Operations Optimization Assignment

A MILP model for the repatriation scheduling problem

Group 26

TUDelft

Operations Optimization Assignment

A MILP model for the repatriation scheduling problem

by

Group 26

Student Name	Student Number
José Cunha	5216087
Pablo Garcia	5270944
Stijn Koelemaij	5089344

Tutors: Alessandro Bombelli
Faculty: Faculty of Aerospace Engineering, Delft

TUDelft

Contents

Introduction

This report describes the implementation of a Gurobi solving algorithm for a mixed-integer linear programming problem. The problem was chosen based on the 2022 article ”A Mixed Integer Linear Programming Model and a Basic Variable Neighborhood Search Algorithm for the Repatriation Scheduling Problem” [1]. A written description of the problem, along with a description of the variables and constraints is given in Chapter 1. Verification of the constraints and the algorithm as a whole is reported in Chapter 2, as well as a comparison to the results from the paper. Before the conclusion, a thorough sensitivity analysis is given in Chapter 3.

1 Problem Description and Implementation

The article ”A mixed integer linear programming model and a basic variable neighbourhood search algorithm for the repatriation scheduling problem” [1] explains the challenges the COVID-19 pandemic posed when needing to repatriate people. With countries closed and mandatory quarantines being implemented, countries had to determine the most efficient way of bringing their citizens back home. The paper introduces a Repatriation Scheduling Problem (RSP) as an optimisation problem, this can essentially be considered a Capacitated Vehicle Routing Problem. The variables and symbols used in this model are described in Table 1.1 below.

Table 1.1: *Notation of the model.*

Group	Symbol	Definition
Indices and sets	i	Index of set of cities (1, 2, ..., m)
	j	Index of set of priority groups (1, 2, ..., n)
	k	Index of set of airplanes (1, 2, ..., u)
	M	Set of cities
	N	Set of priority groups
	U	Set of airplanes
Problem inputs	C_k	Capacity of airplane $k \in U$
	P_{ij}	Number of individuals in city $i \in M$ of group $j \in N$
	P_{ij}^{cum}	Cumulative number of individuals per group $j \in N$ and groups with priority higher than $j \in N$, in city $i \in M$
	ω_j	Importance weight/measure given to repatriating individuals in group $j \in N$
	QC	Quarantine locations capacity
Model variables	x_{ik}	Binary variable that takes value 1 if airplane $k \in U$ is assigned to city $i \in M$, and 0 otherwise
	L_i	Number of repatriated individuals from city $i \in M$
	$\hat{P}_{ij}^{cum}$	Integer variable showing cumulative number of individuals per group $j \in N$ after repatriating L_i individuals from city $i \in M$
	R_{ij}	Number of individuals from group $j \in N$ repatriated from city $i \in M$
	α_{ij}	Auxiliary binary variable that takes value 1 if number of individuals repatriated from city $i \in M$ and group $j \in N$ exceeds P_{ij}^{cum} (i.e. $P_{ij}^{cum} \leq L_i$), otherwise 0
	ϵ_i	Auxiliary binary varible that takes value 1 if number of individuals in city $i \in M$ exceeds the sum of capacities of airplanes assigned t the same city, otherwise 0
	V	'Big M' value, set as 10^6

The problem can be described by the symbols in Table 1.1: there are M cities with P citizens divided into N priority groups, and only U number of aircraft available to bring them home.

1.1. Model Description

The objective, described by Equation 1.1, is to maximise Z, which can be described as the sum of all the individuals repatriated R_{ij} multiplied by their respective priority group importance weight ω_j

$$(\text{max})Z = \sum_{i\in M}\sum_{j\in N}\omega_j \cdot R_{ij} \tag{1.1}$$

Additionally, there are 16 equations describing the constraints of the problem given below.

