bar{x}_{i\ell} + \sum_{\ell=k+1}^{n} x_{i\ell} + \sum_{\ell=1\,:\,\bar{z}_{\ell k}\leq\varepsilon}^{i-1} x_{\ell i} - x_{ij} \geq 0.$$

As in B&C procedure, to determine if two nodes are connected through the edges of a block, we have computed the shortest path between each pair of nodes within the block using the algorithm of Floyd-Warshall. In this subproblem we have considered a complete graph induced by the nodes in the block and the length of the edge between two nodes is defined as

0 if this edge actually exists in the original graph and 1 otherwise. Therefore, if the length of the shortest path between two nodes is 0, these nodes were originally connected.

This incomplete formulation has been reinforced with the following valid inequality:

$$\sum_{\substack{i=1\\ e_{ij}\in E}}^{j-1} x_{ij} + \sum_{\substack{i=j+1\\ e_{ji}\in E}}^{n} x_{ji} \geq 1 - z_{jj}, \quad \forall j = 1, \dots, n. \tag{31}$$

The family of inequalities (31) gives a necessary condition for the connection of the nodes in the same group but it is not sufficient. Actually, it implies that every node of a block except the representative of the block must be connected with another node of the block via an edge of E. Therefore, the first incomplete formulation is:

$$\begin{aligned} (FI_1) \quad \min \quad & \sum_{i=1}^{n}\sum_{j=i+1}^{n} C_{ij}x_{ij} \\ s.t. \quad & (3)-(6), (10)-(17), (19)-(21), (31). \end{aligned}$$

2.4.2 Second incomplete formulation

In the second incomplete formulation, we consider formulation $\overline{F2_{MTZ}}$ where the family of constraints (18) and (22)-(24) have been removed and (31) is included. Therefore, the second incomplete formulation is:

$$\begin{aligned} (FI_2) \quad \min \quad & \sum_{i=1}^{n}\sum_{j=i+1}^{n} C_{ij}x_{ij} \\ s.t. \quad & (10), (12)-(14), (20), (21), (25)-(28), (31). \end{aligned}$$

As before, these constraints ensure that all the x variables between pair of nodes in the same block take the value 1, but the connection between each pair of nodes of the same block is not guaranteed.The difference with respect to the previous one is that this formulation does not use z variables. Therefore, the non-connected solutions obtained in each node of the branch-and-bound tree are cut with the family of valid inequalities (30).

2.4.3 Third incomplete formulation

In the third incomplete formulation, we consider Formulation FI_2 where the family of constraints (12)-(13) have been partially removed. Therefore, the third incomplete formulation is:

$$
\begin{aligned}
(FI_3) \quad \min \quad & \sum_{i=1}^{n} \sum_{j=i+1}^{n} C_{ij} x_{ij} \\
s.t. \quad & (10), (14), (20), (21), (25) - (28), (31), \\
& x_{1j} + x_{j\ell} - x_{1\ell} \leq 1, \quad \forall j, l = 1, \ldots, n,\ i < j < \ell, \qquad (32) \\
& x_{in} + x_{jn} - x_{ij} \leq 1, \quad \forall i, j = 1, \ldots, n,\ i < j < \ell. \qquad (33)
\end{aligned}
$$

In this case, besides to not guarantee the connection of the nodes within the same block, this formulation does not ensure the interlink among all the nodes of the same block. Therefore, in addition to adding the violated cuts of the family (30), we need some additional valid inequalities, namely the violated constraints of the family (12)-(13).

3 Heuristic

The problem complexity requires the development of heuristic methods for two reasons. The first reason is that we want to solve problems of any size, but integer linear programming solves medium-sized problems only. The second reason is that an approximate solution to the problem is needed by integer programming too. In fact, computational times can decrease if integer programming starts with a good feasible solution, with which to discard the nodes that do not contain optimal solutions.

The two method that are proposed here are plain variations on local search heuristics. It can be readily seen from the combinatorial structure of the problem that the clusters of an incumbent solution can be modified re-assigning objects to different clusters. If the connectivity constraints are still satisfied and the objective function improves, then the incumbent solution is updated. The process is then repeated and stops when clustering cannot be improved. In this case it said that a local optimum has been reached. More formally For every node i, let $N[i] = \{j | ij \in E\} \subseteq V$ be the node set of the neighbors of i. Let $\Pi = \{V_1, \ldots, V_p\}$ be a partition of V, the partition is feasible if sets V_j are connected for all $j = 1, \ldots, p$. Let $V_{c(i)}$ be that set such that $i \in V_j$, that is, $c(i) = j$.

