An integer programming approach for solving a generalized version of the Grundy domination number

Manoel Campêlo¹, Daniel Severín¹

¹Dep. Estatística e Matemática Aplicada, Universidade Federal do Ceará, Brazil
²Dep. de Matemática (FCEIA), Universidad Nacional de Rosario, Argentina
³CONICET, Argentina

Abstract

A sequence of vertices in a graph is called a legal dominating sequence if every vertex in the sequence dominates at least one vertex not dominated by those that precede it, and at the end all vertices of the graph are dominated. The Grundy domination number of a graph is the size of a largest legal dominating sequence. In this work, we introduce a generalized version of the Grundy domination problem. We explicitly calculate the corresponding parameter for paths and web graphs. We propose integer programming formulations for the new problem, find families of valid inequalities and perform extensive computational experiments to compare the formulations as well as to test these inequalities as cuts in a branch-and-cut framework. We also design and evaluate the performance of a heuristic for finding good initial lower and upper bounds and a tabu search that improves the initial lower bound. The test instances include randomly generated graphs, structured graphs, classical benchmark instances and two instances from a real application. Our approach is exact for graphs with 20-50 vertices and provides good solutions for graphs up to 10000 vertices.

Key words: Grundy (total) domination number, Legal dominating sequence, Integer programming, Tabu search, Kneser graphs, Web graphs.

2010 MSC: 90C10, 90C57, 05C69

1. Introduction

Covering problems are some of the most studied problems in graph theory and combinatorial optimization due to the large number of applications. Consider a hypergraph \( \mathcal{H} = (X,E) \) without isolated vertices. An edge cover of \( \mathcal{H} \)
is a set of hyperedges \( \mathcal{C} \subset \mathcal{E} \) that cover all vertices of \( \mathcal{H} \), i.e. \( \bigcup_{C \in \mathcal{C}} C = X \). The general covering problem consists in finding the covering number of \( \mathcal{H} \) which is the minimum number of hyperedges in an edge cover of \( \mathcal{H} \) \[1\].

The most natural constructive heuristic for obtaining an edge cover of \( \mathcal{H} \) is as follows. Start from empty sets \( \mathcal{C} \) and \( \mathcal{W} \) (the latter one keeps the already covered vertices). At each step \( i \), pick a hyperedge \( C_i \in \mathcal{E} \), add \( C_i \) to \( \mathcal{C} \) and add all the elements of \( C_i \) to \( \mathcal{W} \). The process is repeated until \( \mathcal{W} = X \). In addition, \( C_i \) can only be chosen if at least one of its elements has not been previously included in \( \mathcal{W} \), i.e. \( C_i \setminus \left( \bigcup_{j=1}^{i-1} C_j \right) \neq \emptyset \).

How bad can a solution given by this heuristic be (compared to the value of an optimal solution)? The answer leads to the concept of the Grundy covering number of \( \mathcal{H} \) which computes the largest number of steps performed by such a constructive heuristic, or equivalently, the largest number of hyperedges used in the resulting covering \[3\].

Let \( G = (V, E) \) be a simple graph. For any \( v \in V \), let \( N(v) \) be the open neighborhood of \( v \), i.e. \( N(v) = \{ u \in V : (u, v) \in E \} \) (the symbol “\( = \)” will be used recurrently to make definitions) and \( N[v] \) be the closed neighborhood of \( v \), i.e. \( N[v] = N(v) \cup \{ v \} \).

A particular case of the Grundy covering problem occurs when \( \mathcal{H} \) is the hypergraph of the closed neighborhoods of vertices in a graph \( G \): \( \mathcal{H} = (V(G), \mathcal{E}) \) where \( \mathcal{E} = \{ N[v] : v \in V(G) \} \). Here, the Grundy covering number of \( \mathcal{H} \) is called Grundy domination number of \( G \) \[3\]. Analogously, the Grundy total domination number of \( G \) is the Grundy covering number of the hypergraph of the open neighborhoods of vertices in \( G \) \[4\].

In order to illustrate these concepts, consider the graph \( G \) of Figure 1 a) and the hypergraph of the closed neighborhoods of \( V(G) \). A possible execution of the heuristic would be to pick \( N[3] \), then \( N[4] \) and finally \( N[5] \). In Figure 1 b), it is displayed from left to right what happens at each step. A vertex \( v \) inside a box means that \( N[v] \) is the chosen set at that step. Filled circles denote those vertices that are already covered (set \( \mathcal{W} \) of the heuristic). Note that, at any step, at least one new vertex is covered: in step 2 vertex 5 is covered while in step 3 vertex 1 is covered. The Grundy domination number of this graph is 3 since it is not possible for the heuristic to perform 4 steps.

The study of these domination parameters was originally motivated by a domination game \[2 \] \[10 \] \[11\], and their associated problems can model some applications where there are two players with opposing interests. For instance, consider a city divided into districts where the municipal government intends to offer a concession per year of a given service (e.g. Internet connection) to companies. Each year, a company is located in a district, which is committed to providing its service to both the district and its neighbors. In return, it requires having at least, within its domain (the district where it is installed and its neighbors), a captive district: only that company offers the service to it, for (at least) a year. The goal of the government is to plan which district should be chosen (for a company to be settled) each year so that, after a while, the city is entirely covered and to maximize the number of companies providing the service (and, thus, to foster long-term competitiveness). The given problem can
be modeled as a Grundy domination problem where each vertex represents a district and two vertices are adjacent if and only if they represent neighboring districts. Indeed, the Grundy domination number gives the maximum number of companies. For instance, if the city is modeled as the graph of Figure 1a), an optimal schedule is to settle a company on district 3 in year 1, another on district 4 in year 2 and the last one on district 5 in year 3. During the 2nd. year, district 5 only receives the service from the company on district 4. The same happens for district 1 and the company on district 5 during the last year. In Section 6, the city of Buenos Aires is considered.

The problems associated to these parameters are hard for general graphs. In [3], it is proven that the Grundy domination problem is \( \mathcal{NP} \)-hard for chordal graphs (it is also proven that this problem is polynomial for trees, cographs or split graphs). Regarding the total version of this problem, it is \( \mathcal{NP} \)-hard when \( G \) is bipartite [4] (but it is polynomial on trees, \( P_4 \)-tidy and distance-hereditary bipartites [5]).

It is known that one of the most powerful tools to solve \( \mathcal{NP} \)-hard problems are branch-and-cut algorithms, which are based on Integer Programming. In this work, we introduce a problem that generalizes the Grundy domination and the Grundy total domination, and we propose integer programming formulations for this new problem. In order to obtain good upper and lower bounds, we also design a heuristic algorithm that combines a greedy strategy with a tabu search. Our approach is exact for instances ranging from 20 to 50 vertices (depending

Figure 1: Example: a) graph \( G \), b) a maximum legal dominating sequence
on the edge density of the graph) and, in particular, by taking only the heuristic
algorithm, one can achieve good solutions on large instances.

In Section 2, we introduce a general version of the problem and show some
useful properties. In particular, we calculate the exact value of the correspond-
ing parameters for two families of graphs (paths and web graphs). In Sections 3
and 4, we introduce an integer programming model together with several fami-
lies of valid inequalities that strengthen its linear relaxation. Besides, we present
constraints that remove unnecessary integer points from the solution space and
whose addition results in several formulations. In Section 5, we propose a heuris-
tic for obtaining an initial upper bound and an initial feasible solution of our
problem, and a tabu search for improving that initial solution. In Section 6, we
perform extensive computational experiments to compare the formulations as
well as to test two families of valid inequalities as cuts in a branch-and-cut frame-
work. We also evaluate the performance of the tabu search. The test instances
include randomly generated graphs, structured graphs, classical benchmark in-
ces and two real instances from the aforementioned application in the city
of Buenos Aires. Another experiment allows us to formulate a conjecture about
the Grundy domination number on Kneser graphs. Finally, in Section 7 some
conclusions are drawn.

Some results contained in this work appeared without proof in the extended
abstract [7].

1.1. Definitions and notation

Let $G = (V, E)$ be a simple graph and $C$ be a subset of vertices of $V$. Define
the function $N(\cdot) : V \rightarrow P(V)$, called neighborhood of $v$, as follows:

$$N(v) = \begin{cases} N[v], & \text{if } v \in C \\ N(v) & \text{if } v \notin C \end{cases}$$

Here, $P(V)$ denotes the powerset of $V$. Assume that no vertex from $V \setminus C$ is
isolated in $G$ so that, for all $v \in V$, $N(v) \neq \emptyset$ and $v \in N(w)$ for some $w \in V$.

Some definitions given in [3, 4] (which only depends on $G$) are rewritten
below in terms of the pair “$G; C$”. These definitions abstract the behavior of
the heuristic mentioned at the beginning of this work.

Let $S = (v_1, \ldots, v_k)$ be a sequence of distinct vertices of $G$. The sequence $S$
is called a legal sequence of $G; C$ if

$$W_i \doteq N(v_i) \setminus \bigcup_{j=1}^{i-1} N(v_j) \neq \emptyset$$

holds for every $i = 2, \ldots, k$. By convention, $W_1 \doteq N(v_1)$ which is trivially non
empty.

If $S$ is a legal sequence of $G; C$, then we say that $v_i$ footprints the vertices
from $W_i$. That is, $v_i$ footprints a vertex $u \in N(v_i)$ if $u$ does not belong to the
neighborhood of $v_j$, for $j = 1, 2, \ldots, i - 1$. 

4
If $\bigcup_{j=1}^k W_j = V$ then $S$ is called a dominating sequence.

For a given sequence $S = (v_1, \ldots, v_k)$, let $\hat{S} = \{v_1, \ldots, v_k\}$. We say that a legal sequence $S$ of $G; C$ is maximal if there is no $v \in V \setminus \hat{S}$ such that $(S, v)$, i.e. the sequence $S$ with $v$ appended at the end, is legal. We say that a legal sequence $S$ of $G; C$ is maximum if there is no legal sequence $S'$ of $G; C$ such that $S'$ is longer than $S$. A maximum legal sequence is also maximal.

Consider again the graph of Figure 1 and let $C = V$. Part b) actually shows that the sequence $(3, 4, 5)$ is legal and dominating: $W_1 = \{2, 3, 4\}$ (2, 3 and 4 are footprinted by 3), $W_2 = \{5\}$ (5 is footprinted by 4), $W_3 = \{1\}$ (1 is footprinted by 5) and $W_1 \cup W_2 \cup W_3 = V$. The sequence is also maximum. On the other hand, the sequence $(5, 3)$ is a maximal legal sequence that is not maximum.

2. The General Grundy Domination Problem

Let $G = (V, E)$ be a simple graph on $n$ vertices and $C \subset V$ such that no isolated vertex is in $V \setminus C$. Let $\mathcal{H}(G; C)$ denote the hypergraph $(V, \mathcal{E})$ with $\mathcal{E} = \{N(v) : v \in V\}$.

We define the general Grundy domination number of $G; C$, denoted by $\gamma_{gr}(G; C)$, as the Grundy covering number of $\mathcal{H}(G; C)$. It gives rise to the following problem:

**General Grundy Domination Problem (GGDP)**

**INSTANCE:** a graph $G = (V, E)$ and a set $C \subset V$ such that no isolated vertex is in $V \setminus C$.

**OBJECTIVE:** obtain $\gamma_{gr}(G; C)$.

Since the Grundy domination problems mentioned in the introduction are particular cases of our problem ($\gamma_{gr}(G; V)$ is indeed the Grundy domination number of $G$ while $\gamma_{gr}(G; \emptyset)$ is the Grundy total domination number of $G$), they can be addressed by a tool that just solves the GGDP.

The following result will be useful to get the general Grundy domination number. It shows that every optimal solution is a maximum legal sequence and the “dominating” property comes for free.

**Proposition 2.1.** Let $G; C$ be an instance of the GGDP and $S$ a maximal legal sequence of $G; C$. Then, $S$ is dominating. Moreover, if $S$ is maximum, then $\gamma_{gr}(G; C)$ is the length of $S$.

**Proof.** If $S$ were not dominating, there would exists a vertex $v$ not footprinted by any element of $S$. As $G; C$ is an instance of the GGDP, there exists a vertex $w$ such that $v \in N(w)$, and so $w \not\in S$. Thus, the sequence $(S, w)$ is legal since $w$ footprints $v$, which contradicts the maximality of $S$. Therefore, $S$ is dominating.

The second part of the statement is derived by the fact that legal dominating sequences are (by definition) in one-to-one correspondence with the solutions provided by the constructive heuristic for $\mathcal{H}(G; C)$. \qed
2.1. Properties on the GGDP

As we will see later, knowing a good upper bound will be fundamental for the performance of the exact algorithm that will be proposed (the worse the bound, the larger the integer linear program that must be solved). The following result gives a bound on the length of a sequence where the first elements are known.

