Integer programming formulations and efficient local search for relaxed correlation clustering

Eduardo Queiroga, Anand Subramanian, Rosa Figueiredo, Yuri Y. Frota

Integer programming formulations and efficient local search for relaxed correlation clustering

Eduardo Queiroga · Anand Subramanian · Rosa Figueiredo · Yuri Frota

Received: date / Accepted: date

Abstract Relaxed correlation clustering (RCC) is a vertex partitioning problem that aims at minimizing the so-called relaxed imbalance in signed graphs. RCC is considered to be an NP-hard unsupervised learning problem with applications in biology, economy, image recognition and social network analysis. In order to solve it, we propose two linear integer programming formulations and a local search-based metaheuristic. The latter relies on auxiliary data structures to efficiently perform move evaluations during the search process. Extensive computational experiments on existing and newly proposed benchmark instances demonstrate the superior performance of the proposed approaches when compared to those available in the literature. While the exact approaches obtained optimal solutions for open problems, the proposed heuristic algorithm was capable of finding high quality solutions within a reasonable CPU time. In addition, we also report improving results for the symmetrical version of the problem. Moreover, we show the benefits of implementing the efficient move evaluation procedure that enables the proposed metaheuristic to be scalable, even for large-size instances.

Keywords Relaxed correlation clustering · unsupervised learning · integer programming · iterated local search

E. Queiroga and Y. Frota
Instituto de Computação, Universidade Federal Fluminense, Rua Passo da Pátria, São Domingos, 24210-240, Niterói-RJ, Brazil.
E-mail: eduardoqueiroga@id.uff.br, eduardovqueiroga@gmail.com, yuri@ic.uff.br
Phone: +55 (21) 2629-5637

A. Subramanian
Departamento de Sistemas de Computação, Centro de Informática, Universidade Federal da Paraíba, Rua dos Escoteiros, Mangabeira, 58055-000, João Pessoa-PB, Brazil.
E-mail: anand@ci.ufpb.br

R. Figueiredo
Laboratoire Informatique d'Avignon, Avignon Université, 339 Chemin des Meinajaries, 84911, Avignon cedex 9, France.
E-mail: rosa.figueiredo@univ-avignon.fr

1 Introduction

In graph theory, signed graphs are those where each arc (or edge) has a positive or negative sign [50, 51]. This type of graph has been extensively used for modeling problems in various fields including biology [16], economy [29, 47], chemistry [38], ecology [15], image segmentation [31], linguistics [46], but mainly in social network analysis [19, 23, 2, 22, 3, 10]. One of these problems is *correlation clustering* (CC) [5], which is a well-known unsupervised learning problem that aims at finding a vertex partitioning in a signed graph so as to minimize the disagreements, given by negative arcs (or edges) within a cluster and positive arcs (or edges) between clusters. Before formally defining the CC problem on a directed signed graph, we shall introduce some notation.

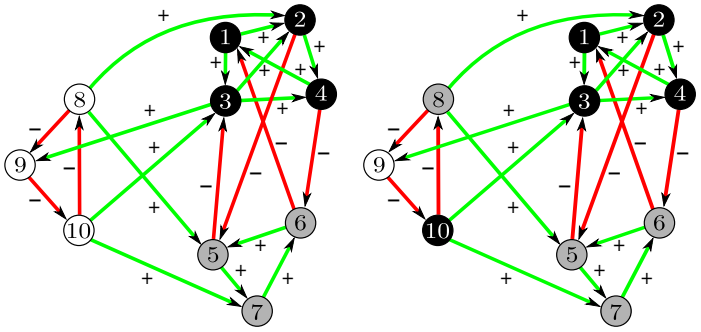
- Let $G = (V, A, s)$ be a signed digraph, where $V = \{1, 2, \ldots, n\}$ is the vertex set, $A \subseteq V \times V$ is the arc set, and $s : A \to \{+, -\}$ is a function that assigns a sign to each arc.
- An arc $a \in A$ is called *negative* if $s(a) = -$ and *positive* if $s(a) = +$.
- For each arc $a \in A$, let w_a be an associated non-negative weight. We will also use w_{ij} and w_{ji} to denote the weight of the arcs (i, j) and (j, i), respectively.
- The set of positive and negative arcs are denoted, respectively, as A^+ and A^-; thus $A = A^- \cup A^+$.
- A partition of V into l disjoint subsets $P = \{S_1, S_2, \ldots, S_l\}$ is called a l-partition of V.
- For $1 \leq p, q \leq l$, let $A[S_p : S_q] = \{(i, j) \in A \mid i \in S_p, j \in S_q\}$.
- $\Omega^+(S_p, S_q) = \sum_{a \in A^+ \cap A[S_p:S_q]} w_a$ and $\Omega^-(S_p, S_q) = \sum_{a \in A^- \cap A[S_p:S_q]} w_a$.

The *imbalance* $I(P)$ of a l-partition P is defined as

$$I(P) = \sum_{1 \leq p \leq l} \Omega^-(S_p, S_p) + \sum_{\substack{1 \leq p \leq l, \\ 1 \leq q \leq l, \\ p \neq q}} \Omega^+(S_p, S_q) \tag{1}$$

The CC problem consists of determining a partition P which minimizes $I(P)$. One of the applications of CC is related to the analysis of structural balance on social networks. In such networks, the arcs represent social relations between actors (i.e. the vertices of the network), whereas the sign represents feelings such as like/dislike and agreement/disagreement. According to the structural balance theory of Heider [30, 14], a network is *balanced* if there is a bipartition of the vertex set so that every positive arc joins actors in a same group and every negative arc joins actors in different groups. Later, Davis [17] generalized the structural balance to support a partition with more than two groups, which is fully compatible with the criterion optimized by the CC. Indeed, an algorithm for CC is a useful tool for evaluating how balanced a social network is. Note that a signed graph is balanced if there is a partition P such that $I(P) = 0$.

Although traditional structural balance works in several scenarios, Doreian and Mrvar [20] pointed out that this concept could not be appropriate for some networks. They argued that Equation (1) penalizes patterns in the partitions associated with relevant social psychological processes. For example, the network in Figure 1a represents a scenario with three groups, where one of them is a group of mutually hostile mediators (vertices 8, 9, and 10). Figure 1b illustrates why CC is



(a) Ideal partition (b) CC partition

Fig. 1: Structural balance in a network with mutually hostile mediators.

not suitable in this case: it is not capable of detecting the group of mediators or any type of subgroup internal hostility. This was illustrated in practice by Levorato et al. [36], where positive/negative mediation was detected in networks describing the United Nations General Assembly Voting Data. The equivalent was also verified for the case of differential popularity (a process in which some actors receive more positive links than others in a group) detected in benchmark instances from the literature [21], and for internal hostility detected in networks describing voting activity of members of the European Parliament [3].

Still in [20], Doreian and Mrvar introduced the concept of generalized structural balance giving rise to a new definition for the imbalance of a vertex partition which corrects the partition patterns penalized in Equation (1). The *relaxed imbalance* of a l-partition partition P, denoted here $RI(P)$, is defined as

$$RI(P) = \sum_{1 \leq p \leq l} \min\{\Omega^+(S_p, S_p), \Omega^-(S_p, S_p)\} + \sum_{\substack{1 \leq p \leq l, \\ 1 \leq q \leq l, \\ p \neq q}} \min\{\Omega^+(S_p, S_q), \Omega^-(S_p, S_q)\}. \quad (2)$$

The main subject of this work is the *relaxed correlation clustering* (RCC) problem [20, 26], which is a CC variant. In the RCC, given an integer parameter $1 \leq k \leq n$, one aims at finding a partition $P \in \cup_{l=1}^{k} \mathcal{P}^l$ that minimizes the *relaxed imbalance* given by Equation (2), such that $\mathcal{P}^l$ is the set of all l-partitions of V. The optimal value of the RCC problem determines how balanced a network is w.r.t. the relaxed structural balance introduced by Doreian and Mrvar[20]. For example, the partition in Figure 1a is an optimal solution for RCC because $RI(P) = 0$. Both CC and RCC problems were proven NP-hard by Bansal et al. [5] and Figueiredo and Moura [26], respectively.

Doreian and Mrvar [20] tackled the RCC by applying a relocation algorithm to analyze four real data sets related to relaxed structural balance, with up to 20 vertices. The authors adapted a heuristic method proposed in Doreian and Mrvar [19] for the CC problem with a fixed number of clusters, which optimizes a generic and parameterized function called *criterion function*. Later, Brusco et al. [9] proposed a branch-and-bound algorithm for solving RCC to optimality. This algorithm was capable of solving instances with up to 29 vertices (hereafter referred

to as small-sized instances) and k varying from 2 to 7. Moreover, an additional set of instances with up to 40 vertices was considered for experiments with $k = \{3, 5\}$. Figueiredo and Moura [26] developed an *integer linear programming* (ILP) formulation that was capable of solving some small-sized instances when $k = \{2, 3\}$ and for high values of k, concluding that the proposed ILP formulation and the existing branch-and-bound algorithm are somewhat complementary approaches. The authors also proposed a symmetrical version of RCC (SRCC). Although different heuristic procedures were proposed in the literature for CC (see [19, 49, 48, 6, 36], among others), to the best of our knowledge, only one heuristic procedure has been applied to the RCC. Levorato et al. [36] adapted their iterated local search (ILS) algorithm, originally developed for CC, to solve SRCC. Thus, there are no heuristic or metaheuristic procedures specifically proposed for RCC problem.

The two main contributions can be summarized as follows:

- We present two novel integer programming formulations for the RCC problem and we investigate their empirical performance in comparison to an existing formulation. The results show that the new formulations appear to produce better results in practice.
- We propose a local search-based metaheuristic that relies on a series of auxiliary data structures to efficiently recalculate the relaxed imbalance after applying an operation that modifies a partition, as well as on a novel perturbation mechanism. The results obtained suggest that the developed algorithm is superior to an existing approach, producing, on average, high quality solutions in a limited amount of CPU time, not only for RCC instances but also for SRCC instances. Finally, we also demonstrate the practical benefits of implementing the move evaluation in an efficient way.

The remainder of the paper is organized as follows. Section 2 presents a small instance for the RCC problem. Section 3 defines the symmetric version of the problem. Section 4 introduces two novel mathematical formulations for the RCC. Section 5 explains the proposed efficient local search-based metaheuristic including the efficient move evaluation schemes. Section 6 presents the results of the computational experiments. Finally, the conclusions are discussed in Section 7.

2 A small RCC example

A small RCC example involving 6 vertices is depicted in Figure 2. In Figure 2(a), the signed graph to be partitioned is illustrated. In Figure 2(b), a feasible solution $P = \{S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}\}$ is presented. This solution has relaxed imbalance $RI(P) = 4$ obtained by adding the following terms.

- $\min\{\Omega^+(S_1, S_1), \Omega^-(S_1, S_1)\} = \min\{\emptyset, w_{12}\} = \min\{0, 1\} = 0$
- $\min\{\Omega^+(S_2, S_2), \Omega^-(S_2, S_2)\} = \min\{w_{34}, \emptyset\} = \min\{1, 0\} = 0$
- $\min\{\Omega^+(S_3, S_3), \Omega^-(S_3, S_3)\} = \min\{w_{65}, w_{56}\} = \min\{1, 1\} = 1$
- $\min\{\Omega^+(S_1, S_2), \Omega^-(S_1, S_2)\} = \min\{w_{14}, w_{23}\} = \min\{1, 1\} = 1$
- $\min\{\Omega^+(S_2, S_1), \Omega^-(S_2, S_1)\} = \min\{\emptyset, \emptyset\} = \min\{0, 0\} = 0$
- $\min\{\Omega^+(S_1, S_3), \Omega^-(S_1, S_3)\} = \min\{\emptyset, \emptyset\} = \min\{0, 0\} = 0$
- $\min\{\Omega^+(S_3, S_1), \Omega^-(S_3, S_1)\} = \min\{w_{52} + w_{62}, w_{61}\} = \min\{2, 1\} = 1$
- $\min\{\Omega^+(S_2, S_3), \Omega^-(S_2, S_3)\} = \min\{\emptyset, \emptyset\} = \min\{0, 0\} = 0$

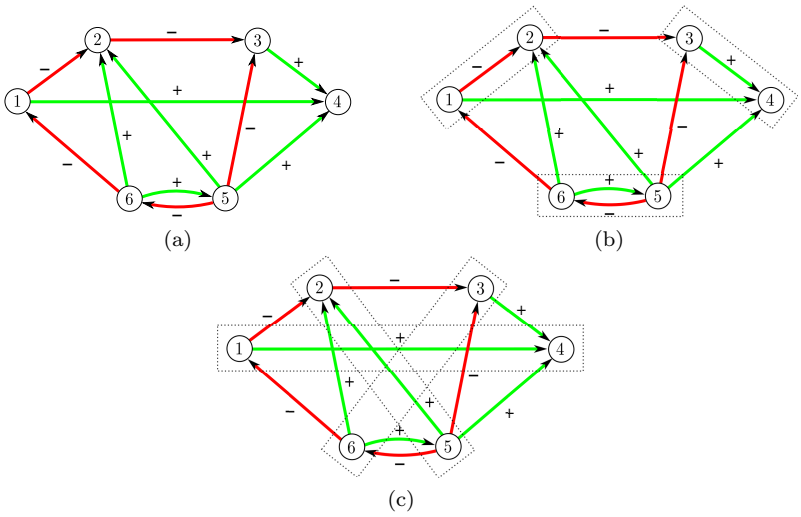


Fig. 2: (a) A small RCC instance with unitary weights and $k = 3$. (b) A feasible solution $P = \{\{1, 2\}, \{3, 4\}, \{5, 6\}\}$ with relaxed imbalance $RI(P) = 4$. (c) An optimal solution $P^* = \{\{1, 4\}, \{2, 5\}, \{3, 6\}\}$ with relaxed imbalance $RI(P) = 1$.

- $\min\{\Omega^+(S_3, S_2), \Omega^-(S_3, S_2)\} = \min\{w_{53}, w_{54}\} = \min\{1, 1\} = 1$

Figure 2(c) depicts an optimal solution $P^* = \{S_1 = \{1, 4\}, S_2 = \{2, 5\}, S_3 = \{3, 6\}\}$ for the problem with $RI(P^*) = \min\{\Omega^+(S_3, S_1), \Omega^-(S_3, S_1)\} = \min\{w_{34}, w_{61}\} = \min\{1, 1\} = 1$.

3 Symmetric RCC

In this work, we also consider the symmetric version of RCC (SRCC) introduced in Figueiredo and Moura [26]. The relaxed imbalance, as given by Equation (2), penalizes non-predominant relations (w.r.t. the signs) inside each cluster q and non-predominant relations *from* a cluster p *to* a cluster q. The difference in the symmetric relaxed imbalance defined in Figueiredo and Moura [26] is that it penalizes non-predominant relations *among* pairs of clusters, i.e., it considers simultaneously all positive (all negative) relations from S_p to S_q and from S_q to S_p. Thus, the SRCC can be defined on an undirected graph in which parallel edges with opposite signs are allowed.

Let $G' = (V, E, s')$ be an undirected signed graph with a positive weight w'_{ij} associated to each edge $\{i, j\} \in E$. Let us denote E^+ and E^-, respectively, the sets of positive and negative edges in E; thus $E = E^+ \cup E^-$. In this work, we transform the SRCC instance defined on $G' = (V, E, s')$ into a RCC instance defined on a directed signed graph $G_d = (V, A, s)$ in which: $A = \{(i, j), (j, i) : \{i, j\} \in E\}$; for each $(i, j) \in A$, $s((i, j)) = s((j, i)) = s'(\{i, j\})$ and the associated weight $w_{ij} = w_{ji} = \frac{w'_{ij}}{2}$. In other words, for each edge $\{i, j\} \in E$, one creates two arcs $(i, j), (j, i) \in A$ with the same signal and half of the weight.

Let $E[S_p : S_q]$ be the set of edges connecting the vertices in S_p and those in S_q. We denote $\Omega'^{+}(S_p, S_q) = \sum_{e \in E^+ \cap E[S_p:S_q]} w'_e$ and $\Omega'^{-}(S_p, S_q) = \sum_{e \in E^- \cap E[S_p:S_q]} w'_e$. The symmetric relaxed imbalance $SRI(P)$ of a l-partition P is defined as

$$SRI(P) = \sum_{1 \leq p \leq l} \min\{\Omega'^{+}(S_p, S_p), \Omega'^{-}(S_p, S_p)\} + \sum_{1 \leq p < q \leq l} \min\{\Omega'^{+}(S_p, S_q), \Omega'^{-}(S_p, S_q)\} \tag{3}$$

The following result relates Equations (2) and (3).

