Expert Systems With Applications 198 (2022) 116728

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

A mixed integer linear programming model and a basic variable neighbourhood search algorithm for the repatriation scheduling problem

Sameh Al-Shihabi [a,b,*], Nenad Mladenović [c]

[a] Industrial Engineering and Engineering Management Department, University of Sharjah, PO Box 27272, Sharjah, United Arab Emirates
[b] Industrial Engineering Department, University of Jordan, Amman 11937, Jordan
[c] Department of Industrial System Engineering, Khalifa University, Abu Dhabi, United Arab Emirates

ARTICLE INFO

Keywords:
Repatriation
Scheduling
Variable neighbourhood search
Optimization
Covid-19

ABSTRACT

Commercial flights nearly halted due to the COVID-19 pandemic in the second quarter of 2020. Consequently, several countries have had to schedule repatriation flights to return their citizens stranded in other countries. Flight routes and schedules are known in normal circumstances, and passengers buy seats on these flights; however, the reverse steps happen in repatriation. Passengers express their need to travel, and flights are scheduled to satisfy their requests. The problem behind this flight schedule can be called the repatriation scheduling problem (RSP), in which we need to repatriate citizens from different countries. The objective of the RSP is to return the most vulnerable citizens first. The capacity of available airplanes and quarantine locations limit the number of repatriated citizens. To address this problem, we have developed a mixed-integer linear program (MILP) to model the RSP. Moreover, we suggest a basic variable neighbourhood search (BVNS) algorithm to solve the problem. We test the BVNS algorithm by creating and solving a set of 108 RSP instances and then comparing the BVNS solutions with the exact ones. Despite allocating only 20 s to run the BVNS algorithm compared to eight hours for a commercial exact solver's branch and bound algorithm, the BVNS algorithm could find better results than the lower bounds for 62 instances and similar values for 17 instances.

1. Introduction

The spread of the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2), which is responsible for the coronavirus disease (COVID-19), has caused the most extensive quarantine in human history (Kharroubi & Saleh, 2020). Starting with the Chinese government's imposition of a lockdown on Hubei province on January 23, 2020 (Lau et al., 2020), most countries followed the same steps after the World Health Organization (WHO) declared that COVID-19 is a pandemic on 11 March 2020 (WHO, 2020b). Thus, many countries restricted all forms of international and domestic travel (Chinazzi et al., 2020) to restrict people's movement, hoping to prevent the spread of the SARS-CoV-2 virus; consequently, commercial flights were temporarily suspended. Therefore, decision-makers in several countries had to schedule repatriation flights to return their citizens from other countries. In this paper, we model and solve the flights scheduling problem that decision-makers have addressed in an impromptu manner to repatriate their citizens.

A good percentage of individuals who requested to be repatriated are expatriate workers who lost their jobs due to the COVID-19 pandemic and could not return to their countries due to the border closure and the suspension of commercial flights. Before the COVID-19 pandemic, the number of expatriate workers was expected to be 87.5 million in 2021, according to Finaccord (2017). In the Middle East alone, one million expatriates were expected to lose their jobs in Saudi Arabia due to the sudden lockdown caused by the COVID-19 pandemic in 2020 (Arab News, 2020). A similar number of expatriates was expected to leave the UAE in 2020 due to COVID-19 (Hashmi, 2020).

In addition to work, several other reasons forced people to be out of their countries after the closure of their borders. Millions of people leave their countries for reasons such as tourism and study. Consequently, countries that tried to prevent the infiltration of the SARS-CoV-2 virus by closing their borders have created another problem. What would happen to their citizens if they were left outside their countries? Due to border closures and the suspension of commercial flights, these citizens can neither return to their countries nor survive in foreign countries. Consequently, countries had to schedule repatriation flights to return their citizens.

* Corresponding author at: Industrial Engineering and Engineering Management Department, University of Sharjah, PO Box 27272, Sharjah, United Arab Emirates.

E-mail addresses: salshihabi@sharjah.ac.ae (S. Al-Shihabi), nenad.mladenovic@ku.ac.ae (N. Mladenović).

https://doi.org/10.1016/j.eswa.2022.116728
Received 5 July 2021; Received in revised form 20 February 2022; Accepted 21 February 2022
Available online 11 March 2022
0957-4174/© 2022 Elsevier Ltd. All rights reserved.