A move $m = (i, q)$ is a pair $i \in V$, $q = 0, \ldots, p$. If $q \geq 1$, then the move represents the possible assignment of unit i to cluster V_q; the move is feasible if both new cluster $V_{c(i)} \setminus i$ and $V_q \cup i$ are connected. If $q = 0$, then the move represents the assignment of i to an an empty cluster, that is i becomes a singleton and eventually $V_{p+1} = \{i\}$. The evaluation of a feasible move is calculated as the value:

$$\delta_{iq} = \begin{cases} \sum_{j \in V_q} c_{ij} - \sum_{j \in V_{c(i)}} c_{ij} \text{ if } q \geq 1; \\ -\sum_{j \in V_{c(i)}} c_{ij} \text{ if } q = 0. \end{cases} \tag{34}$$

The Local Search Algorithm 1 begins with a feasible partition Π, then tries to improve the objective function moving units to different clusters. If there is no feasible move that decreases the objective function, then Π is a local optimum. When a local optimum is found, then the procedure can be applied again starting with new initial partitions, till a maximum t^{max} re-starts are tried. The procedure to determine the local optimum is described as follows: 1.

Algorithm 1 Local Search Algorithm

```
1: procedure LOCAL SEARCH FOR CONNECTED CLUSTER
2:     for do$t$ from 1 to $t^{max}$                          ▷ Local Search is repeated $t^{max}$ times
3:         $(\Pi, f) \leftarrow RandomNetClique(version = "`RR"' \text{ or } "`VNS"')$    ▷ $\Pi$ is a random
   feasible partition, $f$ is the objective function
4:         $loc\_opt = FALSE$                                  ▷ Condition for a local optimum
5:         while $loc\_opt = FALSE$ do
6:             $\Delta \leftarrow Moves\_Compilation(\Pi, \ldots)$           ▷ Generate a list of moves
7:             $(M, d) \leftarrow Choose\_Move(\Delta, \ldots)$      ▷ $M$ a subset of moves, $d$ variation of the
   objective function
8:             if $d < 0$ then
9:                 $f \leftarrow f + d$,                        ▷ Decrease of the Objective Function
10:                Apply $M$ to $\Pi$.                          ▷ Obtain a new partition $\Pi$
11:            else
12:                $loc\_opt = TRUE$.                           ▷ Condition for a local optimum
13:            end if
14:        end while
15:    end for
16: end procedure
```

The algorithm begins calculating a random clustering Π with objective function f. Partition Π is calculated by "'Random Restart"' or "'Variable Neighborhood Search"' (the methods are described later). The while loop of instructions from 4 to 13 leads Π to a local optimum, applying repeatedly the procedures Moves_Compilation and Choose_Move. The output of Moves_Compilation is a matrix $\Delta = [\delta_{iq}], i = 1 \ldots n, q = 0 \ldots p$ listing the improvement of the objective function for each feasible move. The output of the procedure Choose_Move is a list of moves $M = \{m_1, \ldots, m_w\}$ and a value d. The list of moves M, when applied to Π, leads to a new feasible partition with the value of the objective function decreased by d. The moves of M are chosen in a greedy way from the matrix Δ, considering that there are multiple moves improving the objective function without needing to recalculate Δ. Indeed, if a move $m_t = (i, q)$ is chosen and applied to Π, then values δ_{js} remain the same if $s \neq q, c(i)$ and $c(j) \neq q, c(i)$. More precisely, procedure Choose_Move is described as follows:

