The maximum balanced subgraph of a signed graph: applications and solution approaches

Rosa Figueiredo, Yuri Y. Frota

The maximum balanced subgraph of a signed graph: applications and solution approaches

Rosa Figueiredo[a,*], Yuri Frota[b]

[a] *CIDMA, Department of Mathematics, University of Aveiro*
3810-193 Aveiro, Portugal.
rosa.figueiredo@ua.pt
[b] *Department of Computer Science, Fluminense Federal University*
24210-240 Niterói-RJ, Brazil.
yuri@ic.uff.br

Abstract

The Maximum Balanced Subgraph Problem (MBSP) is the problem of finding a subgraph of a signed graph that is balanced and maximizes the cardinality of its vertex set. This paper is the first one to discuss applications of the MBSP arising in three different research areas: the detection of embedded structures, portfolio analysis in risk management and community structure. The efficient solution of the MBSP is also in the focus of this paper. We discuss pre-processing routines and heuristic solution approaches to the problem. a GRASP metaheuristic is developed and improved versions of a greedy heuristic are discussed. Extensive computational experiments are carried out on a set of instances from the applications previously mentioned as well as on a set of random instances.
Keywords: Combinatorial optimization; Balanced signed graph; Heuristics; Portfolio analysis; Community structure.

*Corresponding author. Fax and Telephone number: +351 234370066. Email: rosa.figueiredo@ua.pt
Rosa Figueiredo is supported by FEDER founds through COMPETE-Operational Programme Factors of Competitiveness and by Portuguese founds through the CIDMA (University of Aveiro) and FCT, within project PEst-C/MAT/UI4106/2011 with COMPETE number FCOMP-01-0124-FEDER-022690.

1. Introduction

Let $G = (V, E)$ be an undirected graph where $V = \{1, 2, \ldots, n\}$ is the set of vertices and E is the set of edges connecting pairs of vertices. Consider a function $s : E \to \{+, -\}$ that assigns a sign to each edge in E. An undirected graph G together with a function s is called a *signed graph*. An edge $e \in E$ is called *negative* if $s(e) = -$ and *positive* if $s(e) = +$.

Signed graphs were introduced by Heider [23] with the purpose of describing sentiment relations between people pertaining to a same social group and to provide a systematic statement of social balance theory. Cartwright *et al.* [10] formalized Heider's theory stating that a balanced social group, i.e., a balanced signed graph, could be partitioned into two mutually hostile subgroups each having internal solidarity. In the last decades, signed graphs continued to be a very attractive discrete structure for social network researchers [1, 13, 14, 26, 36] but also for researchers in other scientific areas, including portfolio analysis in risk management [22, 24], biological systems [12, 24], efficient document classification [5], detection of embedded matrix structures [19] and community structure [27, 34], a very prominent area of network science [30]. The common element among these applications is the fact that all of them are defined in a collaborative vs. conflicting environment that can be modeled over a signed graph.

Let $G = (V, E, s)$ denote a signed graph and let E^- and E^+ denote, respectively, the set of negative and positive edges in G. Also, for a vertex set $S \subseteq V$, let $E[S] = \{(i, j) \in E \mid i, j \in S\}$ denote the subset of edges induced by S. A signed graph $G = (V, E, s)$ is *balanced* if its vertex set can be partitioned into sets W (possibly empty) and $V \setminus W$ in such a way that $E[W] \cup E[V \setminus W] = E^+$. The problem that is studied herein can be stated as follows.

Problem 1.1 (MBSP)**.** Let $G = (V, E, s)$ be a signed graph. The *Maximum balanced subgraph problem* is the problem of finding a subgraph $H = (V', E', s)$ of G such that H is balanced and maximizes the cardinality of V'.

The MBSP is known to be an NP-hard problem [7] although the problem of detecting balance in signed graphs can be solved in polynomial time [21]. In the literature, the MBSP has already been applied in the detection of embedded matrix structures [18, 19] and in portfolio analysis in risk management [22].

Gulpinar *et al.* [19] showed that the problem of detecting a maximum embedded reflected network (DMERN) can be reduced to the MBSP. The existing solution approaches to the MBSP were in fact proposed for the solution of the DMERN problem. The literature proposes various heuristics for the solution of the DMERN problem (for references see [19]). In [19], the first signed graph solution approach was proposed for this problem: a greedy heuristic which is able to find the optimal solution whenever the whole matrix is a reflected network matrix. Lately, Figueiredo *et al.* [18] developed the first exact approach for the DMERN problem based on the signed graph reformulation of Gulpinar *et al.* Computational experiments were carried out over a set of instances found in the literature as a test set for the DMERN problem. Almost all these instances were

solved to optimality in a few seconds showing that they were not appropriate for assessing the quality of a heuristic approach to the problem and that more difficult benchmark instances should be provided.

The notion of balance for signed graphs in the context of portfolio analysis was introduced by Harary *et al.* [22]. They showed that a portfolio characterized by a signed graph that is balanced is predictable, in the sense that the structure of the balanced signed graph allows investors to predict the risk of the associated portfolio. These authors used small samples of signed graph structures (up to five vertices) to illustrate how the structural balance and portfolio risk management are linked. Their conclusions were only illustrated by a small real example with four assets.

Another balance subgraph problem defined on signed graphs is studied in [24] from the point of view of edge deletions. The authors proposed a new data reduction scheme and a fixed-parameter algorithm for this problem. Computational experiments were carried out over randomly generated signed graphs and over signed graphs representing financial networks and gene regulatory networks. Their algorithms were used over signed graphs of up to several hundred vertices.

Our contributions are two-fold. First, we group and discuss three applications, coming from different research areas, that can be solved as instances of the MBSP. In doing so we provide a new set of benchmark instances of the MBSP, including a set of difficult instances for the DMERN problem. Second, we contribute to the efficient solution of the MBSP by developing a pre-processing routine, an efficient GRASP metaheuristic, and improved versions of the greedy heuristic proposed in [19].

The remainder of the paper is structured as follows. In Section 2 we discuss three applications of the MBSP arising in different research areas. The reduction rules that compose the pre-processing procedure are described in Section 3. The integer programming formulation and the branch-and-cut algorithm presented in [17, 18] to the MBSP is outlined in Section 4. In Section 5, we present two heuristic approaches to solve the MBSP. First, we describe the greedy heuristic proposed in [19] for the MBSP and propose new strategies to be used in the first step of this heuristic. Then, we develop a GRASP heuristic for the problem. In Section 6, computational results are reported for random instances as well as for instances of the three applications previously described. In Section 7 we present concluding remarks.

We next give some notations and definitions to be used throughout the paper. For an edge set $B \subseteq E$, let $G[B]$ denote the subgraph of G induced by B. Also, for a vertex set $S \subseteq V$, we define $\delta(S) = \{(i,j) \in E \mid i \in S,\ j \in V \setminus S\}$ and $N(S) = \{j \in V \mid (i,j) \in \delta(S)\}$. A set $I \subseteq V$ is called a *stable set* if no pair of vertices in I is joined by an edge. We represent a cycle by its vertex set $C \subseteq V$. In this text, a signed graph is allowed to have parallel edges but no loops. Also, we assume that parallel edges have always opposite signs. We define $G^- = (V, E^-)$ and, for a vertex set $S \subseteq V$, we define $N^-(S) = \{j \in V \mid (i,j) \in \delta(S) \cap E^-\}$, $N^+(S) = \{j \in V \mid (i,j) \in \delta(S) \cap E^+\}$ and G^S the signed graph obtained from G by changing the signs of each edge in $\delta(S)$. We refer the

reader to [37] for a bibliography of signed graphs.

2. Applications

2.1. Detecting embedded matrices

The knowledge of a special structure in a matrix defining a linear or an integer programming problem can be used to solve it in an efficient way. One of these special structures is a network matrix. It is well known [8] that if the constraint matrix of a linear programming problem is a network matrix, then we can use the network simplex algorithm to solve this problem more efficiently.

A matrix B is called a *network matrix* if the elements of B belong to the set {-1,0,+1} and, additionally, if every column of B contains at most one element $+1$ and at most one element -1. Given a row of matrix B, the operation of changing the signs of all non-zero row elements is called a *reflection* of this row. A matrix B is called a *reflected network matrix* if there exists a set of row reflections that transforms matrix B into a network matrix.

Consider a $\{-1, 0, +1\}$-matrix $A = [a_{ik}]$ with n rows. Two rows of matrix A are said to be in conflict if they both have a $+1$ or they both have a -1 in the same column. A signed graph $G(A)$ can be used to represent existing conflicts in A [19]. The vertex set of $G(A)$ is defined as $V = \{1, \ldots, n\}$ and the set of negative and positive edges of $G(A)$ are defined as follows: an edge $(i, j) \in E^-$ if and only if $a_{ik} = a_{jk} \neq 0$ for some column k of matrix A; an edge $(i, j) \in E^+$ if and only if $a_{ik} = -a_{jk} \neq 0$ for some column k of matrix A. In [19], Gulpinar *et al.* showed that the problem of detecting a maximum embedded reflected network (DMENR) of A is equivalent of finding the maximum balanced subgraph of $G(A)$. Figure 1 illustrates this result.

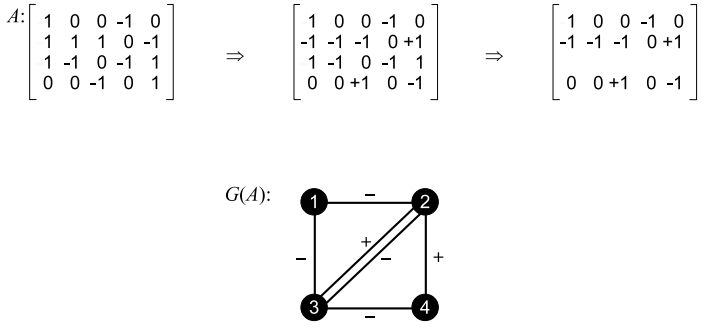

Figure 1: Existing conflicts (negative edges) in matrix A are represented in signed graph $G(A)$, as well as possible conflicts (positive edges) after row reflections. The subgraph of $G(A)$ induced by $\{1, 2, 4\}$ is a maximum balanced subgraph of $G(A)$ associated with a maximum embedded reflected network in A.

