RAIRO-Oper. Res. 55 (2021) 3121–3140
https://doi.org/10.1051/ro/2021150

RAIRO Operations Research
www.rairo-ro.org

A MATHEURISTIC APPROACH FOR THE MAXIMUM BALANCED SUBGRAPH OF A SIGNED GRAPH

Jorge Reynaldo Moreno Ramírez*, Yuri Abitbol de Menezes Frota and Simone de Lima Martins

Abstract. A graph $G = (V, E)$ with its edges labeled in the set $\{+, -\}$ is called a signed graph. It is balanced if its set of vertices V can be partitioned into two sets V_1 and V_2, such that all positive edges connect nodes within V_1 or V_2, and all negative edges connect nodes between V_1 and V_2. The maximum balanced subgraph problem (MBSP) for a signed graph is the problem of finding a balanced subgraph with the maximum number of vertices. In this work, we present the first polynomial integer linear programming formulation for this problem and a matheuristic to obtain good quality solutions in a short time. The results obtained for different sets of instances show the effectiveness of the matheuristic, optimally solving several instances and finding better results than the exact method in a much shorter computational time.

Mathematics Subject Classification. 90C59, 90C27, 90C05, 90C10.

Received January 25, 2021. Accepted September 20, 2021.

1. Introduction

Let $G = (V, E)$ be an undirected graph where V is the set of vertices and E is the set of edges. Consider the set of labels $\{+, -\}$ and the function $s : E \to \{+, -\}$ that assigns a label (or sign) to each edge in E. An edge $e \in E$ is called negative if $s(e) = -$ and positive if $s(e) = +$. For some interesting problems an edge can be both positive and negative. In this case, it is called a parallel edge ($s(e) = \pm$). In this context, an undirected graph G together with a function s is called a signed graph $G = (V, E, s)$ or briefly an s-graph.

Signed graphs were introduced by Heider [13] for describing sentiment relations between people of the same social group. In the conventional approach for modeling a social network, a positive edge represents a friendship or similarity, while a negative edge represents some enmity. Signed graphs have been studied by researchers in different scientific areas, including portfolio analysis in risk management [12, 14], biological systems [5] and detection of embedded matrix structures [11].

The concept of balanced signed graphs was formalized by Cartwright and Harary [4], as an extension of Heider's theory. The authors stated that a balanced signed graph could be partitioned into two vertex sets, being that all edges within the sets are positive, and all those between the sets are negative. It is clear that not every signed graph is balanced, and there are different situations in which finding a balanced signed subgraph

Keywords. Balanced signed graph, local search, matheuristic.

Computer Science Department, Universidade Federal Fluminense, Niterói, RJ, Brazil
*Corresponding author: jorge85.mail@gmail.com

results in a problem of interest. Despite this fact, the problem of finding a balanced subgraph with maximum number of vertices (MBSP) has been applied in almost as many applications as the problem of finding signed graphs [11,12,14].

To formally define the MBSP problem, we present some additional notations used in the rest of the paper. Let $G = (V, E, s)$ denote a signed graph where $E^- \subseteq E$ and $E^+ \subseteq E$ denote, respectively, the set of negative and positive edges in G. Similarly, for a vertex $v \in V$, we define by $\delta^-(v)$ and $\delta^+(v)$ the set of vertices adjacent to v connected by negative and positive edges. Moreover, for a vertex set $V' \subseteq V$, let $E[V'] = \{\{u, v\} \in E \mid u, v \in V'\}$ be the subset of edges induced by V' and $G_{V'} = (V', E[V'], s)$ its vertex induced subgraph of G. To simplify notation we will assume that $|G_{V'}| = |V'|$. We also define that a signed graph $G = (V, E, s)$ is balanced if its vertex set can be partitioned into sets V_1 and V_2 in such a way that $E[V_1] \cup E[V_2] = E^+$. Thus, given a signed graph $G = (V, E, s)$, the MBSP is the problem of finding a subset of vertices $V' \subseteq V$ such that the subgraph $G_{V'}$ is balanced and maximizes the cardinality of V'.

Gülpinar *et al.* [11] proved that MBSP is NP-hard, and the literature presents some heuristic and exact approaches to solve this problem [8,9,11]. Branch-and-cut methods presented in [8,9] use mathematical models based on the fact that 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 [2, 11, 21]. Also, heuristics and metaheuristics were developed for this problem, such as the GGMZ heuristic [11] and a GRASP metaheuristic in [8]. More recently, a new heuristic for this problem, based on negative chordless cycles, was presented by Marinelli and Parente [16].

Different strategies appear in the literature to combine exact and heuristics algorithms. In this context, matheuristics [7, 10, 17] have attracted the attention of the scientific community. Matheuristics may use some features of the mathematical models of one problem to customize heuristics to solve these problems or use heuristics to improve time effectiveness of mathematical programming techniques.

In this work, we propose an alternative optimization formulation for the one proposed in [8,9]. In contrast with their formulation, the size of our optimization model is polynomial in the input size. We also introduce a matheuristic developed to solve the MBSP to obtain good quality solutions quickly.

The remainder of this paper is organized as follows. Section 2 presents a new formulation for the MBSP problem. Section 3 describes the main algorithms used in the proposed matheuristic. In Section 4, we present the results obtained by the matheuristic and the comparison with previous heuristics. Finally, conclusions are discussed in Section 5.

2. A new formulation for the MBSP

Previous mathematical formulations are based in the fact that 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. The branch-and-cut algorithm proposed in [9] and extended in [8] is based on this result and consider a possible exponential set of odd negative cycles in G, denoted as C^-.

The formulation used in this algorithm is the following:

$$\max \sum_{v \in V} y_v \tag{2.1}$$

subject to:

$$y_u + y_v \leq 1, \qquad \forall \{u, v\} \in E^- \cap E^+, \tag{2.2}$$

$$\sum_{v \in C} y_v \leq |C| - 1, \qquad \forall C \in C^-, \tag{2.3}$$

$$y_v \in \{0, 1\}, \qquad \forall v \in V, \tag{2.4}$$

where $y_v = 1$ if and only if vertex v belongs to the balanced subgraph, $y_v = 0$ otherwise. The objective function (2.1) tries to maximize the number of vertices in the balanced subgraph. Constraints (2.2) ensure that the

endpoints vertices u and v of any parallel edge cannot belong together to the balanced subgraph. Constraints (2.3), denoted as odd negative cycle inequalities, prohibit cycles with an odd number of negative edges in the subgraph.

In this work, a different formulation is proposed based on the construction of two disjoint sets $V_1, V_2 \subseteq V$, aiming to maximize the number of vertices in the resulting balanced signed graph $G_{V_1 \cup V_2}$. Our main objective is to employ a polynomial number of constraints in order to reach a polynomial bounded formulation. Thus, we define the binary variables x_v for all $v \in V$, such that $x_v = 1$ if and only if vertex v belongs to the set V_1; otherwise, $x_v = 0$. Similarly, for all $v \in V$, we define binary variables $z_v = 1$ if vertex v belongs to the set V_2; otherwise, $z_v = 0$. In this context, we have formulated the MBSP as the following integer programming problem:

$$\max \sum_{v \in V} (x_v + z_v) \tag{2.5}$$

subject to:

$$x_v + z_v \leq 1, \quad \forall v \in V, \tag{2.6}$$
$$x_u + x_v \leq 1, \quad \forall \{u, v\} \in E^-, \tag{2.7}$$
$$z_u + z_v \leq 1, \quad \forall \{u, v\} \in E^-, \tag{2.8}$$
$$x_u + z_v \leq 1, \quad \forall (u, v) \in A^+, \tag{2.9}$$
$$x_u + z_u + x_v + z_v \leq 1, \quad \forall \{u, v\} \in E^+ \cap E^-, \tag{2.10}$$
$$x_v \in \{0, 1\}, \quad \forall v \in V, \tag{2.11}$$
$$z_v \in \{0, 1\}, \quad \forall v \in V, \tag{2.12}$$

where $A^+ = \{(u, v), (v, u) \mid \{u, v\} \in E^+\}$.