Algorithm 2 Choose Move Procedure

```
1:  procedure CHOOSE MOVE
2:      Input: $\Delta = [\delta_{iq}], j = 1, \ldots, n, q = 0, \ldots, p$
3:      Output: $M$= list of moves; $d$ = variation of the objective function
4:      $V \leftarrow \{1, \ldots, n\}, K \leftarrow \{1, \ldots, p\}, M \leftarrow \emptyset$
5:      $D \leftarrow 0, f_i mprove \leftarrow TRUE$                              ▷ Initialization
6:      while $V \neq \emptyset, K \neq \emptyset, f\_improve = TRUE$ do
7:          Let $\delta_{iq} = \min\{\delta_{jw} | j \in V, w \in K\}$              ▷ Move Choice
8:          if $\delta_{iq} < 0$ then
9:              $d = d + \delta_{iq}$                                            ▷ update $d$
10:             $M \leftarrow M \cup \{(i, q)\}$
11:             $V \leftarrow V \setminus \{i\}$
12:             $K \leftarrow K \setminus \{c(i)\}$
13:             if $q \geq 0$ then
14:                 $K \leftarrow K \setminus q$
15:             end if
16:         else
17:             $delta\_positive = FALSE$                    ▷ no improvement moves are left
18:         end if
19:     end while
20: end procedure
```

It remain to describe how partitions Π are generated in Step 3 of Algorithm 1. Two alternatives are tested. The first procedure works out Π from scratch and randomly at every iteration and is called Random Restart. RR is implemented labeling objects with random integer numbers between 1 and c^{max}, then the clusters of Π are the connected objects having the same label. The final number of clusters can be larger than c^{max}. RR is the simplest of the family of the so-called Multi-Start methods, **?**, and used often as the benchmark to which to compare more elaborate versions, Benati (2008). In this contribution, RR is compared to a second procedure called Variable Neighborhood Search (VNS). The starting solutions of VNS mild randomness with information contained in the best found solution. It is implemented as follows: at the first iteration Π is a random partition and the first local search calculates the first best solution Π^{best}. A new Π is worked out from Π^{best} relocating t objects into different clusters, where t is a varying parameter. At the earliest stages, t is small to search for improved solutions in the immediate proximity of Π^{best}. If no improvement of Π^{best} is found, than t is increased, to explore solutions that are further away from Π^{best}. When t reassignments are to be done, both objects and clusters are chosen randomly and an object is relocated only when the modified clusters remain connected. The qualitative difference between RR and VNS is that the search of the latter is more constrained than the search of the former, as it is designed to take advantage of the (potential) good clusters contained in Π^{best}. The principle of VNS are explained in Hansen et al. (2010), recent applications of VNS to clustering problems are Hansen et al. (2012) and **?**.

Algorithm 3 CH election algorithm

```
1: procedure CH–Election
2:     for each node $i \in N$ do
3:         Broadcast HELLO message to its neighbor
4:         let $k \in N1\ (i)$ U $i$ be s.t
5:         QOS($k$) = max QOS($j$) $|j \in N1(i)$ U $i$
6:         MPRSet($i$) = $k$
7:     end for
8: end procedure
```

4 Computational Results

Algorithms are tested on simulated problems, in which parameters determine the clusters features and the network connectivity. The experiment layout is the one proposed in **?**: Data are composed of n units on which m binary features, $F_i = \{0, 1\}, i = 1, \ldots, m$, are recorded. Units belong to one of two groups, each group is composed of $n/2$ units. If one unit belongs to group 1, then $\Pr[F_i = 1] = p_c$ for all $i = 1, \ldots, m$, otherwise, if the unit belongs to group 2, then $\Pr[F_i = 1] = 1 - p_c$ for all $i = 1, \ldots, m$. If p_c is close to one, then the two groups are well separated, as all units of Group 1 tend to be a vector of ones and all units of Group 2 is a vector of zeros. As p_c gets closer to 0.5, the distinction between the two groups is less and less precise. Then units are connected through arcs: If two units (or nodes) belongs to the same group then the probability that there is an arc between the two units is p_{in} (the probability of an internal arc). If the two nodes belongs to two different groups, then the probability that there is an arc between the two units is p_{out} (the probability of an external arc). vertex i is connected to X_{in} vertices of the same group of i and the X_{out} vertices of the other group, where X_{in} and X_{out} are random variables. With the parameter above, $E[X_{in}] \approx np_{in}/2$ and $E[X_{out}] \approx np_{in}/2$. All experiments are run with $p_{in} > p_{out}$, so that connectivity provides information: If a node i, whose membership is uncertain, is connected with a node j that is known to belong to Group k, then it is likely that i belongs to k as well.