**Proposition 2.2.** Let \( t, k \) be positive integers such that \( t < k \), \( S = (v_1, \ldots, v_t) \) and \( S' = (S, v_{t+1}, \ldots, v_k) \) be legal sequences. Then,

\[
k \leq n - \left| \bigcup_{i=1}^{t} N(v_i) \right| + t.
\]

**Proof.** Let \( W = \bigcup_{i=1}^{t} N(v_i) \). Since \( S' \) is legal, vertices \( v_{t+1}, \ldots, v_k \) must footprint at least one different vertex from \( V \setminus W \), implying that \( k - t \leq n - |W| \). Hence, \( k \leq n - |W| + t \).

**Corollary 2.3.** Let \( t \) be a positive integer such that \( t \leq \gamma_{gr}(G;C) \) and \( \delta_t \) be the minimum cardinality of \( \bigcup_{i=1}^{t} N(v_i) \) for all legal sequences \( (v_1, \ldots, v_t) \). Then, \( \gamma_{gr}(G;C) \leq n - \delta_t + t \).

Let \( m_t = n - \delta_t + t \). For the case \( t = 1 \), finding \( m_1 \) is linear and provides an upper bound which can be tight sometimes. For instance, let \( G \) be the graph of Figure 1 and \( C = V \). Here, \( \delta_1 = 3 \) and, by the corollary, \( \gamma_{gr}(G;V) \leq 3 = m_1 \).

As the sequence \((3,4,5)\) is legal, \( \gamma_{gr}(G;V) = 3 \). The bound \( m_1 \) was previously derived in [3] for the Grundy domination number (i.e. \( C = V \)) and in [4] for the total case (i.e. \( C = \emptyset \)), although their proofs are different from ours, which is valid for any \( C \).

Better bounds can be computed by increasing \( t \). For instance, let \( G \) be the graph of Figure 2 (a) and \( C = V \). Here, \( \delta_1 = 2, \delta_2 = 4, \delta_3 = 6 \) and, by the corollary, we get the upper bounds 7, 6 and 5 respectively. None of them is tight since \( \gamma_{gr}(G;V) = 4 \) (the sequence \((1,2,4,6)\) is optimal). A similar example is given in Figure 2 (b) for the case \( C = \emptyset \). Here, \( \delta_1 = 1, \delta_2 = 3 \) and \( \delta_3 = 5 \), thus obtaining the upper bounds 7, 6 and 5 respectively, but \( \gamma_{gr}(G;\emptyset) = 4 \) (the sequence \((5,2,6,1)\) is optimal).

Another way to shorten the time of optimization is by reducing the size of the input graph.

We say that distinct vertices \( u, v \) are *twins* if \( N(u) = N(v) \). If \( G;C \) does not have twin vertices, then \( G;C \) is called *twin free*. If there exist twin vertices \( u, v \), then

\[
\gamma_{gr}(G;C) = \gamma_{gr}(G - v; C \setminus \{v\}),
\]

where \( G - v \) is the graph obtained from \( G \) by deleting \( v \). This rule can be applied recursively until the instance is twin free.

Now, if \( G \) is the disjoint union of graphs \( G_1 \) and \( G_2 \), then

\[
\gamma_{gr}(G;C) = \gamma_{gr}(G_1; C \cap V(G_1)) + \gamma_{gr}(G_2; C \cap V(G_2)).
\]

This rule allows us to solve each component separately as the overall time is lower than solving \( G;C \) directly.
2.2. The GGDP on paths and web graphs

When a new $\mathcal{NP}$-hard optimization problem on graphs is introduced, a natural question is what happens on simple structures such as paths and circuits. Below, we give $\gamma_{gr}(G; C)$ for two families of graphs (paths and web graphs) with any $C$. Recall that GGDP is already $\mathcal{NP}$-hard on bipartite or chordal graphs.

Let $P_n$ denote an induced path on $n \geq 1$ vertices where $V(P_n) = \{1, \ldots, n\}$. Note that, for any integer $n \geq 2$ and any $C \subset V(P_n)$, $\delta_1 = 2$ when $\{1, n\} \subset C$ and $\delta_1 = 1$ otherwise. Therefore,

**Observation 2.4.** Let $n \geq 2$ and $C \subset V(P_n)$. If $\{1, n\} \subset C$, then $m_1 = n - 1$; otherwise, $m_1 = n$.

We say that $C$ is a good configuration for $P_n$ if

(i) $n = 1$ and $C = \{1\}$,
(ii) $n = 2$ and $C \neq \{1, 2\}$,
(iii) $n \geq 3$ and either
   (iii.1) $1 \notin C$ and $C$ is a good configuration for the subpath induced by $\{3, \ldots, n\}$ or
   (iii.2) $n \notin C$ and $C$ is a good configuration for the subpath induced by $\{1, \ldots, n-2\}$.

For the sake of simplicity, when we say that $C$ is a good configuration for a subpath $P'$ of $P$, we are actually referring to the set $C \cap V(P')$. Moreover, in contrast with the definition of GGDP, we allow the subgraph induced by $V(P') \setminus C$ to have isolated vertices in the good configuration definition.

From the previous observation, we trivially have

**Observation 2.5.** Let $n \geq 1$ and $C \subset V(P_n)$. If $C$ is a good configuration for $P_n$, then $m_1 = n$.

The following results give the general Grundy domination number of a path and, at the same time, show where the upper bound $m_1$ is tight.
Proposition 2.6. Let \( n \geq 1 \), \( G = P_n \) and \( C \subseteq V(G) \). If \( \{1,n\} \subseteq C \) or \( C \) is a good configuration for \( G \), then \( \gamma_{gr}(G;C) = m_1 \); otherwise, \( \gamma_{gr}(G;C) = m_1 - 1 \).

Proof. If \( n = 1 \), we must have \( C = \{1\} \), because \( N(1) \neq \emptyset \). So \( C \) is a good configuration and \( \gamma_{gr}(G;C) = m_1 = 1 \). Now, consider the case \( n = 2 \). If \( C = \{1,2\} \) then \( \gamma_{gr}(G;C) = m_1 = 1 \). And, if \( C \neq \{1,2\} \) (\( C \) is a good configuration for \( G \)) then \( \gamma_{gr}(G;C) = m_1 = 2 \).

For \( n \geq 3 \), note that \((1,2,\ldots,n-1)\) is a legal dominating sequence. Indeed, 1 footprints 2 (and itself, if \( 1 \in C \)), 2 footprints 3 (and 1, if \( 1 \notin C \)), and \( i = 3, \ldots, n-1 \) footprints \( i+1 \). Thus, \( n-1 \leq \gamma_{gr}(G;C) \leq m_1 \). If \( \{1,n\} \subseteq C \), by Observation 2.4 we have \( \gamma_{gr}(G;C) = m_1 = n-1 \). It remains to consider the case \( \{1,n\} \notin C \). By Observation 2.4, \( m_1 = n \). Then, \( \gamma_{gr}(G;C) \in \{m_1-1,m_1\} \), and it is enough to prove that \( C \) is a good configuration for \( P_n \) if and only if there exists a legal dominating sequence of size \( n \).

First, assume that \( C \) is a good configuration for \( P_n \). We use induction on \( n \). Recall that we have already obtained \( \gamma_{gr}(G;C) = 2 \) for \( n = 2 \), and \( \gamma_{gr}(G;C) = 1 \) is trivial for \( n = 1 \). For \( n \geq 3 \), (iii.1) or (iii.2) holds. Assume without loss of generality that (iii.1) holds (the other case is symmetric). The induction hypothesis ensures the existence of a legal dominating sequence \( S \) for \((3,\ldots,n)\), with \(|S| = n - 2 \). Consider the extended sequence \( S' = (1,S,2) \). \( S' \) is legal and dominating for \( P_n \) with \(|S| = n \).

Now, assume that there exists a legal dominating sequence \( S \) such that \(|S| = n \). We use again induction on \( n \). \( C \) is trivially a good configuration for \( P_n \) when \( n \in \{1,2\} \). For \( n \geq 3 \), any dominating legal sequence of length \( n \) must start with an endpoint of \( P_n \), say \( 1 \), and such a vertex must belong to \( V \setminus C \). Moreover, 2 must be the last vertex in the sequence, and the \( n-2 \) vertices between 1 and 2 define a legal dominating sequence for \((3,\ldots,n)\). By the induction hypothesis, \( C \) is a good configuration for \((3,\ldots,n)\). Since \( 1 \notin C \), \( C \) is also a good configuration for \((1,\ldots,n)\).

Corollary 2.7. Let \( n \geq 1 \), \( G = P_n \) and \( C \subseteq V(G) \). If \( C \) is a good configuration for \( G \), then \( \gamma_{gr}(G;C) = n \). Otherwise, \( \gamma_{gr}(G;C) = n - 1 \).

The second (less trivial) family are web graphs [14]. Let \( n,k \) be positive integers such that \( n \geq 2(k+1) \). A web \( W^k_n = (V,E) \) is a graph with \( V = \{0,\ldots,n-1\} \) and \( E = \{(i,j) : 0 < |i-j| \leq k \text{ or } |i-j| \geq n-k \} \). See examples in Figure 3. Note that \( N(i) = \{i \oplus k, i \ominus (k-1), \ldots, i \ominus 1, i \ominus 1, \ldots, i \ominus (k-1), i \ominus k\} \), where \( \oplus \) and \( \ominus \) stand for the addition and subtraction modulo \( n \). Therefore, \( m_1 = n - 2k \) if \( C = V \), and \( m_1 = n - 2k + 1 \) if \( C \neq V \).

Proposition 2.8. Let \( G \) be the web graph \( W^k_n \) and \( C \subseteq V(G) \) (\( C \) possibly empty). Then, \( \gamma_{gr}(G;C) = m_1 \) in the following cases: (i) \( C = V \), or (ii) there is \( i \in V \setminus C \) such that \( V \setminus N[i] \) induces a path \( P_t \), \( t = n - 2k - 1 \), and \( C \) is a good configuration for \( P_t \). Otherwise, \( \gamma_{gr}(G;C) = m_1 - 1 \).

Proof. Consider the sequence \((0,\ldots,n - 2k - 1)\) of length \( n - 2k \). It is a legal dominating sequence since vertex 0 footprints 1, \ldots, \( k \) and \( n - k, \ldots, n - 1 \) (and itself if \( 0 \in C \)), vertex 1 footprints \( k + 1 \) (and vertex 0 if \( 0 \notin C \)) and, if \( n >
2(k + 1), vertex i footprints k + i for all i = 2, . . . , n − 2k − 1. The last footprinted vertex is n − k − 1. Therefore, we obtain m1 − 1 ≤ n − 2k ≤ γgr(G; C) ≤ m1. If C = V, we are done since m1 = n − 2k.

Assume now that C ̸= V. Let i ∈ V \ C, and

\[ V_i \doteq V \setminus N[i] = \{i \oplus (k + 1), i \oplus (k + 2), \ldots, i \oplus (n − k − 1)\}. \]

Note that |Vi| = n − 2k − 1. It suffices to show that γgr(G; C) = m1 if and only if Vi induces a path and C is a good configuration for it. Recall that m1 = n − 2k + 1. The unique way of getting a dominating legal sequence of length m1 is by starting with i and then choosing a vertex that will footprint only one more vertex at each step. This means that, at any step but the last one, we cannot choose a vertex from N(i) because it would footprint i and at least one vertex from Vi, and finally a vertex from N(i). In addition, Vi must induce a path, otherwise some of its vertices would footprint at least 2 vertices. Consider the sequence S′ = (i, S, j) where j ∈ N(i) and S is a maximum legal sequence of G[Vi]. In virtue of Corollary 2.7, |S| = n − 2k − 1 if C is a good configuration for G[Vi] and |S| = n − 2k − 2 otherwise. Therefore, γgr(G; C) = m1 if and only if Vi induces a path and C is a good configuration for it.

For example, consider the web graphs depicted in Figure 3. Filled circles denote the vertices of C, i.e. C = \{1, 2, 3, 4, 5, 7\}. Upper bounds for γgr(G; C) are m1 = 7 (if G = W3 8) and m1 = 3 (if G = W4 8). In a), C is neither a good configuration for V \ N[0] = \{2, 3, 4, 5, 6\} nor for V \ N[6] = \{0, 1, 2, 3, 4\}, so a legal sequence of maximum length is (0, 1, 2, 3, 4, 5). In b), since C is a good configuration for V \ N[0] = \{4\}, a legal sequence of maximum length is (0, 4, 1).

### 3. Integer programming formulation

Let G; C be an instance of the GGDP, and LB and m be a lower bound and upper bound on γgr(G). We present an ILP formulation for the problem by modeling the iterative process performed by the constructive heuristic described...
in the introduction. For the sake of readability, through this section, we also present examples based on the graph of Figure 2 (a) with \( C = V \) and \( m = 7 \).

Legal sequences will be modeled as binary vectors: a given sequence is represented by an array of \( m \) consecutive slots \((s_1, \ldots, s_m)\) where each slot can be empty or store a vertex. An empty slot is skipped when forming the sequence from its corresponding array. For instance, the array \((0, 1, 0, 2, 4, 0, 6)\), where “0” denotes the empty slot, represents the sequence \((1, 2, 4, 6)\). Observe that a sequence \( S \) can be represented by \( \binom{m}{|S|} \) different arrays. An empty slot can be interpreted as an innocuous step (without performing any action) in the process, just to attain \( m \) steps in total.