Notice that, according to the definition of $G(A)$, matrices of different dimensions can yield the same signed graph and very large matrices can define very sparse signed graphs. Thus, signed graphs are an appropriate discrete

structure to model and to solve the DMERN problem. The first signed graph based solution approach to the DMERN problem was described in [19]. This greedy heuristic approach is described in Section 5. In [18], Figueiredo *et al.* introduced an integer programming formulation to this problem in which they explore the relations between network matrices and signed graphs. Based on this formulation, they proposed a branch-and-cut method to solve the MBSP to optimality. An improved version of this algorithm is described in Section 4. In the computational experiments presented in [18], most of the instances found in the literature were solved to optimality in a few seconds attesting that the instances commonly used as benchmark for the DMERN problem are not appropriate for assessing the quality of a heuristic approach to the problem.

2.2. Portfolio analysis in risk management

A portfolio is a collection of securities (assets) held by an investor. Balancing on signed graphs is used in [22] to define a risk-limiting strategy for portfolio definition. In this context, each security is represented by a vertex in the signed graph while the correlation between securities is represented by the set of signed edges. A balanced signed graph with only positive edges represents a speculative portfolio since all its securities tend to move in the same direction, either on the upside or on the downside. On the other hand, a balanced signed graph with at least one negative edge is associated with a limited risk portfolio. Such a portfolio is defined by two sets of securities, each set with a tendency to move in tandem, while some pair of assets (connected by negative edges) tend to move in opposite directions providing the investors with a hedging guarantee. According to [22], an unbalanced signed graph represents an unpredictable portfolio.

With the purpose to illustrate this signed graph approach to portfolio analysis, the authors described a simplistic procedure that involves only complete graphs. The procedure is a constructive one, starting with two securities. Roughly speaking, at each iteration two possible actions could be followed: to switch at least one of the securities or to add a new security to the graph. A small case study with only four vertices was presented and discussed.

In [24], another version of the MBSP is studied where edge-deletions rather than vertex deletions are considered. The authors proposed and implemented a very efficient pre-processing routine as well as a fixed-parameter algorithm to solve this edge-deletion variant of the MBSP. They showed that their algorithms can be used to compute the balancedness of financial networks of up to several hundred vertices. However, we question the meaning of edge deletions in the portfolio analysis application since the correlation between securities cannot be controlled by an investor. On the other hand, the efficient solution of the MBSP defined over a signed graph representing a big set of securities would give a more efficient way to define a limited risk portfolio.

2.3. Community structure

As we have mentioned before, signed graphs have shown to be a very attractive structure for social network researchers [1, 13, 14, 26, 36]. Balancing

and clustering problems defined on signed graphs arise naturally in the study of community structures [3, 27, 30, 34, 36]. In this context, each vertex in the signed graph represents a person in a social group while an edge represents a sentiment relation between two people (mutual liking or disliking, friendship or enmity, cooperation or defection, interaction or avoidance) or says if their attitudes toward an object match.

No matter the measure of balance, social groups are rarely balanced. One big challenge in this area is to evaluate balance in a social network. Once a measure of balance is defined, it can be used as a tool to study whether and how the network evolves to a possible balance state. Interesting questions arise: Could we cluster a social group according their preferences? Which is the minimum number of relations that should change in a group in order to obtain a balanced network? The first question was extensively studied in the literature and continue to be an interesting research topic (see [30] for references). The identified clusters, or communities, are cohesive groups corresponding to circles of friends, business partners or groups playing a similar role or having a similar political position.

The solution of the MBSP is related with another relevant question: Which is the biggest balanced subgroup in a social network? In answering this question, we identify two dominant and opposite communities in the network. This topic has already appeared in the investigation of community structure of networks determined by common voting [27], where different representations of the United Nations General Assembly (UNGA) voting records was proposed. For each network representation, the authors looked for the network partition that better defined the main voting groups, which allowed the identification of the majority ones. The next three sections are dedicated to the description of solution approaches to the MBSP.

3. Pre-processing

Polynomial-time data reduction is a strategy extensively used to deal with very large instances of difficult problems [24, 35]. In the following, we describe very simple reduction rules to the MBSP similar to the ones that have been used in the solution of vertex coloring problems. Consider a signed graph $G = (V, E, s)$.

(1) Let $i \in V$ be a vertex such that $N(i) = \emptyset$. Define $G' = (V \setminus \{i\}, E, s)$, solve the problem over G' and let subgraph $H' = (V', E', s)$ be the obtained optimal solution. An optimal solution for G is given by $H = (V' \cup \{i\}, E', s)$.

(2) Let $i \in V$ be a vertex such that $N^-(i) = \emptyset$, $N^+(i) \neq \emptyset$ and, for each pair $u, v \in N^+(i)$, $(u, v) \in E^+$. Define $G' = [V \setminus \{i\}]$ and solve the problem over G'. Let $H' = (V', E', s)$ be the obtained optimal solution. An optimal solution for the problem defined over G is given by $H = (V' \cup \{i\}, E[V' \cup \{i\}], s)$.

(3) Let $G = (V, E, s)$ be a graph such that $V = V^1 \cup V^2$ and $E = E[V^1] \cup E[V^2] \cup \{(i, j)\}$, with $i \in V^1$, $j \in V^2$ and $(i, j) \notin E^+ \cap E^-$. Define $G^1 =$

$(V^1, E[V^1])$ and $G^2 = (V^2, E[V^2])$. Solve the problem, independently, over graphs G^1 and G^2. Let $H' = (V', E', s)$ and $H'' = (V'', E'', s)$ be the obtained optimal solutions. An optimal solution for the problem defined over G is given by the subgraph $H = (V' \cup V'', E' \cup E'', s)$.

(4) Let $i, j \in V$ be two vertices in G such that $N^+(i) = N^+(j)$, $N^-(i) = N^-(j)$ and $(i, j) \in E^- \cap E^+$. Define $G' = G[V \setminus \{j\}]$. Solve the problem over G' and let $H' = (V', E', s)$ be the optimal solution. This subgraph is also an optimal solution for the problem defined over G.

Reduction rules (1), (2), (3) and (4) can be applied repeatedly until we obtain a graph G that cannot be reduced by any of them.

Besides the rules mentioned above, we implemented a general reduction procedure based on the work of Hüffner *et al.* [24] for the edge-deletion version of the MBSP. The procedure proposed in [24] looks for a small set of vertices S such that removing S from G cuts off a small vertex set C from the rest of the graph. The vertex set S is called a *separator*. The main idea is to enumerate all the possible states concerning the vertices in S and, for each possible state, to determine the size of an optimal solution for the induced subgraph $G[S \cup C]$. Finally, the subgraph $G[S \cup C]$ is replaced in G by a smaller equivalent subgraph such that the value of the optimal solution for the original graph is kept. The definition of this equivalent subgraph depends on the problem definition. In [24], separators with $|S| \leq 4$ and $|C| \leq 32$ were heuristically generated. This reduction procedure applied to the MBSP is illustrated in Figure 2 and explained in the following.

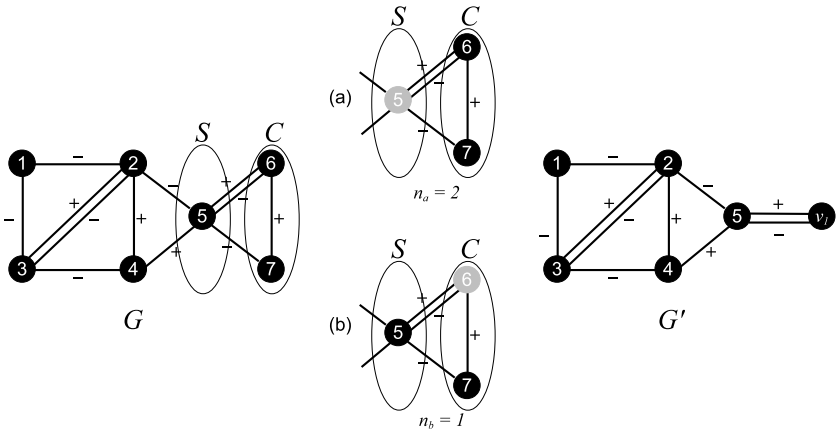

Figure 2: Reduction rule $S1$ applied to signed graph G. For any feasible solution, we have two possible states concerning the vertex set S: (a) vertex 5 does not belong to the feasible solution; (b) vertex 5 belongs to the feasible solution. The optimal solution for the MBSP defined over $G[\{5, 6, 7\}]$ has $n_a = 2$ vertices from C in case (a) and $n_b = 1$ in case (b). Signed graph G' is defined such that $v(G) = v(G') + 1$.

We generate separators for the MBSP such that $|S| = 1$ and $|C| \leq 15$. A separator with $|S| = 1$ can be generated polynomially by searching for a 1-

cut (articulation vertex) in the graph. Next, we describe how the equivalent subgraph is defined in our case.

Let $S = \{i\}$ be an articulation vertex. For any feasible solution of the MBSP, we can have two possibilities concerning vertex i: (a) vertex i does not belong to the solution; (b) vertex i belongs to the solution. Let n_a and n_b be the number of vertices from C in an optimal solution for $G[S \cup C]$ if case (a), respectively, case (b), happens. Notice that $n_a \geq n_b$ and that, no matter if vertex i belongs or not to the optimal solution, the vertex set C contributes with at least n_b vertices to the value of an optimal solution for G. Then, we proceed in the following way. Record value n_b; define a vertex set $V' = \{i, v_1, \ldots, v_{n_a - n_b}\}$; define a set of parallel edges $E' = \{(i, v) \mid \forall\ v \in V' \setminus \{i\}\}$; and define a new graph G' by replacing $G[S \cup C]$ in G by (V', E'). Let $v(G)$ be the value of the optimal solution of the MBSP defined over a signed graph G. Then, $v(G) = v(G') + n_b$. Henceforth, we will denote this reduction rule as S1.

4. Exact solution approach

In this section, we describe briefly an integer linear programming formulation and a branch-and-cut algorithm recently proposed for the MBSP [17, 18]. This exact algorithm was used in the computational experiments presented in Section 6.