The above model is said to be a *clustering formulation*. The objective function (2.5) maximizes the total number of vertices in $V_1 \cup V_2$. Constraints (2.6) indicate that any vertex in the solution has to belong to only one set, V_1 or V_2. Constraints (2.7) and (2.8) ensure that there are no negative edges with both endpoints in V_1 and V_2, respectively. Moreover, constraints (2.9) indicate that there are no positive edges that have one endpoint in V_1 and the other in V_2. Constraints (2.10) ensure that if there is a parallel edge between two vertices, both vertices cannot be in the solution.

3. A matheuristic for the MBSP

In Figueiredo and Frota [8], a GRASP heuristic was proposed to solve the MBSP and provided the best quality results among the heuristics developed for this problem for the instances analyzed. In addition, the NCCH heuristic presented in [16], obtained the best quality results for the instances used in that work. This section details our proposed matheuristic to tackle the MBSP.

3.1. The NCCH heuristic

The greedy heuristic NCCH proposed in [16] is based on the observation that a vertex set $U \subseteq V$ induces a balanced subgraph G_U if and only if G_U contains no negative cycles, *i.e.*, cycles with an odd number of negative edges [2]. Therefore, this greedy approach tries to reduce negative cycles progressively by applying the Negative Cycle Contraction (NCC) rule shown in Algorithm 1 [16]. This procedure preserves the sign of all negative cycles and shortens those containing an edge $\{u, v\}$ that is not a parallel edge. Any balanced induced subgraph of the signed graph obtained by applying the NCC-rule to all the vertices of V is a balanced induced subgraph of G.

Algorithm 2 presents the NCCH heuristic using this rule.

Algorithm 1: NCC_RULE.

Input: The signed graph $G = (V, E, s)$ and edge $\{u, v\} \in E$
1 **if** $s(\{u, v\}) \neq \pm$ **then**
2 **foreach** $w \in V \setminus \{v\}$ *adjacent to* u, *where* $s(\{u, w\}) \neq \pm$ **do**
3 $* \leftarrow s(\{u, v\}) \cdot s(\{u, w\})$
4 **if** $\{v, w\} \notin E^*$ **then**
5 $E^* \leftarrow E^* \cup \{v, w\}$
6 $E = E \backslash \{u, v\}$

Algorithm 2: NCCH.

Input: The signed graph $G = (V, E, s)$, function ω.
Output: The balanced subgraph G_U.
1 $S \leftarrow V$
2 $U \leftarrow \emptyset$
3 **while** $S \neq \emptyset$ **do**
4 $u \leftarrow$ the vertex in S with the smallest positive value of ω
5 Apply NCC-rule to every vertex v adjacent to u
6 **foreach** v *adjacent to* u **do**
7 **if** $s(\{u, v\}) = \pm$ **then**
8 delete v from S
9 $U \leftarrow U \cup \{u\}$
10 $G_U \leftarrow (U, E[U], s)$
11 **return** G_U

This greedy procedure runs from lines 3 to 9, while there are vertices from S to be examined. In each step, it selects a vertex u by using a greedy criterion defined by the function ω (line 4). Then it applies the NCC rule to every vertex v adjacent to u (line 5). Later (lines 6–8), the method deletes all parallel edges generated after applying the NCC-rule. At the end of Algorithm 2 (line 10), the vertex set U induces a balanced signed graph (see details in [16]).

The function ω should prioritize vertices that have few and small negative cycles without chords associated with them. Marinelli and Parente [16] proposed three options for ω, and showed that the performance of the heuristic varies depending on the ω function and instance structure.

3.2. Construction procedure based on negative chordless cycles

Based on the NCCH heuristic, we developed a construction procedure using a Restricted Candidate List (RCL) [19]. This procedure, shown in Algorithm 3, is similar to the above algorithm. The only difference between the original NCCH heuristic and this new one is that the former greedily selects the vertex using the function ω, while the latter creates a list L to select the vertex from that list randomly. The list L contains all vertices with the minimum value for the ω function.

Choosing a random vertex from this list at each step causes the final solution to differ from the solution obtained by the NCCH heuristic. Thus, Algorithm 3 can be used to generate different feasible solutions. In this work $\omega(v) = d_G(v)$ was selected, where $d_G(v)$ represents the degree of vertex $v \in V$. This is one of the functions proposed in [16], and it was chosen because it demands less computational time to be calculated.

Algorithm 3: CONSTRUCTION_NCCH.

Input: The signed graph $G = (V, E, s)$, function ω.
Output: The balanced subgraph G_U.

```
1  S ← V
2  U ← ∅
3  while S ≠ ∅ do
4      L ← {v ∈ S | ω(v) ≤ ω(u),   ∀u ∈ S}
5      u ← random element from L
6      Apply NCC-rule to every vertex v adjacent to u
7      foreach v adjacent to u do
8          if s({u, v}) = ± then
9              delete v from S
10     U ← U ∪ {u}
11 G_U ← (U, E[U], s)
12 return G_U
```

3.3. Local Search

The purpose of this procedure is to find locally optimal solutions with respect to some neighborhood in order to improve the initial solution provided by Algorithm 3. The local search in [8] obtains neighboring solutions by removing one or two vertices from the current solution, and including as many vertices as possible (maintaining feasibility), in order to reach an improved solution.

If we consider the removal of all possible pairs of elements, a quadratic component is incorporated in the local search. This neighborhood would be computationally expensive in large graphs. So, our local search procedure obtains neighboring solutions by randomly removing $p \leq |U'|$ vertices from a feasible solution $G_{U'} = (U', E[U'], s)$ (shrinking phase) and reapplying the NCCH heuristic shown in Algorithm 2 (expansion phase). The parameter p is determined by the input parameter α that indicates a percentage of the number of vertices of the input graph G (line 2). Thereby, we define the p-neighborhood, denoted by N_p, as the family of all solutions obtained by removing p vertices from U' and reapplying the NCCH heuristic described in Algorithm 2. As the number of neighbors grows exponentially, only k neighbors in N_p are analyzed in each local search step.

The local search procedure, shown in Algorithm 4, tries to increase the solution's cardinality returned by the construction phase. Each time it finds an enhancement, the current best solution G_{best} is updated (lines 6–8). The method stops if there is no further improvement of the solution after exploring k neighbors in N_p.

3.4. The MS_NCCH heuristic

Using the constructive procedure and the local search algorithm we propose the MS_NCCH heuristic described in Algorithm 5, based on a multi-start schema.

Each MS_NCCH iteration starts with an initial solution generated for the problem (lines 6–9). For the first iteration, we use the greedy NCCH heuristic described in Algorithm 2. Then, we use the constructive procedure defined in Section 3.2 for the remaining iterations. Once a feasible solution is created, the local search tries to improve the quality of that solution (line 10). The best global solution G_{best} is updated (line 12) if the solution found by the local search improves it. The method maintain a pool of elite solutions EL (line 13) that is updated with new solutions found in the local search procedure that are better than those in EL, respecting the size limit $EliteSize$ of EL. That implies that all neighboring solutions obtained in the local search stage are considered. As EL has a limited size, when a solution has a better value than the worst solution of EL, the new solution enters the set, while the worst solution leaves EL.

From lines 14 to 17 the set of elite solutions is checked. If some new solution enters the set, then the counter for the number of iterations without change in EL is restarted.

Algorithm 4: LOCAL_SEARCH.

Input: The balanced subgraph G, a maximum number of attempts k and the percentage of vertices to remove α.
Output: The balanced subgraph G_{best} with at least as many vertices as G.

$attempt \leftarrow 0$
$G_{best} \leftarrow G$
$p = \alpha \cdot |G|$
while $attempt < k$ **do**
  $G' \leftarrow$ random neighbor of G_{best} in N_p
  if $|G'| > |G_{best}|$ **then**
    $G_{best} \leftarrow G'$
    $attempt \leftarrow 0$
  else
    $attempt \leftarrow attempt + 1$
return G_{best}

Algorithm 5: MS_NCCH

Input: graph G, $MaxTime$, $MaxIter$, $MaxIterElite$, k, α.
Output: The balanced subgraph G_{best}.

