Applications of checkpoint allocation in Covid context

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
Universidade Federal do Ceará - Campus de Russas
R. Felipe Santiago, 411 - Cidade Universitária, Russas - Ceará
marciocs@ufc.br

RESUMO

Motivado pelo início da pandemia da Covid 19, este trabalho estuda formas de definir e localizar pontos de controle rodoviário por vários períodos de tempo, na tentativa de reduzir taxas de infecção, em um ambiente com recursos limitados. A partir de dados abertos da cidade de Nova York sobre volume de tráfego e Covid 19, criamos um conjunto de dados para gerar os pontos de interesse e uma métrica para avaliar aqueles escolhidos como pontos de controle. Construímos um modelo de programação inteira para o cenário proposto e o solucionamos usando um resolvedor matemático. Simulamos diferentes cenários de conflitos entre recursos e pontos de interesse, ao longo de um horizonte. Num conjunto de mais de 15.000 instâncias, resolvemos 13.000 em menos de 2 minutos cada e mostramos casos difíceis de resolver.

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

Pesquisa Operacional na Área de Saúde, Programação Matemática.

ABSTRACT

Motivated by the onset of the Covid 19 pandemic, this paper studies ways to define and locate roadside control points for various time periods in an attempt to reduce infection rates in a resource-limited environment. From open data of New York City on traffic volume and Covid 19, we created a dataset to generate the points of interest and a metric to evaluate those chosen as control points. We built an integer programming model for the proposed scenario and solved it using a mathematical solver. We simulate different scenarios of conflicts between resources and points of interest, over a horizon. In a dataset of over 15000 instances, we solve 13000 in less than 2 minutes each and show hard-to-solve cases.

KEYWORDS. Resource Allocation. COVID 19. Integer Programming.

Operations Research in Health Care, Mathematical Programming.

https://proceedings.science/p/157446?lang=pt-br

1. Introduction

In controlling diseases spread by viruses such as Covid 19, a common tactic, used since the early days of this pandemic, has been temperature screening. It was very common in places with high concentration of people, as malls, to use infra-red to check people temperature. Even though such a strategy contributes to identify cases, it is not effective, given high percentages of asymptomatic patients [Wright e Mackowiak, 2020]. Other control strategies, e.g rapid coronavirus test, are far more useful to detect high viral loads and help diminish contagion rates [Guglielmi, 2021]. In this context, an important factor to consider is urban mobility influence over time, instead of just locating fixed checkpoints in hot zones like malls, hospitals, and schools. So, one should consider a dynamic location over time of checkpoints while having limited availability of resources.

Motivated by this application, in this work we introduce the following problem. Given a set of road checkpoint candidate locations $I = \{1, ..., m\}$, choose a subset $S_t \subseteq I$ for each time t over a time horizon $T = \{1, .., n\}$ so as to maximize $\sum_{t \in T} \sum_{i \in S_t} \mu_{it}$, where μ_{it} is a parameter driven by data (to be explained in Section 2). Such a choice is constrained to some resources' availability. It means that there exist a set of teams[1] $K = \{1, ..., p\}$ to setup the checkpoints, and every team has a work schedule that must be respected. Precisely, the allocation must comply with a set of properties representing desirable features of a practical solution:

- *property 1 (indivisibility of teams)* : A team can be assigned to at most one location at a given time.
- *property 2 (unitary allocation)* : At most one team can be allocated to any location at any given time.
- *property 3 (time schedule)* : Every team $k \in K$ has a restricted time set $\bar{T}(k) \subseteq T$ where it is able to work. In other words, a team is not able to be in any location at at any time in $T \backslash \bar{T}(k)$ (set of times that a team is unavailable).
- *property 4 (team x location compatibility)*: Some teams may not be compatible with a checkpoint location. We denote $\bar{I}(k) \subseteq I$ the set of compatible locations of team $k \in K$.
- *property 5 (time to relocate)*: If a team $k \in K$ is assigned to a location at a given time $t \in T$, it expends at least 1 period to be relocated to a distinct location.

Properties 1-2 are very common in a scenario with scarce resources. Properties 3-4 relates to realistic situations, allowing teams with different working schedules as well as different characteristics (size, competences, experiences). These properties permit to better express the competition (sometimes from different processes or organizations) for resources. At last, property 5 serves as a way to model adversities, e.g. setup times, urban mobility or human factors.