4.1. Integer programming formulation

A signed graph is balanced if and only if it does not contain a parallel edge or a cycle with an odd number of negative edges [6]. Let $C^o(E)$ be the set of all odd negative cycles in G, i.e., cycles with no parallel edges and with an odd number of negative edges. From now on, a cycle $C \in C^o(E)$ is called an *odd negative cycle*. The formulation uses binary decision variables $y \in \{0,1\}^{|V|}$ defined in the following way. For all $i \in V$, y_i is equal to 1 if vertex $i \in V$ belongs to the balanced subgraph, and is equal to 0 otherwise. We use the vector notation $y = (y_i)$, $i \in V$, and the notation $y(V') = \sum_{i \in V'} y_i$ for $V' \subseteq V$. The formulation follows.

$$\text{Maximize } y(V) \tag{1}$$

$$\text{subject to } y_i + y_j \leq 1, \qquad \forall\ (i,j) \in E^- \cap E^+, \tag{2}$$

$$y(C) \leq |C| - 1, \qquad \forall\ C \in C^o(E), \tag{3}$$

$$y_i \in \{0,1\}, \qquad \forall\ i \in V. \tag{4}$$

Consider a parallel edge $(i,j) \in E^- \cap E^+$. Constraints (2) ensure vertices i and j cannot belong together to the balanced subgraph. Constraints (3), called *odd negative cycle inequalities*, forbid cycles with an odd number of negative edges in the subgraph described by variables y. These constraints force variables y to define a balanced subgraph while the objective function (1) looks for a maximum balanced subgraph. The formulation has n variables and, due to constraints (3), might have an exponential number of constraints.

4.2. The branch-and-cut code

The branch-and-cut algorithm developed in [17, 18] has three basic components: the initial formulation, the cut generation and the primal heuristic. The following improvements were added to the version discussed in [17]: a new branching rule and new separation routines. Next, we describe the main ingredients of this improved version, used in the computational experiments described in Section 6. For a detailed description we refer the reader to the original works [17, 18].

Initial formulation and primal heuristic. The initial formulation is composed by a set of clique inequalities that dominate constraints (2), a subset of odd negative cycle inequalities (3), a subset of clique inequalities introduced in [18] and by all the trivial inequalities $0 \leq y_i \leq 1$. A simple rounding heuristic is used every time a fractional solution is found in the branch-and-cut tree.

Separation routines. The cut generation component described in [18] has two separation procedures: an exact separation procedure for odd negative cycle inequalities (3) and a heuristic separation procedure for a family of clique inequalities introduced in [18]. These two separation procedures were also implemented in the improved version presented in [17]. Two new procedures were added to the cut generation component of the improved version. The authors in [17] have implemented the separation procedure described in [28] to the lifted odd hole inequalities defined over the set of parallel edges. Moreover, as indicated in [18], in order to strengthen constraints (3), they have implemented a lifting procedure to the odd negative cycle inequalities with $|C| \leq 20$. In both cases, the subproblems appearing in each iteration of the lifting procedure were solved by simple enumerative algorithms.

Branching on the odd negative cycle inequalities. A standard $0-1$ branching rule was implemented in the original version of the branch-and-cut algorithm [18]. The authors reported that a version of the branching rule proposed in [4] was also implemented but, although it has been successfully applied to the stable set problem [31, 32], better results were obtained to the MBSP with the standard $0-1$ branching rule. A branching rule based on the odd negative cycle inequalities (3) was implemented in [17] and has shown to be more efficient than the standard one. The intuition behind the cycle based branching rule proposed in [17] is the attempt to generate more balanced enumerative trees. The standard $0-1$ branching rule is very asymmetrical and produces unbalanced enumerative trees. The authors in [17] tried to reduce this effect by branching on odd negative cycles and got better computational results with this new branching rule.

5. Heuristic solution approaches

In this section, we describe two heuristic approaches to the MBSP: a greedy heuristic proposed in [19] for the DMERN problem and a GRASP metaheuristic introduced in this work.

5.1. GGMZ greedy heuristic

The heuristic proposed by Gulpinar *et al.* [19] is described in Algorithm 1 and is motivated by the following basic result.

Lemma 5.1. *[19] Every signed tree $T = (V_T, E_T)$ is a balanced graph.*

Algorithm 1: GGMZ GREEDY HEURISTIC.

Input: signed graph $G = (V, E, s)$
Output: stable set I
1 Find a spanning tree T in G;
2 Compute a subset $W \subseteq V$ such that T^W has no negative edges;
3 Find a maximal stable set I in the graph $(G^W)^-$;
4 return I;

It is not difficult to see that there always exists a partition of set I into two subsets that represents a feasible solution for the MBSP. The greedy heuristic proposed by Gulpinar *et al.* has the advantage to find the optimal solution of the MBSP whenever G is a balanced graph.

The first step of Algorithm 1 could be implemented in a number of different ways. In [20], the authors tried three well known strategies to construct a spanning tree: Breadth First Search (BFS), Depth First Search (DFS) and a random method. Following similar ideas, the Step 1 is implemented in this work as a BFS, a DFS and by using other five different spanning tree strategies. In these additional strategies, we calculate a minimum spanning tree based on costs generated for each edge e of the graph. The Kruskal algorithm [11] is used to calculate the minimum spanning tree and the following cost functions were designed:

- $(-,+-,+)$: $f_1(e) = \{3 \text{ if } e \in E^+ \setminus E^-;\ 1 \text{ if } e \in E^- \setminus E^+;\ 2 \text{ if } e \in (E^- \cap E^+)\}$;
- $(+,-,+-)$: $f_2(e) = \{1 \text{ if } e \in E^+ \setminus E^-;\ 2 \text{ if } e \in E^- \setminus E^+;\ 3 \text{ if } e \in (E^- \cap E^+)\}$;
- $(-,+,+-)$: $f_3(e) = \{2 \text{ if } e \in E^+ \setminus E^-;\ 1 \text{ if } e \in E^- \setminus E^+;\ 3 \text{ if } e \in (E^- \cap E^+)\}$;
- Random: a random function $f_{random}(e) \in [0, 1000]$;
- Adaptive: an adaptive function $f_{Adapt}(e) = \{f_1(e) \text{ if } |E^-|/|E^+| < 1;\ f_2(e) \text{ otherwise}\}$.

In Step 2, the subset W is found by using a recursive procedure based on the inductive proof of Lemma 5.1 (see [19] for more details). Finally, in Step 3, we replaced the degree-greedy algorithm [29] used in [19] and [20] by an efficient stable set GRASP [15]. We set the GRASP to halt after 100 iterations have been performed since the last time the best solution was updated or when a time limit of 300 seconds is reached.

5.2. A GRASP metaheuristic

GRASP heuristics have been successfully applied to a large number of combinatorial optimization problems [25]; see [16] for a detailed annotated bibliography. In order to generate better heuristic solutions for the MBSP, a GRASP heuristic is developed in this section. The GRASP heuristic is an iterative procedure that has two phases associated with each iteration: a *construction phase* and a *local search phase*. The construction phase finds an initial solution that later might be improved by the local search phase. Both phases are repeated until a termination criterion is satisfied. In the remainder of this section, we describe these two phases.

The input for the heuristic is the signed graph $G = (V, E, s)$. We assume that a feasible solution for the MBSP is defined as a pair of disjoint sets (V_1, V_2) such that $V_1, V_2 \subseteq V$, $E[V_1] \cup E[V_2] \subseteq E^+$ and $(E[V_1 \cup V_2] \backslash (E[V_1] \cup E[V_2])) \subseteq E^-$.

Construction phase. The overall method attempts to find a maximal feasible solution (V_1, V_2). The *construction phase* begins with $V_1 = V_2 = \emptyset$ and enlarges these sets in a greedy way by adding one new vertex to the solution (to V_1 or to V_2) at a time. Let (V_1, V_2) be a feasible solution. We define $Cand(V_1) = \{i \in V \backslash (V_1 \cup V_2) \mid (V_1 \cup \{i\}, V_2)$ is a feasible solution for the MBSP$\}$, as the set of vertices that are candidate to enter the set V_1. Similarly, we define $Cand(V_2) = \{i \in V \backslash (V_1 \cup V_2) \mid (V_1, V_2 \cup \{i\})$ is a feasible solution for the MBSP$\}$. At each iteration, a vertex $i \in Cand(V_w)$ (for $w = 1, 2$) is randomly selected and a new set V'_w is reached by inserting vertex i into the set V_w. The procedure is repeated until $Cand(V_1) = Cand(V_2) = \emptyset$.

Local search phase. Obviously, there is no guarantee that the construction phase returns a global optimal solution. Therefore, the solution (V_1, V_2) may be improved by a *local search* procedure. The neighborhood of the current solution (V_1, V_2) is defined as the family of all pairs (V'_1, V'_2) obtained by applying one of the following operations for $w \in \{1, 2\}$:

> **(i)** removing a vertex i from V_w and inserting a vertex $j \in (Cand(V_w \backslash \{i\}) \cup Cand(V_{w \mod 2}))$ into the corresponding subset (i.e. V_w if $j \in Cand(V_w \backslash \{i\})$ and $V_{w \mod 2}$ otherwise).
>
> **(ii)** removing two vertices i_1 and i_2 from V_w and inserting vertices j_1 and j_2 belonging to $(Cand(V_w \backslash \{i_1, i_2\}) \cup Cand(V_{w \mod 2}))$ into the corresponding subsets.

The procedure starts with the solution provided by the construction phase and iteratively replaces the current solution by the one with maximum cardinality within its neighborhood, halting when no better solution is found in that way. Similar to the stable set GRASP of Feo *et al.* [15], the method stops after 100 iterations have been performed or when a time limit of 300 seconds is reached.

6. Computational experiments

In this section, we report the results of extensive computational experiments carried out with the different procedures described in the previous sections. Our intention is to evaluate the performance of our heuristics and to provide the interested reader with a set of benchmark instances of each application described in Section 2. We also present numerical results obtained with a set of random instances. Specifically, for the DMERN problem, we intend to fill a gap pointed in the work of Figueiredo *et al.* [18]: we introduce a set of more difficult instances that we believe are more appropriate for assessing the quality of heuristic approaches to this problem. All the instances used here and all the numerical results obtained can be downloaded from `www.ic.uff.br/~yuri/mbsp.html`.

The heuristics are coded in C++ and tested on an Intel Core 2 Duo Computer with a 2.93 GHz and 2 GB of RAM memory. The branch-and-cut (BC) algorithm from [17], described in Section 4, is coded in C++, running on a Intel(R) Pentium(R) 4 CPU 3.06GHz, equipped with 3 GB of RAM. We use Xpress-Optimizer 20.00.21 [33] to implement the components of this enumerative algorithm. All results reported for heuristic procedures and the BC code were obtained over pre-processed instances. Time reported for these procedures does not include pre-processing time. The CPU time limit is set to 1h for the BC and to 300 seconds for the heuristics. We remember the reader that a second termination criterion have been defined in Section 5 for each heuristic procedure.