Proposition 1. *Consider an undirected signed graph G' and the directed signed graph G_d described above. Given any partitioning P, then $SRI(P) = RI(P)$.*

Proof. First, we will show the equivalence of the first terms in (2) and (3), i.e. the intracluster imbalance. Then we will do the same for the second terms, i.e. for the intercluster imbalance. Let S_p be any cluster in P. We have $\Omega'^{+}(S_p, S_p) = \Omega^{+}(S_p, S_p)$ since, for each $e = \{i, j\} \in E^+ \cap E[S_p : S_p]$, $(i, j), (j, i) \in A^+ \cap A[S_p : S_p]$ with $w'_{ij} = w_{ij} + w_{ji}$. The same can be argued for $\Omega'^{-}(S_p, S_p)$ and these two facts imply the equivalence of the first terms in (2) and (3). Now, let S_p and S_q be two clusters in P. By the definition of the directed graph G_d, we have $e = \{i, j\} \in E^+ \cap E[S_p : S_q]$ iff $(i, j) \in A^+ \cap A[S_p : S_q]$ and $(j, i) \in A^+ \cap A[S_q : S_p]$. Since, for each $e = \{i, j\} \in E$, $w'_{ij} = w_{ij} + w_{ji}$, we have that $\Omega^{+}(S_p, S_q) = \Omega^{+}(S_q, S_p) = \Omega'^{+}(S_p, S_q)/2$. The same holds for $\Omega^{-}(S_p, S_q)$ and, since the second term in (3) is written only for $p < q$, the equivalence of the second terms in (2) and (3) follows. □

As a consequence of Proposition 1, solving the RCC over graph G_d is equivalent to solving SRCC over G'.

4 Mathematical formulations

Integer linear programming (ILP) problem formulations have been used to solve CC and other related problems defined on signed graphs [13, 12, 4, 28, 27]. For RCC, an ILP formulation was presented in [26]. When modeling vertex-clustering problems, if there is a need for keeping track the clusters used, two types of formulations can be adopted: cluster-indexed formulation [24, 7, 11] or representatives formulation [1, 4, 13, 28]. Indeed, the ILP formulation of Figueiredo and Moura [26] is a representatives one. Next, we introduce two new formulations for RCC, one of each type.

4.1 Formulation `F1`: a cluster-indexed formulation

Let $K = \{1, \ldots, k\}$ be the set of possible cluster indexes. For each vertex $i \in V$ and $p \in K$ we define,

$$x_i^p = \begin{cases} 1, \text{ if vertex } i \text{ belongs to cluster } S_p, \\ 0, \text{ otherwise.} \end{cases}$$

A set of binary variables is used to describe the set of arcs that will be penalized once, according to Equation (2), the predominant relations are defined. For each arc $(i, j) \in A$, we define,

$$t_{ij} = \begin{cases} 1, \text{ if arc } (i,j) \text{ is penalized,} \\ 0, \text{ otherwise.} \end{cases}$$

A set of binary variables is used to select if the imbalance from cluster S_p to cluster S_q, with $p, q \in K$, is given by negative arcs (predominant relations are positive) or positive arcs (predominant relations are negative). Notice that intracluster imbalance is defined whenever $p = q$. For each pair of cluster indexes $p, q \in K$, we define,

$$s_{pq} = \begin{cases} 1, \text{ if positive arcs from cluster } S_p \text{ to cluster } S_q \text{ are penalized,} \\ 0, \text{ if negative arcs from cluster } S_p \text{ to cluster } S_q \text{ are penalized.} \end{cases}$$

The formulation can be written as

$$\text{minimize} \sum_{(i,j)\in A} w_{ij} t_{ij} \tag{4}$$

$$\text{s.t:} \sum_{p\in K} x_i^p = 1, \qquad \forall i \in V, \tag{5}$$

$$t_{ij} \geq x_i^p + x_j^q - 2 + s_{pq}, \qquad \forall (i,j) \in A^+, \forall p, q \in K, \tag{6}$$

$$t_{ij} \geq x_i^p + x_j^q - 2 + (1 - s_{pq}), \qquad \forall (i,j) \in A^-, \forall p, q \in K, \tag{7}$$

$$x_i^p \in \{0, 1\}, \qquad \forall i \in V, \forall p \in K, \tag{8}$$

$$t_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in A, \tag{9}$$

$$s_{pq} \in \{0, 1\}, \qquad \forall p, q \in K. \tag{10}$$

Objective function (4) minimizes the total relaxed imbalance. Constraints (5) ensure that each vertex is assigned exactly to one cluster. Constraints (6) and (7) define, respectively, if a positive or negative arc will be penalized due to assignment variables x_i^p, x_j^q and the penalizing variables s_{pq}. Note that, when $p = q$, variable t_{pq} defines an intracluster penalty. Finaly, constraints (8)–(10) define the domain of all variables.

Formulations that make use of cluster-indexed variables such as `F1` are considered to be symmetric, as there are a substantial number of ways to represent the same partitioning with a different permutation of indices. In view of this, we add the following set of symmetry breaking inequalities introduced in Bulhões et al. [11] for the p-cluster editing problem:

$$\sum_{l=1}^{p} x_i^l \geq \left(\sum_{\substack{j \in V \\ j<i}} \sum_{l=1}^{p-1} x_j^l \right) - (i-2), \ \forall i \in V, \ \forall p \in K. \tag{11}$$

The inequality above forbids the cluster containing i to use a label superior to p whenever each vertex $j < i$ is assigned to a cluster of index strictly smaller than p. The formulation in the next section adopts a similar strategy in order to break symmetry in the solution space.

4.2 Formulation `F2`: a representatives formulation

A representatives formulation for the RCC is presented as follows. The idea behind this kind of formulation [13] is the unique representation of a cluster by its vertex with the lowest label. Hence, for each pair of vertices $i, j \in V$ satisfying $i \leq j$, we define,

$$x_j^i = \begin{cases} 1, \text{ if the vertex } j \text{ is represented by vertex } i, \\ 0, \text{ otherwise.} \end{cases}$$

Note that when $i = j$, variable x_i^i indicates if i is a representative vertex.

Variables s_{ij}, with $i, j \in V$, used in `F2` are equivalent to variables s_{pq}, with $p, q \in K$, used in `F1`. However now, vertices i and j are used to identify two clusters ($i \neq j$) or one cluster ($i = j$). Variables t_{ij}, with $(i, j) \in A$, are exactly the same as defined in `F1`. Hence, formulation `F2` can be expressed as follows.

$$\text{minimize} \sum_{(i,j) \in A} w_{ij} t_{ij}$$

$$\text{s.t:} \sum_{i \in V: \ i \leq j} x_j^i = 1, \qquad \forall j \in V, \tag{12}$$

$$x_j^i \leq x_i^i, \qquad \forall i, j \in V, i < j, \tag{13}$$

$$\sum_{i \in V} x_i^i \leq k, \tag{14}$$

$$t_{ij} \geq x_i^u + x_j^v - 2 + s_{uv}, \qquad \forall (i,j) \in A^+, \forall u, v \in V, \ u \leq i, v \leq j, \tag{15}$$

$$t_{ij} \geq x_i^u + x_j^v - 2 + (1 - s_{uv}), \qquad \forall (i,j) \in A^-, \forall u, v \in V, \ u \leq i, v \leq j, \tag{16}$$

$$x_j^i \in \{0, 1\}, \qquad \forall i, j \in V, i \leq j, \tag{17}$$

$$t_{ij} \in \{0, 1\}, \qquad \forall (i,j) \in A, \tag{18}$$

$$s_{ij} \in \{0, 1\}, \qquad \forall i, j \in V. \tag{19}$$

Constraints (12) impose that each vertex must be represented by exactly one vertex: either by itself or by another one with a smaller index. Constraints (13) enforce vertex i to be a representative one whenever a vertex j is represented by i. Constraint (14) imposes k as an upper bound on the number of representative vertices, i.e., on the number of clusters in the partition. Constraints (15) and (16) are, respectively, equivalent to constraints (6) and (7) of formulation `F1`. Finally, constraints (17)–(19) are the binary constraints.

The number of variables and constraints of formulations `F1` and `F2` are illustrated in Table 1. Note that since the number of vertices of a graph is usually much greater than the number of clusters, formulation `F1` is more compact than `F2`. On the other hand, formulation `F2` succeeds in eliminating cluster indices from the representation which breaks symmetry from formulation [13].

Table 1: Number of variables and constraints of formulations `F1` and `F2`

	#Variables	#Constraints
`F1`	$\mathcal{O}(nk + \|A\| + k^2)$	$\mathcal{O}(n + k^2\|A^+\| + k^2\|A^-\|)$
`F2`	$\mathcal{O}(2n^2 + \|A\|)$	$\mathcal{O}(n + n^2 + n^2\|A^+\| + n^2\|A^-\|)$

5 Proposed local search-based metaheuristic

In this section, we propose a metaheuristic algorithm for the RCC based on the iterated local search (ILS) method [37]. The key concept of ILS is to combine local search strategies and perturbation mechanisms to escape from local optima. The metaheuristic introduced in Levorato et al. [36] for CC and adapted for the solution of SRCC instances is also an ILS method. Different from Levorato et al. [36], in this work, we propose the use of several complex neighborhoods and the use of advanced data structures which make the search process more efficient.

Algorithm 1 presents a general framework of a multi-start ILS, hereafter referred to as ILS_{RCC}, which has the following input parameters:

a) I_R is the number of restarts of the metaheuristic;
b) I_{ILS} is the maximum number of ILS iterations without improvements;
c) I_P is the maximum number of moves performed by a perturbation mechanism.

For each restart, an initial solution is randomly generated (line 5) and such solution is possibly improved by alternately applying local search (line 9) and perturbation (line 13) strategies until the maximum number of iterations (I_{ILS}) without improvement is achieved. Finally, the best solution found among all restarts is returned (line 18).

The local search procedure is based on variable neighborhood descent (VND) [40], which is a technique that systematically explores a sequence of neighborhood structures (see Section 5.2), searching for better solutions. A neighborhood structure (or simply neighborhood) defines a set of neighbor solutions from a current solution by applying a so-called move. When VND finds an improving move using a particular neighborhood, the solution is updated and the procedure restarts

Algorithm 1: ILS$_{\text{RCC}}$

```
1  Procedure ILS_RCC(I_R, I_ILS, I_P)
2      RI* = ∞
3      P* = ∅
4      for iter = 1 ... I_R do
5          P = ConstructiveProcedure()
6          P' = P
7          iter_ILS = 0
8          while iter_ILS < I_ILS do
9              P = LocalSearch(P)
10             if RI(P) < RI(P') then
11                 P' = P
12                 iter_ILS = 0
13             P = Perturb(P', I_P)
14             iter_ILS = iter_ILS + 1
15         if RI(P') < RI* then
16             P* = P'
17             RI* = RI(P')
18     return P*
```

from the improved solution. The procedure terminates when all neighborhoods fail to improve the current solution. The *best improvement* strategy was adopted, i.e., a neighborhood is fully enumerated and the best improving move (if there is any) is applied. In addition, the neighborhood ordering is defined in a random fashion, which results in a strategy known as Randomized VND (RVND). The combination of ILS and RVND led to state-of-the-art methods for several important combinatorial optimization problems, such as: split-delivery vehicle routing problem [44], minimum latency problem [43] and minimizing weighted tardiness in single machine scheduling with sequence-dependent setup times [45].

The perturbation procedure randomly chooses one of the implemented mechanisms (see Section 5.3) in order to modify the local optimal solution P'. The selected mechanism then applies I_P random consecutive moves over P' in order to generate a solution to continue the search.

In what follows, we provide a detailed description of the auxiliary data structures used for performing efficient move evaluation, as well as on the neighborhood structures and perturbations mechanisms.

5.1 Auxiliary data structures

Assuming that an adjacency matrix is used to access the signed digraph G, and a feasible solution is represented by a set of subsets of indices (*e.g.* $P = \{S_1, S_2, S_3\}$, such that $S_1 = \{1, 2\}, S_2 = \{3, 4\}, S_3 = \{5, 6\}$ for a graph with 6 vertices; see Figure 2), the value of its associated objective function can be straightforwardly computed in $\mathcal{O}(l^2n^2)$ operations, where $l = |P|$. Note that this is due to the complexity of determining the intercluster imbalance. Consequently, performing the move evaluation of a neighbor solution from scratch every time during the local search may turn out to be computationally expensive, especially for large size

instances. However, this can be done in a more efficient manner by precomputing and storing information in auxiliary data structures (ADSs).

We thus propose to implement two classes of ADSs: $\texttt{SumIntra}[S_p]$, which is a set of ADSs that stores the sum of the weights for different subsets of $A[S_p]$ (a.k.a. intracluster arcs of S_p); and $\texttt{SumInter}[S_p][S_q]$, which is a set of ADSs that stores the sum of the weights for different subsets of $A[S_p : S_q]$ (a.k.a. known as intercluster arcs from S_p to S_q). The ADSs are divided according to the sign and arc direction as described in Table 2.

Table 2: Description of the proposed ADSs

ADS	Description
$\texttt{SumIntra}^+[S_p] = \sum_{a \in A^+ \cap A[S_p]} w_a$	Sum of positive weights within S_p
$\texttt{SumIntra}^-[S_p] = \sum_{a \in A^- \cap A[S_p]} w_a$	Sum of negative weights within S_p
$\texttt{SumIntra}^+[S_p][i][\leftarrow] = \sum_{ji \in A^+, j \in S_p \setminus i} w_{ji}$	Sum of positive weights from $S_p \setminus i$ to $i \in S_p$
$\texttt{SumIntra}^+[S_p][i][\rightarrow] = \sum_{ij \in A^+, j \in S_p \setminus i} w_{ij}$	Sum of positive weights from $i \in S_p$ to $S_p \setminus i$
$\texttt{SumIntra}^-[S_p][i][\leftarrow] = \sum_{ji \in A^-, j \in S_p \setminus i} w_{ji}$	Sum of negative weights from $S_p \setminus i$ to $i \in S_p$
$\texttt{SumIntra}^-[S_p][i][\rightarrow] = \sum_{ij \in A^-, j \in S_p \setminus i} w_{ij}$	Sum of negative weights from $i \in S_p$ to $S_p \setminus i$
$\texttt{SumInter}^+[S_p][S_q] = \sum_{a \in A^+ \cap A[S_p:S_q]} w_a$	Sum of positive weights from S_p to S_q
$\texttt{SumInter}^-[S_p][S_q] = \sum_{a \in A^- \cap A[S_p:S_q]} w_a$	Sum of negative weights from S_p to S_q
$\texttt{SumInter}^+[S_p][i][S_q][\rightarrow] = \sum_{ij \in A^+, j \in S_q} w_{ij}$	Sum of positive weights from $i \in S_p$ to S_q
$\texttt{SumInter}^+[S_p][i][S_q][\leftarrow] = \sum_{ji \in A^+, j \in S_q} w_{ji}$	Sum of positive weights from S_q to $i \in S_p$
$\texttt{SumInter}^-[S_p][i][S_q][\rightarrow] = \sum_{ij \in A^-, j \in S_q} w_{ij}$	Sum of negative weights from $i \in S_p$ to S_q
$\texttt{SumInter}^-[S_p][i][S_q][\leftarrow] = \sum_{ji \in A^-, j \in S_q} w_{ji}$	Sum of negative weights from S_q to $i \in S_p$

Given a feasible solution, the `SumIntra` and `SumInter` ADSs can be initially built in $\mathcal{O}(l^2n^2)$ operations as described in Algorithm 2.

5.2 Neighborhood structures

ILS_{RCC} uses three neighborhood structures in the local search, namely: *Insertion*, *Swap* and *Split*. In the following, each of them is described in detail.

5.2.1 Insertion

The *Insertion* neighborhood moves a vertex from a cluster to another one, thus yielding $\mathcal{O}(l^2n)$ possible neighbor solutions to be evaluated.

Algorithm 3 describes how an *Insertion* move is evaluated using the ADSs. This algorithm receives as input the solution P along with its associated cost (relaxed imbalance) RI_P, and the information regarding the move, i.e. S_p, $i \in S_p$ and S_q. At first, auxiliary variables $sum^+_{S_p}$, $sum^-_{S_p}$, $sum^+_{S_p,S_q}$ and $sum^-_{S_p,S_q}$ temporarily store, in $\mathcal{O}(1)$ steps, the sum of the weights associated to the move (lines 2–13). Next, the value of the objective function of the neighbor solution under evaluation, denoted in the algorithm as *cost*, is partially obtained (lines 14–17) by recomputing the penalty decisions using function `UpdateCost` (see lines 31–36). In the loop from lines 18 to 30, a similar procedure is performed for the intercluster cases involving the other clusters and the clusters S_p and S_q. Finally, *cost* is returned and the move yields an improvement if $cost < RI_P$.

