

Integer Programming algorithms for the Domatic Partitioning Problem

Felipe Crispim Fragoso

felipe.cfragoso@gmail.com

RESUMO

Este trabalho propõe uma nova formulação de programação inteira baseada no conceito de representantes e um algoritmo Branch-and-Price para o Problema da Partição Domática (PPD), que é uma variante do clássico problema do conjunto dominante em grafos. Na literatura, o PPD tem sido utilizado como modelo para estender o tempo de vida de redes de sensores sem fio. Uma comparação com a formulação apresentada na literatura mostra que a formulação por representantes supera a formulação da literatura para grafos aleatórios de alta densidade. Além disso, é mostrado que o algoritmo Branch-and-Price tem um melhor comportamento sobre todas as formulações. Esta é a primeira proposta de Branch-and-Price para o PPD. Uma nova forma de aplicar cortes de local branching robustos para uma matheurística SUBMIPping também é proposta neste trabalho.

PALAVRAS CHAVE. domatic partition, domatic number, mathematical formulation, branch-and-cut, branch-and-price.

ABSTRACT

This work proposes a new integer programming formulation based on the representative's concept and a Branch-and-Price algorithm for the Domatic Partitioning Problem (DPP), which is a variant of the classical dominating set problem in graphs. In the literature, the DPP has been used as a model to extend the lifetime of wireless sensor networks. A comparison with the formulation presented in the literature shows that the representative formulation outperforms the literature formulation for high-density random graphs. Furthermore, it is shown that the Branch-and-Price algorithm has a better behavior over all formulations. This is the first proposal of Branch-and-Price for the DPP. A new SUBMIPping matheuristic with robust local branching cuts is also proposed here.

KEYWORDS. domatic partition, domatic number, mathematical formulation, branch-and-cut, branch-and-price.

1. Introduction

Consider a simple undirected graph $G = (V(G), E(G))$ defined by its set of vertices $V(G)$ and its set of edges $E(G)$. Let $N(i) = \{j \in V(G) : \{i, j\} \in E(G)\}$ be the vertices adjacent to i or its *open neighborhood*, and let $N[i] = N(i) \cup \{i\}$ be the *closed neighborhood* of i . In graph domination problems, a vertex $i \in V(G)$ is said to dominate all vertices of its closed neighborhood. Extending this concept to sets of vertices, we say that a set $D \subseteq V(G)$ dominates all vertices of $N[D] = \cup_{i \in D} N[i]$. In particular, when $N[D] = V(G)$ is satisfied, D is called a *dominating set* of G .

The *Graph Domination Problem* (GDP) consists of finding a dominating set with the lowest cardinality possible. The problem was formally defined in 1962 [Berge, 1962; Ore, 1962]. However, its roots go back to challenges proposed to chess players in the 1850s [Haynes, 1998]. Since then, GDP has attracted great theoretical and practical interest, which is reflected in the numerous existing publications associated with it and its various variants.

As an example of a variant of the GDP we have the *Connected Domination Problem* (CDP) [Sampathkumar e Walikar, 1973], where it is imposed a connected dominating set. In more detail, a dominating set $D \subseteq V(G)$ is feasible for the CDP when D induces a connected subgraph of G , i.e. $G[D] = (D, E(D))$, $E(D) = \{(i, j) \in E(G) : \{i, j\} \subseteq D\}$, is connected. Another variant is the *Total Domination Problem* (TDP) [Cockayne et al., 1980], where $G[D]$ must be formed by one or more connected components, always of dimension greater than or equal to 2. Concluding this brief description of variants of the GDP, we have the *Perfect Domination Problem* (PDP) [E.J. Cockayne e Laskar, 1993; Weichsel, 1994], which imposes the domination of each vertex of $V(G) \setminus D$ by a single vertex of D . Note that these three problems are characterized by a particularization of the dominance concept.

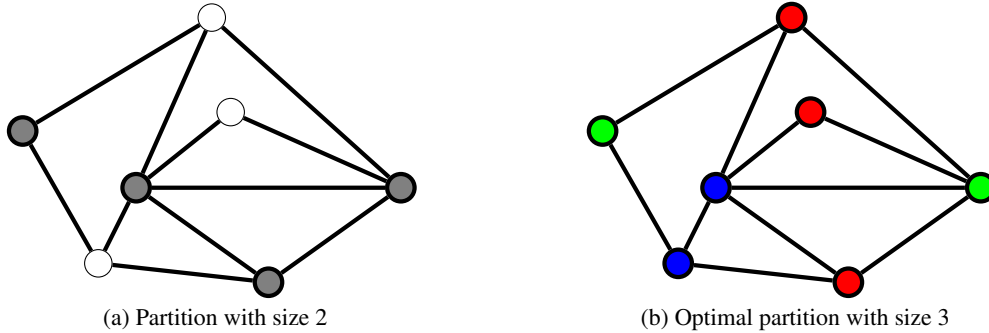
Other kinds of variants of the GDP are also defined with the imposition of a specific graph topology on the admissible dominating set, such as *cliques* [Cozzens e Kelleher, 1990] or *cycles* [Proskurowski e Syslo, 1981]. In the first case, a dominating set $D \subseteq V(G)$ must induce a complete subgraph $G[D]$. In the second, $G[D]$ must correspond to a cycle of G . For more examples of topologies studied in this context, the reader must see the references [Veldman, 1983; Adasme et al., 2018, 2019].