Pre-processed instances are generated by the following procedure. First, reduction rules (1), (2), (3) and (4) are applied repeatedly until we obtain a graph G that cannot be reduced by any of them. Let $B \subseteq V$ be a subset of vertices in G such that, for each $i \in B$, $S = \{i\}$ is a separator such that $|C| \leq 15$. Thus, for each vertex $i \in B$, we apply the reduction rule S1. Results obtained with this pre-processing procedure are reported in the last row (*All rules*) of Tables 1 to 3. The other rows on these tables present numerical results when only one rule is applied repeatedly. Notice that rule 3, as defined in Section 3, eliminates only edges. However, isolated vertices can arise in the graph when only this rule is applied. In that case, we also eliminate them and that explains why non-zero values are reported in row *Rule 3* for vertex reductions. The pre-processing of any of our instances lasted at most 5 seconds, so that pre-processing times are not reported in the next subsections.

6.1. Random instances

We generated two sets of random instances. Group 1 is the set of random signed graphs without parallel ($E^- \cap E^+ = \emptyset$). For this group, we generated graphs by varying the number of vertices $|V|$, the graph density $d = 2 * |E|/(|V|^2 - |V|)$ and the rate $|E^-|/|E^+|$. For this group, we considered a set of 108 random signed graphs having $|V|$ ranging in the set {50, 100, 150, 200}, d varying in the set {0.25, 0.50, 0.75} and having $|E^-|/|E^+|$ varying in the set {0.5, 1.0, 2.0}. For each combination of these values, three different signed graphs were generated. Group 2 is the set of random signed graphs with

parallel edges ($E^- \cap E^+ \neq \emptyset$). For this group, we generated graphs by varying $|V|$, d and the rate $|E^+ \cap E^-|/|E|$ with $|V|$ and d varying as in Group 1 and having $|E^+ \cap E^-|/|E|$ varying in the set $\{0.25, 0.50, 0.75\}$. Again, three different signed graphs were generated for each combination of these values which also totalizes 108 random instances in Group 2.

The pre-processing procedure had almost no effect over the random instances, so numerical results are not reported for this procedure. Now we present the results obtained with the heuristic methods over the sets of random instances. For each random instance, the heuristic procedures terminated because the maximum number of iterations was achieved in less than 20 seconds. Tables 4 and 5 show the results obtained with the seven different versions of GGMZ greedy heuristic discussed in Section 5. Again, average values are presented per $|V|$, d and rates $|E^-|/|E^+|$ and $|E^+ \cap E^-|/|E|$. For heuristic results, the percentage gap of each instance is $100 \times (B^* - BS)/B^*$ where BS is the value of the solution found by the heuristic procedure and B^* is the best feasible solution we got from all solution procedures (heuristics and BC). From Table 4, we conclude that, in average, the Adaptive version of GGMZ algorithm is the best version for instances in Group 1 followed by the DFS version. Results on Table 5 shows that the Adaptive version is also the best one for instances in Group 2. It is not clear which heuristic version has the second place for Group 2. The authors in [20] have concluded that building the spanning tree is the crucial step of GGMZ heuristic and we agree with them. However, they have also reported that DFS is the best strategy for implementing Step 1 of Algorithm 1 among DFS, BFS and a random procedure. Results on Table 5 show that it is not always the case.

Figures 3 and 4 compare the percentage gaps obtained with the Adaptive version and with our GRASP metaheuristic. In these graphics, the x-axis exhibits instances ordered primarily by number of vertices and secondly by number of edges while the y-axis exhibits percentage gaps. Clearly the GRASP achieved small gaps, reaching optimal values for many instances. Figure 4 also shows the percentage gaps obtained with the Adaptive version decreases with the number of edges in the signed graph.

6.2. DMERN instances

As we have mentioned before, some solution approaches based on signed graphs have been proposed in the literature for the solution of the DMERN problem in [18, 19, 20]. These works reported the computational performance of their methods over a set of instances available in the literature. We refer to this set of instances as the GGMZ instances. Before solving each instance of this set, Gulpinar *et al.* applied a pre-processing procedure in order to reduce the size of the coefficient matrix and a scaling procedure in order to increase the dimension of the $\{-1, 0, +1\}$-matrix. This set has around hundred matrices of various sizes. Detailed results were reported by Gulpinar *et al.* for the subset of 44 matrices which have at least 500 rows. Figueiredo *et al.* [18] reported their computational results over the 34 more difficult instances that were made available by Gulpinar *et al.* already pre-processed and scaled. The results obtained with the exact

algorithm developed in [18] showed that GGMZ instances are easy instances except for one. The BC algorithm was able to solve almost all instances in just a few seconds with most of them solved to optimality at the root of the BC tree.

Since this is a set of easy instances, unable to capture the complexity of the MBSP, we do not run computational experiments neither with the improved BC code nor with the heuristic methods. However, we run our pre-processing procedure on this set of instances. We can see from Table 1 that these instances were greatly reduced in the set of vertices as well as in the set of edges. After the application of all the reduction rules, almost all the remaining graphs are highly disconnected. Next, we introduce a set of more difficult instances of the DMERN problem that are more suitable for assessing the quality of heuristic approaches to this problem.

In [2], a large set of general mixed integer programs (MIP) coming from network design problems were used to detect block structures in matrices. In order to compose new DMERN instances, a subset of constraint matrices was selected from these problems (some MIPs had the same constraint matrix only differing in the right-hand side of the inequalities) and the scaling procedure described in [20] was applied. Here, we denote this set of instances as the new DMERN instances. This set is composed by 316 instances having $|V|$ varying from 19 to 8317. Table 3 presents the reductions obtained by running our pre-processing routine over the new DMERN instances. Only reduction rules (1) and (2) were able to reduce instances in this set and together these rules were responsible for halving the number of vertices in this set.

Figure 5 shows the number of instances solved to optimality by the BC code. Graphic (a) gives this information as a function of the graph density while graphic (b) as a function of the number of vertices. We can see that many instances were solved to optimality. Table 9 shows results obtained for the instances remaining unsolved and for the instances solved to optimality in more than one minute. The first three columns give us information about the instances: the Netlib instance name, the number of vertices and the number of edges. The next five columns give us information about the pre-processed signed graphs: the number of vertices, the number of edges and the number of negative, positive and parallel edges. Finally, the last set of columns gives us information about the solution obtained with the BC code: the time (in seconds) spent to solve the instances to optimality ("$-$" means the instance was not solved within the time limit), the percentage final gap, the percentage gap obtained for the initial formulation, the value of the optimal solution (whenever the time limit is reached, this column reports the value of the best integer solution found) and the total number of nodes in the branch-and-bound tree. Information on this table shows us that these instances were the less affected by pre-processing routines. From this set of instances, we can extract 21 instances not solved to optimality by the BC code, some of them with final gap above 20%.

Figures 6, 7 and 8 present the results obtained with the heuristic approaches for all the 316 new DMERN instances. Again, the x-axis exhibits instances ordered primarily by number of vertices and secondly by number of edges while the y-axis exhibits the percentage gaps. Clearly, the GRASP metaheuristic has

found the best heuristic solution for almost all new DMERN instances. With respect to the different versions of the GGMZ heuristic, the percentage gaps varied substantially for a same version of this heuristic approach which makes difficult to establish which version has found the better order to include edges in the spanning tree. The DFS version was the one with less variation on the gaps. Considering only the gaps, from Figure 6, we can conclude that the DFS version got slightly better results than the BFS version (DFS and BFS versions outperformed each other by at least 5% in, respectively, 51% and 34% of the instances). On the other hand, we can see that the three GGMZ heuristics in Figure 7 reached almost the same gaps ($(-,+,+-)$, $(+,-,+-)$ and $(-,+-,+)$ versions outperformed each other by at least 5% in, respectively, 5%, 2.5% and 1% of the instances). From Figure 8, we can conclude that the Adaptive version got better results than the Random version (Adaptive and Random versions outperformed each other by at least 5% in, respectively, 58% and 32% of the instances). From Table 8, we can see that versions $(-,+-,+)$ and $(-,+,+-)$ solved more instances to optimality. Table 8 informs us the number of instances (for each instance set) for which each heuristic method has found the optimal value. The last row on this table informs us how many instances were solved to optimality by the BC code. For the new DMERN instances, the time spent and the termination criteria achieved by the heuristic procedures seems not to be related with the size of the instance to be solved. As we could expect, in general, the GRASP procedure spent more time than the GGMZ heuristics. Average times (in seconds; calculated over the set of all new DMERN instances) spent by the heuristic procedures are shown in Table 6. For 56 instances in this set, each heuristic procedure terminated because the maximum number of iterations was achieved in less than 10 seconds. We run additional experiments on the set defined by the other 260 instances: with time limit set to 10 seconds and set to 30 seconds. Figure 9 (a) and (b) exhibits, respectively, the obtained results. Results obtained by the GRASP procedure are compared with results obtained by the DFS and $(-,+,+-)$ versions. We can see that, after 10 seconds of computation, the GRASP procedure has already found the best results for many instances in this subset extending its advantage as time limit is increased to 30 seconds.

6.3. Portfolio analysis instances

For this application, we considered the market graphs generated by Hüffner *et al.* in [24]. These authors used publicly available historical data (for 2003 and 2004) of 5216 stocks to generate this test set. For each pair of stocks, the correlation coefficient was calculated by using, in a given time range, the corresponding stock price fluctuations. Two stocks are positively correlated if they have similar daily fluctuation; otherwise, they are negatively correlated. So, as it was done in [22], from the correlation coefficients computed, they applied a simple threshold transformation to create a signed graph. Consider a threshold value t. If a correlation coefficient was bigger (smaller) than t $(-t)$, then a positive (negative) edge was added. Different signed graphs were generated by using different threshold values $0.3 \leq t \leq 0.4$. To avoid the generation of trivial

instances (with mostly negative edges) and motivated by a study of Boginski *et al.* [9], they added an offset of 0.05 to all correlation coefficients. Notice that a signed graph generated over the set of 5216 stocks would be a very large graph. Thus, to compose this test set, they randomly chose subgraphs whose sizes range from 30 to 510 vertices in steps of 30. For more details, we refer the reader to the work of Hüffner *et al.* [24].