$$P^{cum}_{ij} = \sum_{t=0}^{t=j} P_{it}, \quad \forall i \in M, \forall j \in N \tag{1.2}$$

$$P^{cum}_{in} - \sum_{k\in U} C_k \cdot x_{ik} \geq (\epsilon_i - 1)\cdot V, \quad \forall i \in M \tag{1.3}$$

$$P^{cum}_{in} - \sum_{k\in U} C_k \cdot x_{ik} \leq \epsilon_i \cdot V, \quad \forall i \in M \tag{1.4}$$

$$L_i \leq \sum_{k\in U} C_k \cdot x_{ik} + (1-\epsilon_i)\cdot V, \quad \forall i \in M \tag{1.5}$$

$$L_i \geq \sum_{k\in U} C_k \cdot x_{ik} + (\epsilon_i - 1)\cdot V, \quad \forall i \in M \tag{1.6}$$

$$L_i \leq P^{cum}_{in} + \epsilon_i \cdot V, \quad \forall i \in M \tag{1.7}$$

$$L_i \geq P^{cum}_{in} - \epsilon_i \cdot V, \quad \forall i \in M \tag{1.8}$$

$$L_i - P^{cum}_{ij} \geq (\alpha_{ij} - 1)\cdot V, \quad \forall i \in M, \forall j \in N \tag{1.9}$$

$$L_i - P^{cum}_{ij} \leq \alpha_{ij} \cdot V, \quad \forall i \in M, \forall j \in N \tag{1.10}$$

$$\hat{P}^{cum}_{ij} \leq (1 - \alpha_{ij})\cdot V, \quad \forall i \in M, \forall j \in N \tag{1.11}$$

$$\hat{P}^{cum}_{ij} \geq (\alpha_{ij} - 1)\cdot V, \quad \forall i \in M, \forall j \in N \tag{1.12}$$

$$\hat{P}^{cum}_{ij} \leq P^{cum}_{ij} - L_i + \alpha_{ij}\cdot V, \quad \forall i \in M, \forall j \in N \tag{1.13}$$

$$\hat{P}^{cum}_{ij} \geq P^{cum}_{ij} - L_i - \alpha_{ij}\cdot V, \quad \forall i \in M, \forall j \in N \tag{1.14}$$

$$R_{ij} = \begin{cases} P^{cum}_{ij} - \hat{P}^{cum}_{ij} \quad \forall i \in M & j = 1 \\ P^{cum}_{ij} - \hat{P}^{cum}_{ij} - P^{cum}_{ij-1} + \hat{P}^{cum}_{ij-1} \quad \forall i \in M & j > 1 \end{cases} \tag{1.15}$$

$$\sum_{i\in M} L_i \leq QC \tag{1.16}$$

$$\sum_{i\in M} x_{ik} \leq 1, \quad \forall k \in U \tag{1.17}$$

Equation 1.2 states that the new variable P_{ij}^{cum}, which represents the cumulative number of people residing in a city i from priority group j or below, is equal to the sum of the number of passengers in a city from each priority group from 1 to an arbitrary t. E.g. $P_{1,3}^{cum} = P_{1,1} + P_{1,2} + P_{1,3}$.

The first constraints in Equations 1.3-1.8 all make use of auxiliary binary variable ϵ_i. This variable takes a value of 1 if there are more passengers than plane capacity in a city i, and 0 otherwise.

In Equation 1.3 and Equation 1.4 P_{in}^{cum} is the total cumulative number of passengers in city i, C_k is the aircraft capacity for plane k and x_{ik} is 1 or 0 depending whether plane k is assigned to city i or not. Additionally, ϵ_i is a binary variable and V is just a large number. Equation 1.3 states that the cumulative sum of passengers in a city i minus the capacity of the planes in city i, should be greater than or equal to 0 if there are more passengers in a city than the aircraft capacity $\epsilon_i = 1$, or should be greater than or equal to $-V$ if $\epsilon_i = 0$. This equation sets a lower bound for the constraint. Equation 1.4 is formulated similarly, and sets an upper bound for this constraint. It states that the cumulative number of passengers in city i minus the aircraft capacity of city i should be less than or equal to 0 if there are fewer passengers in a city than the aircraft capacity ($\epsilon_i = 0$), and less than or equal to V otherwise.