Algorithm 2: Computing the auxiliary data structures

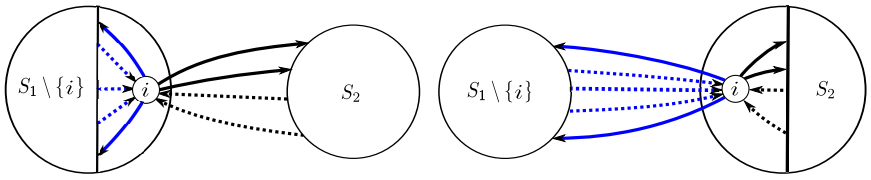
Algorithm `ComputeADSs`(P)
  All ADSs are initialized with 0.0

  ▷ Computing the SumIntra ADSs
  for $\forall p \in \{1, 2, \ldots, l\}$ **do**
    for $\forall i, j \in S_p,\ i < j$ **do**
      if $(i, j) \in A^+$ **then**
        $\texttt{SumIntra}^+[S_p] = \texttt{SumIntra}^+[S_p] + w_{ij}$
        $\texttt{SumIntra}^+[S_p][i][\rightarrow] = \texttt{SumIntra}^+[S_p][i][\rightarrow] + w_{ij}$
        $\texttt{SumIntra}^+[S_p][j][\leftarrow] = \texttt{SumIntra}^+[S_p][j][\leftarrow] + w_{ij}$
      else if $(i, j) \in A^-$ **then**
        $\texttt{SumIntra}^-[S_p] = \texttt{SumIntra}^-[S_p] + w_{ij}$
        $\texttt{SumIntra}^-[S_p][i][\rightarrow] = \texttt{SumIntra}^-[S_p][i][\rightarrow] + w_{ij}$
        $\texttt{SumIntra}^-[S_p][j][\leftarrow] = \texttt{SumIntra}^-[S_p][j][\leftarrow] + w_{ij}$
      if $(j, i) \in A^+$ **then**
        $\texttt{SumIntra}^+[S_p] = \texttt{SumIntra}^+[S_p] + w_{ji}$
        $\texttt{SumIntra}^+[S_p][j][\rightarrow] = \texttt{SumIntra}^+[S_p][j][\rightarrow] + w_{ji}$
        $\texttt{SumIntra}^+[S_p][i][\leftarrow] = \texttt{SumIntra}^+[S_p][i][\leftarrow] + w_{ji}$
      else if $(j, i) \in A^-$ **then**
        $\texttt{SumIntra}^-[S_p] = \texttt{SumIntra}^-[S_p] + w_{ji}$
        $\texttt{SumIntra}^-[S_p][j][\rightarrow] = \texttt{SumIntra}^-[S_p][j][\rightarrow] + w_{ji}$
        $\texttt{SumIntra}^-[S_p][i][\leftarrow] = \texttt{SumIntra}^-[S_p][i][\leftarrow] + w_{ji}$

  ▷ Computing the SumInter ADSs
  for $\forall p, q \in \{1, 2, \ldots, l\}, p \neq q$ **do**
    for $\forall i \in S_p, \forall j \in S_q$ **do**
      if $(i, j) \in A^+$ **then**
        $\texttt{SumInter}^+[S_p][S_q] = \texttt{SumInter}^+[S_p][S_q] + w_{ij}$
        $\texttt{SumInter}^+[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^+[S_p][i][S_q][\rightarrow] + w_{ij}$
        $\texttt{SumInter}^+[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^+[S_q][j][S_p][\leftarrow] + w_{ij}$
      else if $(i, j) \in A^-$ **then**
        $\texttt{SumInter}^-[S_p][S_q] = \texttt{SumInter}^-[S_p][S_q] + w_{ij}$
        $\texttt{SumInter}^-[S_p][i][S_q][\rightarrow] = \texttt{SumInter}^-[S_p][i][S_q][\rightarrow] + w_{ij}$
        $\texttt{SumInter}^-[S_q][j][S_p][\leftarrow] = \texttt{SumInter}^-[S_q][j][S_p][\leftarrow] + w_{ij}$
      if $(j, i) \in A^+$ **then**
        $\texttt{SumInter}^+[S_q][S_p] = \texttt{SumInter}^+[S_q][S_p] + w_{ji}$
        $\texttt{SumInter}^+[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^+[S_q][j][S_p][\rightarrow] + w_{ji}$
        $\texttt{SumInter}^+[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] + w_{ji}$
      else if $(j, i) \in A^-$ **then**
        $\texttt{SumInter}^-[S_q][S_p] = \texttt{SumInter}^-[S_q][S_p] + w_{ji}$
        $\texttt{SumInter}^-[S_q][j][S_p][\rightarrow] = \texttt{SumInter}^-[S_q][j][S_p][\rightarrow] + w_{ji}$
        $\texttt{SumInter}^-[S_p][i][S_q][\leftarrow] = \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] + w_{ji}$

Because Algorithm 3 performs $\mathcal{O}(l)$ steps (due to the loop), finding the best improving move requires $\mathcal{O}(l^3 n)$ operations. Moreover, when P is modified, the ADSs must be updated. However, instead of recomputing the ADSs from scratch in $\mathcal{O}(l^2 n^2)$ operations, one only needs to update the ADSs affected by the vertex that was involved in the move and this can be performed in $\mathcal{O}(n)$ steps, as detailed in the electronic supplementary material of this work.

Figure 3 illustrates an example of an *Insertion* move. Note that the separation of the weights for the adjacent arcs of vertex i in `SumIntra` clearly facilitates the evaluation of the intercluster sums from S_1 to S_2 and from S_2 to S_1. Otherwise, it would be necessary to perform $\mathcal{O}(n)$ operations to compute the weights separately.



(a) Clusters before *Insertion* (b) Clusters after *Insertion*

Fig. 3: Example of an *Insertion* move. The incoming arcs of i are in dashed lines to illustrate the separation of the weights in `SumIntra` and `SumInter`. For the sake of simplicity, the signs were omitted and all arcs have unitary weight.

5.2.2 Swap

The *Swap* neighborhood exchanges two vertices between two different clusters, which leads to $\mathcal{O}(l^2 n^2)$ neighbor solutions if one intends to enumerate all possibilities. The pseudocodes presented in the electronic supplementary material describe how a *Swap* move can be evaluated in $\mathcal{O}(l)$ steps using a similar rationale employed in the *Insertion* neighborhood. Finding the best improving *Swap* move thus require $\mathcal{O}(l^3 n^2)$ operations and the ADSs can be updated in $\mathcal{O}(n)$ steps as also described in the supplementary material.

Figure 4 depicts an example of a swap move, highlighting the arcs connecting exchanged vertices, as they must be treated separately with respect to some ADSs.

5.2.3 Split

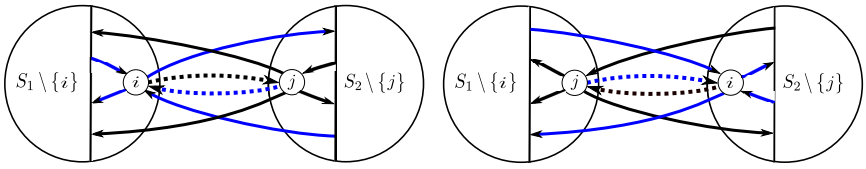
The *Split* neighborhood splits a cluster into two, resulting in a total of $\mathcal{O}(ln)$ neighbor solutions to be examined. Formally, given a cluster $S = \{v_1, v_2, \ldots, v_{|S|}\} \in P$ and an index $c < |S|$, the clusters $S' = \{v_1, v_2, \ldots, v_c\}$ and $S'' = \{v_{c+1}, \ldots, v_{|S|}\}$ are produced to replace S in P. Clearly, a *Split* move can only be applied when $l < k$. The Pseudocodes presented in the electronic supplementary material describes how a *Split* move can use previous evaluations to speedup the next ones. The overall complexity of determining the best improvement is $\mathcal{O}(ln^2)$, as also

Algorithm 3: Using the ADSs to evaluate an insertion move

Algorithm `CompCostInsert`(P, RI_P, S_p, i, S_q)
- ▷ update the sum of the intracluster weights of S_p
- $sum^+_{S_p} = \texttt{SumIntra}^+[S_p] - \texttt{SumIntra}^+[S_p][i][\leftarrow] - \texttt{SumIntra}^+[S_p][i][\rightarrow]$
- $sum^-_{S_p} = \texttt{SumIntra}^-[S_p] - \texttt{SumIntra}^-[S_p][i][\leftarrow] - \texttt{SumIntra}^-[S_p][i][\rightarrow]$
- ▷ update the sum of the intracluster weights of S_q
- $sum^+_{S_q} = \texttt{SumIntra}^+[S_q] + \texttt{SumInter}^+[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$
- $sum^-_{S_q} = \texttt{SumIntra}^-[S_q] + \texttt{SumInter}^-[S_p][i][S_q][\leftarrow] + \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$
- ▷ update the sum of the intercluster weights from S_p to S_q
- $sum^+_{S_p,S_q} = \texttt{SumInter}^+[S_p][S_q] - \texttt{SumInter}^+[S_p][i][S_q][\rightarrow]$
- $sum^-_{S_p,S_q} = \texttt{SumInter}^-[S_p][S_q] - \texttt{SumInter}^-[S_p][i][S_q][\rightarrow]$
- $sum^+_{S_p,S_q} = sum^+_{S_p,S_q} + \texttt{SumIntra}^+[S_p][i][\leftarrow]$
- $sum^-_{S_p,S_q} = sum^-_{S_p,S_q} + \texttt{SumIntra}^-[S_p][i][\leftarrow]$
- ▷ update the sum of the intercluster weights from S_q to S_p
- $sum^+_{S_q,S_p} = \texttt{SumInter}^+[S_q][S_p] - \texttt{SumInter}^+[S_p][i][S_q][\leftarrow]$
- $sum^-_{S_q,S_p} = \texttt{SumInter}^-[S_q][S_p] - \texttt{SumInter}^-[S_p][i][S_q][\leftarrow]$
- $sum^+_{S_q,S_p} = sum^+_{S_q,S_p} + \texttt{SumIntra}^+[S_p][i][\rightarrow]$
- $sum^-_{S_q,S_p} = sum^-_{S_q,S_p} + \texttt{SumIntra}^-[S_p][i][\rightarrow]$
- ▷ Recompute the penalty decisions and updates $RI_{P'}$
- $cost = \texttt{UpdateCost}(RI_P, S_p, S_p, sum^+_{S_p}, sum^-_{S_p})$
- $cost = \texttt{UpdateCost}(cost, S_q, S_q, sum^+_{S_q}, sum^-_{S_q})$
- $cost = \texttt{UpdateCost}(cost, S_p, S_q, sum^+_{S_p,S_q}, sum^-_{S_p,S_q})$
- $cost = \texttt{UpdateCost}(cost, S_q, S_p, sum^+_{S_q,S_p}, sum^-_{S_q,S_p})$
- ▷ update the sum of the intercluster weights involving others clusters
- **for** $S_r \in P \setminus \{S_p, S_q\}$ **do**
  - $sum^+_{S_r,S_p} = \texttt{SumInter}^+[S_r][S_p] - \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$
  - $sum^-_{S_r,S_p} = \texttt{SumInter}^-[S_r][S_p] - \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$
  - $sum^+_{S_r,S_q} = \texttt{SumInter}^+[S_r][S_q] + \texttt{SumInter}^+[S_p][i][S_r][\leftarrow]$
  - $sum^-_{S_r,S_q} = \texttt{SumInter}^-[S_r][S_q] + \texttt{SumInter}^-[S_p][i][S_r][\leftarrow]$
  - $sum^+_{S_p,S_r} = \texttt{SumInter}^+[S_p][S_r] - \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$
  - $sum^-_{S_p,S_r} = \texttt{SumInter}^-[S_p][S_r] - \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$
  - $sum^+_{S_q,S_r} = \texttt{SumInter}^+[S_q][S_r] + \texttt{SumInter}^+[S_p][i][S_r][\rightarrow]$
  - $sum^-_{S_q,S_r} = \texttt{SumInter}^-[S_q][S_r] + \texttt{SumInter}^-[S_p][i][S_r][\rightarrow]$
  - $cost = \texttt{UpdateCost}(cost, S_r, S_p, sum^+_{S_r,S_p}, sum^-_{S_r,S_p})$
  - $cost = \texttt{UpdateCost}(cost, S_r, S_q, sum^+_{S_r,S_q}, sum^-_{S_r,S_q})$
  - $cost = \texttt{UpdateCost}(cost, S_p, S_r, sum^+_{S_p,S_r}, sum^-_{S_p,S_r})$
  - $cost = \texttt{UpdateCost}(cost, S_q, S_r, sum^+_{S_q,S_r}, sum^-_{S_q,S_r})$
- **return** $cost$

Procedure `UpdateCost`$(cost, S_p, S_q, sum^+, sum^-)$
- **if** $S_p = S_q$ **then**
  - **return** $cost - (\min\{\texttt{SumIntra}^+[S_p], \texttt{SumIntra}^-[S_p]\} - \min\{sum^+, sum^-\})$
- **else**
  - **return** $cost - (\min\{\texttt{SumInter}^+[S_p][S_q], \texttt{SumInter}^-[S_p][S_q]\} - \min\{sum^+, sum^-\})$

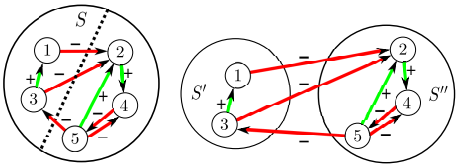
described in the supplementary material. Because of the considerable changes produced by the split move, all ADSs must be updated from scratch using Algorithm 2. It is worth mentioning that implementing a specific procedure to update the ADSs did not pay off the gains in CPU time. Moreover, note that a split move



(a) Clusters before *Swap* (b) Clusters after *Swap*

Fig. 4: Example of a *Swap* move. Arcs (i, j) and (j, i) are in dashed lines to illustrate their importance w.r.t. the ADSs. For the sake of simplicity, the signs were omitted and all arcs have unitary weight.

never worsens a solution, since the imbalance decreases monotonically as k increases [20]. Figure 5 shows an example of a split move considering a cluster with 5 vertices that is split into two with 2 and 3 vertices, respectively.



(a) Cluster before *Split* (b) Cluster after *Split*

Fig. 5: Example of a *Split* move. For the sake of simplicity, all arcs have unitary weight.

5.2.4 Complexity summary

A summary on the complexity of the neighborhoods is provided in Table 3. For each neighborhood, we present its size, as well as the complexity of performing the move evaluation and the overall one using both the efficient best improvement (EBI) and the naive best improvement (NBI) strategies. In EBI, the search for the best improvement move is carried out as described in Section 5.2, whereas in NBI the objective function must be computed from scratch (with no support of ADSs) after each move. We also report the complexity of updating the ADSs in the case of EBI.

Table 3: Complexity summary of the neighborhoods considering both EBI and NBI strategies

Neighborhood	Size	EBI			NBI	
		Move eval.	Overall	Update	Move eval.	Overall
Insertion	$\mathcal{O}(l^2n)$	$\mathcal{O}(l)$	$\mathcal{O}(l^3n)$	$\mathcal{O}(n)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^4n^3)$
Swap	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l)$	$\mathcal{O}(l^3n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^4n^4)$
Split	$\mathcal{O}(ln)$	$\mathcal{O}(n)$	$\mathcal{O}(ln^2)$	$\mathcal{O}(ln^2)$	$\mathcal{O}(l^2n^2)$	$\mathcal{O}(l^3n^3)$

5.3 Perturbation mechanisms

ILS_{RCC} employs three diversification mechanisms to perturb local optimal solutions, namely: *Insertion*, *Merge* and *Sign inversion*. The first one simply performs random insertion moves. In the second, given two clusters S_1 and S_2 chosen at random, one merges them to form a new cluster S_3, that is, $S_3 = S_1 \cup S_2$. The latter perturbation is a novel procedure that considers some RCC specific features, as described in the following.

The proposed *Sign inversion* mechanism enforces the penalized sign in one of the decisions to be changed. More precisely, it modifies the solution in such a way that one of the intracluster or intercluster imbalances becomes defined by the opposite sign. The procedure randomly selects which case (i.e., intracluster or intercluster) is going to be considered. Basically, this is achieved by removing vertices that contribute with the non-penalized sign until the inversion happens. In what follows, we will explain the procedure used to invert an intracluster decision.

Let $+$ be the non-penalized sign for the intracluster imbalance of S_p (this also applies for sign $-$). Formally, the contribution of a vertex $i \in S_p$ is given by Equation (20).

$$\Delta^+(i) = \Omega^+(\{i\}, S_p) + \Omega^+(S_p, \{i\}) - \Omega^-(\{i\}, S_p) - \Omega^-(S_p, \{i\}) \qquad (20)$$

At first, the vertices for which all incident arcs (indegree and outdegree arcs) are positive are removed in non-increasing order of Δ^+. The value of Δ^+ must be updated after each removal. If this does not suffice, the remaining vertices with $\Delta^+ > 0$ are removed using the same sorting criterion. Removals are performed while (i) $\Omega^+(S_p, S_p) \geq \Omega^-(S_p, S_p)$, (ii) there are vertices with $\Delta^+ > 0$ and (iii) $|S_p| > 2$. The removed vertices are randomly added to the other clusters. After applying the perturbation, if $\Omega^+(S_p, S_p) \geq \Omega^-(S_p, S_p)$ (i.e. the sign was not inverted), then the removals are undone and the solution returns to the initial state. Figure 6 illustrates an example involving the application of the *sign inversion* mechanism.

The intercluster sign invertion, e.g., from S_p to S_q, may be easily derived by considering only the arcs $A[S_p : S_q]$ that determine the vertices to be removed from S_p and by changing the condition (iii) to $|S_p| > 1$. Note that no vertices are removed from S_q but it may receive vertices from S_p.

