

Flow-based approach for a Checkpoint Allocation Problem for COVID 2019

Pedro Jorge de Abreu Figueredo

Mestrado e Doutorado em Ciência da Computação
Campus do Pici, Bloco 910, Fortaleza, Ceará
pedrojorgedaf@alu.ufc.br

Manoel Campêlo

Departamento de Estatística e Matemática Aplicada
Mestrado e Doutorado em Ciência da Computação
Campus do Pici, Bloco 910, Fortaleza, Ceará
mcampelo@ufc.br

Márcio Costa Santos

Departamento de Computação - Instituto de Ciências Exatas
Universidade Federal de Minas Gerais
Rua Reitor Píres Albuquerque, ICEx, Pampulha, Belo Horizonte, Minas Gerais
marciocs@dcc.ufmg.br

RESUMO

Utilizando dados de [Figueredo et al., 2022], este estudo emprega um modelo baseado em fluxo para otimizar a alocação de pontos de controle em estradas para a contenção da COVID-19 em cenários com recursos limitados. Ele revisa um modelo de Programação Binária Inteira (BIP) e introduz uma versão mais sofisticada, BIP2, avaliada em mais de 15.000 instâncias com um solucionador matemático. Aproximadamente 13.000 dessas instâncias foram resolvidas com sucesso dentro de um limite de dois minutos. O BIP2 demonstra melhorias significativas na relaxação linear e na eficiência, especialmente em cenários complexos marcados por pseudo-aleatoriedade. A adoção de uma heurística gulosa aumenta ainda mais a eficiência do modelo, possibilitando soluções eficazes em ambientes com múltiplas equipes e destacando a eficácia dos métodos heurísticos na gestão de operações intrincadas. Esta pesquisa sublinha a robustez e adaptabilidade dos modelos, sugerindo sua aplicabilidade em diversos contextos operacionais além da gestão de pandemias.

PALAVRAS CHAVE. Alocação de Recursos, COVID 19, Programação Inteira.

ABSTRACT

Using data from [Figueredo et al., 2022], this study employs a flow-based model to optimize checkpoint allocations on roads for COVID-19 control in resource-scarce settings. It revises a Binary Integer Program (BIP) model and introduces a more sophisticated version, BIP2, evaluated across over 15,000 instances with a mathematical solver. Approximately 13,000 of these were successfully resolved within a two-minute time frame. BIP2 demonstrates significant enhancements in linear relaxation and efficiency, particularly in complex scenarios marked by pseudo-randomness. The adoption of a greedy heuristic further boosts model efficiency, enabling effective solutions in environments involving multiple teams and showcasing the heuristic methods' effectiveness in managing intricate operations. This research underscores the models' robustness and adaptability, suggesting their applicability in diverse operational settings beyond pandemic management.

KEYWORDS. Resource Allocation. COVID 19. Integer Programming.

1. Introduction

In the wake of the COVID-19 pandemic, the deployment of roadside and high population concentration area checkpoints has been identified as a crucial public health measure to control the spread of the virus through massive testing and intelligent lockdown strategies ([Pereira et al., 2022], [Yen et al., 2020]). These checkpoints, primarily aimed at screening and monitoring traffic in urban settings, must be strategically located to maximize their effectiveness in detecting infections and managing public health responses. The overarching challenge involves optimizing the placement of these checkpoints over various periods, considering limited resources and the dynamic nature of urban traffic and disease spread.

The study we previously proposed in [Figueredo et al., 2022] addresses the critical task of deploying roadside checkpoints during the COVID-19 pandemic, aimed at controlling virus transmission through strategic location and timing. The problem involves selecting optimal locations for these checkpoints from a set of potential sites $I = \{1, \dots, m\}$. For each time t in a defined time horizon $T = \{1, \dots, n\}$, we aim to choose a subset $S_t \subseteq I$ that maximizes the function $\sum_{t \in T} \sum_{i \in S_t} \mu_{it}$, where μ_{it} represents a data-driven parameter associated with allocating resources to location i in period t (in this setting, the number of expected infectious people).

Resource constraints play a crucial role in this optimization problem. Specifically, we manage a finite number of teams $K = \{1, \dots, p\}$, comprising police officers, health professionals, and other critical staff – each with specific operational constraints. These constraints are designed to enforce several properties:

- *property 1 (Indivisibility of Teams)*: Each team can be assigned to only one location at any time.
- *property 2 (Unitary Allocation)*: Only one team can be stationed at any location at a given time.
- *property 3 (Time Schedule)*: Each team $k \in K$ can operate only during specific times $\bar{T}(k) \subseteq T$, outside of which they are unavailable.
- *property 4 (Location Compatibility)*: Teams may only be assigned to locations for which they are specifically suited, indicated by $\bar{I}(k)$.
- *property 5 (Relocation Time)*: Teams require a minimum period to relocate – Figueredo et al. [2022] use exactly one period, adding a temporal dimension to resource planning.

These properties are designed to realistically model the logistical challenges of deploying roadside checkpoints, where resource scarcity, team compatibility, and operational timing are critical. By addressing these factors, we provide a structured solution to manage and simulate checkpoint operations during ongoing pandemic conditions efficiently.