Cockayne e Hedetniemi [1977] proposed the notation $\gamma(G)$ to designate the *dominating number* of G , that is, the smallest possible cardinality for a dominating set of that graph. Additionally, Cockayne and Hedetniemi also defined the concept of *domatic number* of G , $d(G)$, as the maximum order for a partition of vertices into dominating sets (see [Cockayne e Hedetniemi, 1975, 1977; Cockayne, 1978]). In this context, the Domatic Partitioning Problem (DPP) consists of finding a partition of $V(G)$ formed by exactly $d(G)$ dominating sets.

Figure 1 describes different domatic partitions for the same graph, G . In particular, Figure 1a presents a partition involving two dominating sets, with different colors applied to each of them. This partition is not optimal, unlike the one indicated in 1b, which involves three dominating sets. DPP is NP-complete for several particular classes of graphs [Chang, 1994; Sprague, 1999; Nguyen e Huynh, 2007] and also for general graphs where $V(G)$ must be partitioned into exactly three dominating sets [Cardei et al., 2002].

In the literature, the DPP has been used as a model to extend the lifetime of wireless sensor networks. In this context, an attempt is made to partition the set of points to be observed in terms of dominating sets of the network(graph) that these points define. Each dominating set of the partition corresponds to an observation system capable of collecting information generated by all points, including those of the set itself. In practical terms, sensors powered by batteries

Figure 1: Examples of Domatic Partitions



are installed at each of the points to be observed. These points, in turn, will play a dual role in the scheme. When they are not part of the dominating set in current operation, they will act as observed points (by other points of the operational dominating set); otherwise, they will act as observation points (individually, also observing themselves). Faced with a limited battery time for the operation of each sensor in the network, different dominating sets are then activated sequentially. More specifically, when the battery charge of a sensor drops to zero, its dominating set is turned off, as a whole, and the network starts to operate with the next dominating set. Consequently, we try to partition the vertices of the graph $G = (V(G), E(G))$, in the largest possible number of dominating sets. This approach has been investigated in several works in the literature [Nguyen e Huynh, 2007; Moscibroda e Wattenhofer, 2005; Ahn e Park, 2011; Balbal et al., 2021; Pino et al., 2018]. In addition to the previous references, the reader should consult the *survey* [Yetgin et al., 2017], for more information on wireless sensor networks, on the proposed models to extend the time of information collection and alternative ways to formulate and solve these models.

DPP and its variants are being explored by the literature since the 1970s. For example, Zelinka [1983] proposed the concept of distance k -domatic numbers as a generalization on the concept of domatic numbers presented in [Cockayne e Hedetniemi, 1977]. 20 years later, using the same context of application of the domatic number, the literature used the distance k -domatic number to maximize the lifetime of sensor networks [Pemmaraju e Pirwani, 2006]. Therefore, algorithmic contributions on this area are justified from a theoretical and also practical point of view.

Most of the works proposed for the domatic number and its variants are from the graph or computer theory domains. This research seeks to contribute with the knowledge on the computational aspects of the problem. More specifically, by analyzing the algorithmic behavior of integer programming based algorithms on connected random graph instances.

2. Literature Formulations for DPP

Two independent works propose a similar formulation [Balbal et al., 2021; Landete e Sainz-Pardo, 2022]. Both works used a undirected graph $G = (V(G), E(G))$ and a *Upper bound* $d(G) \leq \delta(G) + 1$, where $\delta(G)$ corresponds to the smallest degree of a vertex of G . The reader can consult [Cockayne e Hedetniemi, 1977] for a formal demonstration of this upper bound and chapter 13 of [Haynes et al., 1998] for a *survey* with several theoretical results for the DPP. It is noteworthy that the discovery of new upper bounds is a challenging problem for DPP by itself and they also can make the formulation presented here stronger.

In more detail, the formulation of the DPP presented in [Balbal et al., 2021] associate

weights to the vertices (sensors) of a graph G (sensor network) that correspond to the lifetime of each sensor, that is, the authors sought to solve the problem of extending the lifetime of wireless networks through the DPP. In the formulation, the cost of a dominating set would be equal to the weight of the vertex with the shortest lifetime that is part of that set. The second proposal presented in [Landete e Sainz-Pardo, 2022] is a direct formulation for the DPP. It is important to mention that, although apparently different, both proposals for DPP formulations become equivalent when we use unit costs.