As far we could search the literature, the proposed problem has not been investigated yet. However, it can be related to the class of flow capturing problems (FCP). These problems are inspired as alternatives to the Maximal Covering Location Problem (MCLP), and in the same way are NP-Hard. The MCLP aims at locating a prefixed number of facilities in order to cover a maximal number of demand points. In an FCP, it is given a network (set of nodes and arcs) flow structure together with origin and destination (OD) pairs, each one defining a path and an amount of flow. The big difference is that, when a facility is located in a node, all the incident flow is captured, implying that the presence of a facility affects another facility flow capture.

[1] a team may include police officers, health professionals, road officers etc.

https://proceedings.science/p/157446?lang=pt-br

The class of FCP is defined in [Hodgson, 1989] and [Berman et al., 1992]. Depending on the type of facilities considered, one can find different works on our problem context focus on interception of flow, such as vehicle inspecting stations ([Mirchandani et al., 1995], [Hodgson et al., 1996], [Gendreau et al., 2000]), weigth-in-motion systems ([AlGadhi, 2002], [Selmic et al., 2011]) and traffic count points ([Yang e Zhou, 1998]). [Zeng et al., 2010] proposed a generic model for more 30 different FCP applications. In [Arslan et al., 2018], a classification is proposed for FCP variants based on drivers behaviour, namely: the driver is neutral to the facilities locations (most of the works above), the driver may change its path by a small amount (deviation tolerance [Kim e Kuby, 2012]), and the driver desires to evade the facilities with restriction in the number of paths that he/she is willing to take ([Marković et al., 2015]). The first work with a multi-period approach to FCP is due to Chung e Kwon [2015] for a charging stations (eletric cars) allocation problem. Later, we see a multi-period stochastic context to evasive FCP in [Marković et al., 2017]. In some applications, the network information might need a more complex model as a gravity spatial interaction models [Hodgson, 1989], given scarce or incomplete data, tough such models seem to better reflect real information in a real scenario it increases the overall complexity of the model and might turn the problem computationally intractable.

The approach that we propose has a very generic input and throws the specifics of the data to the user side (management side or decision maker). A decision maker can represent distinct resources (teams) and their conflicts (time schedules and compatibility). At the same time, we introduce more complexity by tackling properties 3, 4 and 5, which makes our problem different from a usual scheduling problem.

The remaining of this paper is organized as follows. In Section 2 we show how we generate and process the input data, we remark that there is no benchmark available for the proposed problem and the creation of the instances is a interesting challenge and a contribution itself, since once must combine COVID data with traffic information. In Section 3 we propose a Binary Integer Program (BIP) to solve the proposed problem and, in Section 4, we report on some of the computational experiments. We show that our approach can optimally solve a large amount of the developed instances in few seconds (less than 7s). We provide some insights into which structures make instances harder and provide the benchmark with the instances provide to encourage the search in this problem.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001. This work has been supported by the following Brazilian research agencies with the grants: FUNCAP PS1-0186-00155.01.00/21, FUNCAP PNE 0112.00061.01.00-16, CNPQ Proc. 309315/2019-0.

2. Dataset

As said before, one of the contributions we provide is the data set relating traffic flow information and COVID related data. For the traffic flow information we used New York City traffic data and, for the COVID related data we used COVID19 open data provided also by New York City. In what follows we explain how the data was found and summarize the process employed to produce the final dataset, the final product can be found in [de Abreu Figueredo].

2.1. Gathering Data

We acquired the data through New York City (NYC) open data platform [ope]. Besides being free it offers support to multiple database formats (e.q. csv, xlsx, xml etc).

We make use of the traffic volume data from 2014 to 2019 [tra], as provided by the Department of Transport (DOT) of NYC. This dataset has rows based on daily traffic volume observations divided in 24 intervals of one hour per location. Table 1 shows almost all fields and gives a short

description for each one. The field `Segment ID` is related to the id assigned to each segment of the New York City map, given by the LION dataset.

Column	Short Description
ID	Count ID
Segment ID	LION Segment ID.
Roadway Name	Street name
From	Intersecting street name at one end of street
To	Intersecting street name at another end of street
Direction	Compass direction
Date	Date of the traffic count
Interval of Day	Traffic volume at that time

Tabela 1: Short description of the fields presented on NYC Open Data description.