In our previous work [Figueredo et al., 2022], we used integer programming to address this complex problem, leveraging traffic volume (from 2014 to 2019) and public health data from New York City ([City of New York], [Department of City Planning (DCP) of New York City] and [Health Department of New York City]) to determine optimal locations for these checkpoints. Our model efficiently processed a dataset of over 15,000 instances through computational experiments, optimally solving 13,103 of them within a two-minute time frame per instance. However, instances characterized by pseudo-randomness in teams' time schedules presented significant challenges, exemplifying the model's limitations with highly dynamic and unpredictable scenarios. These challenging instances, particularly under extended time limits of up to one hour, demonstrated the limitations of our

proposed integer programming approach, thus necessitating the exploration of more flexible and efficient methodologies to better adapt to the dynamic environment of a pandemic.

In this work, we detail our structured approach to addressing the COVID-19 pandemic through checkpoint allocation: Section 2 describes our instances, detailing potential locations, periods, and team setups, alongside effectiveness metrics derived from New York City traffic and COVID-19 data. Section 3 revisits our binary integer programming approach (BIP1), focusing on logistical constraints for effective deployment. Section 4 advances this approach by employing a flow-based approach to the possible dynamic resource allocation. It results in a new binary integer programming model (BIP2). Section 5 introduces a greedy algorithm for more complex multi-team scenarios, enhancing our model's efficiency. Section 6 evaluates BIP1 and BIP2 models, addressing the challenges of pseudo-random scheduling. Finally, Section 7 synthesizes these findings, highlighting the superior performance of the flow-based model (BIP2) and proposing future improvements to adapt our models to broader operational environments. This analysis showcases the models' capabilities and maps out directions for future research to improve their applicability and effectiveness.

This study was partly financed by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001 and the Pró-Reitoria de Pesquisa da Universidade Federal de Minas Gerais. This work has been supported by the following Brazilian agencies and grants: FUNCAP (PS1-0186-00155.01.00/21, PGP-0192-00040.01.00/22 and ITR-0214-00096.01.00/23) and CNPQ (312417/2022-5, 404479/2023-5 and 442977/2023-9).

2. Our Instances

This section outlines our method for generating instances from the dataset previously created in [Figueredo et al., 2022]. Each instance is constructed by defining three sets: I , the possible locations; T , the periods within a day; and K , the teams available. Each team k is associated with specific subsets $\bar{I}(k)$ and $T(k)$ which delineate possible locations and available working times, respectively. The costs or scores, μ_{it} , linked to potential allocations, quantify the effectiveness of deploying a team to a location during a specific time slot.

The dataset described in [Figueredo et al., 2022] includes 60 unique dates, each with a predefined set of locations. However, for our analysis, we only consider instances with fewer than 200 locations in the original dataset due to the large size and ample quantity of the instances. This reduces the dataset to 51 dates. For each of them, we set $|T| = 24$, reflecting 24 hours (24 periods of 1 hour in a day), and define I using all available locations during that span. The number of teams $|K|$ is calculated as 30% of the total number of locations, rounded down to the nearest integer number. Each team's schedule is influenced by two main types of restrictions: location and time. These restrictions are detailed as follows:

- **Location Restrictions:**

- *Full*: Teams can be assigned to any location.
- *Neig*: Teams are restricted to operating within a single borough.
- *Rand*: There is a 50% chance that a team is compatible with a given location.

- **Time Restrictions:**

- *Full-time*: Teams work from 8 am to 12 am and from 1 pm to 6 pm.
- *Stage*: Teams work from 8 am to 12 am.
- *Night*: Teams are scheduled from 10 pm to 5 am.

Instance Partition	Description
12-0-0-0-0-0-0-0-0-0-0-0	All teams of the kind (Full) + (Full-time).
0-0-0-6-6-0-0-0-0-0-0-0	Half of teams are (Full) + (Rand), and the other half (Neig) + (Full-time) .
4-0-0-0-4-0-0-0-4-0-0-0	1 third of teams are (Full) + (Full-time), 1 third (Neig) + (Full-time), and 1 third (Rand) + (Full-time)

Tabela 1: Teams partitions with perfect size.

- *Rand*: There is a 50% chance that a team can work at a given hour.

We organized teams into 12 possible combinations of the above restrictions, leading to diverse team schedules. For example, a schedule might combine “Full” location access with “Stage” time restrictions. The teams are distributed among these combinations as defined by a 12-tuple $q_1 - q_2 - \dots - q_{12}$, where the sum of all q_i values equals 12. This tuple dictates the proportion of teams under each scheduling type, ensuring a range of operational scenarios. Due to the vast number of potential combinations, our experiments focus on instances where teams have one, two, or three types of schedules (see Table 1).

The effectiveness metric μ_{it} for each team deployment is calculated by multiplying the traffic flow at the interval of time by the rate of visits to health facilities (All ages with Covid-like illness) for each location. This metric is designed to estimate the potential detection of COVID-19 cases, thereby optimizing the placement of checkpoints to manage and mitigate virus transmission effectively.