The models proposed in [Balbal et al., 2021] and [Landete e Sainz-Pardo, 2022] present, when considering only unitary costs, a formulation through two families of variables. The first family, $x_{d,i}$, indicates whether a vertex $i \in V(G)$ of the graph has been assigned to the dominating set d . The second family, X_d , is defined to say whether a dominating set d is selected to be part of the optimal solution of the problem. As it is not possible to know how many variables X_d will have value 1 in an optimal solution, the authors used the upper bound of $\delta(G) + 1$ to limit the amount of variables X_d present in the model. For a better notation, consider the set $\Delta = \{1, \dots, \delta(G) + 1\}$. Together, both works defined MIP_1 :

$$(MIP_1) \left\{ \begin{array}{ll} \max \sum_{d \in \Delta} X_d & (1) \\ \text{s.t.} \sum_{d \in \Delta} x_{d,i} \leq 1, i \in V(G) & (2) \\ X_d \leq \sum_{j \in N[i]} x_{d,j}, d \in \Delta, i \in V(G) & (3) \\ x_{d,i} \leq X_d, d \in \Delta, i \in V(G) & (4) \\ X_{d+1} \leq X_d, d \in \Delta & (5) \\ X_d, x_{d,i} \in \{0, 1\}, d \in \Delta, i \in V(G) & (6) \end{array} \right.$$

The objective function (1) maximizes the number of selected dominating sets. Constraints (2) prohibit the same vertex from occurring in more than one dominating set, since a requirement of the problem is the disjointness of the vertices. Constraints (3) are the dominance constraints for the sets, they require that the set X_d be a dominating set if $X_d = 1$. Constraints (4) prohibit a vertex from being part of a dominating set that is not selected to be part of the solution ($X_d = 0$). Constraints (5) are symmetry breaking constraints presented only in [Balbal et al., 2021]. They require that if $k < \delta(G) + 1$ dominating sets are selected, they are assigned to the variables X_d where $d \in \{1, \dots, k\}$.

Observe that although the DPP is formally defined as a vertex partitioning problem, constraints (2) do not require a partitioning. However, we can generate a partition from the solution constructed from MIP_1 when we extend some dominating set $X_d = 1$ with all the vertices left out of a dominating set.

2.1. Formulation with Exponential Variables

Two other works present a formulation of the problem with exponentially many variables [Ahn e Park, 2011; Rosati et al., 2023]. In the first work, the linear programming relaxation of the formulation was calculated, but there was no branching in the algorithm. Therefore, the authors did not seek to find the optimal *integer* solution. In the last work, the authors incorporated the exponential formulation into a mathheuristic to solve the integral problem only on a subset of variables. Therefore, the authors are not granting the optimal solution given enough time. As the objective

of this work is to propose exact solutions for the DPP, a Branch-and-Price algorithm based on this formulation will be proposed here (see section 4).

Let $\mathcal{D} = \{D \subseteq V(G) : N[D] = V(G)\}$ be the set of all *minimal* dominating sets of G and $\mathcal{D}_i = \{D \in \mathcal{D} : i \in D\}$ for $i \in V(G)$ to be all *minimal* dominating sets of $\mathcal{D}$ where i is part of it. By assigning binary variables λ_D for all $D \in \mathcal{D}$, we get MIP_3 :

$$(MIP_3) \begin{cases} \max \sum_{D \in \mathcal{D}} \lambda_D & (7) \\ \text{s.t.} \sum_{D \in \mathcal{D}_i} \lambda_D \leq 1, i \in V(G) & (8) \\ \lambda_D \in \{0, 1\}, D \in \mathcal{D} & (9) \end{cases}$$

The objective function (7) maximizes the number of minimal dominating sets in the solution. Constraints (8) guarantee that each vertex will be assigned to at most 1 minimal dominating set. In the same way as MIP_1 , this formulation does not grant a partition but we can build it with a post-processing.

As the number of minimal dominating sets of a graph is limited by $1.71^{|V(G)|}$ [Fomin et al., 2008], this formulation is, in fact, a formulation with exponentially many variables. In order to use a column generation procedure, the authors in [Ahn e Park, 2011] built an *Restricted Master Problem* with a subset of $\mathcal{D}$ generated by the heuristic described in [Cardei et al., 2005]. From then on, a *Pricing* subproblem generating dominating sets (not necessarily minimal) with positive reduced cost was used to iteratively generate additional variables that lead to the optimal linear programming relaxation of MIP_3 . For more references on the column generation procedure the reader can refer to [Desaulniers et al., 2005].

The Pricing subproblem presented in [Ahn e Park, 2011] uses an integer programming formulation for the Minimum Cost Dominating Set Problem (MCDSP), where the costs of the vertices in the objective function are given by the optimal linear programming dual values $\bar{\pi}$ associated with the constraints (8). The MCDSP is formulated as:

$$(MCDSP) \begin{cases} 1 - \min \sum_{i \in V(G)} \bar{\pi}_i w_i & (10) \\ \text{s.t.} \sum_{j \in N[i]} w_j \geq 1, i \in V(G) & (11) \\ w_i \in \{0, 1\}, i \in V(G) & (12) \end{cases}$$

The variables w_i indicate whether a vertex is in the solution ($w_i = 1$) or not ($w_i = 0$). The objective function (10) seeks to find the dominating set with maximum reduced cost, while Constraints (11) ensure that the solution is a dominating set.