The experiments on small and medium sized problems are run on random data generated with the following parameters: $m = 10$, $E[X_{in}] = 3$ and $E[X_{out}] = 1$, $p_c = 0.60$. The value t^{max} the maximum number of starting solutions, has been fixed to $10n$. Results are reported in Table 3. Columns report the objective function and the number of iterations to get to it, iterations calculated as the number of times that the matrix Δ is worked out in line 6 of Algorithm 1. The last column reports the difference between the best found heuristic solution and optimal one: empty spaces means that they are equal, unknown means that the instance has not solved by Branch&Bound.

Both algorithms start from the same initial partition Π, but finally there is a clear evidence that RR is better than VNS. In 26 out of 40 problems RR the objective function of RR is better than the one of VNS, in only 2 occurrences out of 40 VNS is better than RR. Considering only the 26 instances in which RR is better than VNS, one can see that in 23 out of 26 the iteration in which the the best solution is found is larger for RR than for VNS, with an average of 1197 iterations needed by RR and only 358 required by VNS (that remains trapped in a local optimum). It seems to point out that the VNS search is too constrained: It is exploring solutions that are too close to previous ones and does not go where the optimum is found. Unfortunately, RR cannot find the optimal solution to 18 out of 37 solved instances.

Results for graphs with $n = 80, 100$ are reported in Table 4. Here the results of RR and VNS are similar, the avereges of the objective function are equal up to 1 digital points, versus . The best solution found by RR 8 out of 20 times by RR, 11 by VNS and one tie. When we consider the instances in which RR is better, the iterations needed to the best solution found are 10276 on averages for RR, 3876 for VNS. The impressions is again that VNS cannot escape poor qualities initial solutions. This is also supported by the fact that, when the only the 11 instances in which VNS is best are considered, there VNS takes 3895 iterations against 5248 of RR. It seems that having a good starting solution is crucial for VNS to take advantage of its mechanism of neighborhood solutions.

The experiment has been run with fixed values of $n = 50$ and $m = 10$, $p_{out} = 0.04$ and with varying parameters $p_c = 0.65, 0.55, 0.45$, $p_{in} = 0.12, 0.16, 0.20$. Given those probabilities, for a given node the expected number of arcs pointing outside the group is 1, the expected number of arcs pointing within the group can be 3, 4, or 5. The input of a test problem is a matrix D and a graph G, that are randomly generated according to the probabilities described before.
In this section we have computationally tested the formulations presented in Section 2 and their corresponding strengthening. Thus, we have implemented all of them in the commercial solver XPRESS-IVE 1.24.04 running on a Intel(R) Core(TM) i7-4790 CPU @400GHz 32GB RAM. The cut generation option of XPRESS was disabled in order to compare the relative performance of the formulations cleanly.

Algorithms are tested on simulated problems, in which parameters determine the cluster features and the network connectivity. Parameters values are chosen to model application data in which the true clusters are hard to detect without the additional information of the units connectivity. Data are composed of n units on which m binary features, $V_i = \{0, 1\}, i = 1, \dots, m$, are recorded, . Units belong to one of two groups, each group is composed of $n/2$ units. If one unit belongs to group 1, then $\Pr[V_i = 1] = p_c$ for all $i = 1, \dots, m$, otherwise, if the unit belongs to group 2, then $\Pr[V_i = 1] = 1 - p_c$ for all $i = 1, \dots, m$. If p_c is close to one, then the two groups are well separated, as all units of Group 1 tend to be a vector of ones and all units of Group 2 is a vector of zeros. As p_c gets closer to 0.5, the distinction between the two groups is less and less precise. Then units are connected through arcs. If two units (or nodes) belongs to the same group then the probability that there is an arc between the two units is p_{in} (the probability of an internal arc). If the two nodes belongs to two different groups, then the probability that there is an arc between the two units is p_{out} (the probability of an external arc). If $p_{in} > p_{out}$, then connectivity provides information: If a node i, whose membership is uncertain, is connected with a node j that is known to belong to Group k, then it is likely that i belongs to k as well.

The results are reported in Table 2 and they correspond to the average of those obtained after solving ten instances for each size, the time was limited to two hours of CPU. Hence, the first two columns of Table 2 correspond to the size and the formulation used to solve these instances, respectively. The third and fourth columns give the number of instances that were not optimally solved within the time limit and the CPU time in seconds, respectively. The three last columns, GAP, GAP_R and Nodes, stand for the averages of: gap between the best bound and the best solution after 7200 seconds (for the instances whose CPU time exceeded the time limit), the gap in the root node and number of nodes in the B&B tree, respectively. To obtain a general idea of the comparisons among these averaged values, for the results in the column *Time* and for different formulations, we have accounted the value 7200 seconds for those instances that exceed the time limit. In the same way, the values used to computed the average of the column *Nodes* have been the number of nodes of the B&B tree when the CPU time limit was reached.