$G_{best} \leftarrow \emptyset$
$iter \leftarrow 1$
$it \leftarrow 1$
$EL \leftarrow \emptyset$
repeat
  if $iter = 1$ **then**
    $G_0 \leftarrow$ NCCH(G, ω)
  else
    $G_0 \leftarrow$ CONSTRUCTION_NCCH(G, ω)
  $G^* \leftarrow$ LOCAL_SEARCH(G_0, k, α)
  if $|G^*| > |G_{best}|$ **then**
    $G_{best} \leftarrow G^*$
  Update EL with new solutions found in the local search that are better than those in EL.
  if *there is a change in* EL **then**
    $it \leftarrow 1$
  else
    $it \leftarrow it + 1$
  $iter \leftarrow iter + 1$
until $MaxTime$ *limit reached* ***or*** $iter > MaxIter$ ***or*** $it > MaxIterElite$
return G_{best}

The algorithm stops if it reaches the maximum running time $MaxTime$, the maximum number of overall iterations $MaxIter$, or there are no changes in the elite set during $MaxIterElite$ iterations.

3.5. The matheuristic MH_MBSP

The MH_MBSP is a matheuristic developed using the proposed formulation presented in Section 2 and the solutions inserted in the elite set by the MS_NCCH heuristic. The proposed compact formulation can be described using a polynomial number of variables and constraints. This feature allows the formulation to be integrated with the MS_NCCH heuristic in order to quickly find feasible solutions after fixing a subset of variables. The quality of solution depends on the choice of which and how many variables will be selected to be fixed.

In our first approach we try to identify a subset of vertices $U \subseteq V$ that appear in all solutions of the elite set EL in the end of the method MS_NCCH. Then, we include Constraints 3.1 into the formulation, indicating that these vertices must be present in the solution:

$$x_v + z_v = 1, \quad \forall v \in U. \tag{3.1}$$

Unfortunately, this approach turned out to be very restrictive to the formulation, not improving the solutions. So, we decided to use the subset of vertices that appear in all solutions that were ever inserted in the elite set. This strategy improved the results and is adaptive for each instance, as the number of solutions inserted in the elite set varies depending on the instance.

Algorithm 6: MH_MBSP.

Input: The signed graph $G = (V, E, s)$, $MaxTime$, $MaxIter$, $MaxIterElite$, k, α
1 $G_{best} \leftarrow$ MS_NCCH$(G, MaxTime, MaxIter, MaxIterElite, k, \alpha)$
2 $U \leftarrow$ Subset of vertices presented in all solutions ever inserted in the elite set EL
3 $timeMS \leftarrow$ Time used by the MS_NCCH heuristic
4 **if** $timeMS >= MaxTime$ **then**
5 **return** Sol
6 $M \leftarrow$ The formulation described in Section 2 for the input graph G, with Constraints 3.1 defined by vertex set U
7 $G'_{best} \leftarrow$ Solution found by solving M with a time limit of $(MaxTime - timeMS)$
8 **if** G'_{best} *is better than* G_{best} **then**
9 **return** G'_{best}
10 **else**
11 **return** G_{best}

Algorithm 6 shows the MH_MBSP matheuristic for the MBSP problem. It starts by storing the best solution obtained by the MS_NCCH heuristic (line 1) and the subset of vertices presented in all solutions ever inserted in the elite set (line 2).

If the time used by the heuristic reached the time limit $MaxTime$, it returns the solution found by the MS_NCCH heuristic (line 5); otherwise, the model M is created according to Section 2 (line 6), and expanded by Constraints 3.1 defined by vertex set U, indicating that these vertices must be fixed in the solution, belonging to any of solution sets V_1 or V_2.

Subsequently, the formulation is solved using $(MaxTimeMH - timeMS)$ as the time limit (line 7). The matheuristic result is the best solution between the one found by the MS_NCCH heuristic and the one found by solving the model with the initialized variables (lines 8–11).

4. Computational experiments

This section reports the computational experiments carried out with the heuristics methods described in the previous sections. We performed several computational experiments to validate the effectiveness of the proposed methods. First, we analyzed the same set of instances of [8], which contains several groups of instances (DMERN, Portfolio, and Random) having signed graphs with different sizes and densities. Then, we analyzed the set of instances proposed in [16], which contains three groups of instances (N, P, and R instances).

4.1. Exact method

The new proposed mathematical formulation described in Section 2 is the starting point for our exact method used in the matheuristic. We executed it for an hour to verify its performance. Experiments for the exact method were executed on an Intel (R) Core i5-4460S CPU @ 2.90 GHz and 8 Gb of RAM using the operating system

Fedora 22. All methods were programmed in C++ language using the gcc compiler. The IBM ILOG CPLEX 12.6 was used with a single thread of execution, and all other CPLEX parameters were left to their default values.

In [8], the authors showed the results obtained by their exact method based on removing cycles with an odd number of negative edges. These experiments were executed on an Intel (R) Pentium (R) 4 CPU 3.06 GHz, equipped with 3 GB of RAM. In order to establish a more similar environment, we calculated the associated time scaling factor of 0.34 (based on Pass mark benchmark), representing the ratio between the speed of the processor in [8] and the speed of our processor. Based on this, we show their time values multiplied by this associated factor.

Table 1 shows the results obtained for instances set DMERN, composed of 61 instances with the number of vertices $|V|$ varying from 144 to 8317. Columns 2, 3, and 4 present, respectively, the upper bound provided by the root node relaxation, the best lower bound value, and the time to reach this value for the exact model shown in [8]. Columns 5, 6, and 7 have the same meaning, but for our exact method. The symbol "–" means that the optimal solution was not found within the execution time limit.

According to Table 1, our method could solve nine more instances than the method in [8]. On the other hand, the new formulation presents a much weaker relaxation at the root node. However, the resolution times were short, and our exact method presents high-quality lower bounds (feasible solutions). Together with the fact that the new formulation is compact, these features indicate that our proposed formulation may be a good choice to be integrated with a heuristic to search for high-quality and fast feasible solutions.

4.2. Choosing parameters

We used the Irace package [15] to select the parameters for the MS_NCCH heuristic. This package includes the iterated F-race algorithm [3] and several extensions for finding the most appropriate parameter settings for an optimization algorithm. It needs the specification of the parameters to be configured and instances for training. According to Section 3.4, the parameters to be configured are:

- α: The percentage of vertices to be removed from the solution set in the local search phase. Its values ranged from 5% to 50%.
- k: The maximum number of neighbors explored during the local search. Its values were in the range of 10 to 1000.
- it: The number of iterations without a change in the elite set, tested in the range of 1 to 50.

We configure the Irace to run 1000 experiments using 10 instances, none of them present in previous works or used in later comparisons. The instances used were:*boeing1*, *boeing2*, *capri*, *degen2*, *file150.7*, *finnis*, *scfxm1*, *scfxm3*, *scsd6* and *standata*, coming from a set of general mixed integer programs in Netlib (https://www.netlib.org/lp/data). The instances were selected with different sizes and densities to provide parameters that work well with different input structures.

From the results of Irace, the final configurations were: $\alpha = 33.3\%$, $k = 1000$ and $it = 20$. The parameter $EliteSize$ was set to 10, as used in Martins *et al.* [18] and de Holanda Maia *et al.* [6]. For all experiments, we used $MaxIter = 100$ and $MaxTime = 30$ s.

4.3. Analyzing the performance of the heuristics

Tests were developed on an Intel(R) Core(TM) i5-7200U CPU @ 2.50 GHz with 8 Gb of RAM using Ubuntu 16.04. All methods were programmed in C++ language using the gcc compiler. We used the instances presented in [16] to analyze the NCCH, MS-NCCH, and MH-MBSP heuristics performance. The goal of these tests was to verify the impact on the solution of each developed strategy. For all heuristics, the function w is the vertex degree.

The three sets of instances are N-instances, P-instances, and R-instances. N-instances consist of 34 signed graphs obtained from a set of preprocessed and scaled Netlib matrices. P-instances are 50 randomly generated

TABLE 1. Exact results for instance set DMERN.