3. New Formulation for DPP

In order to build a new symmetry-free formulation, this work proposes a *formulation by representatives*. The idea of representatives was initially presented for the Vertex Coloring Problem [Campêlo et al., 2004]. However, it is a powerful and simple idea that can be used to model problems like bin packing [Hadj Salem e Kieffer, 2020] or the DPP itself.

Consider the set $\{z_{ij} : i, j \in V(G), i \leq j\}$ of variables associated with the vertices of a graph. In a domatic partition of a graph $G = (V(G), E(G))$, z_{ii} takes value 1 if vertex $i \in V(G)$

represents a dominating set in the partition of G and $z_{ij} = 1$ when vertex $j \in V(G)$ is being represented by $i \in V(G), i < j$. With the variables z , we can build the formulation MIP_2 :

$$(MIP_2) \left\{ \begin{array}{ll} \max \sum_{i \in V(G)} z_{ii} & (13) \\ \text{s.t. } z_{ii} + \sum_{i < j, j \in V(G)} z_{i,j} \leq 1, i \in V(G) & (14) \\ z_{ii} \leq \sum_{i < k \in N[j]} z_{i,k}, i \in V(G), j \in V(G) \setminus \{i\} & (15) \\ z_{i,j} \leq z_{ii}, i, j \in V(G), i < j & (16) \\ z_{ii}, z_{i,j} \in \{0, 1\}, i \in V(G), j \in V(G) & (17) \end{array} \right.$$

Constraints (14) guarantee a partition of $V(G)$, as each vertex must appear only once as representative or represented. Whenever $i \in V(G)$ is a representative, i.e. $z_{ii} = 1$, it is necessary that the set that i represents is dominating. The dominance constraints (15) guarantee the dominance of the set represented by i , by forcing all other vertices $j \in V(G) \setminus \{i\}$ to be in the represented set of i (i.e. $x_{ij} = 1$) or have some neighboring vertex k in the the set represented by i (Note that i is always the vertex with the smallest identifier). Constraints (16) ensure that only representatives can represent other vertices. The objective function of (13) guarantees the maximum number of dominating sets of the graph.

4. New Branch-and-Price Algorithm for DPP

Consider the set $\mathcal{D}' \subseteq \mathcal{D}$ containing a single dominating set with all vertices of the graph and let $\mathcal{D}'_i = \{D \in \mathcal{D}' : i \in D\}$ for $i \in V(G)$. We can then define the following RMP:

$$(RMP(\mathcal{D}')) \left\{ \begin{array}{ll} \max \sum_{D \in \mathcal{D}'} \lambda_D & (18) \\ \text{s.t. } \sum_{D \in \mathcal{D}'_i} \lambda_D \leq 1, i \in V(G) & (19) \\ \lambda_D \in \{0, 1\}, D \in \mathcal{D}' & (20) \end{array} \right.$$

Now we can solve $MCDSP$ iteratively over the optimal linear programming dual values $\bar{\pi}$ associated with constraints (19). Once the Pricing iterations finish and we have the optimal linear programming relaxation for $RMP(\mathcal{D}')$, we can apply Ryan and Foster's branching rule to $RMP(\mathcal{D}')$ [Ryan e Foster, 1981].

To find a pair of vertices for the Ryan and Foster's branching, this work proposes an algorithm that iterates over each pair of lines $\{r, f\}$ of the optimal linear programming solution of $RMP(\mathcal{D}')$ and finds the pair that has the sum of columns $\sum_{D \in \mathcal{D}_r \cap \mathcal{D}_f} \bar{\lambda}_D$ with the most fractional value possible, that is, with a value closest to $1/2$.

The first (left branch) extends $MCDSP$ by adding a constraint forcing vertices $\{r, f\}$ to occur simultaneously in the solution. The second (right branching) extends $MCDSP$ with a constraint doing the opposite of the left branching, that is, forcing that $\{r, f\}$ do not occur simultaneously. The formulation (a) (resp. (b)) below shows, in red, the modification of $MCDSP$ after one iteration of the left branching (resp. right branching).

In addition to the Pricing modifications highlighted above, modifications to $RMP(\mathcal{D}')$ are made in order to remove infeasible columns for the applied branching constraints. For the left

$$1 - \min \sum_{i \in V(G)} \bar{\pi}_i w_i$$

$$w_r + w_f \leq 1$$

$$\sum_{j \in N[i]} w_j \geq 1, i \in V(G)$$

$$w_i \in \{0, 1\}, i \in V(G)$$

(a) Left Branching

$$1 - \min \sum_{i \in V(G)} \bar{\pi}_i w_i$$

$$w_r = w_f$$

$$\sum_{j \in N[i]} w_j \geq 1, i \in V(G)$$

$$w_i \in \{0, 1\}, i \in V(G)$$

(b) Right Branching

branch, the algorithm remove all columns $D \in \mathcal{D}'$ where $\{r, f\} \not\subseteq D$. As for the right branch, the algorithm removes all columns $D \in \mathcal{D}'$ where $\{r, f\} \subseteq D$. Finally, for the branching search strategy, Depth First Search is used. At first, the left branch is performed until the solution is integer or the Pricing becomes infeasible, only then the right branching occurs.