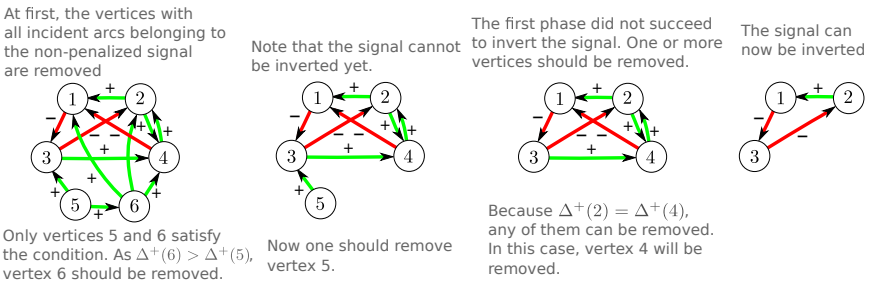


Fig. 6: An example of sign inversion for an intracluster imbalance. For the sake of simplicity, all arcs have unitary weight.

This perturbation allows for exploring some particular regions of the search space that is difficult to be achieved by only using the other mechanisms, including the randomized construction procedure, mainly when sign distribution on arcs is unbalanced and small changes are not likely to invert the sign.

Each time the function `Perturb` is called, a perturbation mechanism is randomly chosen. The selected perturbation then applies from two up to *maxPert* moves. The number of moves is also chosen at random. If $l = 2$, *Merge* is not an eligible perturbation. Therefore, when *Sign inversion* is chosen and no change could be performed, one of the other two remaining mechanisms (or *Insertion* if $l = 2$) is randomly selected.

5.4 Differences between ILS_{RCC} and ILS [36]

Table 4 presents the main differences between ILS_{RCC} and the ILS by Levorato et al. [36] which was developed for the SRCC.

Table 4: Differences between ILS_{RCC} and ILS [36]

	ILS_{RCC}	ILS [36]
Initial solution	Random	Greedy randomized procedure
Local search	*Insertion*, *Swap* and *Split*	*Insertion*
	Best improvement strategy	First improvement strategy
Perturbation	*Insertion*, *Merge* and *Sign Inversion*	*Insertion*

It is worth mentioning that we tried to incorporate the constructive procedure implemented in Levorato et al. [36] into our algorithm, but the experiments reported in Section 6.3.1 indicated that its inclusion did not seem to significantly affect the overall performance of ILS_{RCC} both in terms of solution quality and CPU time.

6 Computational results

All algorithms have been implemented in C++ and executed using a single thread on a PC Intel Core i7-2600 with 3.40 GHz and 16 GB of RAM running Ubuntu 16.04 LTS (64 bits). For results based on ILP formulations, CPLEX 12.7 is used as a MIP solver (single thread) with all other parameters set to their default values.

6.1 Benchmark instances

Regarding the benchmark instances used in our testing, we first present the small-size instances from the literature. Next, we introduce the newly proposed RCC instances and, finally, we describe the existing SRCC instances.

6.1.1 Small-size instances from the literature

The small-size instances considered here were proposed in different works and together they compose a set of nine signed digraphs described as follows.

- *House instances* — In 1952, Lemann and Solomon [32] carried out a sociometric study with students living in three different dormitories (denoted as House A, House B and House C) and obtained four relationship networks per dormitory considering the following information: date, friend, roommate and weekend. Doreian [18] later summed the arc weights of the signed networks associated with each dormitory so as to generate another three networks: House A Sum, House B Sum and House C Sum. We have considered these last three in our experiments.
- *Monastery instances* — In 1868, Sampson [42] studied, in different periods of time, groups of young or novice postulants of a monastery, cataloging data for four types of relationships: affect, esteem, influence, and sanction. From this data, networks were generated for each period of time and type of relationship. Among them, we considered those associated with the relationship affect for different periods of time, namely MonkT2, MonkT3, and MonkT4. In addition, we considered the network Mont4 Sum, generated in Doreian [18] by summing up the arc weights of the four types of relationships in period T4.
- *McKinney instance* — This signed digraph was built by Brusco et al. [9] from the data collected by McKinney [39] in a study about the relationship between children in a classroom. In such study, children were submitted to a test in which they had to choose between the "willing to serve with other children" (labeled as +1), "not being willing to serve" (labeled as -1) and "indifferent" (labeled as 0), defining the class relationship digraph.
- *NewComb instance* — In 1961, Newcomb [41] conducted a well-known sociometric study with University students. A signed digraph was generated in Doreian and Mrvar [20] by slightly modifying the data from this study.

The main characteristics of the aforementioned instances are described in Table 5, where d and d^- indicate the digraph density (given by $d = |A|/(|V|^2 - |V|)$) and the percentage of negative arcs (given by $d^- = |A^-|/|A|$), respectively. For the sake of convenience, we have specified an alias (in parentheses) for each instance.

Table 5: Small-size instance attributes

Name	$\|V\|$	d	d^-	Author(s)
House A Sum (HAS)	21	0.50	0.56	Lemann and Solomon [32], Doreian [18]
House B Sum (HBS)	17	0.59	0.52	Lemann and Solomon [32], Doreian [18]
House C Sum (HCS)	20	0.52	0.53	Lemann and Solomon [32], Doreian [18]
MonkT2 (MT2)	18	0.34	0.47	Sampson [42]
MonkT3 (MT3)	18	0.34	0.46	Sampson [42]
MonkT4 (MT4)	18	0.34	0.46	Sampson [42]
MonkT4 Sum (MT4S)	18	0.50	0.49	Sampson [42], Doreian [18]
McKinney (MK)	18	0.34	0.10	McKinney [39], Brusco et al. [9]
NewComb (NC)	17	0.44	0.43	Newcomb [41], Doreian and Mrvar [20]

6.1.2 Random instances

In order to test the ILS implementations on larger instances, we have generated 48 new signed digraphs with different values of $|V|$, d and d^-. Let $\mathcal{V}_{|V|} = \{100, 200, 400, 600\}$, $\mathcal{V}_d = \{0.1, 0.2, 0.5, 0.8\}$ and $\mathcal{V}_{d^-} = \{0.1, 0.3, 0.5\}$ be the set of values associated with $|V|$, d and d^-, respectively. For each setting obtained by the Cartesian product $\mathcal{V}_{|V|} \times \mathcal{V}_d \times \mathcal{V}_{d^-}$ (represented by a 3-tuple), we have randomly built a signed digraph. Note that larger values of d^- are not used because they are equivalent with respect to the desired sign distribution (e.g., if $d^- = 0.7$, then the percentage of positive arcs will be 0.3). A RCC instance consists of a digraph and a value for the parameter k (maximum number of clusters). For each generated digraph, we consider one instance for each value of k in $\{3, 5, 7, 9\}$. Therefore, this benchmark is composed of 192 instances.

The newly generated digraphs are available at `http://www.ic.uff.br/~yuri/files/rcc_random.zip`.

6.1.3 Symmetric RCC instances

We also considered three sets of symmetric RCC benchmark instances, namely:

- *UNGA instances* — Generated by Levorato et al. [35] and composed of 63 undirected graphs that were built from the voting data of the United Nations General Assembly (UNGA) annual meetings between 1946 and 2008. These networks are weighted versions of UNGA signed digraphs created by Figueiredo and Frota [25].
- *Slashdot instances* — Created by Levorato [33] from subgraphs of the social network Slashdot Zoo containing 200 to 10000 vertices. Such subgraphs were transformed into undirected graphs. Levorato [33] performed experiments with a parallel heuristic for the SRCC. Since we are specifically interested in comparing the performance of sequential implementations, it was thought advisable to consider the instances with up to 2000 vertices.
- *BR Congress instances* — Set of undirected graphs generated by Levorato and Frota [34] from voting sessions of the lower house of Brazilian National Congress. They created two graphs per year between 2011 and 2016, resulting in a total of 12 instances.

The reader is referred to Figueiredo and Frota [25], Levorato et al. [35], Levorato [33], Levorato and Frota [34] for a more detailed description.

6.2 Results for the ILP formulations

Tables 6, 7 and 8 present the results obtained by the formulation proposed in Figueiredo and Moura [26], as well as those determined by `F1` and `F2`. Column $\boldsymbol{z}$ represents the relaxed imbalance, given by $RI(P^*)$, where P^* is the solution (optimal or not) found by the corresponding formulation, **gap** informs percentage gaps calculated between best integer solutions found and final lower bounds (LB) as described in Equation (21), $\boldsymbol{t}$ indicates the CPU time in seconds ("-" means the instance was not solved in the time limit set), and **nodes** is the number of nodes that were solved during the search. Regarding the ILP formulation proposed by

Figueiredo and Moura [26], we report the original results which were found using XPRESS 21.01.00 (Intel Core 2 Duo 2.10 GHz and 3 GB of RAM) and also those determined by CPLEX 12.7 in order to perform a fair comparison. A time limit of 3600 seconds was imposed for each run.

$$gap = 100 \times (BestInteger - LB)/BestInteger \tag{21}$$

We followed the same procedure adopted in Figueiredo and Moura [26] in our testing. For each digraph, we start the experiments with $k = 2$. If the instance is solved to optimality, we then increase the value of k by one unit and attempt to solve the problem again (*forward phase*). If an optimal solution with relaxed imbalance 0 is found, we then interrupt the experiments for that particular digraph since this solution is also optimal for instances with larger values of k. When an instance is not solved to optimality, a similar procedure is carried out to solve instances from $k = n - 1$, where the value of k is decreased by one unit after each successful optimization (*backward phase*). In the backward phase, we use the value of the optimal solution found in the previous run (i.e., for $k + 1$) as a lower bound for the instance with k. The backward phase is finished when an instance is not solved to optimality or when the current instance was solved during the forward phase.

Table 6: Results obtained for instances House A Sum, House B Sum and House C Sum.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
HAS	2	96	0	59	1579	96	57.3	–	14874	96	0	1	688	96	0	22	1362
	3	57	78.9	–	31737					**50**	0	19	10921	**50**	0	2911	103801
	4									**31**	0	92	31468	**31**	0	3115	118591
	5									**27**	0	824	116453	27	78.2	–	94578
	6									**21**	0	1931	185902				
	7									18	29.7	–	270282				
	10									6	33.3	–	99450	6	33.3	–	50536
	11									**4**	0	1415	34689	**4**	0	501	8484
	12									**1**	0	190	5535	**1**	0	73	1544
	13	12	83.3	–	20945					**0**	0	109	2396	**0**	0	70	1240
	14	2[a]	0	1555	16703					0	0	61	1380	0	0	37	730
	15	0	0	3585	30208					0	0	60	831	0	0	7	100
	16	0	0	1162	7358					0	0	18	320	0	0	9	210
	17	0	0	601	2319	6	100.0	–	2499	0	0	91	1300	0	0	19	592
	18	0	0	599	2634	0	0	921	1455	0	0	19	234	0	0	10	248
	19	0	0	23	1	0	0	618	1180	0	0	1	1	0	0	10	260
	20	0	0	< 1	1	0	0	408	1139	0	0	82	671	0	0	10	260

Table 6 – *Continued from previous page*

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
HBS	2	84	0	22	1115	84	69.6	–	28015	84	0	< 1	747	84	0	10	1381
	3	75	47.5	–	9064					**69**	0	56	61150	**69**	0	867	99990
	4									**56**	0	461	203765	56	39.3	–	286043
	5									**43**	0	2161	514692				
	6									**33**	0	2595	383348				
	7									29	44.1	–	409051				
	9									15	26.7	–	138796	15	26.7	–	102548
	10									**11**	0	2536	97956	**11**	0	2186	55691
	11									**8**	0	592	17562	**8**	0	642	20574
	12	5	60.0	–	80375					**5**	0	297	8618	**5**	0	185	6802
	13	2	0	715	13538					2	0	67	2056	2	0	11	883
	14	1	0	279	3761					1	0	14	631	1	0	9	690
	15	0	0	85	584					0	0	28	1026	0	0	2	1
	16	0	0	< 1	1					0	0	18	942	0	0	2	1
HCS	2	64	0	26	615	64	0	1348	7266	64	0	1	363	64	0	8	555
	3	60	83.0	–	42981	96	100.0	–	1323	**53**	0	27	19422	**53**	0	417	25068
	4									**43**	0	341	137992	44	49.1	–	112782
	5									**35**	0	2348	396810				
	6									31	44.2	–	344290				
	12									8	37.5	–	73495	8	37.5	–	81225
	13									**5**	0	2340	37753	**5**	0	1961	40391
	14									**3**	0	1301	18368	**3**	0	476	9800
	15	3	66.7	–	35179					**2**	0	369	4734	**2**	0	84	1900
	16	1	0	2097	12517	5	100.0	–	2556	1	0	103	2270	1	0	16	669
	17	0	0	118	383	0	0	304	500	0	0	34	557	0	0	2	1
	18	0	0	96	153	0	0	255	500	0	0	23	433	0	0	2	1
	19	0	0	< 1	1	0	0	261	500	0	0	59	1024	0	0	2	1

[a] Possible typo

Table 7: Results obtained for instances MonkT2, MonkT3, MonkT4, and MonkT4 Sum.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MT2	2	43	0	13	733	43	0	37	1343	43	0	< 1	818	43	0	2	534
	3	25	0	2238	70771	27	92.6	–	11388	25	0	5	7049	25	0	42	6271
	4	20	85.0	–	121561					**13**	0	9	6998	**13**	0	212	26725
	5									**8**	0	15	7077	**8**	0	116	11601
	6									**4**	0	11	3765	**4**	0	76	8493
	7					14	92.9	–	12097	**2**	0	4	1177	**2**	0	6	718
	8					**1**	0	509	2497	**1**	0	2	236	**1**	0	4	488
	9					**0**	0	666	3835	**0**	0	1	252	**0**	0	4	720
	10					0	0	128	1484								
	11	2	100.0	–	176063	0	0	169	1781								
	12	0	0	2369	102937	0	0	80	1578								
	13	0	0	222	8881	0	0	19	278								
	14	0	0	48	593	0	0	54	1438								
	15	0	0	15	69	0	0	62	1543								
	16	0	0	7	1	0	0	61	1597								
	17	0	0	< 1	1	0	0	40	1082								

Table 7 – *Continued from previous page*

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MT3	2	32	0	6	243	32	0	30	1404	32	0	< 1	126	32	0	2	109
	3	21	0	193	4765	28	92.9	–	8924	21	0	1	876	21	0	7	1964
	4	13	0	2269	54227					13	0	2	1835	13	0	13	2690
	5	8	0	3325	85056					8	0	7	3460	8	0	10	2013
	6	7	71.4	–	100613					**6**	0	4	1840	**6**	0	34	3979
	7	5	60.0	–	140597	4	50.0	–	21250	**4**	0	4	1551	**4**	0	9	1749
	8	2	0	2837	6725	2	0	675	6642	2	0	2	517	2	0	5	739
	9	1	0	438	11577	1	0	56	1610	1	0	1	171	1	0	3	290
	10	0	0	1182	44984	0	0	50	1090	0	0	< 1	1	0	0	2	402
	11	0	0	251	8439	0	0	32	989								
	12	0	0	519	21543	0	0	7	150								
	13	0	0	103	5335	0	0	7	143								
	14	0	0	49	1120	0	0	8	174								
	15	0	0	30	455	0	0	8	197								
	16	0	0	18	119	0	0	8	197								
	17	0	0	< 1	1	0	0	8	197								
MT4	2	25	0	5	149	25	0	11	206	25	0	< 1	194	25	0	1	99
	3	21	0	119	3381	21	75.2	–	12524	21	0	1	923	21	0	6	1459
	4	10	0	563	13945					10	0	1	706	10	0	9	1864
	5	6	0	1463	42782					6	0	3	1283	6	0	64	6038
	6	4	0	2391	68659	5	80.0	–	7519	4	0	2	719	4	0	47	4389
	7	1	0	683	19452	1	0	141	1370	1	0	1	308	1	0	3	476
	8	0	0	414	12469	0	0	31	691	0	0	1	61	0	0	2	331
	9					0	0	66	537								
	10					0	0	35	756								
	11					0	0	4	1								
	12					0	0	11	1								
	13					0	0	11	1								
	14					0	0	9	1								
	15					0	0	8	1								
	16					0	0	9	1								
	17					0	0	8	1								
MT4S	2	86	0	14	347	86	0	66	518	86	0	< 1	217	86	0	2	229
	3	54	0	1539	25379	85	98.8	–	2941	54	0	1	988	54	0	80	5595
	4	43	72.2	–	7483					**36**	0	8	3340	**36**	0	298	21034
	5									**25**	0	23	6128	**25**	0	300	17832
	6									**16**	0	42	7039	**16**	0	225	11659
	7									**12**	0	41	5561	**12**	0	131	5068
	8					9	66.7	–	9104	**8**	0	18	2281	**8**	0	58	2232
	9	6	100.0	–	61861	**3**	0	3407	5188	**3**	0	7	590	**3**	0	8	525
	10	2	0	2582	51491	2	0	3498	5701	2	0	3	172	2	0	6	372
	11	0	0	1740	25094	0	0	79	470	0	0	1	1	0	0	8	602
	12	0	0	925	11838	0	0	126	935								
	13	0	0	58	240	0	0	108	935								
	14	0	0	300	3389	0	0	18	1								
	15	0	0	7	1	0	0	19	1								
	16	0	0	1	1	0	0	19	1								
	17	0	0	< 1	1	0	0	18	1								

Table 8: Results obtained for instances McKinney and NewComb.