This test set is composed by instances with the number of vertices $|V|$ varying in the set $\{30, 60, 90, \ldots, 510\}$ and the threshold value t varying in the set $\{0.300, 0.325, 0.350, 0.375, 0.400\}$. For each combination of these values, 10 different signed graphs were randomly chosen, which means that each signed graph represents a different subset of stocks and totalize 850 instances. We call this set of instances the portfolio instances. From Table 2 we can see that almost all the instances in this set were reduced by at least one reduction rule and that rule 4 was useless for them. If we compare these results with the results obtained for the new DMERN instances, in total, more portfolio instances were reduced but the percentage of eliminated vertices and eliminated edges were higher for the new DMERN instances. The graphics plotted in Figure 10 exhibits the number of instances solved to optimality by the BC code: graphic (a) gives this information as a function of $|V|$ while (b) as a function of t. We can see these instances become more difficult as the number of vertices increases and the threshold value decreases. Remember that, as defined in [22], signed graphs generated for a set of securities in a given time period have more edges as the threshold value decreases. All the instances with up to 210 vertices were easily solved in a few seconds. Table 7 shows average results for signed graphs with $|V| \in \{390, 420, 450, 480, 510\}$. The notations in this table are the same as in Table 9 except for the fact that Table 7 exhibits average values. The entries in column $\%Gap$ are calculated over the instances not solved within the time limit while the entries in columns *Time* and *Nodes* are calculated over the instances solved to optimality. We can see that, in general, the gap increases with the number of vertices and as the threshold value decreases. Also, *Time* and *Nodes* decreases as the threshold value increases.

Now, we turn our attention to the results obtained with the heuristic approaches on the portfolio instances. In Figures 11, 12 and 13, the x-axis exhibits instances ordered primarily by number of vertices and secondly by the threshold value while the y-axis exhibits the percentage gaps. Our GRASP heuristic is the best heuristic also for the portfolio instances. The performance of the GRASP heuristic degrades as the number of vertices exceeds 450 and the performance of GGMZ heuristics degrade as the threshold value decreases. Again, it is difficult to answer which is the best or worst version of GGMZ heuristic for the portfolio instances. Although $(-, +-, +)$ and $(-, +, +-)$ solved more instances in this set to optimality (see Table 8), they were the versions that presented the biggest variations on the gaps (see Figures 11–13). Results in Figure 11 demonstrate that there was less variation in the gaps obtained with the BFS and the DFS version on portfolio instances than the ones observed on the DMERN instances and, for many instances, the BFS version has found a better solution than the DFS version (DFS and BFS versions outperformed each other by at

least 5% in, respectively, 13% and 60% of the instances). Comparing the gaps in Figure 12, we can conclude that the $(+,-,+-)$ version got better results than the other two versions $((-,+,+-)$, $(+,-,+-)$ and $(-,+-,+)$ versions outperformed each other by at least 5% in, respectively, 0%, 58% and 0.1% of the instances). From results in Figure 13 and in Table 8, we can conclude that the Adaptive and Random versions got very similar performance (Adaptive and Random versions outperformed each other by at least 5% in, respectively, 27% and 36% of the instances and solved, respectively, 173 and 187 instances to optimality). For most portfolio instances with up to 210 vertices, all heuristics stop because the maximum number of iterations was achieved in less than 50 seconds. For instances with more than 240 vertices, the heuristics took more time as the threshold value decreased and the number of vertices increased. For most instances with more than 240 vertices, the GRASP heuristic stop because the time limit of 300 seconds was achieved. On the other hand, for most of these instances, the GGMZ heuristics terminated because the maximum number of iterations was achieved in less than 300 seconds (most of them from 50 to 250 seconds). Again, with the aim to detect the preferable heuristic procedure when a quick solution is needed, we run additional computational experiments on a subset of portfolio instances, with number of vertices from 240, considering different time limits (10 and 30 seconds). Figure 14 exhibits the obtained results. We can conclude from Figure 14 (a) that the GRASP procedure is no longer the best option if a quick solution is needed. The Random version of the GGMZ heuristic is the best option followed closely by the $(+,-,+/-)$ GGMZ version. In Figure 14 (b) we see that the results obtained by the GRASP procedure had improved when the time limit was increased from 10 to 30 seconds but the GGMZ heuristic continued to be the best option. From results in column *Best Sol* of Table 7, we noticed that signed graphs associated with portfolio instances contains big balanced subgraphs and that for threshold value equal to 0.400 the signed graphs are almost balanced. We believe this explains the results shown in Figure 14 since the GGMZ heuristic has the great advantage of finding the optimal solution of the MBSP whenever the signed digraph is balanced.

6.4. UNGA instances

In [27], the community structure of networks representing voting on resolutions in the United Nations General Assembly (UNGA) was investigated. The authors constructed networks from the UNGA voting records of the 63 separate annual sessions between 1946 and 2008 in three different ways. The 19th session was not considered since voting occurred on only one resolution in this session. We refer the interested reader to [27] for an introduction to the United Nations General Assembly voting data including an interesting discussion of different ways to represent this data in the form of networks. The records for sessions between 1946 and 2008 are available in Voeten's organization of the UNGA vot-

ing data [1]. In this work, we represent these UNGA voting data records by a set of signed graphs. The optimal solution of the MBSP over this set of signed graphs allows the identification of majority voting groups in each voting session.

Next, we describe how the set of signed graphs is generated. Consider two parameters $\alpha, \beta \in [0, 1]$. The signed graphs are constructed following the steps of Algorithm 2. The sets of positive and negative edges are defined by the number of agreements and disagreements on resolutions. Parameter α specifies the level of agreement and disagreement between two countries that gives rise to positive or negative edges in the signed graph. Parameter β determines if the sum of agreements and disagreements between two countries is sufficient to establish that they cannot be considered as in the same voting group. Following an observation from [27], we treat differently the disagreement between two countries in a *yes-no* pair of votes on a same resolution from a *yes-abstain* pair or a *no-abstain* pair. Also, we normalize counts of agreement and disagreement by the total number of resolutions voted in the session.

Algorithm 2: Algorithm to create a signed graph representing a UNGA voting session.

Input: Voting records {*yes, no, abstention*} for the set of resolutions and countries in a session
Output: Signed graph $G = (V, E, s)$

$V = \{i \mid i$ is a country voting in the section$\}$;
for *Each pair of countries* i *and* j **do**
  $agree = disagree = 0$;
  for *Each resolution* **do**
    if i *and* j *abstain* **then** $agree = agree + 0.5$;
    if i *and* j *vote equal* **and** *do not abstain* **then** $agree = agree + 1$;
    if i *and* j *vote differently* **and** *do not abstain* **then** $disagree = disagree + 1$;
    if *exactly one of them abstain* **then** $disagree = disagree + 0.5$;
  end
  $\%agree = agree/total\_of\_resolutions$;
  $\%disagree = disagree/total\_of\_resolutions$;
  if $(\%agree - \%disagree) \geq \alpha$ **then**
    Create a positive edge (i, j);
  if $(\%agree - \%disagree) \leq -\alpha$ **then**
    Create a negative edge (i, j);
  if $-\alpha < (\%agree + \%disagree) < \alpha$ **and** $(\%agree + \%disagree) \geq \beta$ **then**
    Create a parallel edge (i, j);
end

We generated the set of UNGA instances varying parameters α and β, re-

[1] United Nations General Assembly Voting Data, by Anton Strezhnev and Erik Voeten, http://hdl.handle.net/1902.1/12379. Accessed in June 2012.

spectively, in sets $\{0.1, 0.2, 0.3\}$ and $\{0.5, 0.8\}$. Signed graphs generated have the number of vertices varying from 54 to 192. For each of the 6 combinations of parameters α and β, a set of instances was generated.

The pre-processing procedure had almost no effect on this set of instances that is composed by dense signed graphs. The mean graph density for each of the 6 sets varies from 0.72 to 0.97. On the other hand, UNGA instances were easily solved to optimality by the BC code. This set of instances is composed by signed graphs with at most 192 vertices and that contains large balanced subgraphs. Figure 15 shows the optimal solutions obtained with all the UNGA instances. In this figure the x-axis represents all the sections from 1946 to 2008 numbered from 1 to 63. The dotted dark line in this graphic gives us the number of countries in the session. The other lines shows the optimal solution of the MBSP defined over the associated signed graph. As we could expect, the optimal value increases as parameter α increases and it decreases as parameter β increases. Figure 16 shows the majority groups in the optimal solutions obtained for the UNGA instances generated with $\alpha = 0.3$ and with (a) $\beta = 0.5$ and with (b) $\beta = 0.8$. Interpreting the results in the context of each application is not in the scope of this paper, specially for the UNGA voting records since these data have been widely investigated in the literature. However, it is clear that these voting sessions are characterized by the existence of a majority agreement group that changes whenever we fix one parameter and vary the other. This can explain why these instances are always so easy to solve. Certainly, in other voting contexts, this is not the case and more difficult instances would be generated making the heuristic approaches useful also for this application.

7. Final remarks

The MBSP is a combinatorial problem with applications arising in collaborative vs. conflicting environments that can be modeled over a signed graph. We discussed two applications [19, 22] from the literature and introduced a new application of the problem in community structure. Despite its interesting applications only two solution methods had been proposed in the literature for the MBSP: a greedy heuristic [19] and a BC algorithm [17, 18]. We proposed improved versions for the greedy heuristic and developed a GRASP based algorithm for improving the quality of heuristic solutions obtained for difficult instances. In addition, we discussed simple data reduction rules and implemented a pre-processing procedure for the problem. Numerical experiments were performed over a set of benchmark instances of each application as well as over a set of random instances. A side contribution of this work is the introduction of a set of difficult instances for the DMERN problem filling a gap pointed in the work of Figueiredo *et al.* [18].

The pre-processing procedure had almost no effect over random and UNGA instances. DMERN and portfolio instances were greatly reduced both in the set of vertices and in the set of edges. No reduction rule was useless but their

individual efficiency depends on the set of instances being solved. More portfolio instances were reduced than DMERN instances, however the percentage of eliminated vertices and eliminated edges were higher for the DMERN instances.

All random instances with 50 vertices and all UNGA instances were solved to optimality by the BC algorithm. Most new DMERN instances and portfolio instances were also solved to optimality by this algorithm with a group remaining unsolved. It shows that there is a great number of real applications that can be solved to optimality. This is important since for many problems, the representation of the conflict environment as a signed graph is already an approximation of the real problem where many simplifications were done. In such a scenario, solving the problem heuristically can raise the distance from the obtained solution and the reality.

The GRASP procedure was the best heuristic solution approach when time limit was not a constraint in the solution process. However, when a solution is required in a few seconds, for signed graphs that contains big balanced subgraphs the GGMZ heuristic was the best option. Our computational results also attested that building the spanning tree is an important step in the GGMZ greedy heuristic. The gap associated to the obtained heuristic solution can double depending on the spanning tree used (see Table 4). Moreover, the definition of the best strategy for the implementation of this step depends on the instance being solved. Even if the quality of the heuristic solution was improved by using different strategies in this first step, our GRASP procedure achieved better results for almost all the instances and was able to reach optimal values for many of them. Table 8 shows for how many instances each heuristic procedure has found the optimal value. The last line in this table exhibits the total number of instances for which the optimal solution is known. The GRASP procedure reached optimal values for 65% of the instances the optimal solution is known.