The LION [lio] dataset is provided by the Department of City Planning and contains a single line representation of the streets of NYC as well as additional metadata that can be used to correlate the Traffic Volume Dataset to the Covid19 data.

The Covid-related data, is available in the NYC Department of Health and Mental Hygiene GitHub platform. The repository possesses Covid19 related data since the earlier days of the pandemic [cov]. In particular, we work with the repository `covid-like-illness-by-boro.csv` which keeps track of the number of hospital visits and admissions (per 100,000 people by borough of residence). See Table 2 for a short description of available information.

Variable Name	Definition
Date	Date of visit
Borough	Borough of residence
Admit_X_Y	Rate of hospital admissions among people aged X-Y years
Admit_X	Rate of hospital admissions among people aged X+ years
Admit_All_ages	Rate of hospital admissions among people of all ages
Visit_X_Y	Rate of ED visits among people aged X-Y years
Visit_X	Rate of ED visits among people aged X+ years
Visit_All_ages	Rate of ED visits among people of all ages

Tabela 2: Description `covid-like-illness-by-boro.csv` dataset with rate factor of 100,000 people by borough of residence.

To generate our dataset given the data presented here we use traffic volume data of 2019 and Covid data from 2020. When choosing traffic data, we thought of the most recent year still not affected by movement restrictions due to the pandemic. Regarding Covid data, the option for 2020 is an attempt to get numbers still not largely affected by quarantine and vaccine.

2.2. Reshaping the Data

First, we filter the data from the traffic volume dataset to only 2019 observations. Next, we relate every single row to a Borough with the usage of the LION data.

The reader might notice that the LION2021B dataset is incomplete with relation to the traffic volume data. One way to fix this issue is to observe that the LION2014D dataset is complete

concerning the segments used by the traffic volume. Therefore, we have compared the LION2021B and LION2014D data so as to ensure that we have updated data. We reshaped our LION datasets in order to keep only columns *Street*, *SegmentID* and *Borough* which is computed by the information presented in the columns *LBoro*, and *RBoro*, since *SegmentID* may connect different boroughs or not in that case one must first eliminate duplicates.

Now, we join, by *SegmentID*, the 2019 traffic volume dataset with the LION2014D reshaped dataset. To create our final dataset, we join the previous dataset by date and borough code with the covid dataset. See Table 3 for a description of the fields in the final dataset.

Column	**Short Description**
ID	Count ID
Segment ID	LION Segment ID.
Roadway Name	Street name
From	Intersecting street name at one end of street
To	Intersecting street name at another end of street
Direction	Compass direction
Date	Date of ED visit
Interval of Day	Traffic volume at that time
Borough	Borough Code from 1 (Manhattan) to 5 (Staten Island)
Visit_All_ages	Rate of ED visits by borough of residence
Admit_All_ages	Rate of hospital admissions by borough of residence
locationID	New unique location id by segmentID, Borough Code and Direction grouping

Tabela 3: Final version dataset table.

2.3. Building the Instances

Now we describe how the instances are created from the dataset described before. To generate an instance, we need to define the set I of possible locations, the set T of periods, the set K of teams, the subsets $\bar{I}(k)$ and $T(k)$ of possible locations and working time of each team k as well as the costs (scores) μ_{it} associated with the possible allocations.

The created dataset has a total of 60 different dates, each with a predefined number of locations. In this work, we consider the time horizon of a day ($|T| = 24$) and use all locations that are available on that period to define I. We fix the number of teams to the floor of 30% of the number of locations, i.e. $|K| = \lfloor 0.3|I| \rfloor$. Each team in a given instance has a particular schedule given by a time restriction and a location restriction, whose are defined as follows:

- Possible location restrictions:
  - Full: each team may go to any location;
  - Neig: each team may work in only one Borough.
  - Rand: each team has 50% probability to be compatible with a location.
- Possible time restrictions:
  - Full-time: each team works from 8 am to 12 am and from 1 pm to 6 pm.
  - Stage: each team works from 8 am to 12 am.

https://proceedings.science/p/157446?lang=pt-br

– Night: each team works from 10 pm to 5 am.
– Rand: each team has 50% probability to be able to work in a certain hour.