Now, for a given array \((s_1, \ldots, s_m)\), let \( y_{vi} \) be a binary value such that \( y_{vi} = 1 \) if and only if \( v \) is in the \( i \)-th. slot, i.e. \( s_i = v \). In order these variables to reflect an array composed of distinct vertices, the following constraints must hold:

\[
\sum_{v \in V} y_{vi} \leq 1, \quad \forall \ i = 1, \ldots, m
\]

\[
\sum_{i=1}^{m} y_{vi} \leq 1, \quad \forall \ v \in V
\]

Constraints (1) ensure that at most one vertex is chosen in each slot. Constraints (2) guarantee that each vertex can be chosen no more than once.

In order to model the legality of sequences, we consider another set of binary variables that keeps a record of footprinted vertices. First, let \( A_0 = V \), \( A_i = A_{i-1} \setminus N(s_i) \) otherwise, for \( i = 1, \ldots, m \). In other words, set \( A_i \) contains those vertices available to be footprinted in step \( i + 1 \) and subsequent ones. In case the slot \( s_i \) is empty, no vertex is footprinted at step \( i \). Now, observe that the sequence represented by an array is legal if, for every non-empty \( s_i \), \( A_{i-1} \cap N(s_i) \neq \emptyset \).

Let \( x_{ui} \) be a binary value such that \( x_{ui} = 1 \) if and only if \( u \in A_i \). In particular, \( x_{u0} = 1 \) for all \( u \in V \) meaning that \( x_{u0} \) can be treated as a constant.

The next constraints model the chain of inclusions \( A_i \subset A_{i-1} \):

\[
x_{ui} \leq x_{ui-1}, \quad \forall \ u \in V, \ i = 2, \ldots, m
\]

(3)

Indeed, suppose that \( s_i \) is empty or \( s_i \in V \setminus N(u) \). Then, the l.h.s. of (4) is zero, implying \( x_{ui} \geq x_{ui-1} \), as desired. On the other hand, if \( s_i \in N(u) \), then (4) becomes redundant.
In order to model $A_i \subset A_{i-1} \setminus N(s_i)$ when $s_i$ is not empty, every $u \in A_{i-1}$ such that $s_i \in N(u)$ must not belong to $A_i$. Constraints (5), presented below, remove those elements from $A_i$:

$$x_{ui} + \sum_{v \in N(u)} y_{vi} \leq 1, \quad \forall \ u \in V, \ i = 1, \ldots, m$$

Therefore, constraints (3)-(5) guarantee the correct valuation of the $x$ variables.

Finally, in order the sequence (represented by the array) to be legal, for any $i \geq 2$ such that $s_i$ is not empty, there must be at least one available vertex of $N(s_i)$ at step $i - 1$, which becomes footprinted at $i$:

$$y_{vi} \leq \sum_{u \in N(v)} (x_{ui-1} - x_{ui}), \quad \forall \ v \in V, \ i = 2, \ldots, m$$

Indeed, if $y_{vi} = 1$ (i.e. $s_i = v$), then the r.h.s. of (6) is positive implying that some $u \in N(v)$ must satisfy $x_{ui-1} > x_{ui}$, that is $x_{ui-1} = 1$ and $x_{ui} = 0$.

The previous families of constraints are enough to model legal sequences but not all of them are necessary. In fact, we can dispense with constraints (4) (as proved below). Omitting them allows marking a vertex $u \in A_{i-1}$ as non-available to be footprinted at step $i$ even if it was not indeed footprinted, that is $x_{ui-1} = 1$ and $x_{ui} = 0$ but $y_{vi} = 0$ for all $v \in N(u)$. This does not affect the sequence legality. It leads to the following formulation, which we call $F_1$, that obtains the array that maximizes the amount of non-empty slots:

$$\max \sum_{i=1}^{m} \sum_{v \in V} y_{vi}$$

subject to

$$\sum_{v \in V} y_{vi} \leq 1, \quad \forall \ i = 1, \ldots, m$$

$$\sum_{i=1}^{m} y_{vi} \leq 1, \quad \forall \ v \in V$$

$$x_{ui} \leq x_{ui-1}, \quad \forall \ u \in V, \ i = 2, \ldots, m$$

$$x_{ui} + \sum_{v \in N(u)} y_{vi} \leq 1, \quad \forall \ u \in V, \ i = 1, \ldots, m$$

$$y_{vi} \leq \sum_{u \in N(v)} (x_{ui-1} - x_{ui}), \quad \forall \ v \in V, \ i = 2, \ldots, m$$

$$x_{vi}, y_{vi} \in \{0, 1\} \quad \forall \ v \in V, \ i = 1, \ldots, m$$

**Theorem 3.1.** Formulation $F_1$ gives the value of $\gamma_{gr}(G; C)$. 

11
Proof. Let \((s_1, \ldots, s_m)\) be the array defined by an optimal solution \((x^*, y^*)\) of \(F_1\). Precisely, for each \(i = 1, \ldots, m\),

\[
s_i = \begin{cases} 
\ast & \text{if } \sum_{v \in V} y_{vi} = 0, \\
v & \text{if } y_{vi} = 1.
\end{cases}
\]

Note that \(s_i\) is uniquely defined by (1). Consider the sequence \(S = (v_1, \ldots, v_k)\) represented by that array. We have to show that \(S\) is legal and has maximum length. Since the non-empty slots have distinct vertices by (2), so do \(S\). Now, consider \(v_i \in S\) for some \(i \geq 2\). There is some \(j \geq i\) such that \(s_j = v_i\) (i.e. \(y_{v_i,j} = 1\)) in the array. By constraints (3) and (6), there exists \(u \in N(s_j)\) such that \(x^*_{uj} = 1\). Besides, by constraints (3) and (5), we have \(x^*_{ul} = 1\) and \(\sum_{v \in N(u)} y_{vl} = 0\), for all \(l = 1, \ldots, j - 1\). It follows that \(s_l \notin N(u)\) for all \(l = 1, \ldots, j - 1\). Therefore,

\[
u \in N(s_j) \setminus \bigcup_{n=1}^{j-1} N(s_l) = N(v_i) \setminus \bigcup_{l=1}^{i-1} N(v_l),
\]

which implies that \(S\) is legal.

Now, suppose that \(S'\) is a legal sequence longer than \(S\), e.g. \(S' = (v'_1, \ldots, v'_{k+1})\). As \(m \geq \gamma_{gr}(G; C) \geq k + 1\), \(S'\) fits in an array. Consider \((v'_1, \ldots, v'_{k+1}, \ldots)\) and the sets \(A'_0 = V, A'_i = A'_{i-1} \setminus N(v'_i)\) for all \(i = 1, \ldots, k + 1\) and \(A'_i = A'_{k+1}\) for all \(i = k + 2, \ldots, m\) (if \(m > k + 1\)). Clearly, the array and the sets correspond to a feasible integer point \((x, y)\) with objective value of \(k + 1\), which leads to a contradiction. \(\square\)

4. Strengthening the formulation

In this section, we explore several ways of strengthening the linear relaxation of \(F_1\) by the addition of inequalities.

4.1. Symmetry-breaking inequalities

In an ILP model, two integer solutions are called symmetric if they share the same objective value and one can be obtained from the other by swapping its components [12]. In a broader sense, symmetric solutions have a common characteristic (for instance, both represent the same solution of the problem) so that one can be eliminated without losing the correctness of the model. Sometimes, ILP models with several symmetric solutions may have poor performance and one way to reduce the number of symmetric solutions is to incorporate symmetric-breaking constraints to the formulation. Certainly, constraints (4) defined above remove some symmetric solutions. Below, we present two families of constraints that also do this job.

- For a given legal sequence \((v_1, \ldots, v_k)\), consider an array such that the first elements are those from the sequence and the remaining slots are
empty, i.e. \((v_1, \ldots, v_k, \ldots, \ldots)\). We call it canonical. For instance, the sequence \((1,2,4,6)\) is represented by the canonical array \((1, 2, 4, 6, \ldots, \ldots)\).

Those solutions associated to non-canonical arrays can be removed by

\[
\sum_{v \in V} y_{vi} \leq \sum_{v \in V} y_{vi-1}, \quad \forall \ i = 2, \ldots, m
\]

which means that, if for some \(i \geq 2\), \(s_i\) is not empty, then \(s_{i-1}\) is not empty too.

- As optimal sequences are dominating, we can impose that every integer solution represents a dominating sequence:

\[
\sum_{i=1}^{m} \sum_{v \in N(u)} y_{vi} \geq 1, \quad \forall \ u \in V
\]

Different combinations of the symmetry-breaking constraints presented above give rise to 8 formulations listed in the following table. For instance, \(F_2\) is the same as \(F_1\) plus constraints \([4]\):

| Form. | \([1]\) | \([2]\) | \([3]\) | \([4]\) | \([5]\) | \([6]\) | \([7]\) | \([8]\) |
|-------|--------|--------|--------|--------|--------|--------|--------|--------|
| \(F_1\) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_2\) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_3\) | ✔️ (only \(i = 1\)) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_4\) | ✔️ (only \(i = 1\)) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_5\) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_6\) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_7\) | ✔️ (only \(i = 1\)) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |
| \(F_8\) | ✔️ (only \(i = 1\)) | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ | ✔️ |

In some rows, constraints \([1]\) are omitted (except for the case \(i = 1\)) since they are dominated by \([7]\).

All of the enumerated formulations are correct (in the sense that they deliver \(\gamma_{gr}(G; C)\)) since \(F_1\) is correct and the other ones just delete unnecessary solutions. In particular, two of them deserve a special attention:

**Observation 4.1.** There is a one-to-one correspondence between legal sequences of \(G; C\) and integer solutions of \(F_4\).

**Observation 4.2.** There is a one-to-one correspondence between dominating legal sequences of \(G; C\) and integer solutions of \(F_8\).

Although one would expect that the fewer the number of solutions is, the smaller the size of the branch-and-bound tree will be, the addition of symmetry-breaking inequalities not always help to improve the optimization. Therefore, we carried out computational experiments in order to determine which formulation performs better. Results are reported in Section \([6]\).
4.2. Removing non-optimal solutions

If a lower bound $LB$ is known, solutions corresponding to sequences of size less than $LB$ are not optimal and, thus, are not needed. They can be removed by forcing $\sum_{i=1}^{m} \sum_{v \in V} y_{vi}$ to be at least $LB$. Even better, we can impose that the first $LB$ slots must be used:

$$\sum_{v \in V} y_{vi} = 1, \quad \forall \ i = 1, \ldots, LB$$

These constraints can be incorporated to any of the 8 formulations. In that case, some constraints may become dominated and can be omitted: $\square$ for $i = 1, \ldots, LB$ and $\blacklozenge$ for $i = 2, \ldots, LB$.

4.3. Valid inequalities

It is known that a cutting-plane algorithm is one of the most efficient tools to deal with an integer linear programming problem [15]. The main idea is to consider the linear relaxation and try to strengthen it by adding violated valid inequalities. The algorithm can use general cuts that do not take advantage of the structure (such as Gomory cuts), or specially developed ones that exploit the properties of the problem. In this section, we present some inequalities that are valid for the formulations given above, which will be the basis for a cutting-plane algorithm.

Let $P_i$ be the convex hull of the set of integer feasible solutions in $F_i$.

Given a non-empty set of vertices $U$ and a positive integer $r \leq |U|$, let $N^r \langle U \rangle$ denote the subset of vertices with exactly $r$ neighbors in $U$, i.e. $N^r \langle U \rangle = \{ v \in V : |N(v) \cap U| = r \}$. Observe that $V$ can be partitioned in sets $N^r \langle U \rangle$ for $r = 0, \ldots, |U|$. For example, consider the graph of Figure 4 where $C = V$. Filled circles represent vertices of $U$, i.e. $U = \{2, 8\}$. Then, $V = N^0 \langle U \rangle \cup N^1 \langle U \rangle \cup N^2 \langle U \rangle$, where $N^0 \langle U \rangle = \{7\}$, $N^1 \langle U \rangle = \{1, 4, 5, 6\}$ (vertices inside a rhombus) and $N^2 \langle U \rangle = \{2, 3, 8\}$ (vertices inside a square).

Now, we present a general family of valid inequalities. Consider:
• an integer \( i \in \{2, \ldots, m\} \),
• an integer \( k \in \{1, \ldots, i\} \),
• a non-empty set \( U \) of \( p \) vertices,
• a set \( N \) (possibly empty) such that \( N \subset N^p(U) \)
• a non-empty set \( W \) of \( t \) vertices, \( W = \{w_1, \ldots, w_t\} \), such that
  \[ H1) \quad W \subset N^p(U) \setminus N, \]
  \[ H2) \quad N\langle w_r+1 \rangle \subset N\langle w_r \rangle \text{ for all } r = 1, \ldots, t-1, \]
  \[ H3) \quad N\langle v \rangle \subset N\langle w_t \rangle \text{ for all } v \in N. \]

• integers \( j_1, \ldots, j_{t+1} \in \{1, \ldots, i\} \) such that \( j_1 = 1 \), \( j_{t+1} = i \) and \( j_r \leq j_{r+1} \) for all \( r = 1, \ldots, t \).