The first repatriation flights due to COVID-19 were from Wuhan in China. Karim et al. (2020) have shown that between 29 January 2020 and 27 February 2020, 56 flights repatriated a total of 8597 individuals from Wuhan to 55 countries. These countries wanted their citizens to leave the epicentre of the problem. After the WHO declaration and borders closure in March 2020, the Philippines, as an example, organized a large-scale repatriation program that was called "Bring them home" (Liao, 2020). It was stated in Liao (2020) that "the coronavirus pandemic has drawn attention to the role of sending states in protecting labour migrants during disruptions, particularly by returning them to their countries of origin".

Other countries that have organized repatriation programs similar to the Philippines include Greece (Lytras et al., 2020), India (Syal, 2020), and Australia (Haydar, 2020), to name a few. These countries have organized similar large-scale repatriation programs in collaboration with their flag-carrying airlines. Since a single airline company was responsible for carrying their citizens, the number of repatriated citizens was limited by the capacities of these airlines, in addition to policies imposed by hosting countries. The availability of quarantine locations also limited the number of repatriated citizens. Most countries have followed the recommendations of the WHO regarding the two-week quarantine duration for all repatriated citizens (WHO, 2020a). Consequently, dedicated locations with limited capacity, typically hotels (New South Wales Government, 2020), were used to quarantine and monitor returned citizens to guarantee that the SARS-CoV-2 virus would not spread in the country. Youssef et al. (2021) describes the practical aspects of Lebanon's repatriation program.

Both the quarantine and airline fleet capacities have forced decision-makers to develop repatriation programs, for example, India (Krishna Kumar, 2020) and Jordan (The Jordan Times, 2020). A repatriation program consists of several phases, and in each phase, a limited number of citizens are repatriated based on the citizen's circumstances. Thus, most countries have used platforms where citizens interested in return need to complete applications in which they mention the reason for returning. Examples of countries using this procedure include Jordan,[1] the Netherlands,[2] and India.[3] For example, India's repatriation registration form for returning its citizens from Dubai to India https://www.cgidubai.gov.in/covid_register/ allows users to select one of the following reasons for returning:

1. Medical emergency (self)
2. Medical emergency (family)
3. Death of a family member
4. Stranded tourist/visitor
5. Deportation
6. Loss of employment
7. Expiry of visa
8. Stranded student
9. Pregnancy
10. Other

Decision-makers had to choose the countries from which citizens need to be repatriated based on the received cases. Henceforth, we call the decision-maker's problem of repatriating citizens for one phase the RSP. Fig. 1 shows the sequence of processes to repatriate citizens from foreign countries to the origin country. Each circle in Fig. 1 represents a citizen who wants to return. Circles in black represent citizens of the highest priority to return, followed by moderate priority citizens in dashed circles and low priority citizens in solid circles. Fig. 1 shows that we have two airplanes in stage I, via which we

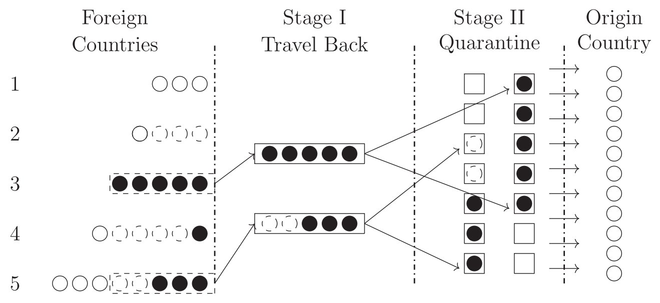

Fig. 1. An illustration of the repatriation scheduling problem (RSP).

repatriate citizens in countries 3 and 5. Repatriated citizens need to stay in dedicated quarantine locations, as shown in stage II. Unlike the stage I operation—flying citizens—which is a batch operation, stage II—quarantining citizens—is not a batch operation, and each citizen would have a separate room. Note that airplanes need to be full for social and economic reasons. Readers interested in more details about the repatriation process and the possible stakeholders involved in this process are referred to Meena et al. (2021).