4.1. Mathuristic for Integer Solutions

Primal heuristics are essential components to improve the enumeration time of a Branch-and-Price tree. One method to generate primal solutions through the Branch-and-Price tree is the Sub-MIPping method, which consists in optimally solve $RMP(\mathcal{D}')$ up to integrality and then compare the integer solution obtained with the incumbent integer solution available, if there is an improvement, the new solution is setup as an incumbent solution.

Instead of naively applying the Sub-MIPping method as described above, this work proposes to interactively price columns, solve $RMP(\mathcal{D}')$ up to integrality and add soft fixing constraints [Fischetti e Lodi, 2003] to guide the next pricing iteration. Algorithm 1 presents the main idea to separate soft fixing constraints.

Algorithm 1 Robust Sub-MIPping for DPP

```

1:  $\alpha := 0.5$ 
2:  $\mathcal{D}' := \mathcal{D}' \cup \text{price}(\mathcal{D}')$ 
3:  $\lambda^* := \text{solve}(RMP(\mathcal{D}'))$ 
4: while TRUE do
5:    $S = \{D : D \in \mathcal{D}', \lambda_D^* = 1\}$ 
6:    $\mathcal{D}' := \mathcal{D}' \cup \text{price}(RMP(\mathcal{D}') \cup \{\sum_{D \in S} \lambda_D \geq |S| * \alpha\})$ 
7:    $\lambda' := \text{solve}(RMP(\mathcal{D}'))$ 
8:   if  $v(\lambda') > v(\lambda^*)$  then
9:      $\lambda^* = \lambda'$ 
10:     $\alpha = 0.5$ 
11:   else
12:      $\alpha = \alpha + 0.2$ 
13:     if  $\alpha > 1$  then
14:       return  $\lambda^*$ 
15:     end if
16:   end if
17: end while

```

Line 1 initializes a α value that will be used in the soft fixing constraints. Line 2 represents a full pricing iteration that returns all new columns necessary to reach the linear programming optimality. Line 3 set an incumbent solution using integer SUB-MIPping. The mathheuristic to generate better integer solutions starts at Line 4. Lines 5-7 separate a new soft fixing cut based on the incumbent integer solution λ^* , price new columns that will be generated with the addition of the soft fixing cut

to $RMP(\mathcal{D}')$ and gets the SUB-MIPping solution λ' . Lines 8-10 update the incumbent solution if a better solution is found and also restarts the α value so the algorithm can iterate more times on the new incumbent solution. Case a better solution is not found, the α value gets increased on line 12. If the α value increases to a value higher than 1, the algorithm stops returning the best incumbent integer solution found (lines 13-14).

The benefit of soft fixing inequalities is that through them an extension of the RMP is made by incorporating new columns that would not be generated without the presence of such cuts. Furthermore, soft fixing inequalities are simple and can be integrated as robust cuts, i.e., without changing the pricing structure.

5. Computational Experiments

All algorithms were implemented in C++ with the *concert* library of CPLEX 20.1. The experiments were performed on a computer with an Intel i7 processor at 3.4 GHz and 16 GB of RAM. The operating system used was the Windows Subsystem for Linux.

The integer problems MIP_1 and MIP_2 were solved directly with the CPLEX solver. For MIP_1 , a higher branching priority was imposed on the variables $X_d, d \in \Delta$. For MIP_2 , branching priority was given to the representative variables $z_{ii}, i \in V(G)$.

The Branch-and-Price algorithm for MIP_3 consists of using CPLEX to solve the linear relaxation of the restricted master problem, apply the mathheuristic described in section 4.1 and solve the pricing problem over all nodes of the tree. All the other aspects of the algorithm were developed with customized data structures.

5.1. Generated Instances

To generate random graph instances, this work used PreZER [Nobari et al., 2011] algorithm, which is an implementation of the Gilbert's random graph generation model [Gilbert, 1959]. The algorithm demands two input parameters: $n \in \mathbb{Z}$ for the number of vertices of the graph $G = (V(G), E(G))$ that will be generated and $p \in [0, 1]$ for the probability of inserting an edge between any pair of vertices of G . Fifty-four instances were generated. The parameters used to generate the instances were $n \in \{50, 100\}$ and $p \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$, where three instances were generated for each possible combination of p and n . To ensure that the generated graph was connected, the PreZER algorithm was re-executed whenever the generated graph was not connected.

5.2. Comparison of results obtained for exact formulations

Tables 1 and 2 present the results obtained with the computational experiments of the formulations presented for the DPP in the instances generated for each value of n .