Instance	Relax_C	ILP_C	Time_C	Relax	ILP	Time
danoint	126	97	165.0	136	97	0.5
bienst1	118	90	2772.0	184	90	1.0
stein45	30.17	29	–	331	30	0.1
disctom	397	299	643.0	399	299	44.0
fc.60.20.1	384	371	171.0	414	371	7.3
air05	240.5	63	–	426	69	–
neos17	2	2	61.0	485	2	0.4
p100x588	638	633	63.0	688	633	0.4
air04	383	113	–	823	133	–
r80x800	840	828	699.0	880	828	19.1
nug08	301	128	150.0	912	128	0.2
p50x864	889	884	107.0	914	884	4.4
n5-3	941	912	82.0	1012	912	0.3
neos21	924	191	782.0	1085	191	0.4
n4-3	1109	1062	165.0	1178	1062	0.7
dano3mip	1150	588	–	1203	760	–
n8-3	1197	1176	120.0	1300	1176	0.3
neos20	595.5	627	107.0	1320	627	0.1
p200x1188	1288	1272	–	1388	1273	–
p200x1188c	1288	1273	–	1388	1276	365.1
roll3000	583	824	170.0	1205	824	0.9
janos-us-ca–D-D-M-N-C-A-N-N	1554	1521	214.0	1643	1521	0.5
pioro40–D-B-M-N-C-A-N-N	1573	1560	126.0	1649	1560	0.5
n13-3	1562	1537	214.0	1661	1537	0.7
n2-3	1669	1656	259.0	1752	1656	0.7
zib54–U-U-E-N-C-A-N-N	1728	1728	62.0	1728	1728	0.5
qap10	200	200	427.0	1820	200	1.9
ns1688347	468.43	375	–	1806	448	–
germany50–U-U-M-N-C-A-N-N	2000	2000	91.0	2000	2000	0.4
protfold	611.61	383	–	1990	574	–
cap6000	2074	2074	111.0	2169	2074	103.7
n7-3	2197	2162	1191.0	2278	2162	2.4
n9-3	2187	2112	1330.0	2280	2112	4.6
acc-1	567	480	–	2286	491	41.3
n3-3	2192	2059	2827.0	2303	2059	8.5
zib54–D-B-E-N-C-A-N-N	2277	2268	211.0	2347	2268	0.8
n12-3	2265	2214	1067.0	2358	2214	3.3
neos818918	2019.83	2018	800.0	2350	2018	6.2
germany50–D-B-M-N-C-A-N-N	2355	2350	262.0	2438	2350	0.8
acc-2	567	454	–	2520	491	60.0
ta2–U-U-M-N-C-A-N-N	2470	2471	175.0	2470	2471	0.5
n6-3	2602	2538	2785.0	2686	2538	5.1
berlin	2678	2653	–	2704	2653	44.4
neos11	752.11	702	–	2686	742	3.3
ta2–D-B-M-N-C-A-N-N	2748	2731	470.0	2837	2731	2.9
acc-6	1011.28	899	–	3040	939	–
acc-5	1002.93	875	–	3045	944	–
acc-3	1125	1125	221.0	3240	1071	–

TABLE 1. continued.

Instance	Relax_C	ILP_C	Time_C	Relax	ILP	Time
acc-4	1125	1125	240.0	3276	1071	–
brasil	3335	3307	–	3364	3307	32.6
mkc	2964	3220	341.0	3124	3223	0.5
p500x2988	3238	3199	–	3488	3206	–
p500x2988c	3238	3098	–	3488	3194	–
mod011	3208	3854	416.0	3240	3854	0.8
neos1	1194	1099	–	2712	1186	29.3
seymour	472.04	559	–	4794	610	–
seymour1	472.04	599	–	4794	610	–
n370a	5100	5100	1374.0	5150	5100	1.4
rentacar	3280	4662	2403.0	4257	4693	1.5
manna81	3382	2322	1178.0	6480	2322	–
neos12	1546.41	1401	–	8275	1546	–

power-law signed graphs, five instances for each size $|V| = 100k$ with $k \in \{1, \ldots, 10\}$. R-instances are random signed graphs generated with the function $RandomGraph(n, p)$ provided by Mathematica [20], with the number of vertices ranging from 50 to 200 and density in the set $\{0.1, 0.3, 0.5, 0.7\}$.

Table 2 shows the results for N-instances. Column 2 shows the results of the greedy heuristic NCCH. The time in seconds, used by this heuristic, is presented in Column 3. Columns 4 and 5 show the average result and time of MS-NCCH heuristic, respectively, after 10 runs. Next two columns show the same information but for the MH-MBSP matheuristic. The final column shows the gap between the best solution found by the exact method (ILP) and the average result found by the matheuristic (Col. 6). The gap is calculated by the following equation:

$$\text{gap} = \text{Best}_{\text{ILP}} - \text{Avg}_{\text{MH-MBSP}}$$

where Best_{ILP} is the best solution found by our exact method in the time limit of 1 h.

Since the exact optimization phase of the matheuristic starts when the MS-NCCH heuristic finishes, the results and time of MS-NCCH are always worse and less than or equal to those obtained by MH-MBSP. For the same reason, the NCCH results are always worse than or equal to those obtained by the MS-NCCH heuristic. Cells with an asterisk in the last column indicate that the problem was not optimally solved in 1 h and the gap is calculated related to the best feasible solution found. Moreover, the value "–" in the time column indicates that the method reached the maximum assigned time of 30 s.

The N-instances set contains "easy" instances, and the greedy heuristic NCCH was able to solve optimally 22 of the 34 instances. For those 12 instances not solved optimally by the NCCH heuristic, the MS-NCCH heuristic solved optimally eight new instances and improved the results for other two unsolved instances. The exact optimization phase of the matheuristic did not improve the results of the remaining four unsolved instances. When analyzing these instances, we can see that they are instances in which the MS-NCCH heuristic used all execution time, and there was no available time to perform the exact optimization phase in three of them.

Table 3 shows the results for P-instances. Columns of this table have the same meaning as in Table 2. These instances are a little more complex than N-instances, and the results reflect that. In this set, the NCCH heuristic found seven optimal results. For the remaining 43 instances, the MS-NCCH heuristic always obtained a better result than the NCCH heuristic and found 10 additional optimal solutions. For the remaining 33 unsolved instances, the matheuristic MH-MBSP improved all results and found 19 new optimal values.

The R-instances are classified in [16] as hard instances because they are graphs connected with many negative cycles. Table 4 shows the results for these instances, and its columns have the same meaning as in the previous tables. The MS-NCCH heuristic improved all results found by the NCCH heuristic. Moreover, the MS-NCCH

TABLE 2. Heuristics results for N-instances.

	NCCH		MS-NCCH		MH-MBSP		
Instance	Result	Time	Avg	Time	Avg	Time	Gap
25fv47	212	0.00	212.0	0.22	212.0	0.23	0.00
80bau3b	1328	0.01	1329.1	12.72	1329.1	12.76	16.90
agg2	62	0.00	62.0	0.09	62.0	0.10	0.00
agg3	62	0.00	62.0	0.09	62.0	0.10	0.00
bnl1	261	0.00	261.0	0.23	261.0	0.24	0.00
bnl2	1364	0.01	1367.0	1.25	1367.0	1.28	0.00
cycle	504	0.00	504.0	1.06	504.0	1.06	0.00
czprob	717	0.00	717.0	–	717.0	–	1.00
d2q06c	802	0.00	804.0	0.45	804.0	0.47	0.00
degen3	776	0.01	796.0	2.83	796.0	3.39	0.00
dfl001	4175	0.57	4409.1	–	4409.1	–	*103.90
fffff800	125	0.00	125.0	0.29	125.0	0.30	0.00
ganges	534	0.00	534.0	1.96	534.0	1.98	0.00
gfrd-pnc	522	0.00	522.0	1.77	522.0	1.82	0.00
greenbea	884	0.00	890.0	3.92	890.0	3.95	0.00
greenbeb	884	0.00	890.0	3.96	890.0	3.98	0.00
maros	300	0.00	300.0	0.37	300.0	0.38	0.00
modszk1	130	0.00	130.0	0.02	130.0	0.02	0.00
nesm	190	0.00	190.0	0.17	190.0	0.17	0.00
perold	143	0.00	143.0	0.07	143.0	0.08	0.00
pilot87	306	0.00	306.0	0.39	306.0	0.40	0.00
pilot	252	0.00	252.0	0.24	252.0	0.24	0.00
pilot.ja	198	0.00	198.0	0.13	198.0	0.14	0.00
pilotnov	203	0.00	203.0	0.13	203.0	0.13	0.00
pilot.we	202	0.00	202.0	0.11	202.0	0.12	0.00
scfxm2	250	0.00	252.0	0.03	252.0	0.05	0.00
sctap2	470	0.00	470.0	0.84	470.0	0.84	0.00
seba	140	0.00	140.0	1.04	140.0	1.24	0.00
shell	480	0.00	482.0	4.13	482.0	4.15	0.00
ship12l	732	0.00	732.0	6.14	732.0	6.23	0.00
ship12s	360	0.00	360.0	1.31	360.0	1.34	0.00
stocfor2	1104	0.01	1106.0	3.07	1106.0	3.11	0.00
stocfor3	8512	0.42	8513.2	–	8513.2	–	2.80
woodw	301	0.00	301.0	0.24	301.0	0.24	0.00