We define the \((i, k, U, N, W, j_1, \ldots, j_{t+1})\)-inequality as:

\[
\sum_{u \in U} x_{ui} + \sum_{v \in N} y_{vi} + \sum_{r=1}^{t} \sum_{j=j_r}^{j_{r+1}} y_{w_r,j} + \sum_{v \in N^p(U)} (p-1)y_{vk} + \sum_{q=1}^{p-1} \sum_{v \in N^q(U)} qy_{vk} \leq p. \tag{10}
\]

Before formally proving its validity, we present an example that can make it easier to understand. Consider again the instance of Figure 4 with \( C = V \) and \( m = 7 \). Let \( i = 5 \), \( k = 4 \), \( U = \{2, 8\} \) (thus, \( p = 2 \), \( N = \{2\} \), \( W = \{w_1 = 8, w_2 = 3\} \) (thus, \( t = 2 \), \( j_1 = 1 \), \( j_2 = 3 \) and \( j_3 = 5 \)). As \( N \) and \( W \) are disjoint subsets of \( N^p(U) \) and \( N(2) \subset N(3) \subset N(8) \), the \((5, 4, \{2, 8\}, \{2\}, \{8, 3\}, 1, 3, 5)\)-inequality, i.e.

\[
\sum_{u \in U} x_{ui} + \sum_{v \in N} y_{vi} + \sum_{r=1}^{t} \sum_{j=j_r}^{j_{r+1}} y_{w_r,j} + \sum_{v \in N^p(U)} (p-1)y_{vk} + \sum_{q=1}^{p-1} \sum_{v \in N^q(U)} qy_{vk} \leq 2
\]

is well defined. Below, there is a representation of variables “\( y \)” from the inequality as a matrix having a row per vertex and a column per step.

| Vertex | Step 1 | Step 2 | Step 3 | Step 4 | Step 5 | Step 6 | Step 7 |
|--------|--------|--------|--------|--------|--------|--------|--------|
| 8      | \( y_{s1} \) | \( y_{s2} \) | \( y_{s3} \) | \( y_{s4} \) | \( y_{s5} \) |        |        |
| 3      |        |        |        |        |        |        |        |
| 2      |        |        |        |        |        |        |        |
| 1      |        |        |        |        |        |        |        |
| 4      |        |        |        |        |        |        |        |
| 5      |        |        |        |        |        |        |        |
| 6      |        |        |        |        |        |        |        |
| 7      |        |        |        |        |        |        |        |

15
Variables “y” associated to vertices from \( N \cup W \) form a ladder \((y_{s1}, y_{s2}, y_{s3}, y_{s4}, y_{s5} \text{ and } y_{s6})\). Due to \( N(2) \subset N(3) \subset N(8) \), only one of these variables can be set to 1, so they contribute at most one unit to the l.h.s. Also, there are variables “y” in the 4th. column \((y_{t4}, \ldots, y_{t8} \text{ except } y_{t4})\). They can contribute at most one unit since any feasible solution satisfies constraints \([1]\). Note that the coefficient of \( y_{s4} \) is 2 as this variable belongs to both the ladder and the column.

If one variable from the ladder is set to 1, vertices 2 and 8 are footprinted \((x_{25} = x_{s5} = 0)\) and the inequality is valid. If none of these variables contributes to the l.h.s but some from the column, vertex 8 is footprinted \((x_{s5} = 0)\) and the inequality is again valid (the l.h.s. has one unit from the variable “y” of the column and at most one unit from \( x_{25} \)). If no variable “y” from the inequality is set to 1, then it reduces to \( x_{25} + x_{s5} \leq 2 \) which is trivially valid.

Now that we know it is valid, we wonder if it is not redundant for a given formulation \( F_i \). We can check it with a linear programing, say LP, that maximizes the l.h.s. of the inequality subject to the linear constraints of \( F_i \). We prove \( \Sigma \), the number of vertices \( \Sigma \) that maximizes the l.h.s. of the inequality subject to the linear constraints of \( F_i \). Let \( z_i^* \) denote the objective value of LP. Such value must be greater than 2 for the inequality to cut the linear relaxation of \( F_i \). For \( i \in \{1, 2, 5, 6\} \), we obtain \( z_i^* = 3.89 \). For \( i \in \{3, 4, 7, 8\} \), we obtain \( z_i^* = 3.80 \).

**Theorem 4.3.** The \((i, k, U, N, W, j_1, \ldots, j_{t+1})\)-inequality is valid for \( P_i \).

**Proof.** Let \((x, y)\) be a feasible integer point of \( P_i \). Define \( \Sigma_N = \Sigma_{v \in N} y_{vi}, \Sigma_W = \sum_{r=1}^t \sum_{j=r}^{j=r+1} y_{wj, j}, \Sigma_p = \sum_{v \in N \setminus \{U\}} (p-1) y_{vk}, \Sigma_q = \sum_{q=1}^{p-1} \sum_{v \in N \setminus \{U\}} q y_{vk} \) and \( s = p - \sum_{v \in U} x_{ui} \) (\( s \) is the number of vertices \( u \in U \) such that \( x_{ui} = 0 \)).

We prove \( \Sigma_N + \Sigma_W + \Sigma_p + \Sigma_q \leq s \).

First, note that, if a vertex \( v \in N^q \) is chosen in a step from \( 1, \ldots, i \), then \( v \) footprinted \( q \) vertices from \( U \) implying that \( 1 \leq q \leq s \). Since no more than one vertex can be chosen in step \( k \) and \( N^1 \), \( N^2 \), \ldots, \( N^p \) are disjoint sets of vertices, \( \Sigma_q \leq s \) and \( \Sigma_p + \Sigma_q \leq p - 1 \).

Suppose that \( s < p \). Then, some vertex from \( U \) is not footprinted in step \( i \). Therefore, no vertex from \( N^p \) can be chosen at steps \( 1, \ldots, i \), implying that \( \Sigma_N = \Sigma_W = \Sigma_p = 0 \). As \( \Sigma_N \leq s \), the inequality is valid.

Now, suppose that \( s = p \). Then, all vertices from \( U \) are footprinted along steps \( 1, \ldots, i \). If some vertex from \( N \) is chosen at step \( i \), i.e. \( \Sigma_N = 1 \), then it is not possible to choose any vertex from \( W \) in a step previous to \( i \) due to hypotheses H2-H3, implying that \( \Sigma_W = 0 \). On the other hand, if \( \Sigma_N = 0 \) and \( y_{w, j} = 1 \) for some \( r \in \{1, \ldots, t\} \) and \( j \in \{j_r, \ldots, j_{r+1}\} \), H2 guarantees that it is not possible to choose vertices from \( w_1, \ldots, w_{r-1} \) in steps \( 1, \ldots, j \) nor vertices from \( w_{r+1}, \ldots, w_t \) in steps \( j, \ldots, i \). Therefore, \( \Sigma_N + \Sigma_W \leq 1 \). Since \( \Sigma_p + \Sigma_q \leq p - 1 = s - 1 \), validity follows.

These inequalities are, sometimes, stronger versions of constraints from \( F_i \). For instance, constraint \([2]\) for any \( v \) is dominated by the \((m,m,\{u\},0,\{v\},1,m)\)-inequality for any \( u \in N(v) \), which is \( x_{um} + \sum_{j=1}^{m} y_{vj} \leq 1 \).

Let \( u \in V \) and \( i = 2, \ldots, m \). Another example arises when there exists a vertex \( w \in N(u) \) such that \( N(x) \subset N(w) \) for every \( x \in N(u) \setminus \{w\} \). The
(i, i, \{u\}, N(u)\setminus\{w\}, \{w\}, 1, i)-inequality, i.e. \(x_{ui} + \sum_{v\in N(u)} y_{vi} + \sum_{j=1}^{i-1} y_{wj} \leq 1\) dominates the constraint \([6]\) corresponding to that \(u\) and \(i\).

The following two subfamilies of \((i, k, U, N, W, j_1, \ldots, j_{t+1})\)-inequalities deserve special attention. They can be efficiently separated and have shown to be effective to strengthen the relaxation in our computational experiments (see Section 6).

- **Type I**: Let \(i \in \{2, \ldots, m\}\), \(u \in V\) and \(w \in N(u)\). The \((i, i, \{u\}, \emptyset, \{w\}, 1, i)\)-inequality is
  \[x_{ui} + \sum_{j=1}^{i} y_{wj} \leq 1.\]

- **Type II**: Let \(i \in \{2, \ldots, m\}\), \(k \in \{1, \ldots, i\}\), \(u_1, u_2 \in V\) such that \(u_1 \neq u_2\), and \(w \in N^{\cap} = N(u_1) \cap N(u_2)\). The \((i, k, \{u_1, u_2\}, \emptyset, \{w\}, 1, i)\)-inequality is
  \[x_{u_1i} + x_{u_2i} + \sum_{j=1}^{i} y_{wj} + \sum_{v\in N^\cup} y_{vk} \leq 2,\]
  where \(N^\cup = N(u_1) \cup N(u_2)\). Note that the coefficient of \(y_{wk}\) is 2 in the l.h.s.

5. Obtaining lower and upper bounds

In this section, we devise an algorithm that provides an initial lower and upper bound of \(\gamma_{gr}(G; C)\), which we call \textsc{GetInitialBounds}, and a metaheuristic based on a tabu search that tries to improve the length of the legal sequence associated with the initial lower bound. Both algorithms make heavy use of the following greedy heuristic:

\begin{algorithm}[H]
\caption{Maximize}
\textbf{Input} : A legal sequence \(S = (v_1, \ldots, v_k)\) of \(G; C\).
\textbf{Output} : A maximal legal sequence \((v_1, \ldots, v_{k'})\) with \(k' \geq k\).
\begin{algorithmic}
\State \(W \leftarrow \bigcup_{j=1}^{k} N(v_j)\);
\While {\(W \neq V\)}
\State \(\text{Cand} \leftarrow \{v \in V \setminus S : N(v) \setminus W \neq \emptyset\}\);
\State Choose \(v \in \text{Cand}\) such that \(|N(v) \setminus W| + \beta\) is minimum;
\State \(W \leftarrow W \cup N(v)\);
\State Append \(v\) to \(S\);
\EndWhile
\State Return \(S\) and exit;
\end{algorithmic}
\end{algorithm}

Let \(f(v) = |N(v) \setminus W|\), i.e. the amount of neighbors that can be footprinted by \(v\) at the current step. The heuristic basically chooses those vertices that minimize \(f(v)\). Above, \(\beta\) is a function returning a random real number from \([0, 1]\) distributed uniformly (each call generates a different number). Adding \(\beta\) to \(f(v)\) acts as a tie breaker mechanism: for two different vertices with the same image under \(f\), the choice of one of them will be decided randomly.
5.1. Initial bounds

The procedure GetInitialBounds basically explores every legal sequence of 3 vertices and finds \( m_3 \) (see Corollary 2.3) together with a maximal legal sequence. From now on, and for the sake of readability, we assume that \( \gamma_{gr}(G; C) \geq 3 \) as the pseudocode of GetInitialBounds can be easily modified to handle instances such that \( \gamma_{gr}(G; C) \leq 2 \).

For any \( v_1, v_2 \), let \( v_1 \prec v_2 \) denotes the expression \( N(v_2) \setminus N(v_1) \neq \emptyset \) which means that the sequence \( (v_1, v_2) \) is legal. Note that the relation \( \prec \) can be stored in memory as a boolean matrix for fast access. The algorithm is displayed below:

**Algorithm 2: GetInitialBounds**

| Input | An instance \( G; C \). |
|-------|-------------------------|
| Output | A maximal legal sequence \( S \) of \( G; C \) and the upper bound \( m_3 \). |
| \( \delta_3 \leftarrow n; \) | |
| \( S \leftarrow \{ v_1 \}; \) | |
| for every \( v_1, v_2 \in V \) such that \( v_1 \prec v_2 \) do | |
| for every \( v_3 \in V \) such that \( v_1 \prec v_3 \) and \( v_2 \prec v_3 \) do | |
| \( W \leftarrow N(v_1) \cup N(v_2); \) | |
| if \( N(v_3) \setminus W \neq \emptyset \) then | |
| \( W \leftarrow W \cup N(v_3); \) | |
| if \( |W| < \delta_3 \) then | |
| \( \delta_3 \leftarrow |W|; \) | |
| \( T \leftarrow \text{Maximize}(v_1, v_2, v_3); \) | |
| if the length of \( T \) is greater than \( S \) then | |
| \( S \leftarrow T; \) | |
| Return \( m_3 = n - \delta_3 + 3 \) and \( S \), and exit; |

Note that the previous algorithm looks for a legal sequence by maximizing \( (v_1, v_2, v_3) \) each time \( \delta_3 \) is updated and keeping the longest one. We also experimented by maximizing every legal sequence of size 3 and, although this procedure yields a little better lower bound, it consumes a long CPU time. In fact, the tabu search reaches the same lower bound in less time.