We can combine one time restriction and one location restriction so as to get 12 kinds of team schedules (3 locations restrictions $\times$ 4 time restrictions). For example, a team schedule can be of the kind "Full" for locations compatibility and of kind "Stage" for time restriction. Here, for a given day in the dataset, we partition the teams among the possible 12 kinds of teams' schedules. Then, an instance is identified by a 12-tuple $q_1 - q_2 - q_3 - q_4 - q_5 - q_6 - q_7 - q_8 - q_9 - q_{10} - q_{11} - q_{12}$ with $\sum_{i=1}^{12} q_i = 12$ representing that there are $\lfloor \frac{q_i}{12} |K| \rfloor$ teams of kind i (the remaining teams are of kind 12, i.e. Rand-Rand schedules). Because of the big number of possible tuples, we restrict the experiments for instances where all teams have only one kind, have two kinds or have three kinds of schedules. More precisely, $q_i \in \{0, 4\}$ or $q_i \in \{0, 6\}$ or $q_i \in \{0, 12\}$, as illustrates the table below:

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 4: Teams partitions with perfect size.

Finally, μ_{it} must be a mean-full metric to represent the hability of the facilities placed in a location to identify covid cases. Hence, the μ_{it} value is defined as the traffic flow (column `Interval of Day` relative to t) $\times$ the rate of visits to health facilities (column `Visit_All_ages`) of location i, hoping that this reflects the possible number of infected people in that location.

3. A Mathematical Programming Model

To model the problem, consider a binary decision variable $x_{k,i}^t \in \{0, 1\}$ to indicate whether ($x_{k,i}^t = 1$) or not ($x_{k,i}^t = 0$) a team $k \in K$ is assigned to location $i \in \bar{I}(k)$ in the time period $t \in \bar{T}(k)$.

Through such set of variables we can present a integer programming model to solve the problem as presented below.

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

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

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

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

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

The objective function $(1a)$ maximizes the sum of values μ_{it} for each location and time in each one placed a team in that specific location at that specific time. Constraints $(1b)$ forces

that a team can be in at most one location at any period, thus ensuring (*property 1*). Constraints $(1c)$ guarantee that at most one team can be at one location i at a given time t, thus ensuring (*property 2*). Constraints $(1d)$ forces that a team cannot be allocated to distinct locations within the relocation time, thus ensuring (*property 5*). Finally, Constraints $(1e)$ define the variables as binary and properties 3 and 4 are already ensured by the variables definitions.

4. Computational Experiments

In this section, we report on some of the computational results. Our computational environment uses Cplex 20.1.0 solver and a implementation on JULIA 1.6.3 JuMP 0.21.10 of (BIP1) model. The tests were carried out on an Intel Core i5-7400(6MB Cache, up to 3.50 GHz clock speed turbo mode) 4 Threads, (2x8)16GB HyperX Fury DDR4 RAM 2400M Hz and 1TB standard Barracuda HD. Because of the instances size, we only execute instances with less than 200 locations. On the covid database, only 51 days are valid for this restriction. In total, we have 15198 instances.

First, we run the model on all these instances but with a very limited execution time – 2 minutes. That way we can validate our model and get some intuition on which partitions lead to harder instances. Those will then be executed with a longer time limit – 1 hour.