heuristic found 27 optimal solutions. The exact optimization phase of the matheuristic improved the MS-NCCH heuristic results in 14 instances and found six new optimal solutions. The MH-MBSP average results improved the NCCH results by 31.0%. A remarkable result of the matheuristics MH-MBSP was for instance *rn200d05-2*, as it obtained, in 7.65 s, a better result than the exact method in one hour of processing.

The above experiments showed the impact that each NCC-rule-based heuristic has on the quality of the solutions. The next section compares the MH-MBSP matheuristic with the heuristic GRASP developed in [8], which is the best-known metaheuristic for the problem.

4.4. DMERN, portfolio and random graphs

This section compares the MH-MBSP matheuristic to the GRASP method proposed in [8], which is the best metaheuristic in the literature for the problem. As the authors kindly provided the GRASP source code, we

TABLE 3. Heuristics results for P-instances.

	NCCH		MS-NCCH		MH-MBSP		
Instance	Result	Time	Avg	Time	Avg	Time	Gap
g0100_1	93	0.00	94.0	0.06	94.0	0.08	0.00
g0100_2	94	0.00	95.0	0.05	95.0	0.06	0.00
g0100_3	96	0.00	97.0	0.02	97.0	0.03	0.00
g0100_4	95	0.00	95.0	0.05	95.0	0.05	0.00
g0100_5	94	0.00	94.0	0.04	94.0	0.05	0.00
g0200_1	188	0.00	190.0	0.02	190.0	0.04	0.00
g0200_2	188	0.00	190.0	0.03	190.0	0.13	0.00
g0200_3	188	0.00	188.0	0.01	188.0	0.03	0.00
g0200_4	190	0.00	190.0	0.12	190.0	0.13	0.00
g0200_5	189	0.00	190.0	0.04	190.0	0.07	0.00
g0300_1	281	0.00	283.9	0.11	284.0	0.27	0.00
g0300_2	281	0.00	283.5	0.05	284.0	0.41	0.00
g0300_3	283	0.00	285.0	0.09	285.0	0.24	0.00
g0300_4	283	0.00	283.0	0.31	283.0	0.33	0.00
g0300_5	283	0.00	285.0	0.04	285.0	0.10	0.00
g0400_1	377	0.00	379.7	0.08	380.0	0.41	0.00
g0400_2	379	0.00	380.0	0.26	380.0	0.51	0.00
g0400_3	377	0.00	378.5	0.11	379.0	0.52	0.00
g0400_4	381	0.00	381.0	0.20	381.0	0.23	0.00
g0400_5	378	0.00	380.0	0.10	380.0	0.27	0.00
g0500_1	471	0.00	474.8	0.12	475.0	0.67	0.00
g0500_2	473	0.00	476.6	0.16	477.0	0.47	0.00
g0500_3	470	0.00	472.4	0.15	473.0	1.23	0.00
g0500_4	473	0.00	473.0	0.16	473.0	0.50	0.00
g0500_5	476	0.00	476.9	0.13	477.0	0.27	0.00
g0600_1	564	0.00	567.5	0.27	568.0	6.36	0.00
g0600_2	569	0.00	571.2	0.24	571.9	1.40	0.10
g0600_3	566	0.00	566.8	0.25	568.0	12.86	0.00
g0600_4	567	0.00	568.3	0.21	569.0	2.26	0.00
g0600_5	567	0.00	570.6	0.24	571.0	0.80	0.00
g0700_1	661	0.00	665.5	0.39	666.0	2.00	0.00
g0700_2	658	0.00	667.5	0.33	668.0	1.27	0.00
g0700_3	661	0.00	662.5	0.29	663.0	3.30	0.00
g0700_4	657	0.00	661.0	0.33	662.0	1.77	0.00
g0700_5	664	0.00	664.0	0.36	664.5	0.82	0.50
g0800_1	750	0.00	754.6	0.44	756.6	16.98	0.40
g0800_2	751	0.00	756.5	0.44	758.9	–	0.10
g0800_3	759	0.00	761.9	0.46	763.0	2.52	0.00
g0800_4	757	0.00	758.6	0.52	759.0	6.15	0.00
g0800_5	754	0.00	759.3	0.39	760.0	4.15	0.00
g0900_1	844	0.00	849.6	0.58	851.7	–	0.30
g0900_2	845	0.00	850.5	0.57	852.8	–	0.20
g0900_3	848	0.00	852.8	0.48	854.4	14.52	0.60
g0900_4	846	0.00	849.5	0.59	852.8	20.59	0.20
g0900_5	848	0.00	851.3	0.60	852.9	–	0.10
g1000_1	937	0.00	942.0	0.59	945.2	–	*0.80
g1000_2	941	0.00	946.8	0.59	949.0	–	*1.00
g1000_3	940	0.00	944.6	0.68	947.3	–	*0.70
g1000_4	942	0.00	947.2	0.80	948.7	26.93	0.30
g1000_5	945	0.00	948.5	0.51	949.8	2.58	0.20

TABLE 4. Heuristics results for R-instances.

	NCCH		MS-NCCH		MH-MBSP		
Instances	Result	Time	Avg	Time	Avg	Time	Gap
rn50d01-1	39	0.00	41.0	0.06	41.0	0.07	0.00
rn50d01-2	35	0.00	40.0	0.01	40.0	0.14	0.00
rn50d01-3	36	0.00	43.0	0.06	43.0	0.07	0.00
rn50d03-1	18	0.00	33.0	0.08	33.0	0.09	0.00
rn50d03-2	32	0.00	33.0	0.08	33.0	0.09	0.00
rn50d03-3	31	0.00	32.9	0.03	32.9	0.04	0.10
rn50d05-1	27	0.00	30.0	0.08	30.0	0.09	0.00
rn50d05-2	27	0.00	28.0	0.05	28.0	0.06	0.00
rn50d05-3	29	0.00	30.0	0.08	30.0	0.09	0.00
rn50d07-1	23	0.00	26.0	0.07	26.0	0.08	0.00
rn50d07-2	24	0.00	27.0	0.08	27.0	0.09	0.00
rn50d07-3	11	0.00	24.9	0.02	24.9	0.05	0.10
rn100d01-1	60	0.00	75.0	0.33	75.0	0.40	0.00
rn100d01-2	52	0.00	72.0	0.16	72.0	0.42	0.00
rn100d01-3	54	0.00	78.0	0.32	78.0	0.37	0.00
rn100d03-1	40	0.00	54.0	0.29	54.0	0.35	0.00
rn100d03-2	46	0.00	56.0	0.19	56.0	0.21	0.00
rn100d03-3	40	0.00	50.0	0.08	50.0	0.52	0.00
rn100d05-1	38	0.00	43.9	0.04	44.0	0.09	0.00
rn100d05-2	21	0.00	41.0	0.14	41.0	1.16	0.00
rn100d05-3	13	0.00	42.0	0.07	42.0	1.73	0.00
rn100d07-1	31	0.00	38.4	0.28	39.0	1.77	0.00
rn100d07-2	36	0.00	39.1	0.22	39.1	0.34	0.90
rn100d07-3	26	0.00	36.3	0.14	36.6	2.06	0.40
rn150d01-1	71	0.00	96.0	0.87	96.0	1.18	0.00
rn150d01-2	91	0.00	103.0	0.80	103.0	0.83	0.00
rn150d01-3	93	0.00	103.0	0.85	103.0	0.87	0.00
rn150d03-1	58	0.00	71.0	0.24	71.0	0.30	0.00
rn150d03-2	54	0.00	67.0	0.45	67.0	1.26	0.00
rn150d03-3	41	0.00	67.8	0.31	68.0	2.64	0.00
rn150d05-1	40	0.00	55.5	0.26	56.0	5.77	0.00
rn150d05-2	46	0.00	55.7	0.11	55.9	1.88	0.10
rn150d05-3	30	0.00	52.2	0.26	53.0	5.64	0.00
rn150d07-1	25	0.00	45.1	0.28	45.3	20.61	0.70
rn150d07-2	25	0.00	46.8	0.17	47.0	8.68	0.00
rn150d07-3	31	0.00	44.8	0.20	44.9	8.22	1.10
rn200d01-1	79	0.00	119.0	0.56	119.0	0.79	0.00
rn200d01-2	121	0.00	131.0	1.88	131.0	1.92	0.00
rn200d01-3	102	0.00	121.0	0.76	121.0	0.83	0.00
rn200d03-1	59	0.00	74.3	0.26	74.9	3.46	*0.10
rn200d03-2	61	0.00	76.9	0.31	77.3	5.27	0.70
rn200d03-3	53	0.00	77.9	0.40	77.9	14.58	*0.10
rn200d05-1	45	0.00	59.4	0.25	59.4	26.76	*0.60
rn200d05-2	45	0.00	58.8	0.24	59.1	7.65	*-1.10
rn200d05-3	46	0.00	61.7	0.29	61.7	22.00	*0.30
rn200d07-1	33	0.00	48.0	0.12	48.0	26.56	*2.00
rn200d07-2	28	0.00	48.9	0.15	48.9	22.31	*0.10
rn200d07-3	37	0.00	52.0	0.24	52.0	22.45	*0.00