3. A Previous Mathematical Model

In our previous work [Figueredo et al., 2022], we developed a compact binary integer programming model to tackle the problem presented. To frame the model, we define a binary decision variable $x_{k,i}^t \in \{0, 1\}$ that indicates whether ($x_{k,i}^t = 1$) or not ($x_{k,i}^t = 0$) team $k \in K$ is assigned to location $i \in \bar{I}(k)$ during time period $t \in \bar{T}(k)$. This variable forms the basis of our integer programming model, structured as follows:

$$(BIP1) \max \sum_{k \in K} \sum_{i \in \bar{I}(k)} \sum_{t \in \bar{T}(k)} \mu_{it} x_{k,i}^t \quad (1a)$$

$$\text{s.t.} \quad \sum_{i \in \bar{I}(k)} x_{k,i}^t \leq 1 \quad \forall k \in K, \forall t \in \bar{T}(k), \quad (1b)$$

$$\sum_{k \in K: i \in \bar{I}(k), t \in \bar{T}(k)} x_{k,i}^t \leq 1 \quad \forall i \in I, \forall t \in T, \quad (1c)$$

$$x_{k,i}^{t-1} + \sum_{j \in \bar{I}(k) \setminus \{i\}} x_{k,j}^t \leq 1 \quad \forall k \in K, \forall i \in \bar{I}(k), \forall t - 1, t \in \bar{T}(k), \quad (1d)$$

$$x_{k,i}^t \in \{0, 1\} \quad \forall k \in K, \forall i \in \bar{I}(k), \forall t \in \bar{T}(k). \quad (1e)$$

The objective function (1a) aims to maximize the sum of values μ_{it} , which quantifies the impact of deploying a team to a specific location at a particular time. Constraint (1b) ensures that each team can be assigned to only one location per period, fulfilling *property 1*. Constraint (1c) ensures that no more than one team occupies any given location at the same time, thus satisfying

property 2. Constraint (1d) prevents teams from being reassigned too quickly, addressing the logistical challenge of relocation, encapsulated in *property 5*. Lastly, Constraint (1e) enforces the binary nature of the decision variables, inherently maintaining *properties 3* and *4* based on the variable definitions.

4. A Mathematical Programming Model using Flow

In this section, we introduce an alternative approach to modeling our problem using a flow-based framework. This method hinges on constructing a directed graph, G_k , for each team $k \in K$, which encapsulates all possible allocations of k to the locations in $\bar{I}(k)$ along the periods in $\bar{T}(k)$. These individual graphs are then merged into a larger digraph, G^* , where the goal is to find $|K|$ internally disjoint paths, where the one using vertices of G_k describes the allocations of team k along the day.

The directed graph G^* is structured as shown in Figure 1a. It has a source vertex s that dispatches a single unit of flow through every arc (s, s_k) , where s_k is the source vertex of digraph G_k . Similarly, it has a target vertex l that receives a unit of flow from every arc l_k, l , where l_k is the target vertex of digraph G_k . In other terms, s_k and l_k are the starting point and ending point of the path in G_k . By its turn, G_k has a vertex v_{kit} for each location $i \in \bar{I}(k)$ and time $t \in \bar{T}(k)$ as well as a vertex v_{kat} to represent the possibility of k be not allocated in time $t \in T$. Thus, a can be seen as an artificial location (See Figure 1b). The horizontal and vertical layers in G_k correspond to locations and times respectively. Thus, the arcs in G_k represent the possible transitions of team k between locations during a day. To get smaller graphs, we can contract the vertices l_k and l .

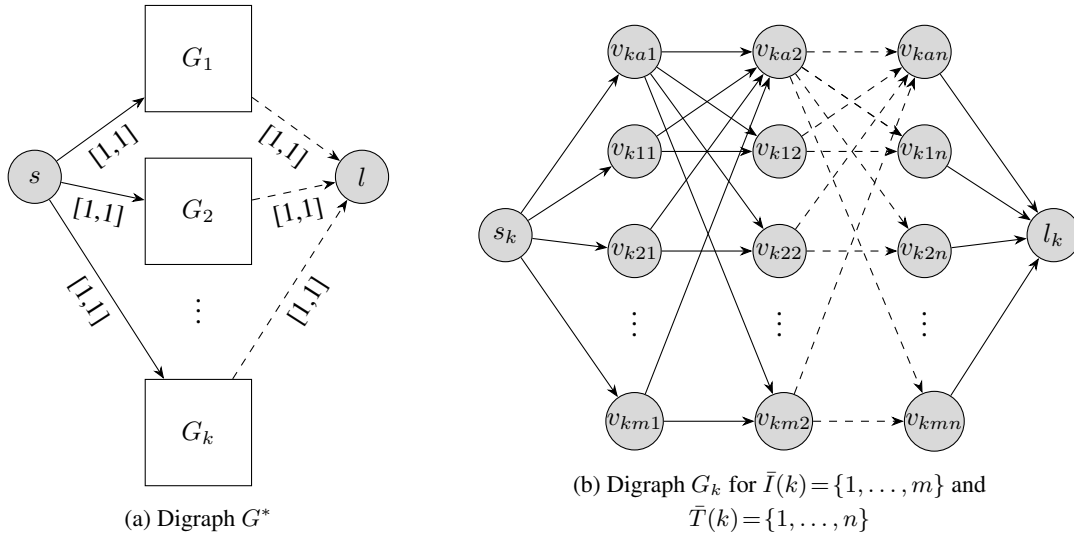


Figura 1: Illustration of digraphs that base the flow-model (BIP2).

Precisely, the vertex set of G_k is

$$V(G_k) = \{v_{k,i,t} : i \in \bar{I}(k), t \in \bar{T}(k)\} \cup \{v_{k,a,t} : t \in T\} \cup \{s_k, l_k\}$$

The arcs of G_k are of the following types:

Movement Arcs: $(v_{k,i,t-1}, v_{k,i,t})$ for $i \in \bar{I}(k)$ and $t-1, t \in \bar{T}(k)$.