The formulations described in the second column correspond to the ones described in Section 2. The last one, "$FI_3 + sol\_heu$" corresponds to the third incomplete formulation where the solver has been fed with the solution provided by the heuristic approach. As we can see, among the three first formulations, $\overline{F_{flow}}$, $\overline{F_{MTZ}}$ and $\overline{F2_{MTZ}}$, the third one provides much better results than the other ones. Regarding the branch and cut approach, although it does not provide an uniform improvement with respect to the running times, it slightly improves the gap in the instances where the optimal solution is not achieved within the limit time. With respect to the incomplete formulations, the third-incomplete formulation reports much better computational results than the remaining ones. Observe that the best computational results have been obtained when the solution obtained by the heuristic approach is fed to the solver using the third incomplete formulation. As a conclusion, we can see that the successive improvements from the first to the last formulation have provided a remarkable reduction of CPU time (around two orders of magnitude).

Table 1: Add caption

n	t_exact	t_heur	#	GAP_exact/heur
20	0.9		9	0.2
30	29.4		7	1.2
36	798.8		2	3.3
40	2774.6		3	4.6

5 Computational Results

The main motivation of the new clustering model proposed here is that connection data improves the quality of the clustering algorithms. The following experiment proves that it is so. The experiment has been run with fixed values of $n = 50$ and $m = 10$, $p_{out} = 0.04$ and with varying parameters $p_c = 0.65, 0.60, 0.55$, $p_{in} = 0.12, 0.16, 0.20$. Given those probabilities, for a given node the expected number of arcs pointing outside the group is 1, the expected number of arcs pointing within the group are approximately 3, 4, or 5. Varying p_c, data more and more confounded are generated. The mode of Group 1 is vector of ones, the mode of Group 2 is a vector of zeros. Denoting with Z the number of ones of a unit of Group 1, with $m = 10$ and $p_c = 0.65$, $\Pr[Z \leq 4] \approx 0.10$, that is the probability of being closer to the mode of the other group. The probability of of the same distance is $\Pr[Z = 5] \approx 0.20$. With $p_c = 0.55$, $\Pr[Z \leq 4] \approx 0.25$, $\Pr[Z = 5] \approx 0.24$.

Three clustering methods are considered. The first two methods are the k-means and the clique-partition, two methods that do not consider the network structure of the data. The third method is the connected-clique-partition presented in this paper. The k-means algorithm is run with k equal to the exact value of 2, an assumption that seldom can be made in practice. The methods are compared in terms of the ARI, the adjusted Rand index Hubert and Arabie (1985). The ARI is an index that compares the true and the estimated partitions of a population. It is equal to 1 if the true and the estimated partition are equal, it is close to 0 (with the possibility of being negative) if the estimated partition is equivalent to the random one.

The result of the experiment are reported in Table 5. Looking at the averages, it can be seen that the best ARI has been obtained by the net-clique clustering, with a value of