Both tables have 13 columns. The first 4 columns present information for each instance: instance identifier, density of the graph, the optimal value of the instance (or the best known interval that contains the optimal value), and the upper bound of $\delta(G) + 1$. Then, for each algorithm, 3 columns are presented. They correspond, respectively, to the total running time of the algorithm, the final GAP of the algorithm and the number of nodes in the tree. The TLE value was placed in the column corresponding to the total running time of the algorithm for cases in which the algorithm was not able to complete the search process for the optimal integer solution in less than 1 hour.

Observing Table 1 it is evident an advantage of MIP_2 over MIP_1 for instances with density greater than 50%, since these instances were all resolved in the root node of the tree. Furthermore, for instances with density between 30% and 50%, it is also possible to observe a smaller final GAP for MIP_2 . Finally, it can be concluded that the Branch-and-Price algorithm for MIP_3 is the best option for resolution in general. The Branch-and-Price algorithm was able to solve all the

Table 1: Comparison of Formulations n = 50

Inst.	dens.	OPT	$\delta(G) + 1$	MIP_1			MIP_2			Branch-and-Price		
				t(s)	GAP	#Nodes	t(s)	GAP	#Nodes	t(s)	GAP	#Nodes
1	10%	2	2	0.01	0%	1	0.05	0%	1	0.08	0%	17
2	10%	2	2	0.01	0%	1	0.03	0%	1	0.08	0%	5
3	10%	2	2	0.01	0%	1	0.03	0%	1	0.11	0%	18
4	20%	2	2	0.05	0%	1	0.06	0%	1	0.05	0%	12
5	20%	6	6	0.02	0%	1	0.59	0%	1	1.58	0%	24
6	20%	5	5	0.09	0%	1	0.15	0%	1	1.33	0%	29
7	30%	7	7	0.03	0%	1	0.17	0%	1	0.55	0%	29
8	30%	8	9	TLE	11%	5237168	TLE	11%	2981013	46.34	0%	66
9	30%	8	9	TLE	11%	3755707	TLE	11%	1945801	19.99	0%	14
10	40%	12	14	TLE	21%	1712972	TLE	12%	907224	24.47	0%	30
11	40%	12	15	TLE	26%	1190754	TLE	12%	736729	7.27	0%	12
12	40%	11	16	TLE	37%	771098	TLE	21%	1013943	19.33	0%	19
13	50%	14	19	TLE	31%	741942	2813.49	0%	470912	7.97	0%	12
14	50%	15	20	TLE	25%	440068	45.68	0%	6393	8.28	0%	11
15	50%	14	17	TLE	23%	1342506	262.32	0%	71297	11.07	0%	33
16	60%	17	20	TLE	15%	1302907	2.78	0%	10108	1.06	0%	12
17	60%	16	24	TLE	40%	339412	1.86	0%	1	1.61	0%	31
18	60%	16	24	TLE	40%	290700	1.75	0%	1	2.45	0%	29
19	70%	19	29	TLE	24%	568831	1.05	0%	1	1.07	0%	28
20	70%	18	29	TLE	28%	696631	1.26	0%	1	0.97	0%	23
21	70%	18	27	TLE	28%	509538	1.14	0%	1	1.06	0%	22
22	80%	24	34	479.58	0%	937335	0.15	0%	1	0.42	0%	18
23	80%	24	33	TLE	4%	4319009	0.09	0%	1	0.47	0%	9
24	80%	24	35	TLE	4%	4084303	0.19	0%	1	0.56	0%	1
25	90%	25	40	7.48	0%	1	0.04	0%	1	0.15	0%	18
26	90%	25	39	10.06	0%	1	0.04	0%	1	0.16	0%	10
27	90%	25	41	8.21	0%	1	0.05	0%	1	0.18	0%	18