Instance	k	Literature ILP formulation								F1				F2			
		XPRESS				CPLEX											
		z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes	z	gap	t	nodes
MK	2	8	0	118	6531	8	100.0	–	7559	8	0	< 1	28	8	0	112	2802
	3	6	100.0	–	43762					**2**	0	< 1	197	**2**	0	227	7766
	4									**0**	0	< 1	1	**0**	0	50	1567
	13					2	100.0	–	1432								
	14					0	0	57	1								
	15					0	0	59	1								
	16	2	100.0	–	33562	0	0	53	1								
	17	0	0	81	169	0	0	53	1								
	18	0	0	2	1	0	0	52	1								
	19	0	0	19	1	0	0	53	1								
	20	0	0	1	1	0	0	55	1								
	21	0	0	16	1	0	0	58	1								
	22	0	0	2	1	0	0	55	1								
	23	0	0	5	1	0	0	60	1								
	24	0	0	6	1	0	0	58	1								
	25	0	0	1	1	0	0	53	1								
	26	0	0	95	49	0	0	57	1								
	27	0	0	2	1	0	0	59	1								
	28	0	0	< 1	1	0	0	54	1								
NC	2	10	0	4	167	10	0	34	468	10	0	< 1	66	10	0	1	101
	3	7	0	475	9869	8	100.0	–	4988	7	0	1	1193	7	0	43	6188
	4	5	34.6	–	90604					**5**	0	7	5570	**5**	0	83	10538
	5					6	83.3	–	9349	**3**	0	11	4690	**3**	0	80	8054
	6					**1**	0	1299	6295	**1**	0	1	331	**1**	0	35	3561
	7					**0**	0	2646	7542	**0**	0	3	1093	**0**	0	12	1177
	8	1	100.0	–	146619	0	0	846	2861								
	9	0	0	172	9807	0	0	594	2624								
	10	0	0	123	5969	0	0	500	2276								
	11	0	0	27	405	0	0	46	1005								
	12	0	0	37	162	0	0	142	1815								
	13	0	0	8	1	0	0	294	2075								
	14	0	0	< 1	1	0	0	116	1172								
	15	0	0	< 1	1	0	0	85	1505								
	16	0	0	< 1	1	0	0	64	1505								

The results obtained show that `F1` outperforms the other formulations with respect to the number of optimal solutions achieved. When a formulation obtained an optimal solution with relaxed imbalance 0 for a given value of k, then we assume that all optima were found for executions with larger values of k (e.g., `F1` and `F2` solved instances from MT2 to optimality). While this formulation found 38, 64, 27, 15 optimal solutions for each group (House, Monastery, McKinney, NewComb), respectively, `F2` found 29, 64, 27, 15, respectively, and the formulation by Figueiredo and Moura [26] obtained 18, 48, 13, 10 using XPRESS, and 7, 44, 15, 12 using CPLEX, respectively. Overall, a total of 40 new optimal solutions (counting only once the optimal solutions with relaxed imbalance 0 for each digraph) were found and all instances of 6 of the 9 groups were solved to optimality.

In addition, it can be observed that `F1` is generally faster, but it is outrun by `F2` on instances HAS, HBS and HCS for larger values of k. Note that for the latter two, `F2` solves some instances at the root node. The formulation by Figueiredo

and Moura [26] is clearly slower, even taking into account the hardware difference for the results found using XPRESS. In general, more nodes are solved when running such formulation, but in some cases, `F2` is the one to solve more nodes.

The results also demonstrate that RCC has the expected behavior for k-partition problems, where problems with k close to 2 and $n-1$ are easier than those problems with k close to $n/2$. This can be explained by the number of possible partitions, which is given by the Stirling number of the second type [8] as described in Equation (22).

$$S(n,k) = \frac{1}{k!}\sum_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n \tag{22}$$

Finally, we report in Table 9 a comparison between the optimal solutions for RCC and CC. In addition to comparing $I(P)$ with $RI(P)$ and the correction $\delta = I(P) - RI(P)$ obtained with RCC (recall that RCC was proposed in order to correct *wrong penalties* in CC), we also decompose the total imbalance into minimum, average, and maximum penalties for intracluster and intercluster cases. With the exception of one instance (MT4 with $k = 3$), a positive correction is obtained by RCC. It is worth mentioning that most of the imbalance (and hence the correction) occurs in intercluster cases.

Table 9: Comparison of optimal solutions for RCC and CC in small instances.

Name	k	$I(P)$							$RI(P)$							δ
		total	intra			inter			total	intra			inter			
			min	avg	max	min	avg	max		min	avg	max	min	avg	max	
HAS	4	64	0	4.8	16	0	7.5	19	31	0	2.3	5	0	1.8	5	33
HBS	4	81	0	3.5	14	1	11.2	23	56	0	5.8	11	0	2.8	10	25
HCS	3	59	0	5.3	10	4	14.3	27	53	3	5.7	8	3	6.0	10	6
MT2	3	35	0	0.7	1	3	11.0	16	25	0	0.3	1	1	4.0	7	10
MT3	3	22	0	0.3	1	4	7.0	11	21	0	1.0	3	0	3.0	7	1
MT4	3	21	0	1.7	5	3	5.3	9	21	0	1.7	5	0	2.7	6	0
MT4S	3	62	1	3.0	7	12	17.7	25	54	1	3.0	7	2	7.5	15	8
MK	2	12	0	3.0	6	6	6.0	6	8	0	2.0	4	2	2.0	2	4
NC	4	20	0	1.3	5	0	2.5	7	5	0	0.5	1	0	0.3	1	15

6.3 ILS implementations

We used the same values adopted in Subramanian and Farias [45] for the main parameters of ILS_{RCC}, that is, $I_R = 20$ and $I_{ILS} = \min\{100, 4 \times n\}$. The only difference is that we imposed a minimum value of 100 for the latter as in Silva et al. [43]. Moreover, we set $maxPert = 6$ after conducting some experiments (see Section 6.3.1). The algorithms were executed 10 times on each instance in all experiments. Hereafter, the percentage gap of a solution P' is computed as $gap = 100 \times (f(P') - f(P_{best}))/f(P')$, such that f is the objective function (i.e., $f(P) = RI(P)$ for RCC and $f(P) = SRI(P)$ for SRCC) and P_{best} is the best solution among all solutions found by the ILS_{RCC} and the ILS of Levorato et al. [36].

6.3.1 Impact of the different components of the algorithm

We evaluate the ILS_{RCC} concerning the impact of: (i) using the greedy constructive algorithm by Levorato et al. [36]; (ii) the parameter $maxPert$; (iii) the neighborhood structures; (iv) perturbation mechanisms. To this end, we conducted experiments involving all 100-vertex random instances.

At first, we assess the impact of replacing the completely random construction (line 5 in Algorithm 1) with the greedy construction of Levorato et al. [36]. Table 10 shows the average gaps and CPU times obtained by the two versions of ILS_{RCC} (using two different constructive procedures) for different values of k. It can be seen that none of the two versions are significantly superior than the other w.r.t. both criteria. The only exception occurs when $k = 9$, where using the completely random construction produces a superior average gap and CPU time. In general, using the greedy construction in ILS_{RCC} leads to an improvement of only 0.01% in terms of average gap, and an increase of around 3% on the average CPU time. Therefore, it is reasonable to conclude that the effort to implement a more sophisticated constructive procedure may not be worthwhile for the RCC when the algorithm contains an effective local search.

Table 10: Comparison of two constructive heuristics in ILS_{RCC} for different values of k.

Construction	k	Avg. gap (%)	Avg. time (s)
Greedy [36]	3	**0.14**	7.09
	5	**0.92**	13.76
	7	**1.77**	19.78
	9	2.84	24.20
Mean		**1.42**	16.21
Random	3	0.17	**6.83**
	5	1.03	**13.36**
	7	1.96	**19.20**
	9	**2.56**	**23.31**
Mean		1.43	**15.68**

The impact of varying the parameter $maxPert$ is illustrated in Figure 7, where values between 4 and 8 are considered to compare different versions of ILS_{RCC}. The results show that the values 5 and 7 are dominated by the remaining ones; the value 4 produces the faster execution but with a poor average gap, whereas the value 8 achieves the best average gap but the worst CPU time. Therefore, we decided to use $maxPert = 6$ because it yields a good balance between solution quality and CPU time.

We assess the impact of the neighborhood operations by considering 7 different configurations. In this case, we consider $I_R = 20$ and $I_{ILS} = 1$ (i.e. only one iteration of the RVND procedure is executed) and we measure the percentage improvement over the initial solution. The average results are shown in Figure 8. We can observe that the configurations that yields the most promising results are those 5 and 7. The difference between both settings is that the latter includes the neighborhood *Swap*, which led to a slight improvement despite the additional CPU time.

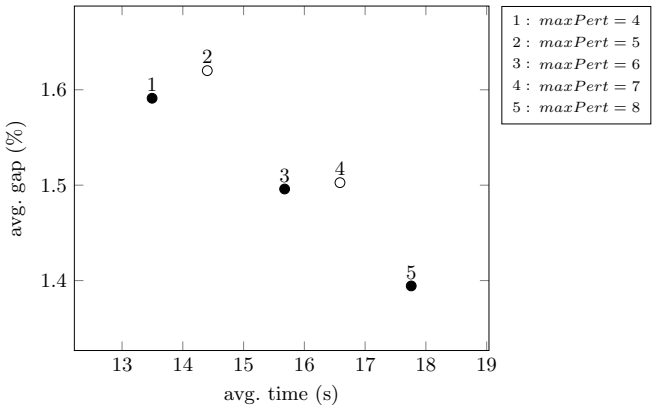


Fig. 7: Impact of the parameter $maxPert$. Each point represents a configuration and points with no fill represent those dominated by one or more settings.

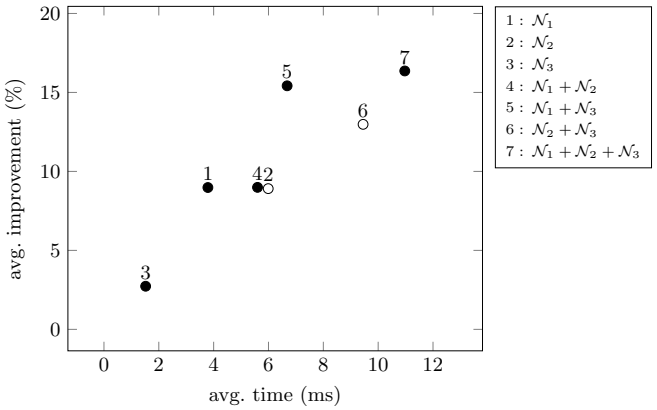
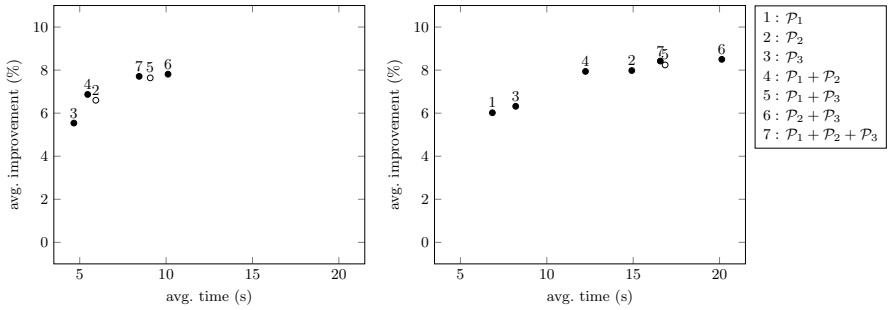


Fig. 8: Impact of the neighborhood operations. Each point represents a configuration and points with no fill represent those dominated by one or more settings. $\mathcal{N}_1$, $\mathcal{N}_2$ and $\mathcal{N}_3$ denote neighborhoods *Insert*, *Swap* and *Split*, respectively.

In order to measure the impact of the perturbation mechanisms, we run ILS_{RCC} using the default values of the parameters and store the percentage improvement over the best solution found in the previous testing for each instance. To choose an interesting configuration, we perform experiments considering two scenarios: with and without the neighborhood *swap*. The average results obtained are depicted in Figure 9. The best results are obtained by settings 6 and 7 for both scenarios. The scenario that considered *Swap* achieves better improvements at the expense of CPU time. Although not reported in Figure 9, the settings tested in the second scenario (i.e., the one including *Swap*) systematically found more best solutions than their corresponding counterpart in the first scenario. We, therefore, decided to select configuration 7 of the second scenario because it appears to offer an interesting compromise between solution quality and CPU time.



(a) (b)

Fig. 9: Impact of the perturbation operations. Each point represents a configuration and points with no fill represent those dominated by one or more settings. $\mathcal{P}_1$, $\mathcal{P}_2$ and $\mathcal{P}_3$ denote perturbations *Insert*, *Split* and *Sign Inversion*, respectively. Part (a) does not include the neighborhood *Swap* whereas part (b) does.

6.3.2 Comparison with the literature

The implementation by Levorato et al. [36] was originally devised for the symmetric version of the problem. Therefore, we had to slightly modify the source code, which was provided by the authors, to cope with the asymmetric case. We will refer to this method as $\text{ILS}_{\text{adapt}}$.

Concerning the small-size instances considered in Section 6.1.1, it was observed that ILS_{RCC} and $\text{ILS}_{\text{adapt}}$ are capable of consistently finding the optimal solutions in a fraction of a second.

Table 11 shows the aggregate results obtained by the ILS implementations on the set of random instances. For each digraph, we report the minimum, average and maximum values of the average percentage gaps (there is an average gap for each value of $k \in \{3, 5, 7, 9\}$), as well as the average CPU time in seconds. Detailed results are reported in Appendix A. Moreover, for an appropriate comparison, we have imposed the average CPU time obtained by ILS_{RCC}, for each instance, as a stopping criterion for $\text{ILS}_{\text{adapt}}$.

Table 11: Aggregate results for each digraph. Each row reports statistics on the average gaps obtained for the group of four instances (one for each value of $k \in \{3, 5, 7, 9\}$).

$\lvert V\rvert$	d	d^-	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
			min	avg	max	min	avg	max	
100	0.1	0.1	**0.82**	5.02	11.00	3.23	17.65	34.73	8.99
100	0.1	0.3	**0.02**	1.57	2.49	1.18	8.17	16.24	12.60
100	0.1	0.5	**0.02**	2.27	4.59	0.76	6.29	10.99	13.25
100	0.2	0.1	**0.69**	1.35	1.66	1.62	4.92	7.46	10.57
100	0.2	0.3	**0.00**	0.71	1.36	0.10	2.78	4.56	15.64
100	0.2	0.5	**0.11**	1.14	2.67	0.62	3.08	5.73	15.19
100	0.5	0.1	**0.12**	0.35	0.69	0.35	1.33	2.53	12.23
100	0.5	0.3	0.00	0.21	0.53	0.00	0.93	1.98	21.70
100	0.5	0.5	**0.15**	0.43	0.74	0.27	1.27	2.17	20.98
100	0.8	0.1	**0.00**	0.11	0.28	0.22	0.95	1.88	8.91

Table 11 – *Continued from previous page*

$\|V\|$	d	d^-	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
			min	avg	max	min	avg	max	
100	0.8	0.3	0.05	0.22	0.37	**0.03**	0.53	1.08	23.18
100	0.8	0.5	**0.00**	0.21	0.34	0.29	1.10	1.81	24.85
200	0.1	0.1	**0.52**	0.87	1.13	1.23	6.24	13.75	66.34
200	0.1	0.3	**0.03**	0.81	1.29	0.94	2.75	3.65	97.29
200	0.1	0.5	**0.39**	0.60	0.88	1.03	3.13	5.10	91.26
200	0.2	0.1	**0.15**	0.73	1.40	0.37	2.34	4.23	73.44
200	0.2	0.3	**0.16**	0.28	0.41	0.18	1.35	2.12	116.49
200	0.2	0.5	**0.19**	0.44	0.67	0.73	1.47	2.01	102.96
200	0.5	0.1	**0.04**	0.20	0.32	0.22	0.78	1.11	66.67
200	0.5	0.3	**0.05**	0.13	0.21	0.07	0.52	0.81	155.52
200	0.5	0.5	**0.05**	0.21	0.33	0.34	0.75	1.05	142.09
200	0.8	0.1	**0.01**	0.11	0.19	0.12	0.52	0.84	44.26
200	0.8	0.3	**0.01**	0.15	0.35	0.14	0.50	0.83	161.70
200	0.8	0.5	**0.04**	0.19	0.28	0.52	0.85	1.47	165.27
400	0.1	0.1	0.34	0.56	0.73	0.34	1.01	1.49	508.86
400	0.1	0.3	0.16	0.31	0.57	**0.09**	0.84	1.45	870.00
400	0.1	0.5	**0.09**	0.42	0.71	0.32	1.26	2.56	699.45
400	0.2	0.1	**0.07**	0.17	0.20	0.12	0.38	0.65	394.25
400	0.2	0.3	**0.04**	0.17	0.26	0.08	0.81	1.83	988.04
400	0.2	0.5	**0.11**	0.23	0.32	0.26	0.59	0.76	778.40
400	0.5	0.1	**0.05**	0.10	0.17	0.14	0.26	0.41	312.75
400	0.5	0.3	**0.03**	0.06	0.07	0.06	0.14	0.24	978.23
400	0.5	0.5	**0.08**	0.12	0.18	0.21	0.47	0.92	1093.10
400	0.8	0.1	**0.02**	0.05	0.08	0.07	0.13	0.20	183.53
400	0.8	0.3	**0.02**	0.05	0.07	0.12	0.15	0.19	881.24
400	0.8	0.5	**0.06**	0.11	0.16	0.13	0.30	0.40	1304.41
600	0.1	0.1	0.27	0.40	0.52	**0.20**	0.65	1.03	1338.98
600	0.1	0.3	**0.14**	0.21	0.35	0.18	0.69	1.27	3173.07
600	0.1	0.5	**0.15**	0.29	0.50	0.94	1.43	1.94	2470.35
600	0.2	0.1	**0.12**	0.15	0.20	0.26	0.32	0.39	1034.79
600	0.2	0.3	**0.06**	0.11	0.14	0.10	0.42	0.76	3313.68
600	0.2	0.5	**0.14**	0.22	0.27	0.38	0.56	0.73	2824.27
600	0.5	0.1	**0.03**	0.07	0.10	0.06	0.16	0.25	782.57
600	0.5	0.3	**0.02**	0.05	0.08	0.05	0.12	0.17	2822.82
600	0.5	0.5	**0.04**	0.06	0.11	0.21	0.30	0.36	3866.90
600	0.8	0.1	**0.02**	0.04	0.06	0.03	0.06	0.10	474.08
600	0.8	0.3	**0.02**	0.05	0.06	0.04	0.08	0.11	2170.92
600	0.8	0.5	**0.05**	0.09	0.13	0.18	0.30	0.40	4681.35

The results obtained show that, on average, ILS_{RCC} clearly outperforms $\text{ILS}_{\text{adapt}}$. When evaluating the performance of each individual instance, the ILS_{RCC} found the best solution (one with a gap of 0%) for 179 instances (93.2% of the cases), where among them 166 (86.5% of the cases) are strictly better than the best ones achieved by $\text{ILS}_{\text{adapt}}$. Furthermore, we can also see that the average runtime increases with the value of d^-.