5.2. A tabu search based heuristic

Now, we present the tabu search algorithm. Basically, for a given \( k \), it explores (not necessarily legal) sequences of size \( k \) and finish its execution when:

- a maximal legal sequence of size (at least) \( k \) is found,
- a limit in elapsed time or number of iterations is reached (it fails for short).

Then, the initial sequence \( S \) given by GetInitialBounds can be improved by invoking repeatedly the tabu search with \( k \) equal to the length of \( S \) plus one.

We recall that tabu search is a metaheuristic method proposed by Glover [9] that guides a local search algorithm equipped with an additional mechanism to
prevent from visiting a solution twice and get stuck in a local optimum. Next, we define these concepts briefly.

Let $S$ be the solution space of the problem and $f: S \rightarrow \mathbb{R}$ be the objective function. The goal is to obtain a solution $s \in S$ such that $f(s)$ is minimum. For each $s \in S$, consider a neighborhood $N(s) \subset S$ with two desirable (but not exclusionary) properties:

- if $s, s' \in S$ are neighbors, it is inexpensive (from the computational point of view) to obtain $s'$ from $s$, and $f(s')$ from $f(s)$,
- for any $s, s' \in S$, there exists a path $s = s_1, s_2, \ldots, s_r = s'$ such that $s_i$ and $s_{i+1}$ are neighbors for all $i = 1, \ldots, r - 1$.

Consider a set of features $P$ and a set $R \subset S \times P$ such that $(s, p) \in R$ if $s$ presents feature $p$. In general, neighboring solutions share most of their features.

Starting from an initial solution $s_0 \in \mathcal{S}$, tabu search consists of generating a sequence of solutions $s_1, s_2, \ldots$ such that $s_{i+1} \in \arg \min_{s \in N'(s_i)} f(s)$, where $N'(s_i)$ is a subset of $N(s_i)$ described below. In each iteration, a movement from $s_i$ to $s_{i+1}$ is performed and some feature of $s_{i+1}$ is stored in a tabu list $L$. This list indicates whether a movement is allowed or forbidden:

$$N'(s) = \{ s' \in N(s) : (s', p) \notin R \forall p \in L \}.$$ 

However, after several iterations, old tabu features are no longer needed and it is better to remove them from the tabu list. This mechanism is implemented by assigning a time of live to each feature of the tabu list. Consider a vector live where live($p$) is the number of remaining iterations that $p$ belongs to $L$. When a new feature $p$ is inserted into $L$, live($p$) is assigned a value referred to as tabu tenure. Then, in each iteration, the value of live($p$) is decreased by one unit until it reaches zero and $p$ is removed from $L$.

In our case, these concepts are instantiated as follows:

- **Search space and objective function.** $\mathcal{S}$ contains every sequence of $k$ different vertices. For any $S = (v_1, \ldots, v_k) \in \mathcal{S}$, define the set of conflicting indexes as

  $$\mathcal{I}(S) = \{ i : N(v_i) \setminus \bigcup_{j=1}^{i-1} N(v_j) = \emptyset \}.$$ 

Any vertex $v_i$ such that $i \in \mathcal{I}(S)$ is called conflicting. The objective function is defined as $f(S) = |\mathcal{I}(S)|$.

- **Stopping criterion.** The algorithm stops when $f(S) = 0$, meaning that $S$ is a legal sequence, or an iteration/time limit is reached, meaning that the algorithm fails. In the first case, $S$ is maximalized and returned.

- **Initial solution.** It is generated by randomly picking $k$ different vertices with a uniform distribution.
• **Set of features and tabu list.** Here, vertices are features. The tabu list \( L \) contains those vertices marked as tabu.

• **Neighborhood of a solution.** From a sequence \( S = (v_1, \ldots, v_k) \), a neighbor \( S' = (v'_1, \ldots, v'_k) \in N'(S) \) is constructed through two movements, within the restricted neighborhoods \( N'_1(S) \) and \( N'_2(S) \) respectively, which define \( N'(S) = N'_1(S) \cup N'_2(S) \). In both movements, two vertices are involved and, for the movement to be allowed, one of them should not be marked as tabu. First, define the following set

\[
J(S) = \{ j : N(v_i) \cap N(v_j) \neq \emptyset, \text{ for some } i \in \mathcal{I}(S), \ i > j \}.
\]

If \( j \in J(S) \) due to some \( i \in \mathcal{I}(S) \) then \( v_j \) conflicts with \( v_i \) in the sense that \( v_j \) is a vertex that occurs before \( v_i \) in \( S \) and both share elements from their neighborhoods. Now, we can define the two movements:

- \( N'_1 \): Swap a vertex from the sequence with one from outside. Let \( l \in \mathcal{I}(S) \cup J(S) \) and \( z \in V \setminus \{v_1, \ldots, v_k\} \cup L \) such that \( z \prec v_l \) (if \( l \in J(S) \)) and \( v_l \prec z \) (if \( l \in \mathcal{I}(S) \)). The latter conditions prevent from generating solutions that certainly do not improve the objective function. Then, consider the solution \( S' \) equal to \( S \) except for swapping vertices \( v_l \) and \( z \), i.e. for \( r = 1, \ldots, k \),

\[
v'_r = \begin{cases} 
z, & \text{if } r = l, \\
v_r, & \text{otherwise.}
\end{cases}
\]

- \( N'_2 \): Swap two vertices from the sequence. Let \( l_2 \in \mathcal{I}(S) \cup J(S) \) and \( l_1 < l_2 \) such that \( v_{l_2} \prec v_{l_1} \) and \( v_{l_1} \notin L \). Then, consider the solution \( S' \) equal to \( S \) except for swapping \( v_{l_1} \) and \( v_{l_2} \), i.e. for \( r = 1, \ldots, k \),

\[
v'_r = \begin{cases} 
v_{l_2}, & \text{if } r = l_1, \\
v_{l_1}, & \text{if } r = l_2, \\
v_r, & \text{otherwise.}
\end{cases}
\]

• **Selection of the next solution.** For every \( S' \in N'_1(S) \cup N'_2(S) \), \( f(S') \) is computed and the solution with the lowest value is chosen for the next iteration. In case of a tie, the solution is randomly chosen from those ones with the same objective value.

• **Update of the tabu list and tabu tenure.** If the new solution comes from the first movement by swapping some \( v_l \) with \( z \), then \( v_l \) is added to \( L \). If the new solution was constructed by swapping some \( v_{l_1} \) and \( v_{l_2} \), then \( v_{l_2} \) is added to \( L \). In both cases, the tenure assigned to the incoming vertex in \( L \) is a random integer from the interval \([5, 20]\) with uniform distribution.

Let \( S, S', S'' \) be solutions at three consecutive iterations, i.e. \( S' \in N'(S) \) and \( S'' \in N'(S') \). It is easy to see that, regardless of the movement performed, it
is not possible to obtain \( S'' = S \) due to the tabu mechanism. The algorithm and, in particular, the fact that \( S'' \neq S \) can be illustrated with an example. Consider the graph of Figure [4] with \( C = V \), and \( k = 4 \). Suppose that \( L = \emptyset \) and \( S_0 = (1, 8, 2, 6) \). Since 2 is a conflicting vertex, i.e. \( N(2) \setminus (N(1) \cup N(8)) = 0 \), this sequence is not legal. Also, \( I(S_0) = \{3\} \) and \( J(S_0) = \{1, 2\} \) (vertices 1 and 8 conflict with 2). An allowed movement is to swap \( v_{l_2} = 2 \) with \( v_{l_1} = 8 \), giving rise to the legal sequence \((1, 2, 8, 6)\), which ends the search. However, another movement is to swap \( v_{l_2} = 2 \) with \( v_{l_1} = 1 \). Suppose that we perform the latter movement so as to obtain \( S_1 = (2, 8, 1, 6) \). Then, vertex 2 is marked as tabu. In the next iteration, \( I(S_1) = \{3\} \) (1 is a conflicting vertex) and \( J(S_1) = \{1, 2\} \) (2 and 8 conflict with 1). At this point, it should be noted that the next solution, \( S_2 \), cannot be equal to \( S_0 \) because of \( 2 \in L \) (implying that the swap between \( v_{l_2} = 1 \) and \( v_{l_1} = 2 \) is forbidden). An allowed movement is to swap \( v_l = 8 \) with \( z = 4 \), giving rise to the legal sequence \((2, 4, 1, 6)\), which ends the search.

Below, a pseudocode of the whole algorithm is displayed. There, \( \beta \) is a function returning a random real number from \([0, 1)\) (the same as in \textsc{Maximize}).
Algorithm 3: TabuSearch

Input: An instance \( G; C \) and a positive integer \( k \).
Output: A maximal legal sequence of size at least \( k \) or fail.

Initialize \( S \) with a random sequence of \( k \) different vertices;
\( L \leftarrow \emptyset \);
while iteration/time limit is not reached do
  Compute \( I(S) \);
  if \( |I(S)| = 0 \) then
    Return Maximize \( S \) and exit;
  Compute \( J(S) \);
  for every \( l \in I(S) \cup J(S) \) do
    Compute \( |I(S')| + \beta \) and keep in \( S^* \) the solution with the least value;
  for every \( l_2 \in I(S) \cup J(S) \) such that \( l_2 \geq 2 \) do
    for every \( l_1 = 1, \ldots, l_2 - 1 \) such that \( v_{l_2} \preceq v_{l_1} \) and \( v_{l_1} \not\in L \) do
      Compute \( |I(S')| + \beta \) and keep in \( S^* \) the solution with the least value;
  if no \( S^* \) was found (every movement is forbidden) then
    Fail;
  for every \( v \in L \) do
    \( \text{live}(v) \leftarrow \text{live}(v) - 1; \)
    if \( \text{live}(v) = 0 \) then
      \( L \leftarrow L \setminus \{v\}; \)
  Let \( v_{\text{tabu}} \) be \( v_i \) if \( S^* \) was taken from \( N_i^*(S) \), and \( v_{l_2} \) if \( S^* \) was taken from \( N_{l_2}^*(S) \);
  \( L \leftarrow L \cup \{v_{\text{tabu}}\}; \)
  Assign a random number from [5, 20] to \( \text{live}(v_{\text{tabu}}); \)
  \( S \leftarrow S^*; \)
  Fail;

5.3. Improving search with generation of unrelated solutions

We devised a procedure that provides alternative solutions during the search. It consists in picking at random some non-conflicting vertices from the current solution \( S \) to make a subsequence with them (maintaining the order in which they appear in \( S \)). Clearly, this subsequence of \( S \) is legal. Then, it can be maximalized and, if the resulting sequence has at least \( k \) elements, it is returned as the solution of the search.

The following pseudocode brings details of this procedure, named GenAlternativeSol, and should be added to TabuSearch before the line that computes \( J(S) \):
Algorithm 4: \texttt{GenAlternativeSol}

\begin{algorithm}
\caption{GenAlternativeSol}
\begin{algorithmic}
\If{$k - |\mathcal{I}(S)| \geq |\mathcal{R}| + 1$}
\For{$r \in \mathcal{R}$}
\State Let $v_{l_1}, \ldots, v_{l_r}$ be $r$ random non-conflicting vertices from $S$
\State (i.e. $l_i \notin \mathcal{I}(S)$);
\State $S' \leftarrow \text{Maximize}(v_{l_1}, \ldots, v_{l_r})$;
\EndFor
\If{length of $S'$ is $k$ or greater}
\State Return $S'$ and exit;
\EndIf
\EndIf
\end{algorithmic}
\end{algorithm}

where $\mathcal{R}$ is a set of natural numbers that will be chosen experimentally in the next section.

The CPU time spent by \texttt{GenAlternativeSol} per iteration is negligible and, in most cases, it is able to find legal sequences of size $k$ unexpectedly, without the need to wait for $f(S)$ to converge to zero.

6. Computational experiments

This section is devoted to present computational experiments in order to answer several questions: which formulation performs better?, does the addition of inequalities (10) as cuts improve the performance?, how effective is the tabu search to find good solutions?, how large are the instances that our approach can tackle?

The experiments have been carried out over several random and benchmark instances. A computer equipped with an Intel i7-7700 3.6GHz CPU, 8Gb of RAM, and IBM ILOG CPLEX 12.7 has been used. Each run has been performed on one thread of the CPU.

Random instances are generated as follows. For given numbers $n \in \mathbb{Z}_+$ and $p \in [0, 1]$, a graph is generated by starting from the empty graph of $n$ vertices and adding edges with probability $p$. For example, if $p = 1$, then a complete graph is obtained. It is expected that the resulting graph has an edge density similar to $p$, so graphs with $p = 0.1$ and 0.9 are referred as low and high density respectively, while those ones with $p \in \{0.3, 0.5, 0.7\}$ are referred as medium density.