In Equation 1.5 and Equation 1.6 the number of repatriated citizens L_i is constrained. If ϵ_i is 0, the number of repatriated citizens in city i must be less than or equal to the aircraft capacity in the city i plus V, but must be greater than or equal to the aircraft capacity of city i minus V.This makes sense as ϵ_i is 0 when there's enough capacity to relocate all citizens of city i. If ϵ_i is 1, i.e. there's not enough capacity to relocate all citizens, then the number of repatriated citizens will always be equal to the aircraft capacity of the city.

The next two constraints are very straightforward. If $\epsilon_i = 0$, the number of repatriated citizens from city i is equal to the cumulative sum of passengers in city i in both Equation 1.7 and Equation 1.8. Else, if there's not enough plane capacity to relocate all citizens ($\epsilon_i = 1$), the number of repatriated citizens L_i is less than a large number V plus the sum of passengers in city i, P_{in}^{cum}, and greater than $-V$ plus the sum of passengers in city i, thus setting an upper and lower bound for the constraint.

Equations 1.9 - 1.14 all use the auxiliary variable α_{ij}, which has a value of 1 if the number of repatriated individuals in city i exceeds the number of citizens in need of repatriation in city i and priority group j, P_{ij}^{cum}, and 0 otherwise.

Equation 1.9 states that the number of repatriated citizens L_i minus the cumulative sum of passengers P_{ij}^{cum} must be greater than or equal to $-V$ if α_{ij} is 0, and greater than or equal to 0 if α_{ij} is 1. This creates a lower bound which can be complimented by Equation 1.10. Equation 1.10 states that if α_{ij} is 1, the total number of repatriated citizens minus the number of citizens in need of repatriation is less than V, and if α_{ij} is 0, then this number is less than 0.

Equation 1.11 and Equation 1.12 are constraints for the cumulative number of individuals in city i and priority group j after repatriation has already taken place, $\hat{P}_{ij}^{cum}$. If α_{ij} is 1, meaning, there are more citizens repatriated than in need of repatriation, then the leftover number of citizens equals 0 for both constraints. Additionally, if this is not the case and $\alpha_{ij} = 0$, then Equation 1.11 serves as an upper bound for the number of citizens left behind after the repatriation, and Equation 1.12 serves as a lower bound. The next two constraints also follow a similar principle, if α_{ij} is 0, both Equation 1.13 and Equation 1.14 state that the leftover passengers equals the total number of passengers that needed repatriation minus the number of repatriated passengers. Additionally, if α_{ij} is 1, Equation 1.13 is once again an upper bound and Equation 1.14 is a lower bound for the number of leftover passengers.

Equation 1.15 deals with R_{ij}, the number of passengers in group j repatriated from city i. If j is one, then the value of is R_{ij} is equal to the cumulative number of passengers P_{ij}^{cum} minus the number of leftover passengers $\hat{P}_{ij}^{cum}$. If j is greater than one we perform this same operation but subtract from it $P^{cum}_{ij-1} - \hat{P}^{cum}_{ij-1}$.

Equation 1.16 states that the sum of repatriated citizens from all cities must be less than the capacity of quarantine locations QC. Finally, Equation 1.17 ensures that the sum of x_{ik} for all values of i does not exceed one. x_{ik} represents an airplane k assigned to a city i and has a value of 0 if this is not the case or 1 if it is. The sum of x_{ik} for all values of i should always have a value of 1, as each aircraft is assigned to only one city.

It is important to note that this is the general formulation for the constraints, but the number of constraints is subject to the number of cities M, the number of priority groups N and the number of aircrafts U. With these constraints clearly defined, it is now possible to formulate the model in a solver and find the optimal solution. The implementation of this model can be found in the codebase at github.com, described in thew module *model.py*.