Acknowledgements

We would like to thank the authors of [24] for kindly making the portfolio instances available for us. We also thank Luidi Simonetti for providing helpful comments during the development of this work.

[1] P. Abell and M. Ludwig. Structural balance: a dynamic perspective. *Journal of Mathematical Sociology*, 33:129–155, 2009.

[2] T. Achterberg and C. Raack. The MCF-separator – detecting and exploiting multi-commodity flows in MIPs. *Mathematical Programming C*, 2:125–165, 2010.

[3] G. Agarwal and D. Kempe. Modularity-maximizing network communities using mathematical programming. *The European Physical Journal B*, 66:409–418, 2008.

[4] E. Balas and C.S. Yu. Finding a maximum clique in an arbitrary graph. *SIAM Journal on Computing*, 14:1054–1068, 1986.

[5] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proceedings of the 43rd annual IEEE symposium of foundations of computer science*, pages 238–250, Vancouver, Canada, 2002.

[6] F. Barahona and A.R. Mahjoub. Facets of the balanced (acyclic) induced subgraph polytope. *Mathematical Programming*, 45:21–33, 1989.

[7] J.J. Barthold. A good submatrix is hard to find. *Operations Research Letters*, 1:190–193, 1982.

[8] M. S. Bazaraa, J. Jarvis, and H. D. Sherali. *Linear Programming and Network Flows*. Wiley-Interscience, 2009.

[9] V. Boginski, S. Butenko, and P.M. Pardalos. Mining market data: A network approach. *Computers and Operations Research*, 33:3171–3184, 2006.

[10] D. Cartwright and F. Harary. Structural balance: A generalization of heiders theory. *Psychological Review*, 63:277–293, 1956.

[11] T.H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009.

[12] B. DasGupta, G. A. Encisob, E. Sontag, and Y. Zhanga. Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *BioSystems*, 90:161–178, 2007.

[13] P. Doreian and A. Mrvar. A partitioning approach to structural balance. *Social Networks*, 18:149–168, 1996.

[14] P. Doreian and A. Mrvar. Partitioning signed social networks. *Social Networks*, 31:1–11, 2009.

[15] T.A. Feo, M.G.C. Resende, and St.H. Smith. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research*, 42:860–878, 1994.

[16] P. Festa and M.G.C. Resende. *GRASP: an annotated bibliography*. Kluwer Academic Publishers, 2002.

[17] R. Figueiredo and Y. Frota. An improved branch-and-cut code for the maximum balanced subgraph of a signed graph. *CoRR*, 2013. arXiv/1312.4345.

[18] R. Figueiredo, M. Labbé, and C.C. de Souza. An exact approach to the problem of extracting an embedded network matrix. *Computers & Operations Research*, 38:1483–1492, 2011.

[19] N. Gülpinar, G. Gutin, G. Mitra, and A. Zverovitch. Extracting pure network submatrices in linear programs using signed graphs. *Discrete Applied Mathematics*, 137:359–372, 2004.

[20] G. Gutin, D. Karapetyan, and I. Razgon. Fixed-parameter algorithms in analysis of heuristics for extracting networks in linear programs. In *Proceedings of the 4th International Workshop on Parameterized and Exact Computation*, pages 222–233, Copenhagen, Denmark, 2009.

[21] F. Harary and J.A. Kabell. A simple algorithm to detect balance in signed graphs. *Mathematical Social Sciences*, 1:131–136, 1980.

[22] F. Harary, M. Lim, and D. C. Wunsch. Signed graphs for portfolio analysis in risk management. *IMA Journal of Management Mathematics*, 13:1–10, 2003.

[23] F. Heider. Attitudes and cognitive organization. *Journal of Psychology*, 21:107–112, 1946.

[24] F. Huffner, N. Betzler, and R. Niedermeier. Separator-based data reduction for signed graph balancing. *Journal of Combinatorial Optimization*, 20:335–360, 2010.

[25] R. Martí, M.G.C. Resende, and C. C. Ribeiro. Multi-start methods for combinatorial optimization. *European Journal of Operational Research*, 226:1–8, 2013.

[26] T. Inohara. On conditions for a meeting not to reach a deadlock. *Applied Mathematics and Computation*, 90:1–9, 1998.

[27] K.T. Macon, P.J. Mucha, and M.A. Porter. Community structure in the united nations general assembly. *Physica A: Statistical Mechanics and its Applications*, 391:343–361, 2012.

[28] M. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.

[29] V.Th. Paschos. A $\delta/2$-approximation for the maximum independent set problem. *Information Processing Letters*, 44:11–13, 1992.

[30] M.A. Porter, J.-P Onnela, and P.J. Mucha. Communities in networks. *Notices of the AMS*, 56:1082–1166, 2009.

[31] S. Rebennack, M. Oswald, D.O. Theis, H. Seitz, G. Reinelt, and P.M. Pardalos. A branch and cut solver for the maximum stable set problem. *Journal of Combinatorial Optimization*, 21:434–457, 2011.

[32] F. Rossi and S. Smriglio. A branch-and-cut algorithm for the maximum cardinality stable set problem. *Operations Research Letters*, 28:63–74, 2001.

[33] FICO Xpress Optimization Suite. Xpress-Optimizer reference manual, 2009. Release 20.00.

[34] V.A. Traag and J. Bruggeman. Community detection in networks with positive and negative links. *Physical Review E*, 80:036115, 2009.

[35] V. Valls, M. Angeles Perez, and M. Sacramento Quintanilla. Pre-processing techniques for resource allocation in the heterogeneous case. *European Journal of Operational Research*, 107:470–491, 1998.

[36] B. Yang, W.K. Cheung, and J. Liu. Community mining from signed social networks. *IEEE Transactions on Knowledge and Data Engineering*, 19:1333–1348, 2007.

[37] T. Zaslavsky. A mathematical bibliography of signed and gain graphs and allied areas. *Electronic Journal of Combinatorics DS8*, 1998.

Rule	Percentage of Instances Vertices	Edges	Reductions %n	%m
Rule 1	61.32	0.00	27.84	0.00
Rule 2	68.87	55.66	47.32	30.00
Rule 3	1.89	1.89	0.73	1.98
Rule 4	65.09	65.09	24.69	32.95
Rule S1	64.15	64.15	20.65	24.83
All rules	77.58	65.09	55.25	39.93

Table 1: Results obtained with each reduction rule on GGMZ instances.

Rule	Percentage of Instances Vertices	Edges	Reductions %n	%m
Rule 1	87.76	0.00	17.08	0.00
Rule 2	92.12	89.29	24.85	10.30
Rule 3	93.06	93.06	14.03	10.77
Rule 4	0.00	0.00	0.00	0.00
Rule S1	92.71	92.71	12.41	6.99
All rules	94.00	93.29	36.60	16.36

Table 2: Results obtained with each reduction rule on portfolio instances.

Rule	Percentage of Instances Vertices	Edges	Reductions %n	%m
Rule 1	37.97	0.00	54.05	0.00
Rule 2	52.53	20.57	44.17	13.63
Rule 3	0.00	0.00	0.00	0.00
Rule 4	0.00	0.00	0.00	0.00
Rule S1	0.00	0.00	0.00	0.00
All rules	57.28	30.70	49.11	22.27

Table 3: Results obtained with each reduction rule on new DMERN instances.

Method	n				d			$\lvert E^-\rvert/\lvert E^+\rvert$		
	50	100	150	200	.25	.50	.75	.5	1	2
BFS	**10.32**	12.56	9.47	7.41	11.37	10.34	8.10	7.50	**9.13**	11.80
DFS	15.72	15.30	13.99	11.13	12.67	15.04	14.39	10.06	14.26	15.76
(-.+-.+)	13.02	14.91	12.26	10.89	13.93	13.65	10.74	12.79	11.29	12.24
(+.-.+-)	20.66	19.05	16.75	15.03	18.44	17.75	17.44	2.69	15.78	33.61
(-.+.+-)	13.02	13.20	11.33	10.34	13.44	12.63	9.85	12.00	11.06	**10.94**
Random	15.75	17.05	12.70	11.14	14.01	14.23	14.24	10.54	14.02	15.86
Adaptive	11.20	**8.95**	**8.33**	**5.54**	**9.77**	**8.53**	**7.22**	**2.93**	10.06	11.46

Table 4: Results for GGMZ heuristics on random instances in Group 1 ($E^- \cap E^+ = \emptyset$).

Method	n				d			$\lvert E - \cap E + \rvert / \lvert E \rvert$		
	50	100	150	200	.25	.50	.75	.25	.50	.75
BFS	11,68	13,65	14,86	14,18	13,84	13,81	13,13	12,73	13,76	12,08
DFS	15,14	13,83	15,77	14,97	14,45	15,23	15,10	15,35	14,83	12,08
(-,+-,+)	9,96	**10,62**	14,38	13,40	11,99	**12,05**	12,23	11,51	**12,53**	10,23
(+,-,+-)	13,60	15,85	16,72	15,43	15,86	15,19	15,14	16,33	14,72	12,57
(-,+,+-)	10,08	11,66	14,40	13,20	11,91	12,19	12,90	11,42	12,61	10,98
Random	12,77	14,65	15,46	15,88	13,83	15,23	15,00	13,70	13,90	14,17
Adaptive	**9,12**	12,07	**13,25**	**12,83**	**11,09**	12,44	**11,92**	**11,37**	12,99	**9,06**

Table 5: Results for GGMZ heuristics on random instances in Group 2 ($E^- \cap E^+ \neq \emptyset$).

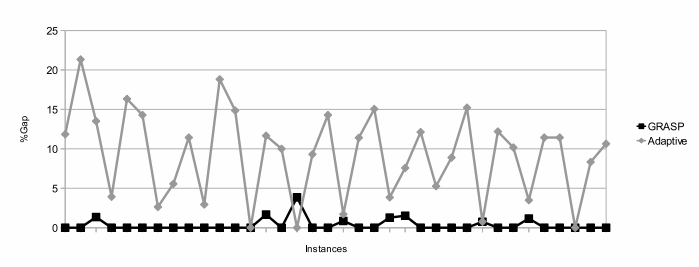

Figure 3: GRASP results on random instances with $E^- \cap E^+ = \emptyset$.