ran both heuristics in the same machine and executed each instance ten times with the same time limit as MH-MBSP (30 s).

Table 5 shows the results obtained for the set of instances DMERN. Columns 2 and 3 show the average GRASP solution and the gap between the best solution found by the exact method (ILP) and the average result found by the GRASP. Columns 4 and 6 have the same meaning as columns 2 and 3 but for the MH-MBSP matheuristic. Moreover, column 5 shows the average execution time for MH-MBSP, because the optimization phase could stop before the time limit. Those results in bold font are strictly better than the results from the other methods. An asterisk in the gap cells indicates that the exact method could not find an optimal solution in one hour, and the gap is calculated using the best solution found. The value "–" in the time cells indicates that the method reached the maximum assigned time of 30 s.

Table 5 shows that the GRASP heuristic obtained the best average results for two instances, which are the smallest in the group and correspond to graphs with 144 vertices (danoint) and 184 vertices (bienst1). In these cases, the GRASP heuristic always found the optimal value, while the matheuristic MH-MBSP found the optimal value sometimes. For the 53 remaining instances, the MH-MBSP got better results than GRASP.

In general, as the instances' size increases, it is more difficult for GRASP to find good quality solutions in a limited time. It may happen because GRASP has a strictly quadratic and more intensive local search. Thus, for small instances, this more intensive search can find good values in a short time, but it would take much more time to achieve a better result for larger ones. In contrast, our local search explores a limited number of neighbors that is a fraction of the total number of vertices. Thus, we work with a local search with a linear behavior, which obtains good quality results for large instances without spending so much time.

We also analyzed the performance of the proposed methods on the Portfolio instances set, which is composed of instances with the number of vertices $|V|$ varying in the set $\{390, 420, 450, 480, 510\}$, and the threshold value t used in the creation of the instances varying in the set $\{0.300, 0.325, 0.350, 0.375, 0.400\}$ (see Huffner *et al.* [14] for more details). For each combination of these values, there are ten different signed graphs, resulting in 250 instances.

Table 6 shows the results for the Portfolio instances. This table is presented in a format similar to that used by Figueiredo and Frota [8]. In this case, each line represents a group of 10 instances generated with the same threshold t. Columns 3–7 in this table have the same meaning as columns 2–6 in Table 5.

The matheuristics MH-MBSP always got a better result than the GRASP heuristic. Moreover, the value obtained by the matheuristic was very accurate as the gap column values are at a maximum distance of 1 from the optimal solution. In three groups of instances, the matheuristic MH-MBSP found all optimal values. For instances with 420 vertices and parameter $t = 350$, it found a better solution than the one found by the exact algorithm in one hour of processing. Additionally, the average resolution time obtained by MH-MBSP was much less than the time limit of 30 s.

In the last experiment, we consider the results obtained for the Random set of instances, which is composed by randomly generated instances classified into two groups. Group 1 is the set of random signed graphs without parallel edges ($E^- \cap E^+ = \emptyset$), with the number of vertices $|V|$ varying in the set $\{50, 100, 150, 200\}$, density d varying in the set $\{0.25, 0.50, 0.75\}$, and $|E^-|/|E^+|$ varying in the set $\{0.5, 1.0, 2.0\}$. For each combination of these values, there are 3 different signed graphs, resulting in 108 instances. Group 2 is the set of random signed graphs with parallel edges ($E^- \cap E^+ \neq \emptyset$), with the number of vertices and the density varying as in Group 1, and $|E^- \cap E^+|/|E|$ varying in the set $\{0.25, 0.50, 0.75\}$. For each combination of these values, there are 3 different signed graphs, resulting in other 108 instances.

Table 7 presents the results where each row represents a group of 27 graphs. Similarly to Table 6, we give the average information per group of instances. For these instances, the GRASP heuristic obtained the best average results in all groups. However, the gaps were small for both heuristics, being at an average distance of only 1 from the best value obtained by the exact algorithm. We verified that the time used by the MS-NCCH heuristic is 0.6% of the time used by the exact phase of the MH-MBSP for these instances. We believe that if the local search of the MS-NCCH is more exhaustive for small instances like these, the elite set's quality may improve and accelerate the exact phase.

TABLE 5. Comparisons of heuristics methods on DMERN instances. For Tables 5–7, results in bold font indicate that are strictly better than the results from the other methods.

	GRASP		MH-MBSP		
Instance	Avg	Gap	Avg	Time	Gap
danoint	**97.0**	0.00	96.4	0.45	0.60
bienst1	**90.0**	0.00	89.2	0.48	0.80
stein45	30.0	0.00	30.0	0.62	0.00
disctom	299.0	0.00	299.0	14.47	0.00
fc.60.20.1	366.2	4.80	**371.0**	0.35	0.00
air05	66.8	3.20	**68.5**	16.54	*1.50
neos17	2.0	0.00	2.0	2.64	0.00
p100x588	624.2	8.80	**633.0**	2.21	0.00
air04	122.1	8.90	**129.8**	–	*1.20
r80x800	816.4	11.60	**828.0**	1.40	0.00
nug08	128.0	0.00	128.0	7.08	0.00
p50x864	874.1	9.90	**884.0**	3.95	0.00
n5-3	686.2	225.80	**912.0**	6.14	0.00
neos21	188.1	2.90	**191.0**	5.94	0.00
n4-3	804.6	257.40	**1062.0**	8.21	0.00
dano3mip	629.3	140.70	**649.0**	17.73	*121.00
n8-3	905.1	270.90	**1176.0**	9.31	0.00
neos20	603.7	23.30	**627.0**	2.69	0.00
p200x1188	1250.3	22.70	**1273.0**	9.76	0.00
p200x1188c	1249.8	26.20	**1276.0**	9.50	0.00
roll3000	794.0	30.00	**824.0**	13.71	0.00
janos-us-ca–D-D-M-N-C-A-N-N	1188.8	332.20	**1521.0**	18.16	0.00
pioro40–D-B-M-N-C-A-N-N	1306.5	253.50	**1560.0**	17.50	0.00
n13-3	1230.0	307.00	**1537.0**	25.46	0.00
n2-3	1359.3	296.70	**1656.0**	21.61	0.00
zib54–U-U-E-N-C-A-N-N	1538.9	189.10	**1728.0**	23.95	0.00
qap10	200.0	0.00	200.0	–	0.00
ns1688347	374.0	74.00	**444.7**	–	*3.30
germany50–U-U-M-N-C-A-N-N	1660.5	339.50	**2000.0**	–	0.00
protfold	403.2	169.80	**546.0**	12.19	*27.00
cap6000	2074.0	0.00	2074.0	–	0.00
n7-3	1750.3	411.70	**2162.0**	–	0.00
n9-3	1433.5	678.50	**2112.0**	–	0.00
acc-1	380.5	110.50	**456.0**	–	35.00
n3-3	1382.4	676.60	**2059.0**	–	0.00
zib54–D-B-E-N-C-A-N-N	1985.5	282.50	**2268.0**	–	0.00
n12-3	1715.5	498.50	**2214.0**	–	0.00
neos818918	1753.4	264.60	**2018.0**	2.34	0.00
germany50–D-B-M-N-C-A-N-N	1857.0	493.00	**2350.0**	–	0.00
acc-2	375.5	115.50	**450.9**	–	40.10
ta2–U-U-M-N-C-A-N-N	2036.0	435.00	**2471.0**	–	0.00
n6-3	2099.9	438.10	**2538.0**	–	0.00
berlin	1670.5	982.50	**2653.0**	–	0.00
neos11	637.6	104.40	**741.7**	9.50	0.30
ta2–D-B-M-N-C-A-N-N	2097.2	633.80	**2731.0**	–	0.00
acc-6	776.4	200.60	**952.1**	–	24.90
acc-5	787.6	190.40	**952.9**	–	25.10
acc-3	852.3	272.70	**1125.0**	21.41	0.00