Figure 10 illustrates how the average gap between the average solution and the best known solution varies according to different values of d, d^- and k. We can observe that the instances appear to become easier when the value of d increases, as depicted in Figure 10a. Furthermore, from Figure 10b, it is visible that the instances with a smaller value of d^- appear to be harder. Finally, larger values of k seem to increase the difficulty of the instances, as clearly shown in Figure 10c.

In Appendix A, we also report many improved upper bounds w.r.t those obtained in the experiment reported in Table 11. These improved solutions were

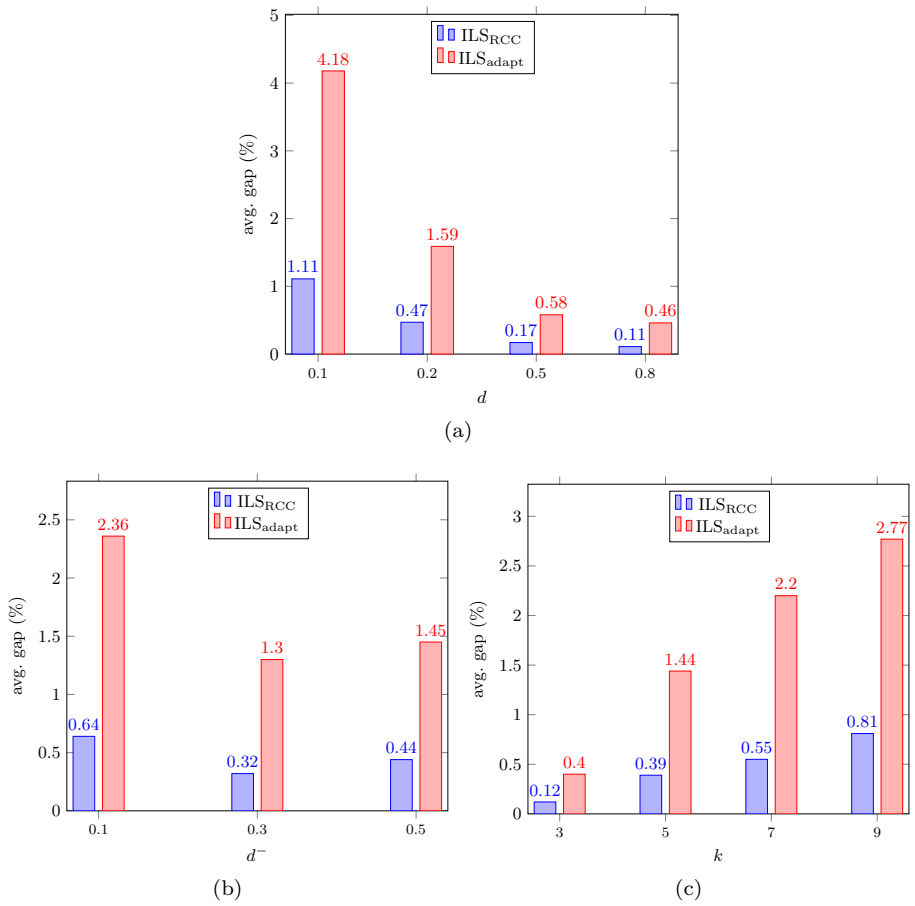


Fig. 10: Average gap performance according to characteristics of the instance.

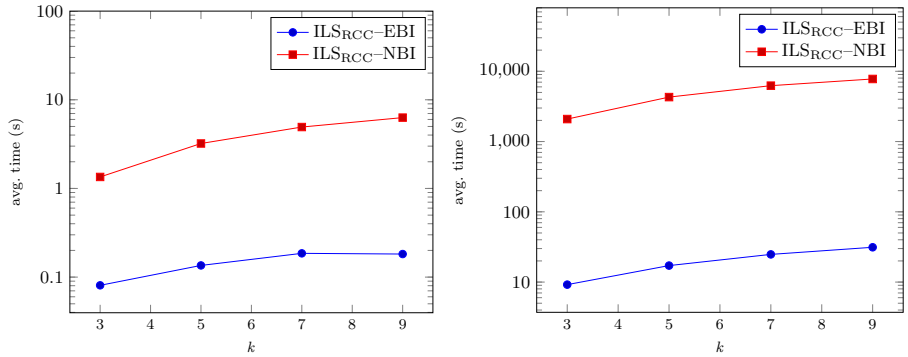
found while experimenting with different settings of the algorithm, and also during the preliminary experiments described in Section 6.3.1.

6.3.3 Impact of the ADSs on the runtime performance

This section examines the average runtime performance of ILS_{RCC} when incorporating the ADSs for efficiently computing the relaxed imbalance value of a neighbor solution during the local search.

Figure 11 depicts the CPU time of the versions of the algorithm using EBI and NBI, respectively, in the log scale. In Figure 11a, we illustrate the comparison for the small-size Monastery instances. Despite the considerable runtime difference, it can be seen that using NBI, i.e., the one that does not make use of ADSs to perform move evaluation, is still doable in practice, as the average CPU time spent by the method is fairly acceptable. However, for the 100-vertex instances, the difference is astonishing, and visibly illustrates the benefits of incorporating

the ADSs proposed in this work. Note that the disparity is likely to become even more prominent for larger instances.



(a) Monastery instances (18 vertices) (b) Random instances with 100 vertices

Fig. 11: Impact of the ADSs on the average CPU time (semi-log plot).

6.3.4 Results for the symmetric RCC instances

Table 12 shows the summary of the results obtained for each set of benchmark instances. In this case, because the original algorithm from the literature is used, we refer to it as "ILS Levorato et al. [36]". Detailed results are provided in Appendix B. We report the number of strictly best solutions found by each version (**#best**), the number of cases in which the best solution found by each algorithm were equal (**ties**), the average percentage gap (**gap$_{avg}$**) and the minimum, average and maximum CPU time, considering the average values of 10 runs for each instance and a time limit of 7200 seconds. The results illustrate that ILS$_{RCC}$ dominates ILS Levorato et al. [36] in terms of strictly best known solutions found, especially in Slashdot and Brazilian Congress benchmarks. To our knowledge, all best solutions found in this experiment are the best known.

Table 12: Summary of results for SRCC benchmarks

Benchmark	total	ILS$_{RCC}$		ILS Levorato et al. [36]		ties	time		
		#best	gap$_{avg}$	#best	gap$_{avg}$		min	avg	max
UNGA	63	**1**	0.02	0	1.72	62	0.5	4.6	13.0
Slashdot	7	**7**	9.72	0	40.08	0	16.9	2293.3	7200.0
BR Congress	14	**8**	0.59	0	1.16	6	61.3	232.7	534.1

7 Concluding remarks

This paper proposes exact and metaheuristic approaches for the relaxed correlation clustering (RCC) problem. In particular, we developed two integer linear

programming formulations that obtained a superior performance when compared to the existing one, as well as an enhanced iterated local search (ILS) algorithm that substantially outperformed the previous ILS implementation from the literature. One key factor of our ILS is the efficient move evaluation scheme, which was crucial for improving the scalability of the method. Moreover, we also put forward a novel perturbation mechanism for the problem that helped the algorithm to find high quality solutions. The performance of ILS was also assessed in benchmark instances of the symmetrical version of RCC (SRCC) and the results achieved were always at least as good as the best known.

Future work includes the development of efficient parallel algorithms for tackling very large instances that may arise in real-life social networks. In addition, as the current integer linear programming formulations are still limited to small-size instances, there is still room for developing enhanced exact algorithms, perhaps in the spirit of Brusco et al. [9], as an attempt to solve larger instances.

Acknowledgments

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Finance Code 001 and by the Conselho Nacional de Desenvolvimento Cientfífico e Tecnológico (CNPq), grants 305223/2015-1 and 303799/2018-8. We would also like to thank Mário Levorato for providing the SRCC source code and Pedro Liguori for the helpful insights on the mathematical models.

Data availability

The datasets analysed during the current study are available from the corresponding author on reasonable request.

References

1. Ales Z, Knippel A, Pauchet A (2016) Polyhedral combinatorics of the k-partitioning problem with representative variables. Discret Appl Math 211:1–14
2. Altafini C (2012) Dynamics of opinion forming in structurally balanced social networks. PLOS ONE 7(6):1–9
3. Arinik N, Figueiredo R, Labatut V (2017) Signed Graph Analysis for the Interpretation of Voting Behavior. In: International Conference on Knowledge Technologies and Data-driven Business (i-KNOW), Graz, Austria, International Workshop on Social Network Analysis and Digital Humanities (SnanDig)
4. Bahiense L, Frota Y, Maculan N, Noronha TF, Ribeiro CC (2009) A branch-and-cut algorithm for equitable coloring based on a formulation by representatives. Electron Notes in Discret Math 35:347–352
5. Bansal N, Blum A, Chawla S (2004) Correlation clustering. Mach Learn 56(1):89–113

6. Beier T, Hamprecht FA, Kappes JH (2015) Fusion moves for correlation clustering. In: CVPR. Proceedings, pp 3507–3516, 1
7. Bonami P, Nguyen VH, Klein M, Minoux M (2012) On the solution of a graph partitioning problem under capacity constraints. In: Mahjoub AR, Markakis V, Milis I, Paschos VT (eds) Combinatorial Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 285–296
8. Broder AZ (1984) The r-stirling numbers. Discrete Mathematics 49(3):241 – 259
9. Brusco M, Doreian P, Mrvar A, Steinley D (2011) Two algorithms for relaxed structural balance partitioning: Linking theory, models, and data to understand social network phenomena. Sociol Methods & Res 40(1):57–87
10. Brusco MJ, Doreian P (2019) Partitioning signed networks using relocation heuristics, tabu search, and variable neighborhood search. Soc Netw 56:70 – 80
11. Bulhões T, de Sousa Filho GF, Subramanian A, Lucídio dos Anjos FC (2017) Branch-and-cut approaches for p-cluster editing. Discret Appl Math 219:51–64
12. Campêlo M, Campos VA, Correa RC (2008) On the asymmetric representatives formulation for the vertex coloring problem. Discret Appl Math 156:1097–1111
13. Campêlo MB, Corrêa RC, Frota Y (2004) Cliques, holes and the vertex coloring polytope. Inf Process Lett 89(4):159–164
14. Cartwright D, Harary F (1956) Structural balance: a generalization of heider's theory. Psychological review 63(5):277
15. Dambacher JM, Li HW, Rossignol PA (2002) Relevance of community structure in assessing indeterminacy of ecological predictions. Ecol 83(5):1372–1385
16. DasGupta B, AEnciso G, Sontag E, Zhang Y (2007) Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. BioSyst 90:161–178
17. Davis JA (1967) Clustering and structural balance in graphs. Human relations 20(2):181–187
18. Doreian P (2008) A multiple indicator approach to blockmodeling signed networks. Soc Netw 30(3):247 – 258
19. Doreian P, Mrvar A (1996) A partitioning approach to structural balance. Soc Netw 18(2):149 – 168
20. Doreian P, Mrvar A (2009) Partitioning signed social networks. Soc Netw 31(1):1 – 11
21. Doreian P, Mrvar A (2014) Testing two theories for generating signed networks using real data
22. Doreian P, Mrvar A (2015) Structural balance and signed international relations. J of Soc Struct 16:2
23. Facchetti G, Iacono G, Altafini C (2011) Computing global structural balance in large-scale signed social networks. Proc of the National Acad of Sci 108(52):20953–20958
24. Fan N, Pardalos PM (2010) Linear and quadratic programming approaches for the general graph partitioning problem. J of Glob Optim 48(1):57–71
25. Figueiredo R, Frota Y (2014) The maximum balanced subgraph of a signed graph: Applications and solution approaches. Eur J of Oper Res 236(2):473 – 487

26. Figueiredo R, Moura G (2013) Mixed integer programming formulations for clustering problems related to structural balance. Soc Netw 35(4):639 – 651
27. Figueiredo R, Frota Y, Labb M (2018) A branch-and-cut algorithm for the maximum k-balanced subgraph of a signed graph. Discret Appl Math
28. Frota Y, Maculan N, Noronha TF, Ribeiro CC (2010) A branch-and-cut algorithm for partition coloring. Netw 55:194–204
29. Harary F, Lim M, Wunsch DC (2003) Signed graphs for portfolio analysis in risk management. IMA J of Manag Math 13:1–10
30. Heider F (1946) Attitudes and cognitive organization. The Journal of Psychology 21(1):107–112, pMID: 21010780
31. Kim S, Nowozin S, Kohli P, Yoo CD (2011) Higher-order correlation clustering for image segmentation. In: Shawe-Taylor J, Zemel RS, Bartlett PL, Pereira F, Weinberger KQ (eds) Advances in Neural Information Processing Systems 24, Curran Associates, Inc., pp 1530–1538
32. Lemann TB, Solomon RL (1952) Group characteristics as revealed in sociometric patterns and personality ratings. Sociom 15(1/2):7–90
33. Levorato M (2015) Efficient solutions to the correlation clustering problem. Master's thesis, Universidade Federal Fluminense, Niterói, Rio de Janeiro, Brazil, available at `http://www.ic.uff.br/PosGraduacao/frontend-tesesdissertacoes/download.php?id=700.pdf&tipo=trabalho`
34. Levorato M, Frota Y (2017) Brazilian congress structural balance analysis. J of Interdiscip Methodol and Issues in Sci
35. Levorato M, Drummond L, Frota Y, Figueiredo R (2015) An ils algorithm to evaluate structural balance in signed social networks. In: Proceedings of the 30th Annual ACM Symposium on Applied Computing, ACM, New York, NY, USA, SAC '15, pp 1117–1122
36. Levorato M, Figueiredo R, Frota Y, Drummond L (2017) Evaluating balancing on social networks through the efficient solution of correlation clustering problems. EURO J on Comput Optim 5(4):467–498
37. Lourenço HR, Martin OC, Stützle T (2010) Iterated local search: Framework and applications. In: Gendreau M, Potvin JY (eds) Handbook of Metaheuristics, Springer US, Boston, MA, pp 363–397
38. Maurya MR, Rengaswamy R, Venkatasubramanian V (2004) Application of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets. Eng Appl of Artif Intell 17(5):501 – 518
39. McKinney JC (1948) An educational application of a two-dimensional sociometric test. Sociom 11(4):356–367
40. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput & Oper Res 24(11):1097–1100
41. Newcomb TM (1961) The acquaintance process. Holt, Rinehart & Winston, New York
42. Sampson SF (1968) A novitiate in a period of change: An experimental and case study of social relationships. PhD thesis, Department of Sociology, Cornell University, NY
43. Silva MM, Subramanian A, Vidal T, Ochi LS (2012) A simple and effective metaheuristic for the minimum latency problem. Eur J of Oper Res 221(3):513 – 520
44. Silva MM, Subramanian A, Ochi LS (2015) An iterated local search heuristic for the split delivery vehicle routing problem. Comput & Oper Res 53:234 –

249
45. Subramanian A, Farias K (2017) Efficient local search limitation strategy for single machine total weighted tardiness scheduling with sequence-dependent setup times. Comput & Oper Res 79:190 – 206
46. Van Gael J, Zhu X (2007) Correlation clustering for crosslingual link detection. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'07, pp 1744–1749
47. Vasanthi B, Arumugam S, Nagar AK, Mitra S (2015) Applications of signed graphs to portfolio turnover analysis. Procedia - Soc and Behav Sci 211:1203 – 1209, 2nd Global Conference on Business and Social Sciences (GCBSS-2015) on Multidisciplinary Perspectives on Management and Society, 17- 18 September, 2015, Bali, Indonesia
48. Wang N, Li J (2013) Restoring: A greedy heuristic approach based on neighborhood for correlation clustering. In: Motoda H, Wu Z, Cao L, Zaiane O, Yao M, Wang W (eds) Advanced Data Mining and Applications, Springer Berlin Heidelberg, Berlin, Heidelberg, pp 348–359
49. Yang B, Cheung W, Liu J (2007) Community mining from signed social networks. IEEE Trans on Knowl and Data Eng 19:1333–1348
50. Zaslavsky T (1982) Signed graphs. Discret Appl Math 4:47–74
51. Zaslavsky T (1998) A mathematical bibliography of signed and gain graphs and allied areas. Electron J of Comb DS8

A Detailed results for the random RCC instances

Table 13: Relaxed imbalance obtained by ILS_{RCC} and $\text{ILS}_{\text{adapt}}$.