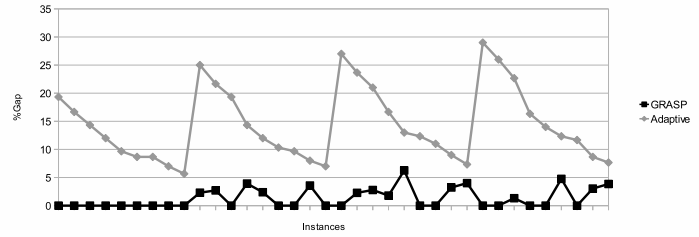

Figure 4: GRASP results on random instances with $E^- \cap E^+ \neq \emptyset$.

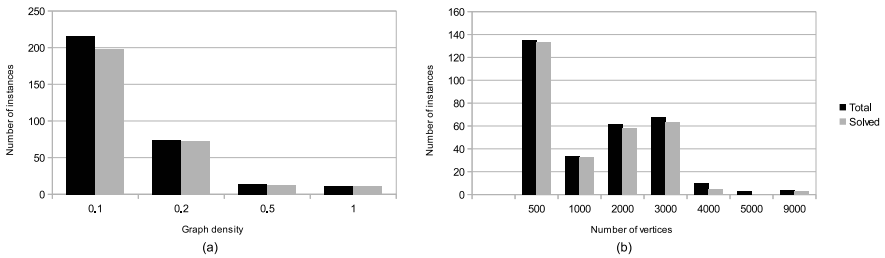

Figure 5: Number of new DMERN instances solved to optimality by the improved BC code.

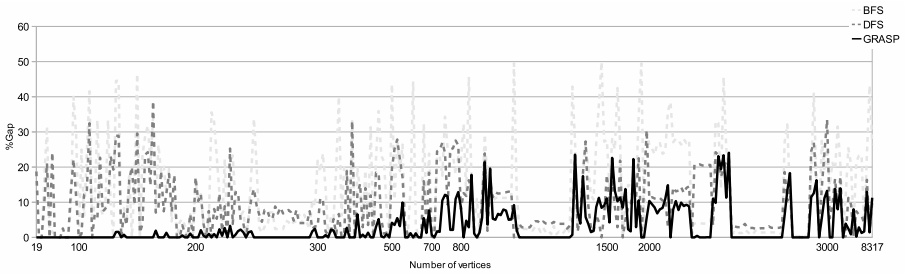

Figure 6: Heuristic results on new DMERN instances.

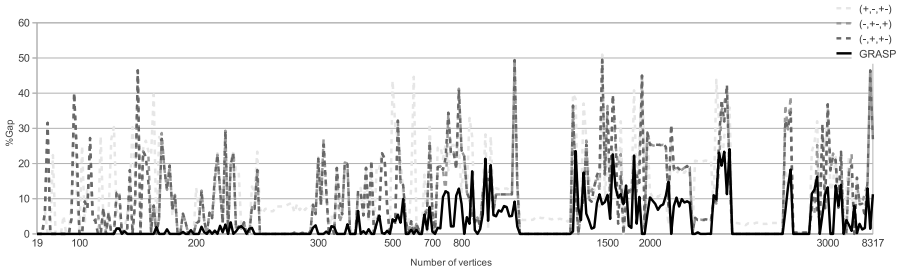

Figure 7: Heuristic results on new DMERN instances.

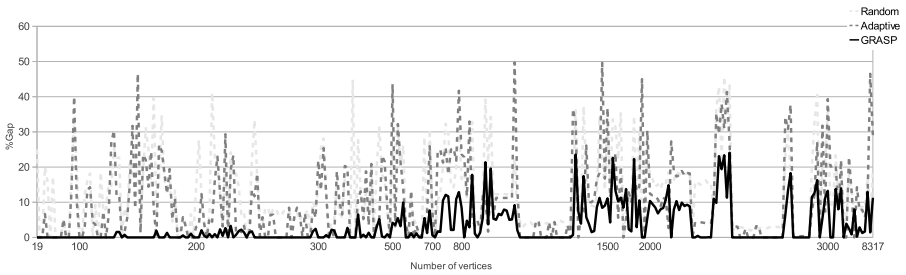

Figure 8: Heuristic results on new DMERN instances.

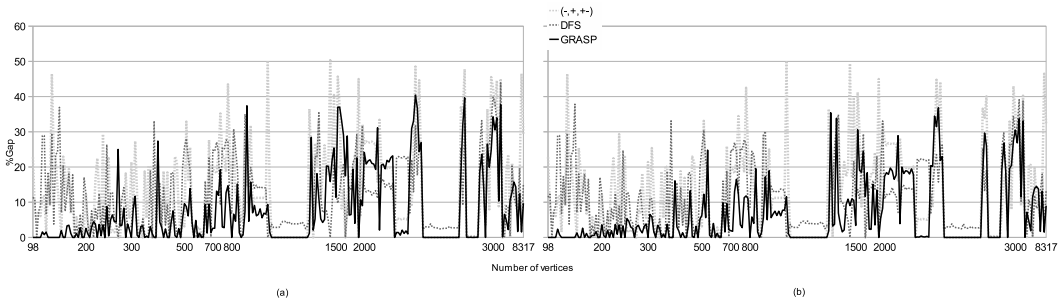

Figure 9: Heuristic results on new DMERN instances with time limit set to (a) 10 seconds and to (b) 30 seconds.

Method	Mean Time
BFS	136.39
DFS	137.94
(-,+-,+)	141.67
(+,-,+-)	142.54
(-,+,+-)	141.72
Random	146.85
Adaptive	142.43
GRASP	184.21

Table 6: Mean time spent by heuristic procedures on new DMERN instances.

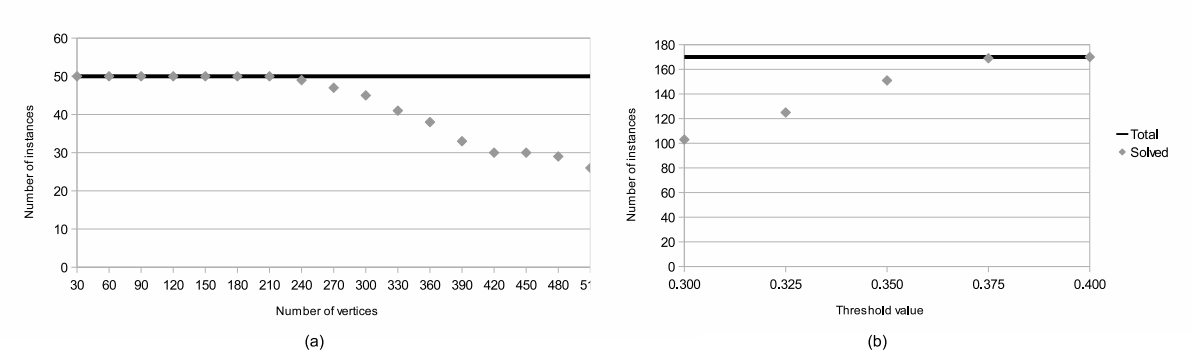

Figure 10: Number of portfolio instances solved to optimality by the improved BC code.

V	t	Time	%Gap	Best Sol	Nodes
390	0.300	650.50	10.25	279.80	510.60
	0.325	101.50	4.69	313.90	497.30
	0.350	29.14	1.60	343.00	576.90
	0.375	5.20	–	367.10	4.40
	0.400	1.40	–	381.30	1.70
420	0.300	–	17.54	280.30	401.70
	0.325	1225.00	7.98	325.20	605.20
	0.350	112.00	4.19	364.10	301.50
	0.375	158.60	–	391.20	209.40
	0.400	4.30	–	407.20	11.50
450	0.300	–	13.66	303.90	335.00
	0.325	121.00	4.93	349.70	391.50
	0.350	381.44	2.65	389.70	249.40
	0.375	23.40	–	418.40	17.20
	0.400	2.70	–	436.20	1.00
480	0.300	725.00	25.30	308.50	255.90
	0.325	527.67	12.53	360.10	299.70
	0.350	231.00	3.30	408.40	343.20
	0.375	162.40	–	441.70	77.50
	0.400	7.20	–	463.70	7.00
510	0.300	815.00	28.70	321.80	182.00
	0.325	457.00	13.55	372.40	252.30
	0.350	58.33	4.27	424.50	292.60
	0.375	647.33	0.53	464.40	552.80
	0.400	7.10	–	491.40	4.40

Table 7: Results obtained with the BC method on portfolio instances.

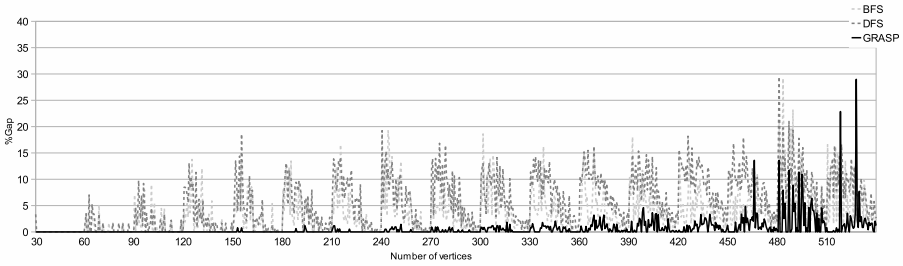

Figure 11: Heuristic results on portfolio instances.

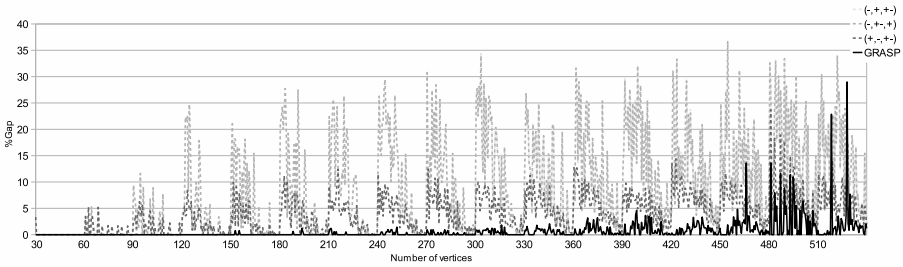

Figure 12: Heuristic results on portfolio instances.

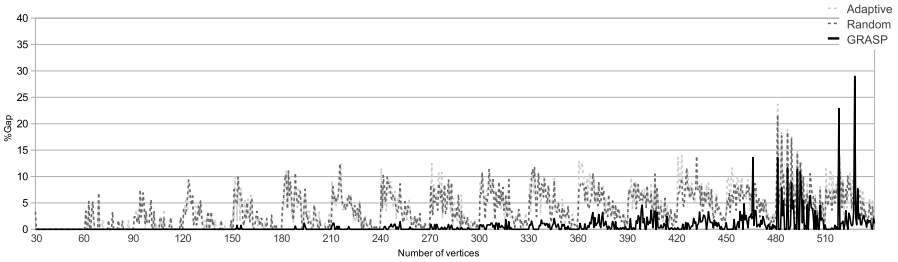

Figure 13: Heuristic results on portfolio instances.