1.2. Data generation algorithm

In the paper by Al-Shihabi and Mladenović [1], alongside the MILP model, a data generation algorithm is provided in order to create relatively representative data to test the model. This algorithm provides methods to create data for the problem inputs, namely the number of cities, number of priority groups, number of aircraft, capacity of the aircraft, importance values of priority groups, number of individuals per group and city and quarantine capacity. The algorithm takes the following steps:

1. Choose number of cities.
2. Choose number of aircraft.
3. According to number of aircraft, randomly select each airplane capacity from 200, 250, 300 or 350 seats.
4. Choose number of priority groups.
5. Assign priority values according to group number (i.e. for three groups $\omega = [3, 2, 1]$)
6. Uniformly generate the values of P_{ij} to be between 0.5 and 1.5 of the average aircraft capacity.
7. Determine the number of quarantine locations by choosing the fleet capacity to quarantine locations ratio (FQ), therefore limiting the number of repatriated individuals by either the fleet capacity or the number of quarantine locations.

The data generated from this algorithm will be later used to test the implementation of the model in Chapter 2, as well as to analyse how sensitive the model is to changes in its variables, in Chapter 3.

2 Verification & Validation

This chapter shows how the verification was performed on the code. This was in the following two ways: firstly, each constraint was analyzed to see if they were functioning properly in Section 2.1. Secondly, a number of datasets were tested of which the outcome could easily be predicted. Section 2.2 shows the datasets used and the code's compliance.

2.1. Constraint Verification

To test that all the constraints were being satisfied by the final results of the model, a simple test case was used (provided by Al-Shihabi and Mladenović [1]), described in Table 2.1.

Table 2.1: *Illustrative small test data given by [1].*

City	**Group**			
	A	B	C	D
1	200	300	100	100
2	300	100	200	0
3	50	100	350	150
4	250	150	200	200
5	500	0	150	50
Priorities	10	6	4	2
U = 2				
QC = 2000				
C_k = 300				

The model was then run on this illustrative dataset, and each of the 17 constraints was verified. The results of these unit tests are tabulated in Table 2.1. All constraints were verified to be met by the results, and hence the implementation of the constraints can be considered to be correct. All constraint tests are described in the module *test_constraints.py* of the codebase (github.com).

Table 2.2: *Constraint verification unit tests.*

Constraint number	**Pass/Fail**
(1)	**PASS**
(2)	**PASS**
(3)	**PASS**
(4)	**PASS**
(5)	**PASS**
(6)	**PASS**
(7)	**PASS**
(8)	**PASS**
(9)	**PASS**
(10)	**PASS**

(11)	**PASS**
(12)	**PASS**
(13)	**PASS**
(14)	**PASS**
(15)	**PASS**
(16)	**PASS**
(17)	**PASS**

2.2. Outlier Tests

The aim of the first test is to see whether the algorithm properly takes the priority values into account. 2 cities were made, 1 including people from each priority group and 1 with people from only the lowest group. The number of planes was set to 2, with a capacity of 300 each, while the amount of quarantine places was set to 4000, simply an arbitrary value exceeding the planes' total capacity.

Table 2.3: *Dataset 1* ($u = 2, C_k = 300, QC = 4000$)

	Group 1	Group 2	Group 3	Group 4
City 1	0	0	0	2000
City 2	100	200	300	400
weighting	10	6	4	2
Expected outcome:	100x10	+ 200x6	+ 300x4	+ 0x2 =3400
Pass/Fail:	**PASS**			

As expected, the higher priority groups are indeed chosen to repatriate first. In the second test, a scenario is created to test whether each plane indeed only visits one city, e.g. the objective function would be higher if a plane would take citizens from multiple cities.

Table 2.4: *Dataset 2* ($u = 1, C_k = 500, QC = 4000$)

	Group 1	Group 2	Group 3	Group 4
City 1	100	100	100	100
City 2	100	0	0	0
weighting	10	6	4	2
Expected outcome:	100x10	+ 100x6	+ 100x4	+ 100x2 =2200
Pass/Fail:	**PASS**			

Some more trivial tests should also be performed, such as setting the plane's capacities or quarantine capacity to zero. The description and results of these are shown in the tables below, starting with an airplane capacity of 0.