n	Nombre	\#	Time	GAP_2	GAP_R_2	Nodes
20	$\overline{F_{flow}}$	0	12.3	0.0	25.7	3174.4
	$\overline{F_{MTZ}}$	0	56.0	0.0	21.2	29193.5
	$\overline{F2_{MTZ}}$	0	9.8	0.0	18.6	11180.5
	B&C_$\overline{F_{flow}}$	0	16.1	0.0	25.6	3160.9
	B&C_$\overline{F_{MTZ}}$	0	15.2	0.0	21.2	6201.3
	B&C_$\overline{F2_{MTZ}}$	0	6.7	0.0	20.4	4747.0
	FI_1	0	2.6	0.0	16.0	257.9
	FI_2	0	2.0	0.0	17.0	55.8
	FI_3	0	1.0	0.0	27.9	116.9
	FI_3+sol_heu	0	0.9	0.0	27.6	36.6
30	$\overline{F_{flow}}$	4	4281.4	13.8	34.1	68943.2
	$\overline{F_{MTZ}}$	4	5040.8	6.9	29.8	282112.6
	$\overline{F2_{MTZ}}$	0	1389.8	0.0	25.9	254538.7
	B&C_$\overline{F_{flow}}$	4	4450.7	11.3	33.2	71208.1
	B&C_$\overline{F_{MTZ}}$	3	4358.2	6.3	29.8	250908.6
	B&C_$\overline{F2_{MTZ}}$	1	2021.1	0.5	25.9	385374.5
	FI_1	2	1615.9	2.8	23.9	3430.8
	FI_2	0	141.4	0.0	24.2	978.8
	FI_3	0	46.9	0.0	34.9	2473.4
	FI_3+sol_heu	0	29.4	0	34.9	1104,8
36	$\overline{F_{flow}}$	10	7199.9	31.8	44.1	26340.1
	$\overline{F_{MTZ}}$	10	7199.8	33.0	46.8	88221.4
	$\overline{F2_{MTZ}}$	7	6121.1	11.0	31.1	506437.0
	B&C_$\overline{F_{flow}}$	10	7200.0	25.9	41.0	29230.4
	B&C_$\overline{F_{MTZ}}$	10	7199.7	47.8	58.6	83581.3
	B&C_$\overline{F2_{MTZ}}$	8	6504.2	16.2	35.4	353671.2
	FI_1	3	3422.4	4.1	24.1	7073.5
	FI_2	1	1503.2	2.0	24.7	3946.7
	FI_3	0	1096.8	0.0	34.8	21787.0
	FI_3+sol_heu	0	798.8	0.0	34.8	11853.0
40	$\overline{F_{flow}}$	10	7200.3	60.8	65.0	12973.8
	$\overline{F_{MTZ}}$	10	7200.1	45.7	53.1	35412.6
	$\overline{F2_{MTZ}}$	8	6103.9	36.1	47.7	105652.6
	B&C_$\overline{F_{flow}}$	10	7199.9	38.2	43.8	14839.6
	B&C_$\overline{F_{MTZ}}$	10	7200.0	40.4	48.7	33290.0
	B&C_$\overline{F2_{MTZ}}$	9	6719.2	35.0	46.9	111553.1
	FI_1	8	6441.8	13.6	26.8	6767.5
	FI_2	4	3857.9	11.2	28.6	4404.3
	FI_3	3	3433.9	6.1	37.2	24669.4
	FI_3+sol_heu	3	2774.6	5.0	36.6	18384.9

Table 2: Computational results of formulations in Section 2.

approximately 0.21, while the k-means and the clique partition gave approximately the same result of 0.13. The net-clique method always improves the partitions of the other methods, with only a few exceptions that are when data are so stranded that no methods is able to recognize any structure (the ARI being approximately equal to 0). Table 6 and 7 reports the ARI averages controlling on values of p_{in} and p_c. Again, a steadily improvement can be observed.

References

Benati, S. Categorical data fuzzy clustering: An analysis of local search heuristics. *Com-*

puters and Operations Research 35, 3 (2008), 766–775. cited By 14.

Cheng, H., Zhou, Y., Huang, X., and Yu, J. X. Clustering large attributed information networks: an efficient incremental computing approach. *Data Mining and Knowledge Discovery 25*, 3 (2012), 450–477.

Combe, D., Largeron, C., Egyed-Zsigmond, E., and Géry, M. Combining relations and text in scientific network clustering. In *International Conference on Advances in Social Networks Analysis and Mining, ASONAM 2012, Istanbul, Turkey, 26-29 August 2012* (2012), pp. 1248–1253.

Grotschel, M., and Wakabayashi, Y. A cutting plane algorithm for a clustering problem. *Mathematical Programming 45*, 1-3 (1989), 59–96. cited By 112.

Hansen, P., Mladenovic, N., and Moreno Perez, J. Variable neighbourhood search: Methods and applications. *Annals of Operations Research 175*, 1 (2010), 367–407. cited By 203.

Hansen, P., Ruiz, M., and Aloise, D. A vns heuristic for escaping local extrema entrapment in normalized cut clustering. *Pattern Recognition 45*, 12 (2012), 4337–4345. cited By 3.

Hubert, L., and Arabie, P. Comparing partitions. *Journal of Classification 2*, 1 (1985), 193–218. cited By 1975.