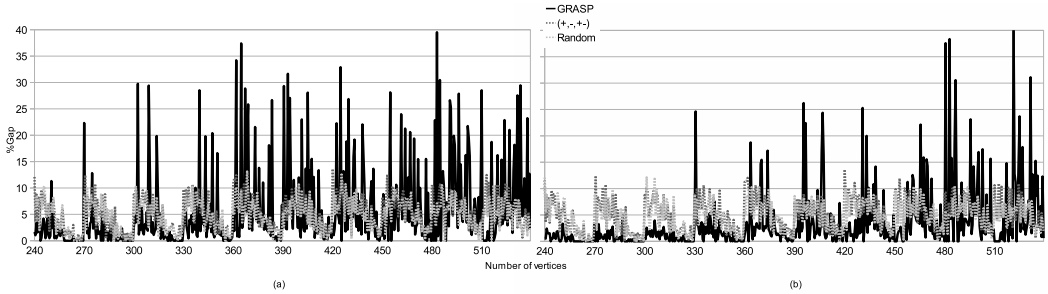

Figure 14: Heuristic results on portfolio instances with time limit set to (a) 10 seconds and to (b) 30 seconds.

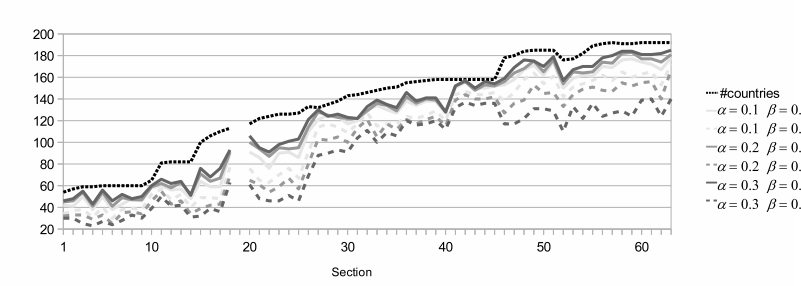

Figure 15: Optimal values obtained on UNGA instances.

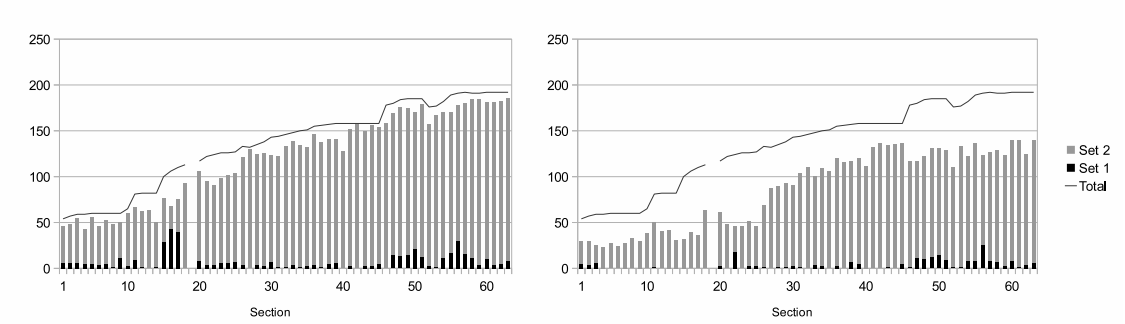

Figure 16: Majority groups in the optimal solutions obtained for UNGA instances with $\alpha = 0.3$ and with (a) $\beta = 0.5$ and with (b) $\beta = 0.8$.

Method	Instance set				Total
	Random Group 1	Random Group 2	new DMERN	Portolio	
BFS	5	5	28	203	241
DFS	2	4	32	171	209
(-,+-,+)	3	9	98	219	329
(+,-,+-)	6	3	35	171	215
(-,+,+-)	3	8	99	219	329
Random	1	4	21	187	213
Adaptive	8	9	76	173	266
GRASP	32	50	159	485	726
BC	34	57	295	718	1104

Table 8: Number of instances solved to optimality.

Name	n'	m'	n	m	$m-$	$m+$	$m-+$	Time	%Gap	%GapLP	Best Sol	Nodes
danoint	145	1457	144	1456	497	903	56	165	0.00	31.25	96	3951
bienst1	184	2548	184	2548	1981	567	0	2772	0.00	31.11	90	39710
stein45	331	10701	331	10701	10701	0	0	–	4.03	4.03	29	516
disctom	399	30000	399	30000	30000	0	0	643	0.00	32.78	299	16
fc.60.20.1	414	1051	414	1051	521	530	0	171	0.00	3.50	371	399
air05	426	30257	426	30257	30257	0	0	–	30.98	281.75	63	95
neos17	486	117855	486	117855	117370	0	485	61	0.00	0.00	2	1
p100x588	688	1470	688	1470	625	845	0	63	0.00	0.79	633	71
air04	823	55592	823	55592	55592	0	0	–	40.43	238.94	113	27
r80x800	880	2000	880	2000	1026	974	0	699	0.00	1.45	828	223
nug08	912	13952	912	13952	13952	0	0	150	0.00	135.16	128	3
p50x864	914	1872	914	1872	895	977	0	107	0.00	0.57	884	53
n5-3	1012	10750	1012	10750	5472	5278	0	82	0.00	3.18	912	1
neos21	1085	37373	1085	37373	37373	0	0	782	0.00	383.77	191	3
n4-3	1178	15341	1178	15341	7670	7671	0	165	0.00	4.43	1062	1
dano3mip	1228	46507	1227	46506	14948	31003	555	–	85.43	95.91	587	42
n8-3	1300	11656	1300	11656	6258	5398	0	120	0.00	1.79	1176	1
neos20	1352	14670	1320	14639	10788	3851	0	107	0.00	0.08	595	10
p200x1188	1388	2970	1388	2970	1256	1714	0	–	0.63	1.26	1272	526
p200x1188c	1388	2970	1388	2970	1228	1742	0	–	0.59	1.18	1273	495
roll3000	1543	60946	1300	60706	25022	31630	4054	170	0.00	0.34	581	2
janos-us-ca–D-D-M-N-C-A-N-N	1643	11651	1643	11651	5491	6160	0	214	0.00	2.17	1521	1
pioro40–D-B-M-N-C-A-N-N	1649	10243	1649	10243	5777	4466	0	126	0.00	0.83	1560	1
n13-3	1661	14725	1661	14725	7579	7146	0	214	0.00	1.63	1537	1
n2-3	1752	14856	1752	14856	7935	6921	0	259	0.00	0.79	1656	1
zib54–U-U-E-N-C-A-N-N	1809	7744	1809	7744	1594	1196	4954	62	0.00	0.00	1728	1
qap10	1820	35200	1820	35200	35200	0	0	427	0.00	0.00	200	3
ns1688347	1866	36800	1866	36800	24983	10195	1622	–	20.49	24.91	375	129
germany50–U-U-M-N-C-A-N-N	2088	10560	2088	10560	1143	2691	6726	91	0.00	0.00	2000	1
protfold	2112	89677	2112	89677	30219	58395	1063	–	53.40	59.69	383	4
cap6000	2174	11167	2174	11167	10297	0	870	111	0.00	0.00	2074	1
n7-3	2278	24476	2278	24476	12220	12256	0	1191	0.00	1.62	2162	3
n9-3	2280	33180	2280	33180	16280	16900	0	1330	0.00	3.55	2112	3
acc-1	2286	44595	2286	44595	30912	13683	0	–	2.86	18.13	480	20
n3-3	2303	38857	2303	38857	18602	20255	0	2827	0.00	6.46	2059	5
zib54–D-B-E-N-C-A-N-N	2349	10028	2347	10025	6991	3034	0	211	0.00	0.49	2266	1
n12-3	2358	26496	2358	26496	12956	13540	0	1067	0.00	2.30	2214	1
neos818918	2400	10130	2400	10130	6485	3195	450	800	0.00	0.09	2018	17
germany50–D-B-M-N-C-A-N-N	2438	12232	2438	12232	6325	5907	0	262	0.00	0.21	2350	1
acc-2	2520	60669	2520	60669	43842	16827	0	–	8.76	24.89	454	23
ta2–U-U-M-N-C-A-N-N	2579	12312	2578	12312	2582	1834	7896	175	0.00	0.00	2470	1
n6-3	2686	31228	2686	31228	14664	16564	0	2785	0.00	2.52	2538	3
berlin	2704	6630	2704	6630	2703	3927	0	–	0.94	0.94	2653	17
neos11	2706	47185	2706	47185	33685	13440	60	–	5.84	7.14	702	7
ta2–D-B-M-N-C-A-N-N	2839	13458	2837	13457	9090	4367	0	470	0.00	0.70	2729	1
acc-6	3047	74184	3047	74184	55567	18571	46	–	11.09	12.49	899	10
acc-5	3052	74312	3052	74312	54569	19697	46	–	13.84	14.62	875	10
acc-3	3249	72072	3249	72072	49812	22179	81	221	0.00	0.00	1125	1
acc-4	3285	75186	3285	75186	52301	22804	81	240	0.00	0.00	1125	1
brasil	3364	8265	3364	8265	3363	4902	0	–	0.85	0.85	3307	9
mkc	3386	6583	3127	6299	3503	2793	3	341	0.00	0.00	2964	1
p500x2988	3488	7470	3488	7470	3064	4406	0	–	1.22	1.22	3199	61
p500x2988c	3488	7470	3488	7470	3650	3820	0	–	4.52	4.52	3098	70
mod011	3886	8245	3240	8186	8186	0	0	416	0.00	0.00	3208	1
neos1	4732	80870	4732	80870	41850	36380	2640	–	7.92	8.64	1099	2
seymour	4944	604020	4794	604007	604007	0	0	–	15.25	15.41	409	0
seymour1	4944	604020	4794	604007	604007	0	0	–	15.25	15.41	409	0
n370a	5150	15000	5150	15000	15000	0	0	1374	0.00	0.00	5100	1
rentacar	5709	16913	4294	16669	7916	8716	37	2403	0.00	0.06	3278	2
manna81	6480	72900	6480	72900	72900	0	0	1178	0.00	45.65	2322	1
neos12	8317	320726	8317	320726	302967	17549	210	–	10.38	10.38	1401	0

Table 9: Results obtained with the BC method on the new DMERN instances.