$\lvert V\rvert$	d	d^-	k	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
				min	avg	max	min	avg	max	
100	0.1	0.1	3	**131**	132.10	134	133	135.40	139	2.82
100	0.1	0.1	5	**94**	98.20	101	103	108.50	112	6.71
100	0.1	0.1	7	**68**	70.90	74	78	84.40	89	11.80
100	0.1	0.1	9	**45**	50.70	54	66	69.00	73	14.65
100	0.1	0.3	3	**408**	408.10	409	409	412.90	416	5.73
100	0.1	0.3	5	**323**	327.50	333	335	342.60	348	11.05
100	0.1	0.3	7	**270**	276.70	284	280	294.00	305	15.84
100	0.1	0.3	9	**233**	239.00	247	252	257.60	263	17.78
100	0.1	0.5	3	533	533.10	534	533	537.10	546	6.93
100	0.1	0.5	5	**427**	430.70	435	435	446.60	457	11.98
100	0.1	0.5	7	**352**	365.30	371	380	387.10	397	15.10
100	0.1	0.5	9	**298**	312.50	324	325	334.90	341	18.99
100	0.2	0.1	3	327	329.30	333	327	332.40	336	2.83
100	0.2	0.1	5	**284**	288.80	291	293	297.00	302	7.22
100	0.2	0.1	7	**253**	256.70	261	263	269.90	275	13.10
100	0.2	0.1	9	**227**	230.70	233	234	245.40	252	19.12
100	0.2	0.3	3	1032	1032.00	1032	1032	1033.00	1035	5.88
100	0.2	0.3	5	**913**	914.90	917	927	937.10	948	14.35
100	0.2	0.3	7	**833**	843.70	854	847	866.80	876	19.09
100	0.2	0.3	9	**775**	785.70	799	797	812.10	821	23.22
100	0.2	0.5	3	**1341**	1342.50	1344	1346	1349.40	1358	7.87
100	0.2	0.5	5	**1162**	1171.40	1177	1182	1194.90	1204	13.43
100	0.2	0.5	7	**1057**	1067.60	1075	1070	1092.30	1105	17.56
100	0.2	0.5	9	**955**	981.30	998	994	1013.10	1023	21.90
100	0.5	0.1	3	940	941.10	943	940	943.30	947	3.81
100	0.5	0.1	5	**899**	901.00	903	902	906.50	910	8.74
100	0.5	0.1	7	**864**	867.30	873	871	878.20	883	14.85
100	0.5	0.1	9	**833**	838.80	843	844	854.70	871	21.54
100	0.5	0.3	3	2741	2741.00	2741	2741	2741.10	2742	7.57
100	0.5	0.3	5	**2629**	2631.20	2634	2632	2636.70	2642	19.69
100	0.5	0.3	7	**2532**	2537.90	2556	2555	2568.80	2581	28.54
100	0.5	0.3	9	**2459**	2472.10	2487	2497	2508.60	2524	31.01
100	0.5	0.5	3	3939	3944.90	3958	3939	3949.80	3966	12.35
100	0.5	0.5	5	**3659**	3677.70	3693	3697	3713.10	3729	19.50
100	0.5	0.5	7	**3507**	3518.90	3529	3537	3548.70	3559	25.56
100	0.5	0.5	9	**3343**	3367.90	3382	3399	3417.10	3435	26.51
100	0.8	0.1	3	1537	1538.00	1541	1537	1540.40	1546	2.82
100	0.8	0.1	5	**1502**	1502.00	1502	1503	1509.50	1515	6.08
100	0.8	0.1	7	**1470**	1471.30	1474	1473	1484.00	1493	10.41
100	0.8	0.1	9	**1439**	1443.10	1445	1448	1456.40	1465	16.35
100	0.8	0.3	3	4599	4601.50	4606	4599	4600.60	4604	7.17
100	0.8	0.3	5	**4462**	4468.90	4476	4468	4478.60	4488	18.14
100	0.8	0.3	7	**4364**	4377.30	4385	4369	4391.50	4403	30.49
100	0.8	0.3	9	**4270**	4286.10	4308	4303	4316.60	4332	36.91
100	0.8	0.5	3	6676	6676.00	6676	6676	6695.60	6741	16.14
100	0.8	0.5	5	**6319**	6330.80	6347	6343	6376.80	6401	23.40
100	0.8	0.5	7	**6095**	6115.60	6141	6136	6166.90	6195	28.12
100	0.8	0.5	9	**5921**	5939.60	5959	5973	5989.00	6008	31.76
200	0.1	0.1	3	**691**	694.60	700	695	699.60	703	16.03
200	0.1	0.1	5	**602**	606.70	611	615	624.30	634	42.44
200	0.1	0.1	7	**539**	545.20	550	565	576.10	582	85.12
200	0.1	0.1	9	**499**	504.30	511	524	535.90	544	121.75
200	0.1	0.3	3	**1993**	1993.50	1995	2002	2011.90	2024	32.91
200	0.1	0.3	5	**1782**	1799.30	1812	1823	1834.90	1848	90.46
200	0.1	0.3	7	**1659**	1680.70	1695	1712	1719.70	1730	123.25
200	0.1	0.3	9	**1574**	1589.70	1607	1619	1633.60	1645	142.54
200	0.1	0.5	3	**2716**	2726.60	2744	2732	2744.20	2754	56.06
200	0.1	0.5	5	**2403**	2414.90	2435	2435	2465.90	2484	81.47
200	0.1	0.5	7	**2226**	2240.50	2260	2274	2299.50	2327	108.14
200	0.1	0.5	9	**2104**	2122.70	2135	2151	2173.70	2189	119.36
200	0.2	0.1	3	**1515**	1517.30	1526	1516	1520.70	1524	15.82
200	0.2	0.1	5	**1422**	1433.80	1445	1437	1452.50	1461	43.79

Table 13 – *Continued from previous page*

$\|V\|$	d	d^-	k	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
				min	avg	max	min	avg	max	
200	0.2	0.1	7	**1357**	1364.40	1371	1384	1394.30	1410	88.66
200	0.2	0.1	9	**1293**	1311.50	1326	1328	1350.10	1357	145.51
200	0.2	0.3	3	**4376**	4383.10	4387	4377	4384.10	4392	32.41
200	0.2	0.3	5	**4136**	4144.30	4154	4161	4176.80	4194	102.09
200	0.2	0.3	7	**3957**	3971.30	3988	4027	4042.70	4054	163.24
200	0.2	0.3	9	**3856**	3871.90	3885	3916	3938.90	3960	168.21
200	0.2	0.5	3	**6142**	6153.80	6178	6172	6187.30	6206	66.70
200	0.2	0.5	5	**5689**	5715.00	5729	5740	5772.30	5818	92.31
200	0.2	0.5	7	**5436**	5460.90	5501	5449	5528.80	5558	114.75
200	0.2	0.5	9	**5229**	5264.30	5283	5298	5336.40	5362	138.09
200	0.5	0.1	3	**3950**	3951.50	3957	3952	3958.90	3963	20.22
200	0.5	0.1	5	**3876**	3887.10	3899	3893	3903.90	3912	43.74
200	0.5	0.1	7	**3814**	3826.40	3835	3835	3855.60	3869	77.27
200	0.5	0.1	9	**3769**	3774.70	3782	3802	3811.20	3824	125.47
200	0.5	0.3	3	11624	11629.30	11639	11624	11632.30	11643	43.78
200	0.5	0.3	5	**11368**	11384.70	11398	11398	11416.20	11436	112.73
200	0.5	0.3	7	**11169**	11193.00	11217	11227	11254.90	11273	213.97
200	0.5	0.3	9	**11052**	11066.70	11089	11122	11142.10	11175	251.60
200	0.5	0.5	3	**17090**	17098.40	17124	17121	17147.80	17192	104.95
200	0.5	0.5	5	**16367**	16401.30	16462	16434	16496.40	16540	136.04
200	0.5	0.5	7	**15969**	16006.10	16038	16054	16099.80	16137	157.54
200	0.5	0.5	9	**15621**	15673.30	15710	15734	15787.00	15821	169.82
200	0.8	0.1	3	6348	6348.80	6350	6348	6355.50	6366	14.59
200	0.8	0.1	5	**6284**	6291.10	6295	6297	6310.60	6322	29.45
200	0.8	0.1	7	**6226**	6237.90	6247	6243	6263.90	6279	52.70
200	0.8	0.1	9	**6181**	6189.70	6202	6199	6214.10	6234	80.28
200	0.8	0.3	3	**18810**	18811.80	18816	18817	18823.60	18831	41.21
200	0.8	0.3	5	**18546**	18556.70	18569	18573	18588.90	18612	116.73
200	0.8	0.3	7	**18339**	18374.70	18400	18381	18415.50	18450	197.81
200	0.8	0.3	9	**18148**	18211.30	18241	18261	18291.50	18319	291.05
200	0.8	0.5	3	**28291**	28301.70	28311	28305	28364.70	28410	129.39
200	0.8	0.5	5	**27342**	27400.10	27458	27497	27544.10	27589	157.90
200	0.8	0.5	7	**26793**	26867.10	26911	26921	27000.30	27078	180.17
200	0.8	0.5	9	**26416**	26474.00	26534	26521	26591.40	26676	193.61
400	0.1	0.1	3	2989	2996.30	3006	**2986**	2996.10	3003	104.42
400	0.1	0.1	5	**2812**	2826.60	2839	2813	2833.70	2853	284.54
400	0.1	0.1	7	**2669**	2688.60	2711	2687	2708.50	2756	625.85
400	0.1	0.1	9	**2569**	2586.10	2605	2578	2608.00	2633	1020.60
400	0.1	0.3	3	8843	8851.50	8862	**8837**	8845.30	8854	233.79
400	0.1	0.3	5	**8360**	8374.40	8383	8389	8413.10	8475	811.53
400	0.1	0.3	7	**8022**	8047.40	8074	8040	8119.80	8206	1280.75
400	0.1	0.3	9	**7835**	7880.20	7934	7893	7950.30	7990	1153.92
400	0.1	0.5	3	12523	12532.80	12544	**12522**	12562.00	12610	508.94
400	0.1	0.5	5	**11611**	11648.60	11679	11622	11734.00	11783	660.74
400	0.1	0.5	7	**11099**	11178.80	11209	11179	11243.30	11309	815.20
400	0.1	0.5	9	**10807**	10868.90	10928	10869	10929.00	10985	812.93
400	0.2	0.1	3	6296	6300.10	6306	6296	6303.30	6315	91.57
400	0.2	0.1	5	**6147**	6159.20	6170	6157	6167.90	6186	247.20
400	0.2	0.1	7	**6028**	6039.90	6071	6035	6054.00	6085	462.77
400	0.2	0.1	9	**5912**	5923.90	5934	5927	5950.60	5975	775.45
400	0.2	0.3	3	18278	18284.70	18298	**18277**	18291.60	18305	235.19
400	0.2	0.3	5	**17735**	17758.10	17779	17764	17802.00	17849	786.29
400	0.2	0.3	7	**17385**	17431.00	17467	17509	17555.30	17597	1370.41
400	0.2	0.3	9	**17163**	17207.10	17236	17244	17321.80	17373	1560.29
400	0.2	0.5	3	26898	26921.80	26941	**26892**	26960.90	27041	567.47
400	0.2	0.5	5	**25577**	25655.80	25711	25707	25774.00	25805	749.25
400	0.2	0.5	7	**24947**	24994.40	25055	25016	25108.90	25199	846.28
400	0.2	0.5	9	**24472**	24550.60	24590	24530	24639.40	24727	950.62
400	0.5	0.1	3	**15853**	15860.50	15867	15865	15875.60	15885	100.59
400	0.5	0.1	5	**15758**	15767.10	15777	15775	15786.70	15811	208.04
400	0.5	0.1	7	**15642**	15668.50	15684	15665	15689.90	15717	361.93
400	0.5	0.1	9	**15561**	15580.80	15610	15602	15625.00	15641	580.47
400	0.5	0.3	3	**47486**	47500.60	47521	47488	47516.40	47536	228.52
400	0.5	0.3	5	47004	47020.20	47038	**46996**	47035.40	47074	666.00
400	0.5	0.3	7	**46604**	46636.50	46677	46632	46682.40	46759	1295.69
400	0.5	0.3	9	**46319**	46352.50	46381	46353	46429.70	46474	1722.73
400	0.5	0.5	3	**71795**	71852.80	71898	71858	71969.60	72029	955.35