Inglehart, R., and Baker, W. Modernization, cultural change, and the persistence of traditional values. *American Sociological Review 65*, 1 (2000), 19–51. cited By 1527.

Johnson, E., Mehrotra, A., and Nemhauser, G. Min-cut clustering. *Mathematical Programming 62*, 1-3 (1993), 133–151. cited By 115.

McPherson, M., Smith-Lovin, L., and Cook, J. Birds of a feather: Homophily in social networks. *Annual Review of Sociology 27* (2001), 415–444. cited By 3794.

Neville, J., Adler, M., and Jensen, D. D. Clustering relational data using attribute and link information. In *Proceedings of the Workshop on Text Mining and Link Analysis, Eighteenth International Joint Conference on Artificial Intelligence* (Acapulco, Mexico, 2003).

Wasserman, M., and Faust, K. *Social Networks Analysis: Methods and Applications.* Cambridge University Press.

Xu, Z., Ke, Y., Wang, Y., Cheng, H., and Cheng, J. GBAGC: A general bayesian framework for attributed graph clustering. *ACM Transactions on Knowledge Discovery from Data 9*, 1 (2014), 5:1–5:43.

name	RR	it	VNS	it	Optimal
20-1	-114	11	-114	64	
20-2	-74	28	-34	4	
20-3	-122	34	-122	67	
20-4	-112	14	-110	8	
20-5	-128	233	-102	8	
20-6	-102	112	-96	13	
20-7	-154	100	-102	442	
20-8	-94	895	-92	226	- 96
20-9	-116	9	-116	28	
20-10	-140	11	-140	10	
30-1	-254	1210	-248	1661	
30-2	-152	171	-152	109	
30-3	-200	1265	-144	12	-210
30-4	-200	1378	-170	1003	
30-5	-288	1675	-276	296	
30-6	-260	382	-260	101	
30-7	-228	377	-222	280	
30-8	-122	906	-108	95	-126
30-9	-276	973	-136	16	
30-10	-168	2274	-154	21	-176
36-1	-296	3437	-296	1033	-300
36-2	-300	588	-300	33	-304
36-3	-356	1372	-340	15	-390
36-4	-326	1573	-304	1015	-340
36-5	-300	767	-300	420	
36-6	-286	121	-286	1045	
36-7	-310	1723	-324	452	-344
36-8	-230	1988	-204	25	-246
36-9	-260	1643	-242	518	-268

n	p_c	$E[X_{in}]$	$E[X_{out}]$	label	RR	it	VNS	it
80	0.6	3.0	1.0	1	-1030	2389	-1086	4245
80	0.6	3.0	1.0	2	-968	7601	-1042	2670
80	0.6	3.0	1.0	3	-1274	14858	-1260	141
80	0.6	3.0	1.0	4	-976	8314	-900	1035
80	0.6	3.0	1.0	5	-1234	1487	-1370	415
80	0.6	3.0	1.0	6	-936	10286	-818	4512
80	0.6	3.0	1.0	7	-1246	10875	-1286	3260
80	0.6	3.0	1.0	8	-998	10262	-904	1954
80	0.6	3.0	1.0	9	-1190	5695	-1196	1562
80	0.6	3.0	1.0	10	-1416	2514	-1440	1732
100	0.6	3.0	1.0	1	-1630	5027	-1482	10911
100	0.6	3.0	1.0	2	-1730	2097	-1908	7597
100	0.6	3.0	1.0	3	-1216	3138	-1266	2873
100	0.6	3.0	1.0	4	-2094	4527	-1966	3299
100	0.6	3.0	1.0	5	-1442	13965	-1386	6752
100	0.6	3.0	1.0	6	-2176	2844	-2176	3648
100	0.6	3.0	1.0	7	-1686	2289	-1756	3034
100	0.6	3.0	1.0	8	-1390	15593	-1484	408
100	0.6	3.0	1.0	9	-1934	14973	-1798	2405
100	0.6	3.0	1.0	10	-2136	4052	-2194	15048
					-1435.1	7139.3	-1435.9	3875.1