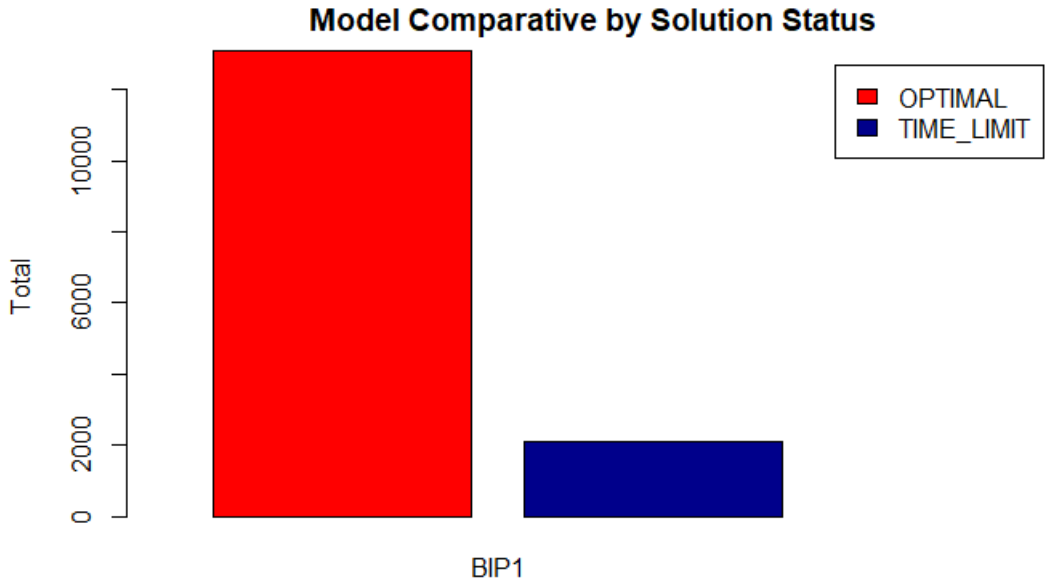


Figura 1: Graphical comparison between the number of instances that achieved the optimal value versus the ones that stops by time limit.

Figure 1 presents the results for the first experiment. It shows that, even restricted to a small time limit, the model can solve a reasonable amount of instances optimally. To be exact, it optimally solves 13,103 instances compared to 2095 instances that hit the time limit. To find a set of hard instances, we selected the instances optimally solved, grouped by partition, and calculated the mean (Mean), standard deviation, minimum (Min) and Maximum value (Max) of the computing time (in seconds).

Table 5 presents the 10 partitions with the greatest Mean of solution times. We can see really high values of standard deviation but with low means, because our instances have a different

	partition	Mean	Std	Min	Max
1	0-0-0-4-0-4-0-0-0-4-0-0	28.64	39.05	0.00	110.19
2	0-4-0-4-0-4-0-0-0-0-0-0	25.81	40.12	0.00	117.96
3	0-4-0-4-0-0-0-0-0-4-0-0	24.07	40.42	0.00	110.24
4	0-0-0-4-0-0-4-0-0-4-0-0	23.26	39.43	0.00	118.42
5	0-0-0-4-0-0-4-0-0-0-4-0	23.01	34.63	0.00	110.69
6	4-4-0-0-4-0-0-0-0-0-0-0	23.00	38.31	0.00	118.92
7	4-0-0-0-4-0-0-0-0-4-0-0	19.05	32.91	0.00	98.41
8	4-0-4-0-4-0-0-0-0-0-0-0	18.70	30.59	0.00	96.04
9	0-0-0-4-4-4-0-0-0-0-0-0	18.48	32.46	0.00	99.34
10	0-4-0-4-4-0-0-0-0-0-0-0	18.00	36.89	0.00	119.50
11	All	6.06	16.42	0.00	119.57

Tabela 5: Computing times for optimally solved instances.

amount of locations by partition. In fact, the average time to solve an instance is about 6.06 seconds, if we look only to instances with an optimal solution. A curious issue is that the IBM Cplex uses a proprietary method known as dynamic search in all the instances tested. It seems that the dynamic search algorithm can really exploit the structure of our problem with heuristics and cuts to solve most of the BIP's in the root of the Branch&Cut tree.

	partition	Mean	Std	Min	Max
1	0-0-0-12-0-0-0-0-0-0-0-0	7.06	18.86	0.07	59.71
2	0-4-0-0-0-0-0-0-4-4-0-0	0.26	0.05	0.22	0.29
3	4-4-4-0-0-0-0-0-0-0-0-0	0.25	0.01	0.25	0.26
4	0-4-0-0-0-4-0-0-0-4-0-0	0.25	0.08	0.19	0.31
5	4-4-0-0-0-0-0-0-0-0-4-0	0.25	0.01	0.24	0.26
6	4-0-4-0-0-4-0-0-0-0-0-0	0.24	0.02	0.23	0.26
7	0-4-4-0-0-0-0-0-0-4-0-0	0.23	0.02	0.21	0.24
8	0-0-4-0-4-0-0-0-0-0-4-0	0.22	0.00	0.22	0.22
9	0-0-6-0-0-0-0-0-6-0-0-0	0.22	0.02	0.21	0.23
10	0-0-4-0-0-0-0-0-0-4-4-0	0.21	0.01	0.21	0.22
11	All	0.12	2.16	0.00	59.71

Tabela 6: Relative Gap for instances reaching time limit by partition

At this moment we start focusing on partitions with teams of the fourth kind (the periods that teams can work are randomized) to find harder instances. By looking at Relative Gaps (Cplex output gap) by partition (Table 6) we found that partition "0-0-0-12-0-0-0-0-0-0-0-0" has instances with really big Gaps (mean of 7%) when compared with the entire set of instances reaching time limit that have an average relative gap of 0.12%.