Table 2.5: *Dataset 3 (airplane capacity = 0)* ($u = 2, QC = 4000$)

	Group 1	Group 2	Group 3	Group 4
City 1	100	100	100	100
City 2	300	200	100	0
weighting	10	6	4	2
Expected outcome:	0 ($C_k = 0$)			
Pass/Fail:	**PASS**			

The following dataset tests what happens when the quarantine locations are set to zero:

Table 2.6: *Dataset 4 (quarantine locations = 0) $(u = 2, C_k = 300)$*

	Group 1	Group 2	Group 3	Group 4
City 1	100	100	100	100
City 2	300	200	100	0
weighting	10	6	4	2
Expected outcome:	0 ($QC = 0$)			
Pass/Fail:	**PASS**			

Setting the weighting factor of a group to zero:

Table 2.7: *Dataset 5 $(u = 2, C_k = 300, QC = 4000)$*

	Group 1	Group 2	Group 3	Group 4	
City 1	100	100	200	100	
City 2	100	0	0	0	
weighting	10	**0**	4	2	
Expected outcome:	200x10	+ 0x0	+ 200x4	+ 0x2	=2800
Pass/Fail:	**FAIL**	(outcome = 2400)			

As can be seen from the table, the test indicated a Fail. This can be explained by analyzing the way that P_{ij}^{cum} is defined in the program. The problem is not that ω_2 is set to zero, but the order of priority groups is wrong. For The algorithm to work correctly, the weighting factors must be ordered from high to low. This is not the case for dataset 5, and it can be seen that the algorithm assigns the 100 passengers of group 2 in city 1 to be repatriated, even though the weight is set to zero. Then the 100 capacity left in the plane is filled with 100 group 3 passengers, to finally end up at 2400 for the objective function value. After performing other similar tests it was clear that keeping the priorities in order is important for the algorithm to properly function.
Note that for some specific small datasets the true optimum solution can still be found, but for most sets the programme will find a sub-optimal solution, especially with bigger datasets.

2.3. Validation

From the article by Al-Shihabi and Mladenović [1], a small dataset was provided (described in Table 2.1), as well as the results for a series of different model configurations, which is shown in Table 2.8. Using the same model configurations, and using the implementation described in this assignment, the model was run (with results seen in Table 2.9) and the differences in objective function value (Z) and aircraft assignment were compared. For both Table 2.8 and Table 2.9, the results in brackets can be read as the number of aircraft followed the number of people being repatriated.

Table 2.8: *Validation set for solutions of the illustrative example for different number of airplanes and quarantine capacities*

Aircraft	QC	Cities					Z
		1	2	3	4	5	
2	2000	0	(1, 300)	0	0	(1, 300)	6000
4	2, 000	(1, 300)	(1, 300)	0	(1, 300)	(1, 300)	11400
6	2000	(1, 300)	(1, 300)	(1, 300)	(1, 300)	(2, 600)	15500
8	2000	(1, 300)	(1, 300)	(1, 300)	(1, 300)	(2, 600)	15500
8	2500	(2, 600)	(1, 300)	(1, 300)	(1, 300)	(2, 600)	18500
10	2500	(2, 600)	(2, 600)	(1, 300)	(1, 300)	(2, 600)	18500
10	3000	(2, 600)	(2, 600)	(2, 600)	(2, 600)	(2, 600)	20900
12	3000	(2, 600)	(2, 600)	(2, 600)	(2, 600)	(2, 600)	20900
12	3500	(2, 600)	(2, 600)	(2, 600)	(3, 800)	(3, 700)	21600
12	4000	(2, 600)	(2, 600)	(2, 600)	(3, 800)	(3, 700)	21600
14	4000	(3, 700)	(2, 600)	(3, 650)	(3, 800)	(3, 700)	21900

Table 2.9: *Solutions of the illustrative example for different number of airplanes and quarantine capacities using own implementation.*