Previous researchers have only studied and documented repatriation due to COVID-19 from a medical perspective. For example, Lytras et al. (2020) have checked the percentages of repatriated citizens who carried the virus to Greece in March 2020. Similarly, in Içduygu (2020), the SARS-CoV-2's spread was checked among expatriates flying from Wuhan to their countries in the first three months of 2020. Karim et al. (2020) have shown that it is vital to have multi-disciplinary teams that understand how the virus might spread in an airplane when repatriating citizens and how to mitigate that spread.

Unlike the medical aspects related to repatriation, the operational aspects were not studied earlier. To the authors' knowledge, no research paper has already analysed or developed optimization models that consider the operational side of international repatriation due to pandemics. To fill this gap, we develop a mixed-integer linear program (MILP) that captures the decision-makers' dilemmas related to repatriating their citizens due to COVID-19 via their flag-carrying airlines. The objective of the suggested MILP is to repatriate the most vulnerable citizens who need to return, given the quarantine locations and airplanes' capacities. We also solve the RSP using a basic variable neighbourhood search (BVNS) algorithm as the RSP is highly dynamic, i.e., new passengers might register in the repatriation platform to return or cancel their requests. The BVNS algorithm, despite its simplicity, is capable of quickly finding competitive solutions compared to solutions found using a commercial solver's branch and bound (B&B) algorithm. We generate and solve a set of 108 instances that represent various repatriation scenarios to compare the B&B and BVNS algorithms.

In brief, the contributions of our paper are: (i) description and definition of a new optimization problem; (ii) a new mathematical programming formulation of that problem; (iii) design of a heuristic method based on Variable neighbourhood search metaheuristic to solve this new problem; (iv) extensive comparative statistical analysis of the exact and heuristic methods on instances simulated by us.

The rest of this paper is organized as follows. Section 2 presents the MILP model for the RSP and explains the model through an illustrative example, whereas Section 3 introduces the BVNS algorithm. A computational study is then presented in Section 4, in which we generate and solve a set of 108 instances. Finally, we discuss our conclusions and possible future research in Section 5.

2. Mathematical model

We start this section by listing the notation used to define the MILP model. We then formulate the MILP model and explain it through an illustrative example.

[1] https://safelyhome.gov.jo/application/visitor.

[2] https://www.netherlandsandyou.nl/documents/frequently-asked-questions/need-help-returning-to-the-netherlands.

[3] https://www.cgidubai.gov.in/covid_register/.

Table 1
List of notation.

Group	Symbol	Definition
Indices and sets	i	Index set of cities in set $M = \{1, 2, \dots, m\}$.
	j	Index set of priority groups in set $N = \{1, 2, \dots, n\}$.
	k	Index set of airplanes in set $U = \{1, 2, \dots, u\}$.
	M	Set of cities, $M = \{1, 2, \dots, m\}$.
	N	Set of priority groups, $N = \{1, 2, \dots, n\}$.
	U	Set of airplanes, $U = \{1, 2, \dots, u\}$.
Problem inputs	C_k	Capacity of airplane $k \in U$.
	P_{ij}	The Number of individuals residing in city $i \in M$ of group $j \in N$.
	QC	Number of available quarantine locations.
	ω_j	The importance measure given to repatriating citizens in group $i \in N$.
MILP model	α_{ij}	An auxiliary variable that has a value of 1 if the number of citizens repatriated from city $i \in M$ of group $j \in N$, L_i, exceeds L_i, i.e., $L_i \geq P^{cum}_{ij}$, 0 otherwise.
	ϵ_i	An auxiliary variable that has a value of 1 if the number of citizens in city $i \in M$ is greater than the sum of capacities of airplanes assigned to the same city, 0 otherwise.
	L_i	A variable that shows the number of citizens repatriated from city $i \in M$.
	V	Very large number.
	P^{cum}_{ij}	A variable showing the cumulative number of people in group $j \in N$ and other groups with priorities greater than $j \in N$, before repatriating any citizen from city $i \in M$.
	$\hat{P}^{cum}_{ij}$	A variable showing the cumulative number of people in group $j \in N$ and other groups with priorities greater than $j \in N$, after repatriating L_i citizen from city $i \in M$.
	R_{ij}	A variable showing the number of citizens repatriated from group $j \in N$ residing city $i \in M$.
	x_{ik}	A binary decision variable that has a value of 1 if airplane $k \in U$ is assigned to city $i \in M$, 0 otherwise.