Table 4: Heuristic results for the largest sized problems

units	features	p_c	p_{in}	p_{out}	k-means	clique	net-clique
50	10	0.60	0.20	0.04	0.038	0.058	0.136
50	10	0.60	0.20	0.04	0.021	0.013	0.121
50	10	0.60	0.20	0.04	0.255	0.275	0.374
50	10	0.60	0.20	0.04	0.020	-0.004	0.083
50	10	0.60	0.20	0.04	-0.006	0.002	0.131
50	10	0.60	0.20	0.04	-0.019	-0.018	0.075
50	10	0.60	0.20	0.04	-0.018	0.081	0.163
50	10	0.60	0.20	0.04	0.039	0.098	0.123
50	10	0.60	0.20	0.04	0.059	0.071	0.135
50	10	0.60	0.20	0.04	0.300	0.050	0.339
50	10	0.60	0.16	0.04	0.177	0.125	0.192
50	10	0.60	0.16	0.04	0.144	0.157	0.231
50	10	0.60	0.16	0.04	0.347	0.195	0.267
50	10	0.60	0.16	0.04	-0.014	0.019	0.084
50	10	0.60	0.16	0.04	-0.006	0.192	0.404
50	10	0.60	0.16	0.04	0.177	0.214	0.257
50	10	0.60	0.16	0.04	0.143	0.122	0.137
50	10	0.60	0.16	0.04	0.215	0.110	0.093
50	10	0.60	0.16	0.04	-0.018	0.017	0.091
50	10	0.60	0.16	0.04	-0.004	0.007	0.216
50	10	0.60	0.12	0.04	0.039	0.003	0.147
50	10	0.60	0.12	0.04	0.006	0.073	0.055
50	10	0.60	0.12	0.04	0.177	0.177	0.298
50	10	0.60	0.12	0.04	0.214	0.140	0.193
50	10	0.60	0.12	0.04	0.060	-0.019	0.225
50	10	0.60	0.12	0.04	0.085	0.215	0.203
50	10	0.60	0.12	0.04	0.215	0.301	0.342
50	10	0.60	0.12	0.04	0.060	0.138	0.130
50	10	0.60	0.12	0.04	-0.006	-0.012	0.145
50	10	0.60	0.12	0.04	0.143	0.017	0.161
50	10	0.65	0.20	0.04	0.020	0.263	0.336
50	10	0.65	0.20	0.04	0.300	0.193	0.328
50	10	0.65	0.20	0.04	0.347	0.372	0.541
50	10	0.65	0.20	0.04	0.509	0.451	0.472
50	10	0.65	0.20	0.04	0.451	0.422	0.490
50	10	0.65	0.20	0.04	-0.006	-0.022	0.060
50	10	0.65	0.20	0.04	-0.019	0.133	0.138
50	10	0.65	0.20	0.04	0.509	0.374	0.429
50	10	0.65	0.20	0.04	0.299	0.203	0.456
50	10	0.65	0.20	0.04	0.215	0.089	0.239
50	10	0.65	0.16	0.04	0.299	0.203	0.328
50	10	0.65	0.16	0.04	0.214	0.237	0.344
50	10	0.65	0.16	0.04	0.508	0.415	0.347
50	10	0.65	0.16	0.04	0.452	0.400	0.430
50	10	0.65	0.16	0.04	0.143	0.144	0.205
50	10	0.65	0.16	0.04	0.397	0.637	0.774
50	10	0.65	0.16	0.04	0.569	0.481	0.637
50	10	0.65	0.16	0.04	0.011	0.206	0.346
50	10	0.65	0.16	0.04	0.397	0.127	0.356
50	10	0.65	0.16	0.04	0.177	0.105	0.208
50	10	0.65	0.12	0.04	0.300	0.200	0.322
50	10	0.65	0.12	0.04	0.397	0.507	0.645
50	10	0.65	0.12	0.04	0.509	0.422	0.484
50	10	0.65	0.12	0.04	0.021	0.283	0.309
50	10	0.65	0.12	0.04	0.398	0.236	0.465
50	10	0.65	0.12	0.04	0.112	0.182	0.461
50	10	0.65	0.12	0.04	0.509	0.449	0.474

	0.55	0.6	0.65
k-means	0.021	0.095	0.289
clique	0.019	0.094	0.290
net-clique	0.083	0.185	0.379

Table 6: ARI avereges controlling for p_c

	0.12	0.16	0.20
k-means	0.135	0.150	0.120
clique	0.146	0.147	0.110
net-clique	0.217	0.233	0.197

Table 7: ARI averages controlling for p_{in}