If we look at the number instances with no exact solution by partition (Table 7) then we find that our initial guess may be really good. So we start to execute the solver on all instances with partition "0-0-0-12-0-0-0-0-0-0-0-0" with time limit of 1 hour. The new results show that the 25 instances that hit the time limit of 2 minutes could not still be solved to optimality, even allowing much more execution time. However, the relative gaps are now considerably smaller (Table 8). It is worth noting that the 26 instances that could be solved to optimality take about

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

Tabela 7: Number of instances reaching time limit by partition.

0.01 seconds. Finally, we draw attention to the fact that our formulation yields small gaps but, in some real applications, the manager might be interested in close even this tiny gaps once they might translate in large economical or/and social losses.

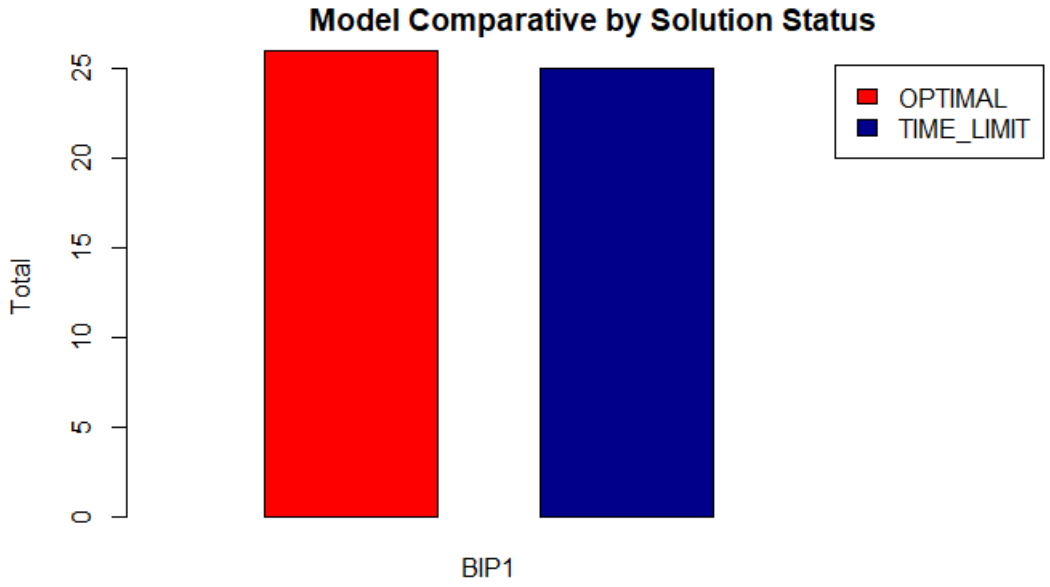


Figura 2: Number of instances “0-0-0-12-0-0-0-0-0-0-0” by status of the solution within 1 hour.

	Parameter	Mean	Std	Min	Max
1	Solution Time	0.01	0.01	0.00	0.03
2	Relative Gap	0.02	0.01	0.01	0.03

Tabela 8: Mean results to OPTIMAL(Solution Time) and TIME_LIMIT(Gap) solutions status by parameter.

5. Conclusions

In this work, we introduced a problem aiming at multi-period allocation of teams to establish mobile checkpoints with emphasis in combating covid 19. We showed a simple way to relate covid 19 information with geographic location and traffic volume real data. As a result, we defined a large dataset of instances to our problem based on real data. Our BIP model were able to provide solutions to 15918 instances (13103 optimal solutions) with a time limit of 2 minutes and relative gaps of 0.12%, what is really important in disaster response scenarios. Our studies show that the introduction of pseudo-randomness to the teams' time schedules make it harder to solve the instances, even when we increase time limits (1 hour).