2.1. Model notations

We divide the notation used in formulating the MILP model into three groups: indices and sets, problem inputs, and MILP model formulation, as shown in column 1 of Table 1. Columns 2 and 3 of Table 1 show the list of symbols and their definitions, respectively.

2.2. Model formulation

Set $M = \{1, 2, \dots, m\}$ represents the set of cities from which citizens are repatriated. Citizens in city $i \in M$ are divided into n priority groups based on their need to return. Set $N = \{1, 2, \dots, n\}$ is the set of priority groups. We assume that groups are ranked in a descending order based on their priorities in set N so that the priority of group 1 is higher than group 2. We use P_{ij} to show the number of individuals stranded in city $i \in M$ from group $j \in N$ who want to return to their country.

The decision-makers try to return all the citizens stranded out of the country; however, they are constrained by the airplane fleet capacity and quarantine location availability. Due to these limiting constraints, it is crucial to return citizens having high priorities first, i.e., repatriate as much as possible from group 1, followed by group 2, 3, until groups n. For group $j \in N$, decision-makers assign weight ω_j to show the importance of returning this group. i.e., ω_j is the importance measure of group $j \in N$. This parameter, ω_j, can also be interpreted as the cost of keeping this group out of the home country. The higher the group priority is, the higher the value of ω_j is.

.

In addition to sets M and N, we also have set $U = \{1, 2, \dots, u\}$ that represents the set of flag-carrying airplanes, and each airplane $k \in U$, has capacity C_k. QC denotes the available quarantine locations. In summary, we have u airplanes that would repatriate citizens who reside in m different cities. The decision-makers need to decide which airplane to send to one of the m cities. Consequently, the main decision variable of the model is x_{ik} that has a value of 1 if airplane k is flying to city i, 0 otherwise. Note that an airplane needs to carry C_k citizens from city i, unless the number of citizens in city $i \in M$ is less than C_k.

Decision makers want to maximize the sum of the repatriated citizens' priorities, as shown in Eq. (1), where R_{ij} is the number of citizens repatriated from group $j \in N$ residing city $i \in M$. To calculate R_{ij}, we define first a new variable, P^{cum}_{ij}, that shows the cumulative number of people in group $j \in N$ and other groups with priorities greater than $j \in N$, as shown in Eq. (2). For example, assuming that we have three priority groups that are ranked in descending order based on their priorities. We use the notation (j, P_{ij}) to show the priority group number and the number of stranded citizens that belong to this group. For this example, assume that we have the following situation: (1400), (2200), and (3150) then $P^{cum}_{i,1} = 400$ $P^{cum}_{i,2} = P_{i2} + P_{i,1} = 600$, and $P^{cum}_{i,3} = P_{i,3} + P_{i,2} + P_{i,1} = 750$.

As stated earlier, if an airplane is assigned to a city, it needs to fly its maximum capacity for economic and social reasons. Airplane $k \in U$ flying to city $i \in M$ need to carry C_k passengers, unless $P^{cum}_{in} < C_k$, i.e., the number of stranded citizens in city $i \in M$ is less than the airplane capacity. Consequently, we use the system of Eqs. (3)–(8), where V in these equations represent a large number. Auxiliary variable ϵ_i has a value of 1 if the number of citizens in city $i \in M$ is greater than the sum of capacities of airplanes assigned to the same city, 0 otherwise. Based on the value of ϵ_i the number of passengers leaving city $i \in M$, L_i, is either equal to either the sum of capacities of the airplanes assigned to city $i \in M$, as shown in Eqs. (5) and (6), or the number of citizens stranded in city $i \in M$, as shown in Eqs. (7) and (8). The constraint represented by Eq. (16) guarantees that the number of citizens repatriated from all cities do not exceed the quarantine capacity QC.