Movement Arcs (Artificial vertices): $(v_{k,a,t-1}, v_{k,a,t})$ for $t-1, t \in T$.

Standby Arcs: $(v_{k,i,t-1}, v_{k,a,t})$ for $i \in \bar{I}(k)$, $t-1 \in \bar{T}(k)$ and $t \in T$.

Activation Arcs: $(v_{k,a,t-1}, v_{k,i,t})$ for $i \in \bar{I}(k)$, $t-1 \in T$ and $t \in \bar{T}(k)$.

Connection Arcs: (s_k, v_{k,i,t_0}) for $i \in \bar{I}(k)$ and $t_0 = \min \bar{T}(k)$.

Connection Arc (Artificial vertex): $(s_k, v_{k,a,1})$.

Sink Arcs: (v_{k,i,t_1}, l) for $i \in \bar{I}(k)$ and $t_1 = \max \bar{T}(k)$.

Sink Arc (Artificial vertex): $(v_{k,a,|T|}, l)$.

Regarding G^* , the vertex and edge sets respectively are

$$V(G^*) = \bigcup_{k \in K} V(G_k) \cup \{s\} \quad \text{and} \quad E(G^*) = \bigcup_{k \in K} (E(G_k) \cup (s, s_k)).$$

We can see that all viable combinations of $K \times \{I \cup \{a\}\} \times T$ are represented in the graph G^* since we have a unique vertex for each combination of location, team, and period, and arcs connecting them according to the problem's constraints. We still have to take care that the chosen paths allocate at most one team to every location at a given time.

Before continuing, we need to define some notation. We denote by $\mathcal{N}_G^-(v)$ the negative neighborhood of a vertex $v \in V$ in a digraph $G = (V, E)$, i.e. $\mathcal{N}_G^-(v) = \{j \in V : (j, v) \in E\}$. Similarly, let $\mathcal{N}_G^+(v)$ be the positive neighborhood of $v \in V$ in a digraph $G = (V, E)$, i.e. $\mathcal{N}_G^+(v) = \{j \in V : (v, j) \in E\}$.

In our model, we represent a feasible solution to the problem by $|K|$ internally disjoint flow paths from s to l in G^* . Thus, if a vertex $v_{k,i,t}$ belongs to a flow path, then team k is allocated to location i during period t in the corresponding feasible solution. Specifically, given a binary flow f on G^* , we can create a solution set $S(f, G_k) = \{(k, i, t) \in K \times I \times T : f^-(v_{k,i,t}) = 1, v_{k,i,t} \in V(G_k)\}$, for each team $k \in K$, where $f^-(v_{k,i,t})$ stands for the amount of flow in f entering vertex $v_{k,i,t}$. Then, we can write the model as follows, where the binary variable f_e^k stores the flow passing through arc $e \in E(G_k)$.

$$(BIP2) \max \sum_{v_{k,i,t} \in V(G^*)} \sum_{j \in \mathcal{N}_{G^*}^-(v_{k,i,t})} \mu_{it} f_{j,v_{k,i,t}}^k \quad (2a)$$

s.t.

$$f_{ss_k} = 1 \quad k \in K, \quad (2b)$$

$$\sum_{j \in \mathcal{N}_{G^*}^-(v)} f_{jv}^k - \sum_{j \in \mathcal{N}_{G^*}^+(v)} f_{vj}^k = 0 \quad \forall v \in V(G_k) \setminus \{l\}, \forall k \in K, \quad (2c)$$

$$\sum_{k \in K: v_{k,i,t} \in V(G_k)} \sum_{j \in \mathcal{N}_{G^*}^-(v_{k,i,t})} f_{j,v_{k,i,t}}^k \leq 1 \quad \forall i \in I, t \in T, \quad (2d)$$

$$f_e^k \in \{0, 1\} \quad \forall k \in K, e \in E(G_k) \quad (2e)$$

The objective function (2a) maximizes the weighted flow, where each flow path represents a team schedule and its flow value is the sum of the μ_{it} associated with those allocations. For each $k \in K$, Constraints (2b)-(2c) define a path on G_k , whose construction guarantees properties 1, 3, 4, and 5 for the corresponding schedule. Constraints (2d) model property 2 by ensuring that no two teams are allocated to the same location simultaneously.

Analyzing Constraints (2b)-(2c), we see that they form a block diagonal structure, with one block related to each team k . Constraints (2d) are the linking constraints that make the model challenging. Indeed, disregarding these constraints, note that we could solve each block separately in polynomial time. As each digraph G_k is acyclic, a search algorithm for the path with maximum cost flow (heaviest path by making arc weights equivalent to the values μ_{it}) between s_k and l would be sufficient.

5. A Greedy Approach

In Section 4, we utilized a graph of feasible schedules and a flow-based model to address the problem. In a specific polynomial case when there is only one team, the problem can be viewed as the longest path problem in a weighted Direct Acyclic Graph (DAG), which is polynomially solvable in linear time [Ahuja et al., 1993]. The solution to the longest path allows us to construct feasible solutions for our second model by assigning flow units to arcs that are incident to vertex $v_{1,i,t}$ ($k = 1$ representing a single team) with the same gain μ_{it} , given the weights of its edges, and zero otherwise.

Let us consider that $M(k, i, t)$ is a Boolean function indicating whether a team $k \in K$ can be at location $i \in I$ at time $t \in T$. We propose an algorithm where each graph G_k is handled individually; the solution with the maximum objective function value is selected, its allocations are blocked for all other teams, and then the process is repeated for the remaining teams until all graphs are utilized to construct a feasible solution. See Algorithm 1.