Our implementation as well as all the instances can be downloaded from:

\url{https://www.fceia.unr.edu.ar/~daniel/stuff/grundy.zip}

6.1. Comparing formulations

In this experiment, we evaluate the 8 formulations presented in Subsection 4.1 over random instances obtained as follows: for each $n \in \{10, 20, 30\}$ and $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, 3 graphs ($G_1, G_2, G_3$) with $n$ vertices and edge probability $p$ are generated, and for each $i \in \{1, 2, 3\}$, two instances are considered: $G_i; V$ (Grundy domination number of $G_i$) and $G_i; \emptyset$ (Grundy total domination number of $G_i$), giving rise to 90 random instances.
Each run consists of invoking GetInitialBounds and solving one of the 8 formulations via a pure branch-and-bound (more precisely, the MIP optimizer of CPLEX is used, with heuristics, cuts and presolve turned off). At the beginning of the optimization, the initial solution is injected as the first incumbent. Neither inequalities (9) nor (10) are considered in this experiment. A time limit of two hours is imposed for each run.

Table 1 reports the results. Each row presents information about the 6 instances with the same number of vertices $n$ and probability $p$, given in the first and second columns. The third column ($RG_{initial}$) shows the average of initial percentage relative gaps (i.e. the value $100(UB - LB)/LB$ where $LB$ and $UB$ are the bounds provided by GetInitialBounds) over the 6 instances. The next columns report the following values: “RelGap” refers to the average over the 6 instances of percentage relative gaps at the end of the optimization, “Solved” is the number of instances solved within the established time limit, and “Time” is the average over the solved instances of time elapsed during optimization (in seconds). In case none of the 6 instances are solved, a mark “−” is displayed. Some values mentioned in the analysis are highlighted in boldface.

All high density instances are not reported since they are satisfactorily solved by GetInitialBounds.

**Analysis.** At first glance, we can observe that the harder instances are those with 30% of density. In fact, the lower the density of the graph is the longer the sizes of maximal legal sequences are, and thus, more variables the models have. Regarding the order of graphs, for 10 vertices the resolution is straightforward, but for 20 and 30 vertices, there are instances that cannot be solved in two hours of CPU time. In general, the formulations could close or significantly reduce the initial gaps of instances up to 20 vertices or more than 50% of edge density. On the other hand, we did not perceive any tendency between instances with $V = C$ and $V = ∅$.

For $n = 20$ and $p = 0.3$, $F_3$ and $F_4$ solve 2 of 6 instances and both formulations report the smallest gap (and $F_3$ uses 7.5% less time than $F_4$, a small difference). For $n = 20$ and $p = 0.5$, $F_4$ and $F_6$ solve all the instances between (roughly) 2 and 3 times faster than the others. For $n = 30$ and $p = 0.5$, $F_3$, $F_4$ and $F_6$ solve almost all instances (compared to the other formulations) and, in particular, $F_4$ presents the smallest gap and elapsed time. For $n = 30$ and $p = 0.7$, $F_4$ and $F_8$ solve all the instances between (roughly) 2 and 5 times faster than the others. By taking these facts into account, we conclude that $F_4$ performs better than the others on average.

**6.2. Reinforcing the relaxations**

As we have pointed out in Subsection 4.3, the addition of violated valid inequalities to the relaxations can improve the performance of the solver. Our cutting-plane algorithm consists of the separation of inequalities (10), specifically Type I and II (see Subsection 4.3). Both have its own routine which is invoked after a linear relaxation is solved. If at least one cut is generated, it is
added to the relaxation and the latter is reoptimized. In particular, the separation of Type I inequalities is performed 10 times in the root node (that means at most 10 reoptimizations), twice in nodes with depths 1 and 2, and once in nodes with depths 3 to 10. The routine that separates Type II inequalities is executed after the one for Type I in nodes with depth at most 5. Below, we describe the implementation of both routines. The current fractional solution is denoted by \((x^*, y^*)\).

- **Separation of Type I inequalities.** Before starting the optimization, create sets

  \[ W_u = \{ w \in N(u) : |N(w)| \geq 2 \text{ and, for all } v \in N(u) \setminus \{w\}, w \prec v \text{ and } v \prec w \} \]

  for each \( u \in V \). Each time the separation routine is invoked, assign \( \mathcal{A} \leftarrow V \). Then, for every \( u \in V \) and \( w \in W_u \cap \mathcal{A} \) do the following. Set \( \text{sum} \leftarrow y^*_w \). For all \( i = 2, \ldots, m \), do \( \text{sum} \leftarrow \text{sum} + y_{wi} \) and check whether \( x^*_{ui} + \text{sum} > 1.1 \). In that case, add \( x_{ui} + \sum_{j=1}^{i} y_{wj} \leq 1 \) as a cut and remove \( w \) from \( \mathcal{A} \).

  Set \( \mathcal{A} \) stores those vertices “w” not used by cuts from previous iterations, thus preventing the generation of cuts with similar support.

- **Separation of Type II inequalities.** Before starting the optimization, create sets

  \[ W_{u_1u_2} = \{ w \in W_{u_1} \cap W_{u_2} : \text{there are } z_1 \in N(u_1) \setminus N(u_2), z_2 \in N(u_2) \setminus N(u_1) \text{ such that } N(w) \setminus (\{u_2\} \cup N(z_1)) \neq \emptyset, N(w) \setminus (\{u_1\} \cup N(z_2)) \neq \emptyset \} \]

  for each pair \( \{u_1, u_2\} \subset V \). The separation routine is executed immediately after the separation of Type I inequalities and makes use of the vertices that remains in \( \mathcal{A} \). For every \( \{u_1, u_2\} \subset V \) and \( w \in W_{u_1u_2} \cap \mathcal{A} \) do the following. Set \( \text{sum} \leftarrow y^*_w \). For all \( i = 2, \ldots, m \), do \( \text{sum} \leftarrow \text{sum} + y_{wi} \) and check whether \( x^*_{u_1i} + x^*_{u_2i} + \text{sum} + \sum_{v \in N \cup Y} y_{vk} > 2.2 \) and, in that case, add \( x_{u_1i} + x_{u_2i} + \sum_{j=1}^{i} y_{wj} + \sum_{v \in N \cup Y} y_{vk} \leq 2 \) as a cut and remove \( w \) from \( \mathcal{A} \).

These routines have been designed in a previous work (an extended abstract) where some polyhedral aspects, such as the dimension of the face define the inequality, have been taken into account [7].

In Subsection 4.2, we presented a set of equalities that can be added to the formulation. We propose two treatments of these equalities:

- **Addition of equalities at the beginning.** Before starting the optimization (i.e. when the model is populated in the memory of CPLEX), add [9] for \( i = 1, \ldots, LB \), where \( LB \) is the lower bound generated by GetInitialBounds. Also, do not add [1] for \( i = 1, \ldots, LB \) nor [7] for \( i = 2, \ldots, LB \).
- **Treatment of equalities as cuts.** The following routine is executed after a linear relaxation is solved. Let \((x^*, y^*)\) be the current fractional solution and \(z \in \mathbb{Z}\) be the objective function value of the best integer solution found so far. In other words, \(z\) is the best available lower bound. For all \(i = z, \ldots, 1\), check whether \(\sum_{v \in V} y^*_{vi} < 0.9\) and, in that case, add \(\sum_{v \in V} y_{vi} = 1\) as a cut. Otherwise, exit the loop.

Preliminary experiments show that the first approach is better, possibly because the lower bound is infrequently improved. Actually, the initial lower bound is usually already close to the optimum. From now on, when we refer to equalities (9), we consider they are added at the beginning of the optimization.

In the next experiment, we evaluate the presence of inequalities (10) (types I and II) and equalities (9) during the optimization, over the same set of instances of the previous experiment. We consider the following 6 variants: Base (i.e. no cuts), \(+T_1\) (i.e. with cuts of type I), \(+T_1 + T_2\) (i.e. with both type of cuts), \(+9\), \(+9 + T_1\), and \(+9 + T_1 + T_2\). Formulation \(F_4\) is used in all cases. Each run consists of invoking \textsc{GetInitialBounds} and solving one of these 6 variants. A time limit of two hours is imposed. Again, CPLEX cuts, heuristics and presolve are turned off.

Table 2 reports the results in the same format as Table 1. Instances of 10 vertices have been omitted as they are too easy for all the variants.

**Analysis.** For \(n = 20\) and \(p = 0.1\), the addition of equalities reduces the average of CPU time to the half and presents a little improvement in the relative gap, and for \(p = 0.3\), it is also able to solve 2 more instances. In particular, the best variant is \(+9 + T_1 + T_2\), and in second place, \(+9 + T_1\). Cuts lose effect in the densest instances, although these ones are also easier to solve. For \(n = 30\) and \(p = 0.5\), \(+9 + T_1\) performs better as it solves all the instances. Again, cuts lose effect for instances of high density \((n = 30\) and \(p = 0.9\)).

From these computational experiments, we propose to add (9) as well as to enable cuts of type I when the density of the graph is less than 60%, and enable cuts of type II when the density is less than 40%.

6.3. Determination of set \(R\)

In Subsection 5.3, we present a procedure that eventually provides legal sequences during the tabu search. These sequences are generated by maximalizing a subsequence of \(r\) non-conflicting vertices taken from the current solution. Here, we carry out an experiment in order to determine which values of \(r\) yield the longest sequences.

Tabu search is executed over random instances of 100 and 200 vertices (and \(C = V\)) for one hour, and with a given initial \(k\). \textsc{GenAlternativeSol} is implemented with \(R = \{2, 3, \ldots, 9\}\) but, each time it finds a sequence of size \(k\) from another of size \(r \in R\), a counter associated to \(r\) is incremented by one unit and that solution is discarded. The values of \(k\) were chosen so that there exists a sequence of size \(k\) but the tabu search does not find anyone within one hour of time. For this reason, only graphs with density up to 50% were considered,
since it seems that the tabu search quickly converges to the optimal solution for higher densities.

Figure 5 shows 6 histogram-like charts, each one corresponding to a graph with \( n \in \{100, 200\} \) and \( p \in \{0.1, 0.3, 0.5\} \). The value of \( k \) is also reported. For each \( r \in \mathcal{R} \), a bar is drawn along with the counter for \( r \), i.e. the number of times GENALTERNATIVESol reaches a sequence of size \( k \) from another of size \( r \).

**Analysis.** The values of \( r \) that yield the longest sequences are 2, 3 and 4. Although \( r = 2 \) is the best in most cases, \( r = 3 \) is better for the hardest case (sequences of size 78, the longest one). Note also that \( r = 4 \) yields sequences of size \( k \) in all tested instances, but this fact does not happen for \( r \geq 5 \). We conclude that \( \mathcal{R} = \{2, 3, 4\} \) is a reasonable setting for GENALTERNATIVESol.

In order to find out how much this procedure improves the search of sequences, we run two versions of the tabu search: one with GENALTERNATIVESol enabled (with \( \mathcal{R} = \{2, 3, 4\} \)), and the other, disabled. Same instances as before are used, plus others with \( n \in \{100, 200\} \) and \( p \in \{0.7, 0.9\} \). Due to the non-deterministic nature of the algorithm, three runs per instance are performed. Each run starts with \( k = 3 \) and each time a sequence of size \( k \) is found, the tabu search is restarted with \( k + 1 \). Table 3 shows the best \( k \) achieved by each run and the time the algorithm took to reach it in brackets. Best run is highlighted in boldface.

**Analysis.** GENALTERNATIVESol dramatically improves the search of long legal sequences. In a matter of seconds, it finds solutions that are not possible to obtain otherwise within one hour of execution. From now on, it is enabled by default.

### 6.4. Limits of our exact algorithm

The goal of this experiment is to estimate the largest size of an instance that can be solved in a fixed amount of time (four hours). In each run, GETINITIALBounds and the tabu search are invoked with a total time limit of 30 seconds, and a maximum of 50000 iterations for the tabu search. Then, formulation \( F_4 \) is solved. Equalities \([9]\) and cuts are added/enabled according to the criterion given in Subection 6.2. In order to differentiate the phases of the algorithm, we call initial phase to the search for initial bounds (GETINITIALBounds and the tabu search) and optimization phase to the resolution of the integer formulation.

The following instances are considered:

- **Graphs from the DIMACS challenge** ([https://mat.gsia.cmu.edu/COLOR04](https://mat.gsia.cmu.edu/COLOR04)).

It is a standard set of benchmark instances which were originally selected for testing graph coloring algorithms, although later it was used for other optimization problems in graphs, in particular dominating set problems \([8]\). We consider those graphs up to 50 vertices, and their complements (names are suffixed with letter c to identify them). For each graph, we
create two instances: one with $C = V$ and the other with $C = \emptyset$, giving a total amount of 32 instances.

• **Random instances.** We consider graphs of 25 and 50 vertices, with edge probability $p \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ and $C \in \{V, \emptyset\}$, giving a total amount of 20 instances. Each instance is identified by $G_{n,p}$.