To find R_{ij}, we need first to calculate the cumulative number of citizens in groups $0, 1, \dots, n$ from each city $i \in M$, after flying back L_i citizens to the home country. We use $\hat{P}^{cum}_{ij}$ to show this number. Consequently, P^{cum}_{ij} and $\hat{P}^{cum}_{ij}$ represent the number of citizens in country $i \in M$ of priority groups $1, 2, \dots, j \in N$, before and after flying L_i passengers, respectively. We use the system of Eqs. (9)–(14) to find $\hat{P}^{cum}_{ij}$. In this system of Eqs. (9)–(14), We use indicator variable α_{ij} to find if the number of repatriated citizens from city i exceeds the number of citizens in groups $1, 2, \dots, j \in N$. Therefore, α_{ij} is 1 if $L_i \geq P^{cum}_{ij}$, 0 otherwise, as shown in Eqs. (9) and (10). Based on the values of α_{ij}, we calculate $\hat{P}^{cum}_{ij}$, as shown in Eqs. (11)–(14). Knowing the value of $\hat{P}^{cum}_{ij}$ enables us to calculate R_{ij} as shown in Eq. (15). Lastly, Eq. (17) guarantees that an airplane can at most fly to a single destination.

In summary, the MILP model of the RSP can be represented using Eqs. (1)–(17) where the main decisions are how to assign airplanes to a cities, as shown in Eq. (18). We use Eqs. (2)–(15) to find the number of citizens repatriated from each group and city. Eq. (16) guarantees that the maximum number of repatriated citizens does not exceed QC, whereas Eq. (17) prevent any airplane from being assigned to more than one destination.

$$(\max) Z = \sum_{i \in M} \sum_{j \in N} \omega_j \times R_{ij}, \tag{1}$$

Table 2
Number and groups of Jordanian citizens expressing their interests to return back.

City	Groups			
	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

$$P_{ij}^{cum} = \sum_{u=0}^{u=j} P_{iu}, \ \forall i \in M, \ j \in N, \tag{2}$$

$$P_{in}^{cum} - \sum_{k \in U} C_k \times x_{ik} \geq (\epsilon_i - 1) \times V, \ \forall i \in M, \tag{3}$$

$$P_{in}^{cum} - \sum_{k \in U} C_k \times x_{ik} \leq (\epsilon_i) \times V, \ \forall i \in M, \tag{4}$$

$$L_i \leq \sum_{k \in U} C_k \times x_{ik} + (1 - \epsilon_i) \times V, \ \forall i \in M, \tag{5}$$

$$L_i \geq \sum_{k \in U} C_k \times x_{ik} + (\epsilon_i - 1) \times V, \ \forall i \in M, \tag{6}$$

$$L_i \leq P_{in}^{cum} + \epsilon_i \times V, \ \forall i \in M, \tag{7}$$

$$L_i \geq P_{in}^{cum} - \epsilon_i \times V, \ \forall i \in M, \tag{8}$$

$$L_i - P_{ij}^{cum} > (\alpha_{ij} - 1) \times V, \ \forall i \in M, \ \forall j \in N, \tag{9}$$

$$L_i - P_{ij}^{cum} \leq \alpha_{ij} \times V, \ \forall i \in M, \ \forall j \in N, \tag{10}$$

$$\hat{P}_{ij}^{cum} \leq (1 - \alpha_{ij}) \times V, \ \forall i \in M, \ \forall j \in N, \tag{11}$$

$$\hat{P}_{ij}^{cum} \geq (\alpha_{ij} - 1) \times V, \ \forall i \in M, \ \forall j \in N, \tag{12}$$

$$\hat{P}_{ij}^{cum} \leq P_{ij}^{cum} - L_i + \alpha_{ij} \times V, \ \forall i \in M, \ \forall j \in N, \tag{13}$$

$$\hat{P}_{ij}^{cum} \geq P_{ij}^{cum} - L_{it} - \alpha_{ij} \times V, \ \forall i \in M, \ \forall j \in N, \tag{14}$$

$$R_{ij} = \begin{cases} P_{ij}^{cum} - \hat{P}_{ij}^{cum}, \ \forall i \in M, \ j = 1 \\ P_{ij}^{cum} - \hat{P}_{ij}^{cum} - P_{ij-1}^{cum} - \hat{P}_{ij-1}^{cum}, \ \forall i \in M, \ j > 1, \end{cases} \tag{15}$$

$$\sum_{i \in M} L_i \leq QC, \tag{16}$$

$$\sum_{i \in M} x_{ik} \leq 1, \ k \in U, \tag{17}$$

$$x_{ik} = \begin{cases} 1 & if\ airplane\ k\ is\ assigned\ to\ fly\ citizens\ from\ city\ i \\ 0 & otherwise. \end{cases} \tag{18}$$