Algorithm 1 A Greedy Heuristic

```

Create vector schedule[k] as a list for  $k \in K$ .
Create vector sol_value[k] as a float for  $k \in K$ .
obj_value  $\leftarrow 0$ 
Create list of all teams  $L \leftarrow \{1, 2, \dots, |K|\}$ 
while  $L \neq \emptyset$  do
  for each team  $k$  in  $L$  do
    Create a weighted DAG  $D_k$  from  $G_k$  with vertices indicated by  $M(k, i, t)$  and edge-weights  $\mu_{it}$ 
    schedule[k], sol_value[k]  $\leftarrow$  LongestPath( $D_k$ )
  end for
  chosen_team  $\leftarrow \arg \max_{k \in L} \{\text{sol\_value}[k]\}$ 
   $L \leftarrow L \setminus \{\text{chosen\_team}\}$ 
  for each team  $k$  in  $L$  do
    for each pair  $(i, t)$ ,  $i \in I, t \in T$ , such that  $M(k, i, t) = \text{True}$  do
      if  $(i, t) \in \text{schedule}[\text{chosen\_team}]$  then
         $M(k, i, t) \leftarrow \text{False}$ 
      end if
    end for
  end for
end while
obj_value  $\leftarrow \sum_{k \in K} \text{sol\_value}[k]$ 
return schedule, obj_value

```

Algorithm 1 creates a feasible schedule (`schedule[k]`) for every team k by applying the Longest Path on DAGs algorithm [Ahuja et al., 1993] to each iteratively modified DAG G_k , and determines the potential contribution (`sol_value[k]`) to the objective function. Then, it selects the team (`chosen_team`) with the highest contribution, locks in the schedule for the `chosen_team`, and repeats the process for the still non-allocated teams. The function $M(k, i, t)$ ensures that all teams' schedules do not overlap, thereby creating a feasible solution and allowing for the calculation of the final objective value (`obj_value`).

We have that Algorithm 1 constitutes a $1/|K|$ -approximation algorithm. Recall that, for maximization problems, an α -approximation algorithm is such that the returned solution has value at least α times the optimal solution value, where $0 < \alpha \leq 1$.

6. Computational Experiments

In this section, we utilize a set of instances previously discussed in Figueredo et al. [2022] concerning our problem to compare the new computational results with earlier results. Our compu-

tational environment consists of the Cplex 20.1.0 solver and an implementation in Julia 1.6.3 using the JuMP 0.21.10 framework. For the *BIP2* model, we configure our continuous optimizer to use the CPLEX Network Solver, taking advantage of the algorithm's suitability for the network-like structure of the model. The tests were conducted on an Intel Core i5-7400 CPU (6MB Cache, up to 3.50 GHz clock speed in turbo mode), with 4 threads, 16GB (2x8) of HyperX Fury DDR4 RAM at 2400 MHz, and a 1TB standard Barracuda HD.

In the instances, the number of teams equals to 30% of the number of locations and, due to data constraints, employ a time interval spanning 24 periods (one hour each). The number of locations varies from 7 to 136, which reflects the variability of the real data on which these instances are based. Rather than focusing solely on the number of locations, our previous work evaluates performance based on the type of team schedule $(\bar{I}(k), \bar{T}(k))$ generated. Each team follows one of the 12 proposed team schedule types, and for each instance, we can partition its teams into at most three different types of team schedules, as indicated by the field partition. Altogether, we have generated 15,198 instances in a combinatorial fashion, based on the partition used and the dimensions of the real data.

In our previous study [Figueredo et al., 2022], we categorized all cases with the pattern “0-0-0-12-0-0-0-0-0-0-0-0” (exclusively type 4 schedules) as “hard instances”. Due to the predominant presence of type 4 schedules on instances that are not solved to optimality on the time limit, big gaps in the integer solutions found, and longer times to solve to optimality. We believe that the pseudo-randomness of the instance schedules generates more complex scenarios.

All tables presented are limited to the top 10 rows, sorted by the parameter *Mean* (mean solution time or gap) or *n* (number of instances) in descending order, depending on which appears on the table. In the case of drawing, we use partitions in lexicographic order.

6.1. Combining the greedy heuristic with *BIP1*

In this subsection, we discuss the results related to our greedy heuristic applied to formulation (*BIP1*). The lower bound provided by the heuristic is used to prune the Branch&Bound tree. Notably, model (*BIP1*) is the same as in [Figueredo et al., 2022]; thus, we use the previous results to evaluate the improvements the greedy algorithm offers.

Previous results analysis reveals that instances featuring a team schedule of type 4 (only the fourth position in the field partition is non-zero) tend to be more challenging. This difficulty may arise because such instances, particularly types 4 and 12, introduce a level of pseudo-randomness: type 12 incorporates randomness in period availability and location compatibility, while type 4 exhibits randomness only in periods but can occur in any location. These characteristics persist in newer tests with the greedy lower bound, as these types of partitions often lead to increased occurrences of the solution statuses `TIME_LIMIT` and `NO_SOLUTION`.

The use of the lower bound introduced an additional solution status (`NO_SOLUTION`), besides those described in [Figueredo et al., 2022]. It indicates the failure to initialize the solution process with an integer solution within a 2-minute time limit using CPLEX's dynamic search proprietary algorithm. This status does not suggest that the solution method becomes worse; rather, our results showed that the overall time limit was reached in fewer cases, thus demonstrating enhanced solver efficiency and yielding 30 more optimal solutions found than before.