• **Real instances** ([https://www.buenosaires.gob.ar/laciudad/barrios](https://www.buenosaires.gob.ar/laciudad/barrios)). We consider two instances based on the map of neighborhoods of the city of Buenos Aires, see Figure 6. In the first one *(full version)*, each district corresponds to a neighborhood, giving an amount of 48. In the second one *(small version)*, each district is associated to one or more neighborhoods, depending on the total area (e.g. Villa Ortúzar, Parque Chas and other small neighborhoods are gathered into one district), giving an amount of 21. Table 7 gives the districts considered in each instance, and the best solutions found by our algorithm (expressed by the allocation order of the companies). In particular, the solution for the second case is optimal.

Table 4 reports the results. Each row corresponds to two instances, one with $C = V$ and the other with $C = \emptyset$. The first, second and third columns report the name of the instance, the number of vertices and the edge density. The next columns give the best upper/lower bounds obtained at the beginning and at the end of the optimization. A star “*” informs that the initial upper bound is provided by the user. A mark “−” is displayed if the time limit is reached. A dagger “†” indicates that the tabu search improves the solution given by GETINITIALBOUNDS. Values in boldface reveal an improvement during the optimization phase. If the optimality is reached by the initial phase, final bounds are not reported.

**Analysis.** Despite the short time allocated to the initial phase, it is very effective, mainly on high density instances. It is able to solve 12 instances (out of 54) in a matter of milliseconds. Also, for every solved instance, the initial phase actually provides the optimal solution. In particular, the tabu search performs very well: such instances where GETINITIALBOUNDS does not provide the optimal solution, the latter is delivered by the tabu search; also it improves the solution generated by GETINITIALBOUNDS on several hard instances such as low-density graphs of 50 vertices.

Regarding the optimization phase, our approach is able to exactly solve more than one-third of the instances (16 out of 42) and to decrease the upper bound (see values in boldface) in half of the cases. A limit of our approach seems to be based on the initial upper bound. It is unlikely that the instance could be solved for $UB \geq 20$ in four hours. On the other hand, instances with $UB \leq 6$ are easily solved regardless of the size of the graph.

As the initial solutions are optimal or near the optimal solution, it is natural to propose the following procedure. Instead of using the upper bound provided by the initial phase, set $UB \leftarrow LB + 1$. Therefore, if the optimization finishes
with objective value equal to the initial $LB$, then the initial solution is optimal. In other words, we are using the solver to decide whether $\gamma_{gr}(G; C) \geq LB + 1$ or not. We evaluated with this procedure those instances where there is a gap of at least two units between $UB$ and $LB$, and we were able to solve one more instance: $\gamma_{gr}(queen6_6c, \emptyset) = 8$.

6.5. Improving bounds of $\gamma_{gr}(K_{n,r}; V)$

For given $r$, $n$ positive integers such that $n \geq 2r$, the Kneser graph $K_{n,r}$ is defined as follows. The vertex set represents all subsets of $\{1, \ldots, n\}$ with $r$ elements, and two vertices are adjacent if and only if the corresponding subsets are disjoint. Since, for $r = 1$, this graph is isomorphic to a complete one and, for $n = 2r$, it is isomorphic to a disjoint union of $\binom{2r}{r}/2$ edges, we assume that $r \geq 2$ and $n \geq 2r + 1$.

Due to its structure and properties, there is interest in knowing the value of different graph parameters of the Kneser graph (see, for example, the famous Lovász’s proof of Kneser conjecture [13]). In particular, in a recent work [9], the authors give the Grundy total domination number of the Kneser graph: $\gamma_{gr}(K_{n,r}; \emptyset) = \binom{n}{r}$. The Grundy domination number is, however, partially characterized. They prove that if $n$ is large enough, then $\gamma_{gr}(K_{n,r}; V)$ coincides with the independence number of $K_{n,r}$:

**Theorem 6.1.** [9] For any $r \geq 2$, there exists $\bar{n}_r \in \mathbb{Z}_+$ such that $\gamma_{gr}(K_{n,r}; V) = \binom{n}{r}$ for any $n \geq \bar{n}_r$. In particular, $\bar{n}_2 = 6$.

They also compute the following cases: $\gamma_{gr}(K_{5,2}; V) = 5$ and $\gamma_{gr}(K_{7,3}; V) = 20$. However, for $r \geq 3$, $\bar{n}_r$ remains unknown and the best bounds in the literature are:

$$\binom{n}{r-1} \leq \gamma_{gr}(K_{n,r}; V) \leq \binom{n}{r} - \binom{n-r}{r}$$

The lower bound comes from the length of a sequence whose set of vertices is a maximum independent set of $K_{n,r}$, while the upper bound is given by Prop. 2.1 of [3] and is equal to $m_1$ (defined in Subsection 2.1).

This subsection intends to improve these bounds for some cases, specifically Kneser graphs with up to 800 vertices. Table 5 reports the results obtained by invoking GetInitialBounds and just after the tabu search, for an hour of CPU time. The first three columns report the parameters $n$ and $r$, and the number of vertices. The next two columns show the value $\binom{n}{r} - \binom{n-r}{r}$ and $m_3$ (given by GetInitialBounds). The last columns have $\binom{n-r}{r-1}$ and the length of the best legal sequence found together with the time it took to reach such a sequence in brackets. A dagger ($\dagger$) indicates that the tabu search improves the solution given by GetInitialBounds. Best values are displayed in boldface.

**Analysis.** Although it is expected that $m_3 \leq m_1$ (see Subsection 2.1), in these instances $m_3$ is strictly lesser. Thus, our procedure provides better upper bounds. Besides, our approach also computes better lower bounds for $K_{8,3}$, $K_{9,4}$, $K_{10,4}$,
\(K_{11,4}, K_{11,5}\) and \(K_{12,5}\), despite the fact that it does not exploit any particular characteristic of the structure of these graphs. In particular, the best sequence is achieved by the tabu search on three of the hardest instances \((K_{10,4}, K_{11,4}\) and \(K_{12,5}\)).

Note that our approach is not able to provide a legal sequence larger than \(\binom{n-1}{r-1}\) for \(r = 3\) and \(n \geq 9\). The same happens for \(r = 4\) and \(n \geq 12\). We believe that \(\binom{n-1}{r-1}\) is the optimal solution for these cases and we conjecture that \(\bar{n}_r = 3r\) for any \(r\) in Theorem 6.1.

6.6. Tabu search on large instances

In the experiment performed in the previous subsection, we observed that \text{GenInitialBounds} spends a considerable amount of time, e.g. for \(K_{12,4}\) (roughly 500 vertices) it takes 170 seconds and for \(K_{12,5}\) (roughly 800 vertices) it takes 613 seconds. For graphs with more than 1000 vertices, this heuristic becomes impractical, e.g. for \(K_{20,3}\) (1140 vertices) it takes 11500 seconds. This behavior is expected since its time complexity is cubic on the number of vertices of the graph.

However, the tabu search (without the initial solution given by \text{GenInitialBounds}) can still generate good solutions for large instances. In this last experiment, we consider some instances up to 10000 vertices where the optimal parameter is already known:

- Grundy total domination on Kneser graphs, \(\gamma_{gr}(K_{n,r}; \emptyset) = \binom{2r}{r} \) \([6]\),
- Grundy domination on Web graphs, \(\gamma_{gr}(W_{2n}; V) = n - 2r\) (Prop. 2.8).

We set \(UB\) with the optimum value and execute the tabu search starting from \(k = 3\) and do not stop until a legal sequence of size \(UB\) is reached. Time in seconds for each instance is reported in Table 6. Observe that, in all cases, the tabu search is able to find the optimal solution within one hour of CPU time.

7. Conclusions

In this work, an optimization problem that generalizes the Grundy domination and Grundy total domination problems is introduced. Some properties of this problem and the exact value of the parameter \(\gamma_{gr}(G; C)\) for two families of graphs are given. This problem is modeled as an integer linear program. Some additional families of constraints are considered in order to provide different formulations of the same model. The validity of another family of inequalities, \([10]\), is proved. Since these inequalities are very generic and its number is exponential, two subfamilies (named Type I and II) are considered and polynomial-time routines for separating them are detailed.

On the other hand, a greedy heuristic, \text{GenInitialBounds}, is proposed. It provides an initial legal sequence and an upper bound of \(\gamma_{gr}(G; C)\) which, for some high density graphs \(G\), is able to certify the optimality of the obtained sequence. The size of the initial legal sequence can be further improved by
a tabu search. This algorithm dramatically improves its performance when an additional mechanism, GenAlternativeSol, is added. For all instances where optimality could be proved, the tabu search with GenAlternativeSol was able to find the optimal solution within one hour. Experiments give evidence that this approach yields optimal or near-optimal solutions for instances up to 10000 vertices.

Our exact approach, i.e. GenInitialBounds plus the tabu search with GenAlternativeSol plus the optimization of one of the formulations with the aid of a cutting-plane algorithm that separates Type I and II inequalities, can exactly solve instances ranging from 20 to 50 vertices depending on the edge density of the graph (this includes one of two real-life instances).

Besides the computational results, the resolution of the Grundy domination problem on Kneser graphs of several sizes allows us to state a theoretical conjecture: that for any \( r \geq 2 \) and \( n \geq 3r \), \( \gamma_{gr}(K_{n,r}; V) = \left( \frac{n-1}{r} \right) \).

**References**

[1] Berge, C., 1989. Hypergraphs. North-Holland, Amsterdam.

[2] Brešar, B., Klavžar, S., Rall, D.F., 2010. Domination game and an imagination strategy. *SIAM J. Discrete Math.* 24, 979–991.

[3] Brešar, B., Gologranc, T., Milanič, M., Rall, D.F., Rizzi, R., 2014. Dominating sequences in graphs. *Discrete Math.* 336, 22–36.

[4] Brešar, B., Henning, M.A., Rall, D.F., 2016. Total dominating sequences in graphs. *Discrete Math.* 339, 1665–1676.

[5] Brešar, B., Kos, T., Nasini, G., Torres, P., 2017. Total dominating sequences in trees, split graphs, and under modular decomposition. *Discrete Optim.* 28, 16–30.

[6] Brešar, B., Kos, T., Torres, P., 2019. Grundy domination and zero forcing in Kneser graphs. *Ars Math. Contemp.* 17, 419–430.

[7] Campêlo, M., Severín, D., 2017. Facets of the polytope of legal sequences. *Electron. Notes Discrete Math.* 62, 15–20.

[8] Chalupa, D., 2018. An order-based algorithm for minimum dominating set with application in graph mining. *Inf. Sci.* 426, 101–116.

[9] Glover, F., McMillan, C., Novick, B., 1985. Interactive decision software and computer graphics for architectural and space planning. *Ann. Oper. Res.* 5, 557–573.

[10] Kinnersley, W.B., West, D.B., Zamani, R., 2013. Extremal problems for game domination number. *SIAM J. Discrete Math.* 27, 2090–2107.

[11] Košmrlj, G., 2014. Realizations of the game domination number. *J. Comb. Optim.* 28, 447–461.
[12] Margot, F., 2010. Chapter 17: Symmetry in Integer Linear Programming. In: Jünger M. et al. (eds) 50 Years of Integer Programming 1958–2008. Springer, Berlin, Heidelberg.

[13] Matoušek, J., 2004. A combinatorial proof of Kneser’s conjecture. Combinatorica 24, 163–170.

[14] Pécher, A., Wagler, A.K., 2006. Almost all webs are not rank-perfect. Math. Program. 105, 311–328.