Table 2: Comparison of Formulations n = 100

Inst.	d	OPT	$\delta + 1$	MIP_1			MIP_2			Branch-and-Price		
				t(s)	GAP	#Nodes	t(s)	GAP	#Nodes	t(s)	GAP	#Nodes
28	10%	4	4	0.03	0%	1	3.51	0%	1	2.64	0%	77
29	10%	6	6	1.68	0%	312	84.84	0%	2582	TLE	16%	1867
30	10%	3	3	0.01	0%	1	0.57	0%	1	0.83	0%	62
31	20%	[9,12]	14	TLE	35%	257272	TLE	35%	28735	TLE	35%	5
33	20%	[9,11]	13	TLE	30%	311856	TLE	30%	23004	TLE	25%	4
34	30%	[15,16]	20	TLE	35%	174078	TLE	35%	28412	TLE	9%	55
35	30%	[12,15]	18	TLE	33%	150509	TLE	38%	25579	TLE	20%	6
36	30%	[13,16]	22	TLE	40%	45069	TLE	39%	9253	TLE	9%	22
37	40%	[19,20]	27	TLE	37%	27847	TLE	36%	6989	TLE	5%	489
38	40%	21	32	TLE	50%	5600	TLE	33%	18503	908.43	0%	24
39	40%	20	28	TLE	42%	28217	TLE	36%	9262	722.23	0%	35
40	50%	25	38	TLE	44%	9062	TLE	26%	10266	469.67	0%	174
41	50%	25	39	TLE	48%	8912	TLE	33%	15954	229.09	0%	49
42	50%	25	41	TLE	48%	8701	TLE	28%	8459	1430.56	0%	874
43	60%	31	45	TLE	42%	5253	TLE	18%	80735	1250.52	0%	256
44	60%	31	51	TLE	50%	2627	TLE	18%	74240	392.25	0%	24
45	60%	31	50	TLE	48%	8343	TLE	18%	81491	355.65	0%	20
46	70%	33	59	TLE	36%	5554	184.51	0%	9074	19.87	0%	53
47	70%	33	60	TLE	34%	5269	39.41	0%	1	27.15	0%	55
48	70%	33	57	TLE	38%	8481	631.32	0%	43044	18.91	0%	52
49	80%	42	71	TLE	16%	88775	8.60	0%	1	7.43	0%	26
50	80%	44	71	TLE	22%	54270	8.97	0%	1	10.14	0%	33
51	80%	43	68	TLE	28%	29024	6.63	0%	1	8.67	0%	23
52	90%	50	80	106.82	0%	1	0.16	0%	1	0.61	0%	19
53	90%	50	82	165.30	0%	1	0.32	0%	1	0.78	0%	8
54	90%	50	79	202.25	0%	1	0.29	0%	1	1.02	0%	25

instances up to integral optimality, although it had a worse execution time than the other formulations for the easiest instances.

Table 2 shows that instances of size $n = 100$ become difficult to resolve. In a comparison between MIP_1 and MIP_2 , it is noticed a better performance of the MIP_1 in the instances of lower density. However, as the density increases, MIP_2 becomes more advantageous. It is also possible to observe a considerable increase for the solution time of the Branch-and-Price. Although it was the algorithm with the highest resolution of instances, it is showing a high difficulty to resolve instances with density between 20% and 30%.

6. Conclusions

In this work, a new formulation based on the representatives concept and a Branch-and-Price algorithm for the Domatic Partitioning Problem is presented. The DPP is a variant of the classical dominating set problem in graphs. We show that the representatives formulation outperforms the classical formulation of the Literature for high density graphs. Furthermore, it is shown that a Branch-and-Price algorithm has a better time execution generally. Additionally, this is the first proposal of Branch-and-Price for the DPP.

As future works, it is interesting to improve the Branch-and-Price algorithm proposed here with pricing heuristics to speed up its enumeration process. It is also important to explore the SUBMIPping matheuristic proposed here in a number of ways. For example by keeping the soft fixing constraints generated in all iterations, so the pricing iterations will consider multiple soft fixing cuts. We can also try a variant of the cut for the 0-valued variables.

References

- Adasme, P., Andrade, R., Leung, J., e Lissner, A. (2018). Improved solution strategies for dominating trees. *Expert Systems with Applications*, 100:30–40. ISSN 0957-4174.
- Adasme, P., Juan, E. S., Soto, I., e Firoozabadi, A. D. (2019). Integer programming models for wireless networks with star backbone topology. In *2019 IEEE CHILEAN Conference on Electrical, Electronics Engineering, Information and Communication Technologies (CHILECON)*, p. 1–6.
- Ahn, N. e Park, S. (2011). An Optimization Algorithm for the Maximum Lifetime Coverage Problems in Wireless Sensor Network. *Management Science and Financial Engineering*, 17(2):39–62.
- Balbal, S., Bouamama, S., e Blum, C. (2021). A greedy heuristic for maximizing the lifetime of wireless sensor networks based on disjoint weighted dominating sets. *Algorithms*, 14(6).
- Berge, C. (1962). *The Theory of Graphs and Its Applications*. Methuen, London.
- Campêlo, M., Corrêa, R., e Frota, Y. (2004). Cliques, holes and the vertex coloring polytope. *Information Processing Letters*, 89(4):159–164.
- Cardei, M., Thai, M., Li, Y., e Wu, W. (2005). Energy-efficient target coverage in wireless sensor networks. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, p. 1976–1984 vol. 3.
- Cardei, M., MacCallum, D., Cheng, M. X., Min, M., Jia, X., Li, D., e Du, D.-Z. (2002). Wireless sensor networks with energy efficient organization. *Journal of Interconnection Networks*, 03 (03n04):213–229.