TABLE 5. continued.

	GRASP		MH-MBSP		
Instance	Avg	Gap	Avg	Time	Gap
acc-4	852.4	272.60	**1125.0**	20.74	0.00
brasil	1044.2	2262.80	**3307.0**	–	0.00
mkc	1575.5	1647.50	**3223.0**	23.37	0.00
p500x2988	1081.9	2124.10	**3206.0**	–	0.00
p500x2988c	1077.3	2116.70	**3194.0**	11.86	0.00
mod011	1822.4	2031.60	**3854.0**	–	0.00
neos1	936.0	250.00	**1127.6**	–	58.40
seymour	541.0	69.00	**582.0**	–	*28.00
seymour1	543.1	66.90	**581.0**	–	*29.00
n370a	457.7	4642.30	**5100.0**	–	0.00
rentacar	2109.1	2583.90	**4684.8**	20.89	8.20
manna81	336.8	1985.20	**2322.0**	–	0.00
neos12	219.9	1326.10	**1505.4**	–	40.60

TABLE 6. Comparisons of heuristics methods on Portfolio instances.

		GRASP		MH-MBSP		
V	t	Avg	Gap	Avg	Time	Gap
390	0.300	255.7	29.53	**284.3**	1.8	*0.93
	0.325	282.3	33.45	**315.2**	1.3	*0.55
	0.350	309.2	34.12	**343.1**	1.5	0.20
	0.375	344.7	22.42	**366.3**	1.3	0.78
	0.400	373.5	7.76	**381.2**	1.5	0.12
420	0.300	276.1	23.41	**298.9**	2.5	*0.63
	0.325	299.9	32.57	**332.3**	2.0	*0.23
	0.350	327.3	37.70	**365.0**	1.6	*−0.01
	0.375	356.4	34.78	**391.1**	0.7	0.10
	0.400	400.4	6.77	**407.2**	1.1	0.00
450	0.300	285.2	32.69	**317.5**	2.4	*0.37
	0.325	314.0	40.02	**353.7**	2.1	*0.28
	0.350	356.7	33.20	**389.8**	1.5	*0.08
	0.375	383.4	35.02	**418.4**	0.8	0.00
	0.400	415.9	20.29	**436.2**	1.0	0.00
480	0.300	300.7	33.24	**333.4**	4.3	*0.52
	0.325	325.9	46.67	**371.4**	3.8	*1.18
	0.350	357.3	52.17	**409.2**	2.7	*0.33
	0.375	393.4	48.29	**441.7**	1.9	0.03
	0.400	428.9	34.82	**463.6**	2.2	0.08
510	0.300	316.4	32.71	**349.1**	7.0	*0.01
	0.325	344.6	45.24	**389.3**	5.8	*0.54
	0.350	374.2	55.02	**428.1**	3.9	*1.13
	0.375	423.0	41.41	**464.3**	2.4	*0.07
	0.400	449.1	42.29	**491.4**	3.4	0.00

TABLE 7. Comparisons of heuristics methods on Random instances.

		GRASP		MH-MBSP		
	V	Avg	Gap	Avg	Time	Gap
Group 1	50	**17.89**	0.000	17.74	0.60	0.153
	100	**23.51**	0.005	23.24	18.87	*0.389
	150	**26.43**	0.010	25.88	25.78	*0.819
	200	**28.14**	0.013	27.80	26.81	*0.716
Group 2	50	**12.44**	0.000	12.18	0.21	0.267
	100	**16.14**	0.000	15.94	9.31	*0.209
	150	**18.17**	0.003	17.79	17.84	*0.435
	200	**19.02**	0.001	18.71	22.66	*0.329

TABLE 8. Instances and targets for convergence analysis.

Instance	$\|V\|$	Target
air05	426	66
air04	823	120
ns1688347	1866	370

We can conclude from Tables 5–7 that, up to 200 vertices, the GRASP and MH-MBSP heuristics are quite similar in terms of the results' quality. But for instances with more than 200 vertices, the heuristic MH-MBSP always achieved better results, both in terms of solution quality and CPU time.

For the last experiment, we evaluated and compared the run-time distributions (also know as time-to-target plots) of the GRASP heuristic and the MH-MBSP matheuristic for some instances. Time-to-target plots display on the ordinate axis the probability that an algorithm will find a solution at least as good as a given target value in a given running time, shown on the abscissa axis.

The heuristics were performed 200 times in each instance, with different seeds for the pseudo-random number generator. Then, the empirical probability distributions of each heuristic time spent to find a target solution value are plotted. The methodology described in [1] was adopted to trace each heuristic's empirical distribution. The run times (t_i) are sorted in ascending order, and a probability $p_i = (i - \frac{1}{2})/200$ is associated with each t_i. Then, the $z_i = (t_i, p_i)$ points for $i = 1, \ldots, 200$ are plotted. The best algorithm is the one that presents the corresponding leftmost plots.

We chose two instances from different sizes to analyze the performance of the methods (*air05* and *ns1688347*). Table 8 shows the instance's name, size, and target value. Both heuristics reached all chosen target values in the previous experiments since our interest was the convergence analysis.

Figure 1 shows the cumulative probability for the MH-MBSP for the instance *air05*, while Figure 2 shows the cumulative probability for both heuristics. In Figure 1, we can see how MH-MBSP reaches the target value very fast, with a high probability of finding the target in a time close to 0.16 s. Also, compared with the GRASP heuristic, the figure shows that GRASP needs much more time to reach the target value. In addition, for a probability of finding the target value greater than 50%, the GRASP heuristic needs at least 50 s.

Figures 3 and 4 show the cumulative probability for the *ns1688347* instance. Again, the MH-MBSP quickly reaches (less than one second) the target value for all iterations. While Figure 4 shows that GRASP presents a probability of 70% of reaching the target value. According to the obtained results, the minimum time to find the target solution for this instance by the GRASP heuristic was 3.3 s. Moreover, it was necessary to use a time close to 8.8 s to reach the target value with a probability greater than 50%.

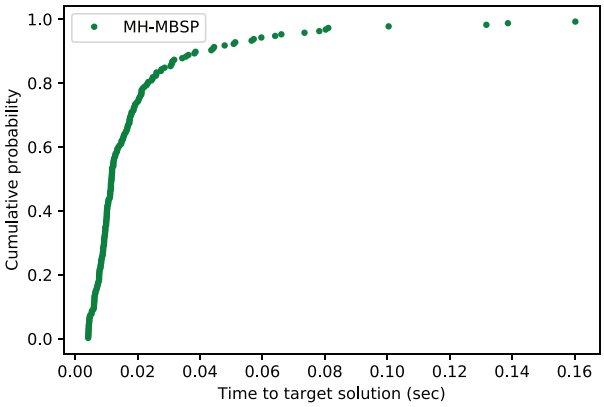

FIGURE 1. Cumulative probability of MH-MBSP for *air05* instance.