[15] Wolsey, L.A., 1998. Integer programming. 1st. edition. Wiley-Interscience, New York.
| $n$ | $p$ | $R_{\text{initial}}$ | Param. | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ |
|-----|-----|-----------------|--------|------|------|------|------|------|------|------|------|
| 10  | 0.1 | 20.73           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 0.9  | 0.8  | 0.5  | 0.4  | 0.8  | 0.8  | 0.4  | 0.4  |
| 10  | 0.3 | 14.88           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 1.2  | 0.9  | 1.0  | 1.1  | 1.2  | 1.1  | 1.4  | 1.4  |
| 10  | 0.5 | 13.75           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 0.7  | 0.6  | 0.6  | 0.6  | 0.7  | 0.6  | 0.6  | 0.5  |
| 10  | 0.7 | 2.78            | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  | 0.1  |
| 20  | 0.1 | 20.21           | RelGap | 4.98 | 4.98 | 6.01 | 6.01 | 4.98 | 4.98 | 7.05 | 6.01 |
|     |     |                 | Solved | 3    | 3    | 3    | 3    | 3    | 3    | 3    | 3    |
|     |     |                 | Time   | 1113.8 | 1444.5 | 1483.5 | 2049.8 | 1286.2 | 1211.5 | 1283.7 | 1696.7 |
| 20  | 0.3 | 18.15           | RelGap | 6.93 | 12.06 | 6.84 | 6.84 | 11.74 | 9.47 | 10.81 | 12.20 |
|     |     |                 | Solved | 1    | 0    | 2    | 2    | 0    | 1    | 0    | 0    |
|     |     |                 | Time   | 5231.9 | − | 4650.8 | 5026.7 | − | 4224.4 | − | − |
| 20  | 0.5 | 19.41           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 289.4 | 356.6 | 272.2 | 98.4 | 290.1 | 97.4 | 312.5 | 183.5 |
| 20  | 0.7 | 16.13           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 18.8 | 10.0 | 11.9 | 5.5  | 11.7 | 10.4 | 15.3 | 11.4 |
| 30  | 0.1 | 18.04           | RelGap | 18.02 | 17.99 | 18.00 | 18.04 | 18.01 | 17.71 | 18.04 | 18.04 |
|     |     |                 | Solved | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
|     |     |                 | Time   | −    | −    | −    | −    | −    | −    | −    | −    |
| 30  | 0.3 | 24.42           | RelGap | 24.42 | 24.42 | 24.42 | 24.42 | 24.42 | 24.42 | 24.42 | 24.42 |
|     |     |                 | Solved | 0    | 0    | 0    | 0    | 0    | 0    | 0    | 0    |
|     |     |                 | Time   | −    | −    | −    | −    | −    | −    | −    | −    |
| 30  | 0.5 | 16.77           | RelGap | 9.60 | 7.97 | 3.03 | 1.52 | 9.18 | 5.98 | 8.06 | 3.03 |
|     |     |                 | Solved | 2    | 2    | 5    | 5    | 2    | 3    | 2    | 5    |
|     |     |                 | Time   | 4241.4 | 3871.1 | 2964.8 | 2480.2 | 5305.1 | 3541.5 | 4305.5 | 2722.3 |
| 30  | 0.7 | 18.65           | RelGap | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
|     |     |                 | Solved | 6    | 6    | 6    | 6    | 6    | 6    | 6    | 6    |
|     |     |                 | Time   | 254.9 | 218.2 | 197.2 | 110.5 | 384.9 | 628.6 | 205.5 | 110.9 |

Table 1: Comparison of formulations
| n  | p  | RelGap | Param. | Base | $F_4$ | $F_4 + T_1$ | $F_4 + T_1 + T_2$ |
|----|----|--------|--------|------|-------|-------------|-----------------|
| 20 | 0.1| 20.21  | Solved | 3    | 4.98  | 4.98        | 4.98            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 2049.8 | 981.7 | 969.2      |                 |
| 20 | 0.3| 18.15  | Solved | 2    | 7.04  | 5.80        | 8.23            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 5026.7 | 2485.1 | 3025.4     | 2572.6          |
| 20 | 0.5| 19.41  | Solved | 6    | 0.00  | 0.00        | 0.00            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 98.4 | 80.7  | 101.5       | 99.5            |
| 20 | 0.7| 16.13  | Solved | 6    | 0.00  | 0.00        | 0.00            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 5.5 | 8.5   | 16.4       | 12.8            |
| 30 | 0.1| 18.04  | Solved | 0    | 0.00  | 0.00        | 0.00            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | --  | --    | --         | --              |
| 30 | 0.3| 24.42  | Solved | 0    | 0.00  | 0.00        | 0.00            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | --  | --    | --         | --              |
| 30 | 0.5| 16.77  | Solved | 5    | 1.52  | 6.36        | 6.36            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 2480.2 | 2585.6 | 3116.4     | 3565.2          |
| 30 | 0.7| 18.65  | Solved | 6    | 0.00  | 0.00        | 0.00            |
|    |    |        |        |      |       |             |                 |
|    |    |        | Time   | 110.5 | 650.3  | 218.3       | 172.1           |

Table 2: Strengthening the relaxation

Figure 5: Number of times a sequence of size $k$ is reached from another of size $r$
| n   | p   | run 1      | run 2      | run 3      | run 1      | run 2      | run 3      |
|-----|-----|------------|------------|------------|------------|------------|------------|
| 100 | 0.1 | 58(593)   | 58(1850)   | 57(257)   | 59(50.50)  | 59(8.47)   | 59(24.6)   |
| 100 | 0.3 | 25(192)   | 26(176)    | 25(147)   | 26(0.09)   | 26(0.70)   | 26(0.09)   |
| 100 | 0.5 | 15(37.6)  | 15(8.06)   | 15(36.7)  | 16(1.36)   | 16(0.14)   | 16(1.36)   |
| 100 | 0.7 | 10(2.16)  | 10(1.03)   | 10(4.2)   | 11(1.75)   | 11(2.08)   | 11(1.75)   |
| 100 | 0.9 | 6(1.44)   | 6(2.09)    | 6(0.47)   | 6(0.09)    | 6(0.14)    | 6(0.14)    |
| 200 | 0.1 | 75(738)   | 75(2352)   | 75(3249)  | 78(7.94)   | 78(9.69)   | 78(7.94)   |
| 200 | 0.3 | 31(598)   | 31(3052)   | 32(3571)  | 33(8.11)   | 33(8.75)   | 33(7.17)   |
| 200 | 0.5 | 18(266)   | 19(1486)   | 19(169)   | 20(3.19)   | 20(0.70)   | 20(0.70)   |
| 200 | 0.7 | 12(235)   | 11(44.4)   | 12(307)   | 12(0.23)   | 12(1.31)   | 12(0.23)   |
| 200 | 0.9 | 7(2.37)   | 7(4.44)    | 7(9.56)   | 7(2.12)    | 7(9.56)    | 7(2.12)    |

Table 3: Evaluation of GenAlternativeSol

| Name     | n   | dens (%) | UB | LB | UB | LB | time |
|----------|-----|----------|----|----|----|----|------|
| myciel3  | 11  | 36.36    | 6  | 6  | 6  | 6  | 0.7  |
| myciel4  | 23  | 28.06    | 13 | 13 | 13 | 13 | 0.50 |
| queen5_5 | 25  | 53.33    | 8  | 8  | 7  | 7  | 288.2|
| 1-FullIns_3| 30 | 22.99    | 24 | 24 | 24 | 24 | 288.2|
| queen6_6 | 36  | 46.03    | 12 | 12 | 12 | 12 | 11   |
| 2-Insertions_3 | 37 | 10.81 | 32 | 32 | 32 | 32 | 19   |
| myciel5  | 47  | 21.83    | 40 | 40 | 40 | 40 | 19   |
| queen7_7 | 49  | 40.48    | 19 | 19 | 19 | 19 | 19   |

| Name     | n   | dens (%) | UB | LB | UB | LB | time |
|----------|-----|----------|----|----|----|----|------|
| myciel3c | 11  | 63.64    | 4  | 4  | 4  | 4  | 0.0  |
| myciel4c | 23  | 71.94    | 5  | 5  | 5  | 5  | 0.5  |
| queen5_5c| 25  | 46.67    | 10 | 9  | 9  | 9  | 266.6|
| 1-FullIns_3| 30 | 77.01    | 8  | 8  | 8  | 8  | 1076 |
| queen6_6c| 36  | 53.97    | 10 | 9  | 9  | 9  | 7530 |
| 2-Insertions_3c | 37 | 89.19 | 4 | 4 | 4 | 4 | 8334 |
| myciel5c | 47  | 78.17    | 8  | 8  | 8  | 8  | 80.5 |
| queen7_7c| 49  | 59.52    | 10 | 10 | 10 | 10 | 112.5|

| Name     | n   | dens (%) | UB | LB | UB | LB | time |
|----------|-----|----------|----|----|----|----|------|
| buenosaires_full | 48 | 10.20    | 44 | 44 | 44 | 44 | 0.1  |
| buenosaires_small | 21 | 20.48    | 18 | 18 | 18 | 18 | 11952|

Table 4: Evaluation on benchmark instances
| n  | r  | \( |V| \) | \( m \) \_known | \( m \) \_known | Tabu |
|----|----|--------|---------------|---------------|-------|
| 8  | 3  | 56     | 46            | 37            | 21    | 22 (0.03) |
| 9  | 3  | 84     | 64            | 50            | 28    | 28 (0.20) |
| 10 | 3  | 120    | 85            | 65            | 36    | 36 (1.00) |
| 11 | 3  | 165    | 109           | 82            | 45    | 45 (3.92) |
| 12 | 3  | 220    | 136           | 101           | 55    | 55 (13.2) |
| 13 | 3  | 286    | 166           | 122           | 66    | 66 (41.8) |
| 14 | 3  | 364    | 199           | 145           | 78    | 78 (107)  |
| 15 | 3  | 455    | 235           | 170           | 91    | 91 (267)  |
| 16 | 3  | 560    | 274           | 197           | 105   | 105 (624) |
| 17 | 3  | 680    | 316           | 226           | 120   | 120 (1382)|
| 9  | 4  | 126    | 121           | 115           | 56    | 77 (0.44) |
| 10 | 4  | 210    | 195           | 179           | 84    | 93 (4.08) |
| 11 | 4  | 330    | 295           | 265           | 120   | 121 (184) |
| 12 | 4  | 495    | 425           | 375           | 165   | 165 (169) |
| 13 | 4  | 715    | 589           | 512           | 220   | 220 (861) |

Table 5: Improving known bounds for Kneser graphs

| Graph  | \( C \) | \( |V| \) \_known | \( V \) \_known | \( \gamma_{\psi}(G;C) \) | time |
|--------|--------|----------------|----------------|-------------------------|------|
| \( K_{22,4} \) | \( \emptyset \) | 7315           | 41.8           | 70                      | 912  |
| \( K_{23,4} \) | \( \emptyset \) | 8855           | 43.8           | 70                      | 1649 |
| \( K_{17,5} \) | \( \emptyset \) | 6188           | 12.8           | 252                     | 474  |
| \( K_{18,5} \) | \( \emptyset \) | 8568           | 15.0           | 252                     | 1268 |
| \( K_{15,6} \) | \( \emptyset \) | 5005           | 1.7            | 924                     | 222  |
| \( K_{16,6} \) | \( \emptyset \) | 8008           | 2.6            | 924                     | 891  |
| \( K_{15,7} \) | \( \emptyset \) | 6435           | 0.12           | 3432                    | 421  |
| \( W_{10000,10000} \) | \( V \) | 10000          | 20             | 8000                    | 1856 |
| \( W_{10000,10000} \) | \( V \) | 10000          | 40             | 6000                    | 2154 |
| \( W_{10000,10000} \) | \( V \) | 10000          | 60             | 4000                    | 2398 |
| \( W_{10000,10000} \) | \( V \) | 10000          | 80             | 2000                    | 2497 |

Table 6: Performance of tabu search on large instances
Figure 6: Neighborhoods of the city of Buenos Aires
### full version

| Name               | Index | Name               | Index |
|--------------------|-------|--------------------|-------|
| Agronomía          | 26    | Almagro            | –     |
| Balvanera          | 12    | Barracas           | 6     |
| Belgrano           | 14    | Boedo              | 21    |
| Caballito          | 27    | Chacarita          | 17    |
| Coghlan            | 3     | Colegiales         | 15    |
| Constitución       | 5     | Flores             | 31    |
| Floresta           | 33    | La Boca            | 2     |
| La Paternal        | –     | Liniers            | 37    |
| Mataderos          | 38    | Monte Castro       | –     |
| Montserrat         | 9     | Nueva Pompeya      | 24    |
| Núñez              | –     | Palermo            | 16    |
| Parque Avellaneda  | –     | Parque Chacabuco   | 22    |
| Parque Chas        | 20    | Parque Patricios   | 7     |
| Puerto Madero      | 10    | Recoleta           | 13    |
| Retiro             | 11    | Saavedra           | 1     |
| San Cristóbal      | 8     | San Nicolás        | –     |
| San Telmo          | 4     | Vélez Sársfield    | 35    |
| Versalles          | 36    | Villa Crespo       | 18    |
| Villa del Parque   | 29    | Villa Devoto       | 32    |
| Villa General Mitre| 28    | Villa Lugano       | –     |
| Villa Luro         | –     | Villa Ortúzar      | 19    |
| Villa Pueyrredón   | 25    | Villa Real         | 34    |
| Villa Riachuelo    | 39    | Villa Santa Rita   | 30    |
| Villa Soldati      | –     | Villa Urquiza      | 23    |

### small version

| Name               | Index |
|--------------------|-------|
| (Agronomía, La Paternal, Parque Chas, Villa General Mitre, Villa Ortúzar, Villa del Parque, Villa Santa Rita) | 13 |
| (Almagro, Boedo)   | 7     |
| (Balvanera, Constitución, Montserrat, San Cristóbal, San Nicolás, San Telmo) | 4 |
| (Barracas, La Boca, Parque Patricios) | 5 |
| Belgrano           | –     |
| Caballito          | 9     |
| (Chacarita, Colegiales, Villa Crespo) | 8 |
| (Coghlan, Núñez, Saavedra) | 11 |
| Flores             | 14    |
| (Floresta, Liniers, Monte Castro, Vélez Sársfield, Versalles, Villa Luro, Villa Real) | – |
| Mataderos          | 16    |
| (Nueva Pompeya, Parque Chacabuco) | 10 |
| Palermo            | 6     |
| Parque Avellaneda  | –     |
| Puerto Madero      | 3     |
| Recoleta           | 1     |
| Retiro             | 2     |
| Villa Devoto       | –     |
| (Villa Lugano, Villa Riachuelo) | – |
| Villa Soldati      | 15    |
| (Villa Urquiza, Villa Pueyrredón) | 12 |

Table 7: Best sequences for both versions (optimal for “small”)