- Chang, G. J. (1994). The domatic number problem. *Discrete Mathematics*, 125(1):115–122.
- Cockayne, E. e Hedetniemi, S. (1975). Optimal domination in graphs. *IEEE Transactions on Circuits and Systems*, 22(11):855–857.
- Cockayne, E. J., Dawes, R. M., e Hedetniemi, S. T. (1980). Total domination in graphs. *Networks*, 10(3):211–219.
- Cockayne, E. J. (1978). Domination of undirected graphs — a survey. In Alavi, Y. e Lick, D. R., editors, *Theory and Applications of Graphs*, p. 141–147, Berlin, Heidelberg. Springer Berlin Heidelberg.
- Cockayne, E. J. e Hedetniemi, S. T. (1977). Towards a theory of domination in graphs. *Networks*, 7(3):247–261.
- Cozzens, M. B. e Kelleher, L. L. (1990). Dominating cliques in graphs. *Discrete Mathematics*, 86(1):101–116.
- Desaulniers, G., Desrosiers, J., e Solomon, M. M., editors (2005). *Column Generation*. Springer, New York, NY, 2005 edition.
- E.J. Cockayne, S. H., B.L. Hartnell e Laskar, R. (1993). Perfect domination in graphs. *J. Combin. Inform. System Sci.*, 18:136–148.
- Fischetti, M. e Lodi, A. (2003). Local branching. *Mathematical Programming*, 98(1-3):23–47.
- Fomin, F. V., Grandoni, F., Pyatkin, A. V., e Stepanov, A. A. (2008). Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. *ACM Trans. Algorithms*, 5(1).
- Gilbert, E. N. (1959). Random Graphs. *The Annals of Mathematical Statistics*, 30(4):1141 – 1144.
- Hadj Salem, K. e Kieffer, Y. (2020). New symmetry-less ilp formulation for the classical one dimensional bin-packing problem. In Kim, D., Uma, R. N., Cai, Z., e Lee, D. H., editors, *Computing and Combinatorics*, p. 423–434, Cham. Springer International Publishing. ISBN 978-3-030-58150-3.
- Haynes, T. (1998). *Fundamentals of domination in graphs*. Marcel Dekker, New York.
- Haynes, T. W., Hedetniemi, S. T., e Slater, P. J. (1998). *Domination in Graphs: Advanced Topics*. Chapman & Hall/CRC Pure and Applied Mathematics. CRC Press, Boca Raton, FL.
- Landete, M. e Sainz-Pardo, J. L. (2022). The domatic partition problem in separable graphs. *Mathematics*, 10(4):640.
- Moscibroda, T. e Wattenhofer, R. (2005). Maximizing the lifetime of dominating sets. In *19th IEEE International Parallel and Distributed Processing Symposium*, p. 8.
- Nguyen, T. N. e Huynh, D. T. (2007). Extending sensor networks lifetime through energy efficient organization. In *International Conference on Wireless Algorithms, Systems and Applications (WASA 2007)*, p. 205–212.

- Nobari, S., Lu, X., Karras, P., e Bressan, S. (2011). Fast random graph generation. In *EDBT 2011, 14th International Conference on Extending Database Technology, Uppsala, Sweden, March 21-24, 2011, Proceedings*, p. 331–342.
- Ore, Ø. (1962). *Theory of graphs*. American Mathematical Society, Providence.
- Pemmaraju, S. V. e Pirwani, I. A. (2006). Energy conservation via domatic partitions. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 06*, p. 143–154, New York, NY, USA. Association for Computing Machinery.
- Pino, T., Choudhury, S., e Al-Turjman, F. (2018). Dominating set algorithms for wireless sensor networks survivability. *IEEE Access*, 6:17527–17532.
- Proskurowski, A. e Syslo, M. (1981). Minimum dominating cycles in outerplanar graphs. *International Journal of Computer & Information Sciences*, 10(2):127–139.
- Rosati, R. M., Bouamama, S., e Blum, C. (2023). Construct, merge, solve and adapt applied to the maximum disjoint dominating sets problem. In Di Gaspero, L., Festa, P., Nakib, A., e Pavone, M., editors, *Metaheuristics*, p. 306–321, Cham. Springer International Publishing. ISBN 978-3-031-26504-4.
- Ryan, D. e Foster, B. (1981). An integer programming approach to scheduling. In Wren, A., editor, *Computer Scheduling of Public Transport*, p. 269–280. North-Holland, Amsterdam.
- Sampathkumar, E. e Walikar, H. (1973). The connected domination number of a graph. *J. Math. Phy. Sci.*, 13(6):607–613.
- Sprague, A. P. (1999). Np-completeness of the domatic number problem on circular arc graphs. In *Proceedings of the 37th Annual Southeast Regional Conference (CD-ROM)*, ACM-SE 37, p. 43–48, New York, NY, USA. Association for Computing Machinery.
- Veldman, H. (1983). Existence of dominating cycles and paths. *Discrete Mathematics*, 43(2):281–296. ISSN 0012-365X.
- Weichsel, P. M. (1994). Dominating sets in n-cubes. *Journal of Graph Theory*, 18(5):479–488.
- Yetgin, H., Cheung, K. T. K., El-Hajjar, M., e Hanzo, L. H. (2017). A survey of network lifetime maximization techniques in wireless sensor networks. *IEEE Communications Surveys Tutorials*, 19(2):828–854.
- Zelinka, B. (1983). On k -domatic numbers of graphs. *Czechoslovak Mathematical Journal*, 33(2):309–313.