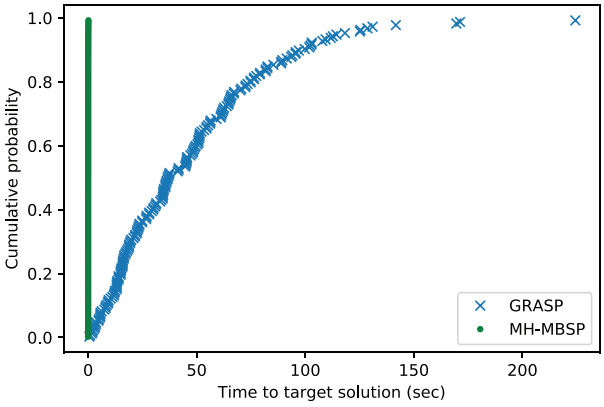

FIGURE 2. Cumulative probability of MH-MBSP and GRASP for *air05* instance.

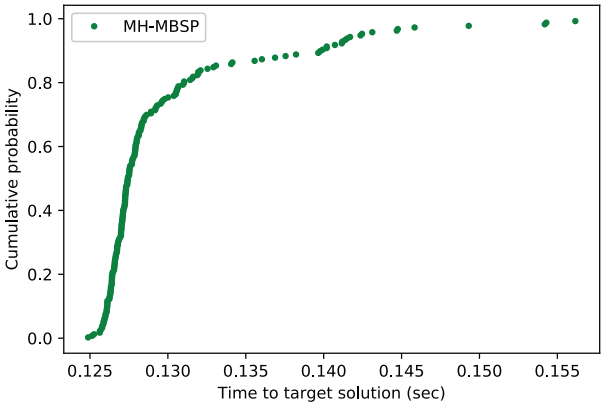

FIGURE 3. Cumulative probability of MH-MBSP for *ns1688347* instance.

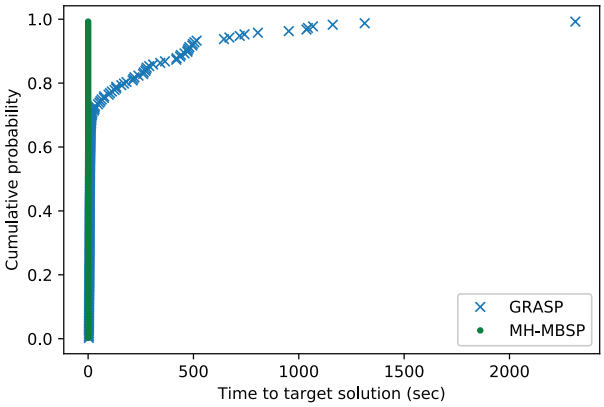

FIGURE 4. Cumulative probability of MH-MBSP and GRASP for *ns1688347* instance.

5. CONCLUSIONS

In this work, we developed methods to obtain heuristic solutions for the MBSP problem. We also provided an alternative formulation for the problem based on the basic definition of a balanced signed graph, whose size grows polynomially as a function of the input size.

Our main contribution in this work is the development of a matheuristic for the MBSP problem. Using different strategies as a multi-start scheme and a fast local search combined with an exact optimization phase allowed our method to obtain quickly good quality solutions for the tested sets of instances.

The results show that the proposed matheuristic is an appropriate choice to find good quality solutions for the MBSP problem. Furthermore, for large instances (with more than 200 vertices), the matheuristic MH_MBSP achieved in 30 s similar or better results than those obtained by solving exactly the problem using the proposed formulation in one hour of processing.

Our current research focus on the investigation of the structure of the polynomial size formulation in order to develop robust and efficient exact methods for the MBSP. Another direction for future research involves extending the solution method to similar balanced subgraph problems, such as KMBSP, where the goal is to find k-balanced components in the graph.

Acknowledgements. This study was financed in part by Fundação Carlos Chagas Filho de Amparo à Pesquisa do Estado de Rio de Janeiro and Conselho Nacional de Desenvolvimento Científico e Tecnológico. The authors would also like to thank the anonymous reviewer, who helped improve the manuscript with constructive comments.

REFERENCES

[1] R.M. Aiex, M.G.C. Resende and C.C. Ribeiro, Ttt plots: a perl program to create time-to-target plots. *Optim. Lett.* **1** (2007) 355–366.

[2] F. Barahona and A.R. Mahjoub, Facets of the balanced (acyclic) induced subgraph polytope. *Math. Program.* **45** (1989) 21–33.

[3] M. Birattari, Z. Yuan, P. Balaprakash and T. Stützle, F-race and iterated f-race: An overview. In: Experimental methods for the analysis of optimization algorithms. Springer (2010) 311–336.

[4] D. Cartwright and F. Harary, Structural balance: a generalization of heider's theory. *Psychol. Rev.* **63** (1956) 277.

[5] B. DasGupta, G. Enciso, E. Sontag and Y. Zhang, Algorithmic and complexity results for decompositions of biological networks into monotone subsystems. *Biosystems* **90** (2006) 161–178.

[6] M.R. de Holanda Maia, A. Plastino and P.H.V. Penna, Minereduce: an approach based on data mining for problem size reduction. *Comput. Oper. Res.* **122** (2020) 104995.

[7] N. Dupin and E.-G. Talbi, Parallel matheuristics for the discrete unit commitment problem with minstop ramping constraints. *Int. Trans. Oper. Res.* **27** (2020) 219–244.

[8] R. Figueiredo and Y. Frota, The maximum balanced subgraph of a signed graph: Applications and solution approaches. *Eur. J. Oper. Res.* **236** (2014) 473–487.
[9] R.M.V. Figueiredo, M. Labbé and C.C. De Souza, An exact approach to the problem of extracting an embedded network matrix. *Comput. Oper. Res.* **38** (2011) 1483–1492.
[10] M. Fischetti and M. Fischetti, Matheuristics. In: Handbook of Heuristics. Springer (2018) 121–153.
[11] N. Gülpinar, G. Gutin, G. Mitra and A. Zverovitch, Extracting pure network submatrices in linear programs using signed graphs. *Discrete Appl. Math.* **137** (2004) 359–372.
[12] F. Harary, M.-H. Lim and D.C. Wunsch, Signed graphs for portfolio analysis in risk management. *IMA J. Manage. Math.* **13** (2002) 201–210.
[13] F. Heider, Attitudes and cognitive organization. *J. Psychology* **21** (1946) 107–112.
[14] F. Hüffner, N. Betzler and R. Niedermeier, Separator-based data reduction for signed graph balancing. *J. Comb. Optim.* **20** (2010) 335–360.
[15] M. López-Ibáñez, J. Dubois-Lacoste, L.P. Cáceres, M. Birattari and T. Stützle, The irace package: Iterated racing for automatic algorithm configuration. *Oper. Res. Perspect.* **3** (2016) 43–58.
[16] F. Marinelli and A. Parente, A heuristic based on negative chordless cycles for the maximum balanced induced subgraph problem. *Comput. Oper. Res.* **69** (2016) 68–78.
[17] B. Martın, Á. Sánchez, C. Beltran-Royo and A. Duarte, A matheuristic approach for solving the edge-disjoint paths problem. *Matheuristics* **2016** (2016) 25.
[18] D. Martins, G.M. Vianna, I. Rosseti, S.L. Martins and A. Plastino, Making a state-of-the-art heuristic faster with data mining. *Ann. Oper. Res.* **263** (2018) 141–162.
[19] M.G.C. Resende and C.C. Ribeiro, Greedy randomized adaptive search procedures: advances and extensions. In: Handbook of metaheuristics. Springer (2019) 169–220.
[20] S. Wolfram, Wolfram Research. *Inc., Mathematica, Version* **8** (2013) 23.
[21] T. Zaslavsky, A mathematical bibliography of signed and gain graphs and allied areas. *Electron. J. Comb.* **1000** (2012) DS8.