Regarding the performance of the exact method, our analysis indicates that the (*BIP1*) model with the lower bound takes 6.04 seconds to solve to optimality. Furthermore, the gap for instances reaching `TIME_LIMIT` is less than or equal to 0.06%, with an average gap of 0.01%.

Comparisons of mean solution times for optimal solutions and gaps for `TIME_LIMIT` instances illustrate very similar outcomes when employing our methods with and without the greedy

algorithm. The consistent improvement observed using the lower bound justifies its application in subsequent tests for both *BIP1* and *BIP2* models.

6.2. Comparison between *BIP1* and *BIP2*

This subsection delves into a comparative analysis between *BIP1* and *BIP2*, focusing on their performance under a 2-minute time constraint. We discovered that *BIP2* not only consistently outperformed *BIP1* in terms of finding optimal solutions but also demonstrated enhanced efficiency in handling integer solutions within the time limit.

Figure 2 highlights the performance differential between the two models, showing that *BIP2* found 13,932 optimal solutions compared to 13,136 by *BIP1*, and managed to find 1,177 integer solutions within the time limit compared to 1,216 by *BIP1*. Significantly, *BIP2* exhibited a much lower occurrence of the status NO_SOLUTION.

The trend of more challenging instances associated with team schedules of type 4 continues to be evident, as reflected in Tables 2 and 3. *BIP2* demonstrates superior performance metrics: it resolves instances to optimality in approximately 3.62 seconds (Table 4) and exhibits a mean relative gap of less than 0.01% for TIME_LIMIT instances (Table 5). Furthermore, *BIP2* spends 1.79 seconds on average per instance for achieving optimality, significantly faster than the 6 seconds spent by *BIP1* (Table 6). This enhanced efficiency suggests that *BIP2*'s flow problem structure provides better linear relaxation, contributing to its faster resolution times and improved performance in evaluating relative gaps (Table 7).

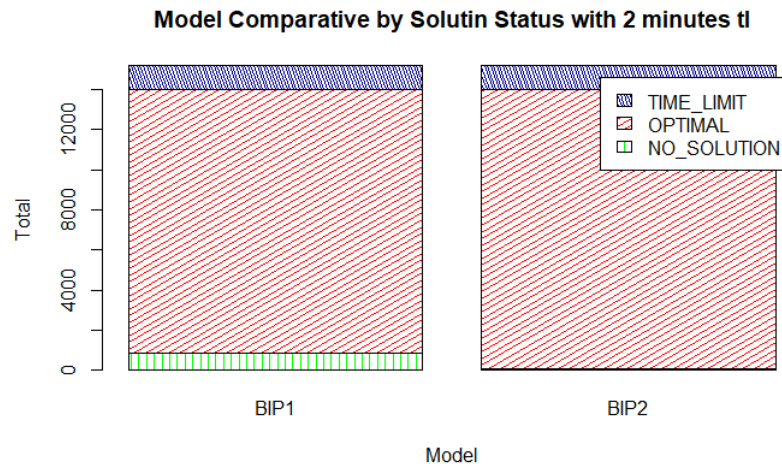


Figura 2: Comparative *BIP1* and *BIP2* solution status.

6.3. Results for *BIP1* and *BIP2* with 1-hour runs

Despite *BIP2*'s superior performance in shorter trials, extending the time limit to one hour yielded minimal gains in optimality. Specifically, *BIP2* managed to solve only one more instance to optimality compared to *BIP1*, achieving a total of 26 optimal instances as illustrated in Figure 3. This marginal improvement required significantly more time, thus increasing the mean solution time of the *BIP2* (Table 8). However, *BIP2* demonstrated better efficiency in minimizing the relative gaps compared to *BIP1*, particularly in these challenging scenarios with full type 4 schedules, as shown in Table 8. This indicates that while *BIP2* excels in more complex setups, the incremental benefits diminish as the time allowed for solving these hard instances increases.

	partition	n
1	0-0-0-4-0-0-0-0-4-0-4	25
2	0-0-0-4-0-0-0-0-4-0-4	25
3	0-0-0-4-0-0-0-4-0-0-4	25
4	0-0-0-4-0-0-4-0-0-0-4	25
5	0-0-0-4-4-0-0-0-0-0-4	25
6	0-0-0-6-0-0-0-6-0-0-0	24
7	0-0-0-6-6-0-0-0-0-0-0	24
8	0-4-0-4-0-0-0-0-0-0-4	24
9	0-0-0-0-0-0-0-0-0-0-12	23
10	0-0-0-4-0-0-0-0-0-0-4-4	23

Tabela 2: Number of instances *BIP2* reached time limit of 2 minutes.

	partition	n
1	0-0-0-12-0-0-0-0-0-0-0	22
2	0-0-0-6-0-0-0-0-0-0-6	10
3	6-0-0-6-0-0-0-0-0-0-0	10
4	4-0-0-4-0-0-0-0-0-0-4	8
5	4-0-0-4-0-0-0-0-4-0-0	5
6	0-0-6-6-0-0-0-0-0-0-0	3
7	4-0-0-4-0-0-0-4-0-0-0	3
8	4-0-4-4-0-0-0-0-0-0-0	3
9	0-0-4-4-0-0-0-0-0-0-4	2
10	0-0-4-4-0-0-0-0-0-4-0	2