Table 13 – *Continued from previous page*

$\|V\|$	d	d^-	k	ILS_{RCC}			$\text{ILS}_{\text{adapt}}$			t_{avg}
				min	avg	max	min	avg	max	
400	0.5	0.5	5	**69825**	69888.40	69958	69973	70148.90	70274	1105.49
400	0.5	0.5	7	**68747**	68873.20	68939	68945	69082.80	69205	1116.31
400	0.5	0.5	9	**68093**	68167.30	68262	68103	68239.80	68522	1195.26
400	0.8	0.1	3	**25285**	25289.20	25293	25295	25302.50	25310	68.99
400	0.8	0.1	5	**25192**	25203.60	25210	25205	25220.40	25242	134.38
400	0.8	0.1	7	**25086**	25106.80	25124	25095	25135.10	25170	209.22
400	0.8	0.1	9	**25027**	25038.30	25051	25054	25064.40	25081	321.54
400	0.8	0.3	3	**76029**	76044.80	76075	76039	76084.70	76157	214.40
400	0.8	0.3	5	**75601**	75627.60	75674	75605	75673.40	75748	547.72
400	0.8	0.3	7	**75228**	75280.40	75332	75237	75315.60	75396	1099.86
400	0.8	0.3	9	**74910**	74954.10	74982	74960	75032.90	75096	1662.96
400	0.8	0.5	3	**117476**	117543.50	117626	117527	117631.00	117773	1109.08
400	0.8	0.5	5	**114831**	114990.90	115126	115077	115261.10	115489	1294.47
400	0.8	0.5	7	**113596**	113681.80	113790	113660	113938.50	114126	1407.46
400	0.8	0.5	9	**112546**	112725.80	112904	112836	112996.20	113288	1406.61
600	0.1	0.1	3	6842	6854.10	6861	**6833**	6847.00	6858	317.03
600	0.1	0.1	5	6621	6632.10	6643	**6614**	6642.60	6678	736.46
600	0.1	0.1	7	**6405**	6438.80	6474	6438	6464.60	6501	1656.70
600	0.1	0.1	9	**6251**	6283.30	6298	6291	6316.40	6359	2645.73
600	0.1	0.3	3	**20349**	20382.00	20398	20356	20385.00	20412	783.90
600	0.1	0.3	5	**19655**	19682.20	19706	19712	19755.70	19788	2518.19
600	0.1	0.3	7	**19168**	19201.50	19228	19238	19324.00	19372	4519.86
600	0.1	0.3	9	**18775**	18840.30	18875	18913	19016.70	19078	4870.33
600	0.1	0.5	3	**29305**	29349.80	29379	29379	29451.00	29516	2034.70
600	0.1	0.5	5	**27696**	27765.20	27866	27923	27959.40	28017	2364.64
600	0.1	0.5	7	**26880**	26956.00	27029	27007	27129.90	27196	2621.14
600	0.1	0.5	9	**26244**	26374.90	26442	26352	26501.40	26600	2860.93
600	0.2	0.1	3	14166	14171.30	14180	**14154**	14172.20	14178	262.04
600	0.2	0.1	5	13976	13996.50	14030	**13969**	14007.40	14053	626.76
600	0.2	0.1	7	**13812**	13830.10	13862	13830	13866.80	13894	1196.27
600	0.2	0.1	9	**13662**	13683.90	13707	13674	13711.30	13758	2054.11
600	0.2	0.3	3	**42106**	42129.40	42160	42115	42149.60	42189	751.15
600	0.2	0.3	5	**41340**	41395.50	41442	41425	41464.50	41531	2294.29
600	0.2	0.3	7	**40801**	40856.90	40912	40951	41016.00	41092	4377.53
600	0.2	0.3	9	**40406**	40450.90	40490	40584	40716.60	40825	5831.76
600	0.2	0.5	3	**62367**	62455.70	62532	62487	62603.90	62704	2301.21
600	0.2	0.5	5	**60041**	60203.50	60337	60316	60483.90	60597	2740.62
600	0.2	0.5	7	**58939**	59089.90	59212	59165	59296.90	59363	3095.99
600	0.2	0.5	9	**58124**	58245.80	58399	58270	58433.80	58576	3159.25
600	0.5	0.1	3	**35888**	35897.40	35910	35893	35911.10	35920	265.25
600	0.5	0.1	5	**35748**	35764.60	35779	35784	35799.60	35814	574.68
600	0.5	0.1	7	**35614**	35649.50	35677	35659	35678.20	35701	874.83
600	0.5	0.1	9	**35493**	35527.30	35551	35543	35580.60	35608	1415.54
600	0.5	0.3	3	106944	106962.10	106982	**106940**	106989.50	107042	664.15
600	0.5	0.3	5	**106282**	106328.70	106368	106297	106375.70	106493	1943.99
600	0.5	0.3	7	**105701**	105777.80	105879	105767	105879.30	106002	3598.85
600	0.5	0.3	9	**105256**	105341.80	105417	105298	105424.20	105562	5084.31
600	0.5	0.5	3	**165124**	165197.50	165260	165337	165464.60	165612	3580.68
600	0.5	0.5	5	**161478**	161564.70	161669	161907	162056.50	162160	3874.33
600	0.5	0.5	7	**159537**	159707.10	159876	159824	160116.70	160347	3929.42
600	0.5	0.5	9	**158301**	158384.50	158506	158513	158763.00	159063	4083.18
600	0.8	0.1	3	57334	57346.90	57357	**57333**	57352.90	57369	166.32
600	0.8	0.1	5	**57208**	57225.60	57248	57231	57242.50	57251	352.95
600	0.8	0.1	7	57044	57075.50	57106	57044	57099.50	57179	524.58
600	0.8	0.1	9	**56968**	56990.30	57006	56971	57003.20	57061	852.47
600	0.8	0.3	3	**172024**	172049.90	172084	172064	172094.00	172142	575.18
600	0.8	0.3	5	171515	171559.10	171605	**171459**	171590.80	171695	1375.56
600	0.8	0.3	7	**170996**	171088.70	171165	171044	171165.60	171259	2608.47
600	0.8	0.3	9	**170559**	170648.70	170738	170651	170742.30	170905	4124.47
600	0.8	0.5	3	**269158**	269304.00	269465	269353	269646.40	269850	4247.83
600	0.8	0.5	5	**264210**	264560.20	264822	265003	265276.20	265560	4722.79
600	0.8	0.5	7	**261978**	262158.00	262388	262582	262903.30	263149	4998.81
600	0.8	0.5	9	**260310**	260584.20	260937	260654	260983.80	261283	4755.96

Table 14: Best solution of all experiments.

$\|V\|$	d	d^-	k	$RI(P)$
100	0.1	0.1	7	64
100	0.1	0.1	9	44
100	0.1	0.3	5	319
100	0.1	0.3	7	265
100	0.1	0.3	9	220
100	0.1	0.5	5	421
100	0.1	0.5	7	350
100	0.1	0.5	9	297
100	0.2	0.1	7	250
100	0.2	0.1	9	223
100	0.2	0.3	7	832
100	0.2	0.3	9	772
100	0.2	0.5	7	1047
100	0.5	0.1	7	863
100	0.5	0.3	7	2526
100	0.5	0.3	9	2455
100	0.5	0.5	7	3486
100	0.5	0.5	9	3326
100	0.8	0.1	7	1468
100	0.8	0.3	7	4356
100	0.8	0.5	7	6089
100	0.8	0.5	9	5882
200	0.1	0.1	3	689
200	0.1	0.1	9	492
200	0.1	0.3	7	1656
200	0.1	0.5	5	2397
200	0.1	0.5	9	2092
200	0.2	0.1	7	1354
200	0.2	0.3	5	4133
200	0.2	0.5	5	5686
200	0.2	0.5	9	5212
200	0.5	0.3	9	11012
200	0.5	0.5	7	15944
200	0.8	0.5	5	27326
400	0.1	0.1	7	2666
400	0.1	0.1	9	2566
400	0.1	0.3	5	8356
400	0.1	0.3	7	8003
400	0.1	0.5	3	12506
400	0.1	0.5	5	11597
400	0.2	0.1	7	6014
400	0.2	0.1	9	5900
400	0.2	0.3	3	18265
400	0.2	0.3	5	17715
400	0.2	0.3	9	17127
400	0.2	0.5	3	26860
400	0.2	0.5	5	25572
400	0.2	0.5	7	24909
400	0.2	0.5	9	24460
400	0.5	0.1	5	15756
400	0.5	0.3	3	47475
400	0.5	0.3	9	46291
400	0.5	0.5	5	69762
400	0.5	0.5	9	68037
400	0.8	0.1	5	25186
400	0.8	0.3	7	75216
400	0.8	0.3	9	74907

Table 14 – *Continued from previous page*

$\|V\|$	d	d^-	k	$RI(P)$
600	0.1	0.1	5	6612
600	0.1	0.3	5	19649
600	0.1	0.3	7	19145
600	0.1	0.5	3	29275
600	0.2	0.1	7	13810
600	0.2	0.1	9	13652
600	0.2	0.3	3	42100
600	0.2	0.3	5	41333
600	0.2	0.3	7	40755
600	0.2	0.3	9	40391
600	0.2	0.5	3	62337
600	0.2	0.5	9	58122
600	0.5	0.1	3	35886
600	0.5	0.3	3	106938
600	0.5	0.3	5	106272
600	0.5	0.3	9	105211
600	0.5	0.5	3	165046
600	0.5	0.5	5	161377
600	0.5	0.5	9	158300
600	0.8	0.1	9	56940
600	0.8	0.3	7	170971

B Detailed results for the SRCC instances

Table 15: Symmetric relaxed imbalance obtained by ILS_{RCC} and ILS Levorato et al. [36].

Instance	$\|V\|$	d	d^-	k	ILS_{RCC}			ILS Levorato et al. [36]			t_{avg}
					min	avg	max	min	avg	max	
UNGA-1946	54	0.484	0.27	2	9.338	9.338	9.338	9.338	9.338	9.338	0.5
UNGA-1947	57	0.490	0.42	3	18.698	18.697	18.698	18.698	18.697	18.698	0.7
UNGA-1948	59	0.494	0.34	4	4.399	4.399	4.399	4.399	4.399	4.399	1.2
UNGA-1949	59	0.496	0.28	2	37.748	37.748	37.748	37.748	37.748	37.748	0.6
UNGA-1950	60	0.496	0.25	2	25.028	25.028	25.028	25.028	25.028	25.028	0.6
UNGA-1951	60	0.490	0.37	2	58.960	58.960	58.960	58.960	58.960	58.960	0.8
UNGA-1952	60	0.495	0.26	2	46.099	46.099	46.099	46.099	46.099	46.099	0.6
UNGA-1953	60	0.488	0.34	2	31.288	31.288	31.288	31.288	31.288	31.288	0.8
UNGA-1954	60	0.492	0.30	2	32.823	32.823	32.823	32.823	32.823	32.823	0.8
UNGA-1955	65	0.464	0.11	4	5.377	5.377	5.377	5.377	5.377	5.377	1.5
UNGA-1956	81	0.480	0.30	4	17.181	17.181	17.181	17.181	17.181	17.181	2.3
UNGA-1957	82	0.495	0.32	3	37.512	37.512	37.512	37.512	37.512	37.512	2.0
UNGA-1958	82	0.489	0.25	2	122.536	122.536	122.536	122.536	122.536	122.536	1.2
UNGA-1959	82	0.497	0.35	2	102.881	102.881	102.881	102.881	102.881	102.881	2.2
UNGA-1960	100	0.488	0.39	3	45.464	45.464	45.464	45.464	45.464	45.464	3.4
UNGA-1961	106	0.467	0.35	3	37.395	37.395	37.395	37.395	37.395	37.395	3.9
UNGA-1962	110	0.468	0.33	2	154.412	154.412	154.412	154.412	154.412	154.412	4.4
UNGA-1963	113	0.490	0.18	4	20.639	20.639	20.639	20.639	20.639	20.639	3.7
UNGA-1964[1]	115	0.500	0.28	3	0.000	0.000	0.000	0.000	31.200	39.000	1.0
UNGA-1965	117	0.495	0.21	4	29.482	29.482	29.482	29.482	29.482	29.482	6.3
UNGA-1966	122	0.484	0.23	2	213.680	213.680	213.680	213.680	213.680	213.680	2.3
UNGA-1967	124	0.490	0.29	4	42.298	42.298	42.298	42.298	42.298	42.298	7.4
UNGA-1968	126	0.490	0.25	3	86.239	86.239	86.239	86.239	86.239	86.239	6.1
UNGA-1969	126	0.495	0.21	3	66.277	66.277	66.277	66.277	66.277	66.277	4.9
UNGA-1970	127	0.497	0.21	3	69.316	69.316	69.316	69.316	69.316	69.316	4.9
UNGA-1971	133	0.493	0.09	4	19.306	19.306	19.306	19.306	19.306	19.306	4.8
UNGA-1972	132	0.499	0.04	2	16.294	16.294	16.294	16.294	16.294	16.294	1.4
UNGA-1973	135	0.499	0.09	3	14.142	14.142	14.142	14.142	14.142	14.142	2.5
UNGA-1974	138	0.499	0.10	3	18.608	18.608	18.608	18.608	18.608	18.608	3.1
UNGA-1975	143	0.472	0.21	4	53.707	53.707	53.707	53.707	53.707	53.707	6.6
UNGA-1976	144	0.460	0.16	4	34.606	34.606	34.606	34.606	34.606	34.606	5.6
UNGA-1977	146	0.465	0.09	6	15.548	15.548	15.548	15.548	15.548	15.548	12.7
UNGA-1978	148	0.483	0.14	3	**74.755**	74.755	74.755	75.445	76.629	77.812	3.1
UNGA-1979	150	0.470	0.16	5	21.520	21.520	21.520	21.520	21.520	21.520	9.6
UNGA-1980	151	0.474	0.18	6	29.303	29.303	29.303	29.303	30.054	31.807	11.7
UNGA-1981	155	0.477	0.18	5	29.121	29.121	29.121	29.121	29.121	29.121	11.3
UNGA-1982	156	0.432	0.15	4	31.378	31.378	31.378	31.378	31.378	31.378	9.8
UNGA-1983	157	0.474	0.22	4	29.523	29.523	29.523	29.523	36.168	62.750	7.4
UNGA-1984	158	0.439	0.20	4	14.033	14.033	14.033	14.033	17.197	45.679	6.9
UNGA-1985	158	0.431	0.12	2	53.630	53.630	53.630	53.630	53.630	53.630	2.9
UNGA-1986	158	0.499	0.08	2	44.673	44.673	44.673	44.673	44.673	44.673	2.4
UNGA-1987	158	0.499	0.05	3	9.119	9.119	9.119	9.119	9.119	9.119	2.8
UNGA-1988	158	0.499	0.08	2	33.905	33.905	33.905	33.905	33.905	33.905	2.5
UNGA-1989	158	0.500	0.05	2	17.302	17.302	17.302	17.302	17.302	17.302	2.3
UNGA-1990	158	0.498	0.10	3	15.024	15.024	15.024	15.024	15.024	15.024	3.5
UNGA-1991	178	0.467	0.10	3	15.480	15.480	15.480	15.480	15.569	15.692	3.6
UNGA-1992	180	0.493	0.08	4	12.201	12.201	12.201	12.201	12.770	17.889	8.0
UNGA-1993	184	0.496	0.09	3	24.689	24.972	25.003	24.689	24.689	24.689	4.4
UNGA-1994	185	0.497	0.13	3	23.573	23.573	23.573	23.573	23.573	23.573	5.5
UNGA-1995	185	0.489	0.11	3	27.624	27.624	27.624	27.624	28.248	28.663	5.1
UNGA-1996	185	0.499	0.07	3	9.541	9.541	9.541	9.541	9.541	9.541	5.6
UNGA-1997	176	0.471	0.16	5	27.018	27.018	27.018	27.018	27.018	27.018	13.0
UNGA-1998	177	0.492	0.14	4	39.300	39.300	39.300	39.300	39.300	39.300	11.2
UNGA-1999	182	0.487	0.10	4	14.375	14.375	14.375	14.375	14.386	14.412	9.0
UNGA-2000	189	0.495	0.13	4	25.099	25.100	25.099	25.099	25.100	25.099	8.6
UNGA-2001	191	0.495	0.16	2	33.531	33.531	33.531	33.531	33.531	33.531	6.8
UNGA-2002	192	0.495	0.10	3	12.687	12.687	12.687	12.687	12.687	12.687	4.1

[1] In the 19th session, voting occurred on only one resolution which explains the signed digraph with very low relaxed imbalance.

Table 15 – *Continued from previous page*

Instance	$\|V\|$	d	d^-	k	ILS_{RCC}			ILS Levorato et al. [36]			t_{avg}
					min	avg	max	min	avg	max	
UNGA-2003	191	0.489	0.06	2	7.466	7.466	7.466	7.466	7.466	7.466	4.2
UNGA-2004	191	0.498	0.05	2	20.638	20.638	20.638	20.638	20.638	20.638	3.6
UNGA-2005	192	0.482	0.06	3	25.516	25.516	25.516	25.516	25.516	25.516	6.0
UNGA-2006	192	0.498	0.05	2	28.954	28.955	28.954	28.954	28.955	28.954	3.6
UNGA-2007	192	0.498	0.06	2	45.570	45.570	45.570	45.570	45.570	45.570	3.7
UNGA-2008	192	0.495	0.06	2	36.889	36.889	36.889	36.889	36.889	36.889	3.8
Slashdot1	200	0.022	0.07	5	**11.0**	11.700	12.0	16.0	17.30	19.0	16.9
Slashdot2	300	0.012	0.08	8	**4.0**	4.600	5.0	8.0	11.10	12.0	61.1
Slashdot3	400	0.008	0.07	4	**15.0**	16.100	18.0	20.0	22.20	24.0	67.7
Slashdot4	600	0.005	0.08	9	**8.0**	10.700	13.0	16.0	19.20	21.0	438.5
Slashdot5	800	0.005	0.11	20	**14.0**	16.300	20.0	32.0	35.60	41.0	2161.5
Slashdot6	1000	0.006	0.14	11	**182.0**	187.200	194.0	206.0	211.10	218.0	6107.6
Slashdot7	2000	0.005	0.15	43	**626.0**	650.700	686.0	712.0	751.90	777.0	7200.0
BR-2010-v1	545	0.490	0.01	4	**309.692**	309.692	309.692	316.091	375.860	596.137	135.3
BR-2010-v2	545	0.490	0.02	4	**332.493**	332.493	332.493	338.836	343.849	345.102	160.9
BR-2011-v1	553	0.488	0.20	4	562.540	562.541	562.541	562.540	584.250	589.677	534.2
BR-2011-v2	553	0.486	0.21	4	566.950	566.950	566.950	566.950	594.462	601.993	517.5
BR-2012-v1	555	0.489	0.04	4	**658.613**	658.613	658.613	672.986	755.450	1072.030	202.8
BR-2012-v2	555	0.488	0.04	4	**681.168**	681.168	681.168	697.567	703.277	704.705	202.2
BR-2013-v1	540	0.489	0.04	4	483.511	503.611	514.011	483.511	757.230	1266.900	117.1
BR-2013-v2	540	0.489	0.04	4	**504.918**	522.200	539.912	631.341	692.390	732.912	115.6
BR-2014-v1	556	0.496	0.01	4	**130.020**	131.001	131.968	131.973	136.706	168.493	61.3
BR-2014-v2	556	0.495	0.01	4	**137.165**	137.629	138.326	140.349	148.078	182.796	65.4
BR-2015-v1	552	0.486	0.12	4	536.724	536.724	536.724	536.724	676.785	1147.860	271.7
BR-2015-v2	552	0.484	0.18	4	585.038	585.038	585.038	585.038	707.996	830.955	294.7
BR-2016-v1	544	0.484	0.10	4	1241.520	1241.520	1241.520	1241.520	1411.645	1610.620	291.9
BR-2016-v2	544	0.482	0.11	4	**1285.950**	1285.950	1285.950	1377.020	1511.614	1677.130	287.7