Aircraft	QC	Cities					Z
		1	2	3	4	5	
2	2000	0	(1, 300)	0	0	(1, 300)	6000
4	2000	(1, 300)	(1, 300)	0	(1, 300)	(1, 300)	11400
6	2000	(1, 300)	(1, 300)	(1, 300)	(1, 300)	(2, 600)	15500
8	2000	(3, 700)	(1, 300)	0	(1, 300)	(3, 700)	15900
8	2500	(2, 600)	(1, 300)	(1, 300)	(2, 600)	(2, 600)	18500
10	2500	(2, 600)	(2, 600)	(1, 300)	(1, 300)	(3, 700)	18800
10	3000	(2, 600)	(2, 600)	(2, 600)	(2, 600)	(2, 600)	20900
12	3000	(2, 600)	(2, 600)	(2, 600)	(2, 600)	(2, 600)	20900
12	3500	(2, 600)	(2, 600)	(2, 600)	(3, 800)	(3, 700)	21600
12	4000	(2, 600)	(2, 600)	(2, 600)	(3, 800)	(3, 700)	21600
14	4000	(3, 700)	(2, 600)	(3, 650)	(3, 800)	(3, 700)	21900

Comparing Table 2.8 and Table 2.9, one can observe that for most values of the number of aircraft and number of quarantine locations tested, the results of the validation set match the ones from this assignment's implementation. However, there are two instances where results vary, namely for 8 aircraft and 2000 QC, and for 10 aircraft and 2500 QC, where Z is 15900 and 18800 instead of 15500 and 18500 respectively. This shows that a more optimal solution was found compared to the literature [1]. This hints at a possible discrepancy between the solvers or perhaps the solution method used by the validation set, as the constraints in this implementation have been verified and are met, and the solution is feasible.

3 Sensitivity Analysis

To perform a sensitivity analysis, the main model parameters must first be identified. From Table 1.1, the problem inputs that can be varied for sensitivity analysis are the importance measure given to each priority group (ω_j), the capacity of each airplane (C_k), the number of airplanes (u), the number of quarantine locations (QC), and the population distribution between the different cities and priority groups (P_{ij}). For the analysis, the initial small test case described in Table 2.1 and by [1] will be used, where the different problem parameters will be varied, and the sensitivity of the final result shall be analysed.

Sensitivity of importance measure ω_j

To test the sensitivity of the importance measure, a new dataset is generated, using the data generation algorithm described in Chapter 1. This consists of a set of five cities with four priority groups, where eight aircraft are available for repatriation. Initially, the importance measure has a distribution of $\omega = [4, 3, 2, 1]$. For simplified notation, the results of the plane scheduling (mostly seen in the variable x_{ik}) will be shown as a list, where each element corresponds to the number of planes allocated to a certain city (corresponding to the list index). For example, if 2 planes are scheduled for city 1 and 3 planes for city 2, it shall be shown as $x = [2, 3]$. It would not be interesting to show the changes in objective value, as this is directly linked to the value of the importance measure, hence a change in objective value may not be representative of a different scheduling result.
If all importance values are multiplied by a constant (i.e. all values doubled or halved) then this will have the same effect on the objective value, however, this will not change the the allocation of planes to cities.

$$\omega = [4, 3, 2, 1] \rightarrow x = [1, 2, 2, 1, 2]$$

$$\omega = [8, 6, 4, 2] \rightarrow x = [1, 2, 2, 1, 2]$$

What may have an effect on the plane scheduling is the ratio between the importance values, even though this depends on the population distribution between the different cities and groups. To analyse this, each importance value was randomly varied with $\Delta\omega \in [-1, 1]$ for many repetitions, resulting in around 88% of combinations with the same airplane scheduling. In this sense, the model can be shown to be relatively insensitive to changes in the importance values, as long as they are placed in order from highest to lowest.