Tabela 3: Number of instances *BIP2* reached no-solution status within 2 minutes.

	partition	Mean	Std	Min	Max
1	0-0-0-4-4-0-0-0-4-0-0-0	21.87	39.71	0.01	119.97
2	0-0-0-4-0-0-4-0-0-4-0-0	21.83	35.14	0.00	113.29
3	0-4-0-4-0-0-0-0-0-4-0-0	21.43	34.52	0.00	113.99
4	0-0-0-4-0-0-0-4-0-4-0-0	20.72	37.63	0.01	117.07
5	4-4-0-4-0-0-0-0-0-0-0-0	20.08	33.65	0.00	115.84
6	0-4-0-4-0-4-0-0-0-0-0-0	19.82	32.50	0.01	111.82
7	0-0-0-4-0-4-4-0-0-0-0-0	18.90	35.48	0.00	119.17
8	0-4-0-4-0-0-4-0-0-0-0-0	17.32	33.84	0.00	116.13
9	0-0-4-4-0-0-0-0-0-0-4-0	17.07	28.24	0.00	119.92
10	0-0-4-4-0-0-0-4-0-0-0-0	17.01	35.19	0.01	111.29
11	All	3.62	12.87	0.00	119.97

Tabela 4: Computing times for optimally solved instances on *BIP2* within 2 minutes.

	partition	Mean	Std	Min	Max
1	0-0-0-12-0-0-0-0-0-0-0-0	0.02	0.01	0.01	0.04
2	0-0-0-6-0-0-0-0-0-0-0-6	0.01	0.01	0.01	0.05
3	0-4-0-4-0-0-4-0-0-0-0-0	0.01	0.02	0.00	0.06
4	4-0-0-4-0-0-0-4-0-0-0-0	0.01	0.01	0.00	0.03
5	0-0-0-6-0-0-0-0-6-0-0-0	0.01	0.01	0.00	0.03
6	6-0-0-6-0-0-0-0-0-0-0-0	0.01	0.01	0.00	0.02
7	0-4-0-4-0-0-0-0-0-4-0-0	0.01	0.01	0.00	0.02
8	0-0-0-6-0-0-0-0-6-0-0-0	0.01	0.01	0.00	0.03
9	0-0-0-6-6-0-0-0-0-0-0-0	0.01	0.00	0.00	0.01
10	0-0-0-6-0-6-0-0-0-0-0-0	0.01	0.00	0.00	0.01
11	All	0.00	0.00	0.00	0.06

Tabela 5: Relative Gap for instances reaching time limit by partition.

7. Conclusions

In this work, we build upon an established problem framework to generalize scenarios where data may not be very detailed, although there is still a need to allocate teams to points of interest across multiple periods in resource-constrained environments. We enhanced our approach by using a greedy heuristic to effectively generate feasible solutions and further developed the BIP model based in [Figueredo et al., 2022], which is notable for its substantial dimensions. This advancement allowed us to rigorously test our models. Particularly, *BIP2* has demonstrated significant

	partition	<i>BIP1</i>				<i>BIP2</i>			
		Mean	Std	Min	Max	Mean	Std	Min	Max
1	0-4-0-4-0-0-0-0-4-0-0	26.54	42.52	0.00	119.17	10.57	23.01	0.00	113.99
2	0-0-0-4-0-4-0-0-0-4-0-0	25.99	36.91	0.00	102.89	10.25	21.02	0.00	99.36
3	0-0-0-4-0-0-4-0-0-0-4-0	23.77	35.71	0.00	119.68	9.81	21.78	0.00	87.74
4	4-4-0-0-4-0-0-0-0-0-0-0	22.76	35.22	0.00	108.16	2.27	3.02	0.01	8.81
5	0-4-0-4-4-0-0-0-0-0-0-0	22.00	39.58	0.00	114.01	5.76	16.50	0.01	90.94
6	0-0-0-4-4-4-0-0-0-0-0-0	20.60	34.68	0.00	109.12	7.94	22.17	0.01	108.02
7	0-0-4-4-0-4-0-0-0-0-0-0	20.00	39.96	0.00	117.19	5.55	14.65	0.00	66.52
8	0-4-0-4-0-4-0-0-0-0-0-0	18.60	31.14	0.00	96.09	7.82	18.03	0.01	80.20
9	4-0-0-0-4-0-0-0-0-4-0-0	18.26	32.61	0.00	110.48	1.26	1.89	0.00	8.65
10	4-0-4-0-0-0-0-0-0-0-4	16.70	30.04	0.00	113.02	2.75	4.90	0.00	24.35
11	All	6.00	16.34	0.00	119.90	1.79	6.77	0.00	114.07

Tabela 6: Comparison between BIP 1 and BIP 2 – Computing times for optimally solved instances on BIP2 within 2 minutes.