It is very important to give emphasis that this model can be applied in other contexts, future works in a more generic model would be a good line of research. We so could study the use of metaheuristics (Evolutive Algorithms) to solve our harder cases in viable times.

Referências

Lion single line street base map. https://www1.nyc.gov/site/planning/data-maps/open-data/dwn-lion.page. (accessed: 02.07.2021).

Nyc coronavirus disease 2019 (covid-19) data. https://github.com/nychealth/coronavirus-data/. (accessed: 02.07.2021).

Nyc open data - data. https://opendata.cityofnewyork.us/data/. (accessed: 02.07.2021).

Traffic volume counts (2014-2020). https://data.cityofnewyork.us/Transportation/Traffic-Volume-Counts-2014-2020-/ertz-hr4r/. (accessed: 10.02.2022).

AlGadhi, S. A. (2002). Optimizing truck weigh stationsâ locations on the highway network of saudi arabia. *Journal of King Saud University - Engineering Sciences*, 14(2):147–163. ISSN 1018-3639.

Arslan, O., Jabali, O., e Laporte, G. (2018). Exact solution of the evasive flow capturing problem. *Operations Research*, 66(6):1625–1640.

Berman, O., Larson, R. C., e Fouska, N. (1992). Optimal location of discretionary service facilities. *Transportation Science*, 26:201–211. ISSN 00411655.

Chung, S. H. e Kwon, C. (2015). Multi-period planning for electric car charging station locations: A case of korean expressways. *European Journal of Operational Research*, 242:677–687. ISSN 03772217.

de Abreu Figueredo, P. J. A dataset merging new york city covid 19 and traffic volume information. https://github.com/Huebr/traffic_volume. (accessed: 14.09.2022).

Gendreau, M., Laporte, G., e Parent, I. (2000). Heuristics for the location of inspection stations on a network. *Naval Research Logistics (NRL)*, 47(4):287–303.

Guglielmi, G. (2021). Rapid coronavirus tests: a guide for the perplexed. https://www.nature.com/articles/d41586-021-00332-4. acessed: 2022-01-26.

https://proceedings.science/p/157446?lang=pt-br

Hodgson, M. J. (1989). A flow-capturing location-allocation model. *Geographical Analysis*, 22: 270–279. ISSN 00167363.

Hodgson, M. J., Rosing, K. E., e Zhang, J. (1996). Locating vehicle inspection stations to protect a transportation network. *Geographical Analysis*, 28(4):299–314.

Kim, J.-G. e Kuby, M. (2012). The deviation-flow refueling location model for optimizing a network of refueling stations. *International Journal of Hydrogen Energy*, 37(6):5406–5420. ISSN 0360-3199. Optimization Approaches to Hydrogen Logistics.

Marković, N., Ryzhov, I. O., e Schonfeld, P. (2015). Evasive flow capture: Optimal location of weigh-in-motion systems, tollbooths, and security checkpoints. *Networks*, 65(1):22–42.

Marković, N., Ryzhov, I. O., e Schonfeld, P. (2017). Evasive flow capture: A multi-period stochastic facility location problem with independent demand. *European Journal of Operational Research*, 257:687–703. ISSN 03772217.

Mirchandani, P. B., Rebello, R., e Agnetis, A. (1995). The inspection station location problem in hazardous material transportation: Some heuristics and bounds. *INFOR: Information Systems and Operational Research*, 33(2):100–113.

Selmic, M., Bešinović, N., e Teodorović, D. (2011). Locating weigh-in-motion checkpoints in traffic networks using genetic algorithm. *E-Soc J*, 2:55–66.

Wright, W. F. e Mackowiak, P. A. (2020). Why Temperature Screening for Coronavirus Disease 2019 With Noncontact Infrared Thermometers Does Not Work. *Open Forum Infectious Diseases*, 8(1). ISSN 2328-8957.

Yang, H. e Zhou, J. T. (1998). Optimal traffic counting locations for originâdestination matrix estimation. *Transportation Research Part B-methodological*, 32:109–126.

Zeng, W., Castillo, I., e Hodgson, M. J. (2010). A generalized model for locating facilities on a network with flow-based demand. *Networks and Spatial Economics*, 10. ISSN 1566113X.