Sensitivity of population distribution

To test the sensitivity of the population distribution (i.e. parameter P in the model), a similar approach to the sensitivity analysis of ω_j can be employed. Again, a test case of five cities, four priority groups and eight aircraft was created using the data generation algorithm, resulting in the scheduling corresponding to

$$x = [1, 2, 2, 1, 2]$$

For each P_{ij}, the number of individuals was randomly varied with a $\Delta P \in [-x, x]$ for many repetitions, where the different ranges of ΔP can be found in Table 3.1. The resulting population distributions were then compared to the results of the test case, and the percentage of cases where there is the same aircraft scheduling as the test case was tabulated in Table 3.1. Considering the results, the model shows quite some insensitivity to a change in number of individuals to be repatriated per city and per group, as a $\Delta P \in [-50, 50]$ resulted in 96% of population distributions having the same aircraft scheduling as the test case and even for large values of ΔP ($\Delta P \in [-100, 100]$, the percentage of results equal to the test case drops to 63%).

Table 3.1: *Percentage of population distributions that have the same aircraft scheduling as the test case*

Δ P	% Similarity
5	$\sim$ 100%
10	$\sim$ 100%
20	$\sim$ 100%
30	99.8%
40	99.8%
50	96%
75	84%
100	63%

Sensitivity of QC

The next factor to analyse for its sensitivity is the number of quarantine locations QC. This parameter can reduce the final value of Z if it is lowered further than the total capacity of all aircraft. For this analysis, the data generator was used, since it can generate larger, more varied and realistic datasets, and thus it is of greater interest to test with the generated data. As described in Chapter 1, QC is determined solely by multiplying the total airplane capacity by a manually specified ratio of fleet capacity to quarantine locations, fq. Small changes in fq were tested on these bigger datasets to observe the influence on Z. The remaining parameters for the data generator were as follows: 50 cities, 60 aircraft with varying C_k analogous to Chapter 1, population per group according to Chapter 1, $\omega_j = 5, 4, 3, 2, 1$ and thus 5 groups. The results of varying fq are shown below:

Table 3.2: *fq sensitivity*

fq	Z	Δfq	Δ Z
1	80276	-	-
1.02	78385	+2%	-2.35%
1.05	79123	+5%	-1.44%
1.10	78416	+10%	-2.32%
1.20	78019	+20%	-2.81%
0.98	77669	-2%	-3.25%
0.95	76882	-5%	-4.23%
0.9	71281	-10%	-11.2%
0.8	65751	-20%	-18.1%

From Table 3.2, it can be seen that increasing fq (creating more quarantine locations relative to total aircraft capacity) does not lead to great changes in the final Z value. On the contrary, decreasing fq leads to a significant decrease in Z. The algorithm seems to be sensitive to decreases in fq, but rather insensitive to an increase.

Sensitivity of C_k

The penultimate factor to consider for the sensitivity analysis is the aircraft capacity. Since Z depends on the aircraft's capacities, it does not make sense to increase C_k and compare the final Z value. To make it more comparable, the 4 options available for C_k were given values such that the average was kept constant. With the initial values having an average of 275, this number was chosen as a constant average for C_k. The remaining data parameters were set identically to Table 3, with fq set at 1. The results are displayed in Table 3.3.

Table 3.3: *C_k sensitivity*

Test Description	C_k options	Z	$\triangle$ Z
Standard run	[200, 250, 300, 350]	79106	-
all C_k identical	[275, 275, 275, 275]	79679	+0.72%
Small changes to C_k	[265, 285, 265, 285]	79283	+0.22%
(see above)	[235, 255, 295, 305]	78501	-0.76%
Larger variation in C_k	[150, 225, 325, 400]	79875	+0.97%
(see above)	[100, 125, 425, 450]	81386	+2.88%
Small, medium and large planes increased chance for medium size	[75, 275, 275, 475]	88921	+12.4%
Only small and large planes	[50, 50, 500, 500]	70662	-10.7%
many large planes, some small planes	[50, 350, 350, 350]	66124	-16.4%
many small planes, some large ones	[100, 100, 100, 800]	85310	+7.84%