	partition	<i>BIP1</i>				<i>BIP2</i>			
		Mean	Std	Min	Max	Mean	Std	Min	Max
1	0-0-0-6-0-0-0-0-0-0-0-6	0.01	0.00	0.01	0.02	0.01	0.00	0.01	0.01
2	0-0-0-6-0-0-0-0-6-0-0-0	0.01	0.01	0.00	0.02	0.01	0.00	0.00	0.01
3	0-0-4-0-0-0-0-0-0-4-0-4	0.01		0.01	0.01	0.00		0.00	0.00
4	0-0-0-4-0-0-0-0-4-0-0-4	0.01	0.01	0.01	0.04	0.01	0.00	0.00	0.01
5	4-0-0-4-0-0-0-4-0-0-0-0	0.01	0.01	0.01	0.02	0.00	0.00	0.00	0.01
6	0-4-0-0-0-0-0-0-0-0-4-4	0.01		0.01	0.01	0.00		0.00	0.00
7	0-0-6-6-0-0-0-0-0-0-0-0	0.01	0.00	0.01	0.02	0.00	0.00	0.00	0.01
8	0-0-0-6-0-0-0-0-0-6-0-0	0.01	0.01	0.00	0.07	0.01	0.01	0.00	0.03
9	0-0-0-4-0-0-0-4-0-0-0-4	0.01	0.00	0.00	0.02	0.01	0.00	0.00	0.01
10	0-0-0-6-0-0-6-0-0-0-0-0	0.01	0.01	0.00	0.02	0.01	0.00	0.00	0.01
11	All	0.01	0.01	0.00	0.07	0.00	0.00	0.00	0.03

Tabela 7: Comparison between BIP 1 and BIP 2 – Relative Gap for instances reaching 2 minutes time limit by partition.

	Parameter	<i>BIP1</i>				<i>BIP2</i>			
		Mean	Std	Min	Max	Mean	Std	Min	Max
1	Solve Time	0.01	0.00	0.00	0.02	33.97	176.35	0.00	916.40
2	Relative Gap	0.02	0.01	0.01	0.03	0.01	0.00	0.00	0.02

Tabela 8: Comparative *BIP1* and *BIP2* for mean results to OPTIMAL (Solution Time) and TIME_LIMIT(Gap) solutions status by parameter within 1 hour.

promise, surpassing the previous model with its enhanced linear relaxation capabilities.

Despite these advancements, the instances categorized as hard remained challenging, with many still unresolved within a one-hour time limit. This observation suggests that further improvements in BIP2, possibly through the application of decomposition strategies, could accelerate solution processes for these complex instances. Additionally, exploring mathematical heuristics presents another viable strategy for addressing these difficulties.

It is crucial to underscore that the adaptability of our model allows for its application in a wide range of contexts beyond the initial focus. Future research could fruitfully explore the creation of a more versatile and generic model. The use of metaheuristics, particularly Evolutionary

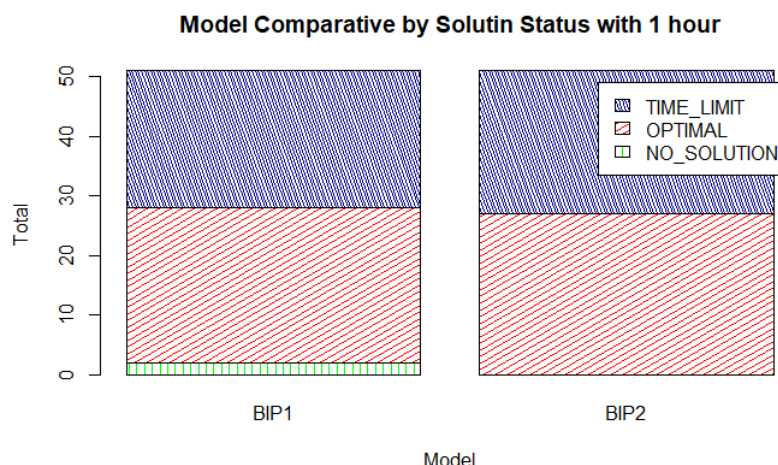


Figura 3: Comparison between *BIP1* and *BIP2* – number of instances by solution status for harder instances.

Algorithms, could offer a robust method to efficiently solve the toughest cases within acceptable time frames. These explorations would not only broaden the applicability of our model but also enhance its effectiveness in dynamic and demanding operational environments.

Referências

- Ahuja, R. K., Magnanti, T. L., e Orlin, J. B. (1993). *Network Flows: Theory, Algorithms, and Applications*. Prentice hall.
- City of New York. Traffic volume counts (2014-2020). <https://data.cityofnewyork.us/Transportation/Traffic-Volume-Counts-2014-2020-/ertz-hr4r/>. (Accessed: 10.02.2022).
- Department of City Planning (DCP) of New York City. Lion single line street base map. <https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-lion.page>. (Accessed: 02.07.2021).
- Figueredo, P. J., Campêlo, M., e Santos, M. C. (2022). Applications of checkpoint allocation in covid context. In *ANAIS DO LIV SIMPÓSIO BRASILEIRO DE PESQUISA OPERACIONAL*, Juiz de Fora. SOBRAPO.
- Health Department of New York City. NYC coronavirus disease 2019 (covid-19) data. <https://github.com/nychealth/coronavirus-data/>. (Accessed: 02.07.2021).
- Pereira, M. A., Dinis, D. C., Ferreira, D. C., Figueira, J. R., e Marques, R. C. (2022). A network data envelopment analysis to estimate nationsâ efficiency in the fight against sars-cov-2. *Expert Systems with Applications*, 210:118362.
- Yen, M.-Y., Schwartz, J., Chen, S.-Y., King, C.-C., Yang, G.-Y., e Hsueh, P.-R. (2020). Interrupting covid-19 transmission by implementing enhanced traffic control bundling: Implications for global prevention and control efforts. *Journal of Microbiology, Immunology and Infection*, 53(3): 377–380.