It must be noted that the initial values for C_k are not optimized values, simply realistic numbers for a plane's capacity, therefore it does not come as a surprise that many of the tests shown in Table 3.3 led to an increase in Z. It can be seen that having a more uniform C_k distribution does not lead to large changes in Z. A first significant change is seen when there is an increased chance for a medium sized airplanes (with C_k exactly equal to the average of the options), and a smaller amount of smaller and larger aircraft. However, a significant increase is also achieved with only $\approx 25\%$ very large planes and $\approx 75\%$ smaller planes. Most likely the algorithm successfully assigns the large planes to cities with a large number of high-priority citizens to still increase Z. In general, a lightly varying C_k distribution does not lead to large changes in Z, while highly varying C_k's can lead to a significant increase as well as a decrease in Z.

Sensitivity of U

The final factor to be considered is the number of aircraft, shown in the variable U. The sensitivity of this variable is intrinsically tied to the sensitivity of both QC and C_k. This occurs because the scheduling of the model is limited by either the number of quarantine locations QC or the total fleet capacity (determined by U and C_k). For example, in the case where the results are limited by the number of quarantine locations (and assuming that QC is lower than the total population to be repatriated), increasing the number of aircraft would have a very large effect on the final result, as there would always be more individuals to be repatriated. This is until the total fleet capacity meets the quarantine capacity, whereupon the final result would not change. For this case, the model is extremely sensitive to the number of aircraft as more or fewer aircraft would directly impact the number of repatriated individuals.

4 Conclusion

This report aimed to reproduce the mixed integer linear programming problem formulated by the article by Al-Shihabi and Mladenović [1] and analyse its implementation in detail. This article describes a Capacitated Vehicle Routing Problem (CVRP) that has the objective of repatriating citizens in different priority groups to different destinations during the COVID-19 pandemic using a limited number of aircraft.

To reproduce this program, the model was described and implemented. Additionally, verification was necessary to ensure the constraints in the model were all effective and working as expected, and the results of our model were compared to the one described in the article [1]. Finally, a sensitivity analysis was performed to investigate the change in the results to changes in the weight of each priority group, the aeroplane capacity, the number of aeroplanes, the quarantine capacity and the population distribution.

To verify the model, each constraint was tested by using a feasible solution and comparing the left-hand side to the right-hand side of the constraint. All constraints passed the verification tests, showing that they are functioning properly within the model. Additionally, system tests were developed for multiple different scenarios. The first four tests were passed; higher priority citizens are repatriated first, each plane only does one trip, and if the aircraft or quarantine capacity is 0, the final result is 0. The fifth one failed since the problem ensures that all the weights are in descending order, so setting the second weight to 0 will give the second priority group the importance weight ω_3, the third ω_3 and the fourth ω_2. This is a result of the formulation of the model itself but contains a logical error. To avoid errors, when formulating a problem, the weighting values should be inputted from highest to lowest.

The sensitivity analysis tested the sensitivity of the model to changes in the importance ratio, the population ratio, the quarantine capacity, the aeroplane capacity, and the number of planes. The model was shown to be insensitive to changes in ω_i, QC, P and C_k. However, when P is changed randomly by at most ± 100 large changes are observed, however, these are expected. The model is very sensitive to changes in U, as the size of the fleet has a direct impact on the number of repatriated passengers. It is important to note that QC, C_k and U all limit each other, therefore a meaningful sensitivity analysis is limited, as for example increasing quarantine capacity is limited by the number of aeroplanes and aeroplane capacity, therefore the sole effect of quarantine capacity cannot be observed.

To further improve this model it would be useful to change the implementation in such a way that the weights could be altered without being in an ascending order. Additionally, to further improve the applications of this model, the model could be defined in such a way that aircraft could make trips to multiple cities, not just one in case their capacity is not filled in one trip. This could make the repatriation process more efficient and possibly reduce its cost, though introduces more complexity.

Bibliography

[1] Sameh Al-Shihabi and Nenad Mladenović. “A mixed integer linear programming model and a basic variable neighbourhood search algorithm for the repatriation scheduling problem”. In: *Expert Systems with Applications* 198 (2022), p. 116728.