2.3. Illustrative example

Assume that we have 5 cities having 4 priority groups as shown in Table 2. The priority values of the four groups are shown in the last row of Table 2. Assume also that we have one type of airplanes where each airplane has a capacity of 300 passengers. Table 3 shows the solutions of this problem for a different number of airplanes and quarantine capacities. Columns 1 and 2 of Table 3 show the number of airplanes and quarantine locations, respectively. Columns 3–7 of Table 3 show the solution of the problem, where for each city we show the number of airplanes assigned to the city and the number of passengers flying out of the city. The objective function value (OFV) is shown in column 8 of Table 3.

Table 3
Solutions of the illustrative example for different number of airplanes.

Airplanes	Quarantine	Cities					OFV
Number	Capacity	1	2	3	4	5	
2	2000	0	(1300)	0	0	(1300)	6000
4	2000	(1300)	(1300)	0	(1300)	(1300)	11,400
6	2000	(1300)	(1300)	(1300)	(1300)	(2600)	15,500
8	2000	(1300)	(1300)	(1300)	(1300)	(2600)	15,500
8	2500	(2600)	(1300)	(1300)	(1300)	(2600)	18,500
10	2500	(2600)	(2600)	(1300)	(1300)	(2600)	18,500
10	3000	(2600)	(2600)	(2600)	(2600)	(2600)	20,900
12	3000	(2600)	(2600)	(2600)	(2600)	(2600)	20,900
12	3500	(2600)	(2600)	(2600)	(3800)	(3700)	21,600
12	4000	(2600)	(2600)	(2600)	(3800)	(3700)	21,600
14	4000	(3700)	(2600)	(3650)	(3800)	(3700)	21,900

Table 3 shows how the OFVs increase with an increase in the number of airplanes or quarantine locations. For example, for 2000 quarantine locations, the OFVs are 11,400 and 15,500 for the four and six airplanes scenarios, respectively. The OFVs did not change when increasing the number of airplanes from 6 to 8 for 2000 quarantine locations since quarantine locations limited the number of repatriated citizens, not the number of airplanes.

3. Variable neighbourhood search

In this section, we describe our BVNS algorithm to solve the RSP. We first review BVNS algorithm variants. We then show how did we apply the BVNS algorithm to the RSP.

3.1. VNS

VNS algorithms are metaheuristics designed to solve combinatorial and global optimization problems. Iteratively and systematically, a VNS algorithm perturbs the incumbent solution of the problem and applies local search algorithm(s) to improve the perturbed solution (e.g., Hansen & Mladenović, 1997, 1999; Hansen, Mladenović, & Pérez, 2010; Mladenović & Hansen, 1997). If the locally optimal solution found in the perturbed solution's neighbourhood is better than the incumbent solution, then the VNS algorithm updates the incumbent solution. According to Hansen et al. (2010), for a minimization problem, VNS is based on the following facts:

1. Fact 1. A local minimum with respect to one neighbourhood structure is not necessarily a local minimum for another neighbourhood structure.
2. Fact 2. A global minimum is a local minimum with respect to all possible neighbourhood structures.
3. Fact 3. For many problems, local minima with respect to one or several neighbourhoods is relatively close.

A generic algorithm that shows the BVNS steps is shown in Algorithm 1. In the initialization phase, lines 3–5, we start by selecting a set of neighbourhoods $\mathcal{N}_k, (k = 1, 2, \dots, k_{max})$ that we use in the shaking step of the search algorithm to perturb the incumbent solution. $\mathcal{N}_k(x)$ represents the set of solution in neighbourhood k. neigbhorhouds $\mathcal{N}_k(x), (k = 1, 2, \dots, k_{max})$ are usually nested. In addition to selecting the neighbourhoods in the initialization phase, we generate an initial solution and choose a stopping condition before starting the search phase. The initial solution is considered the incumbent one.

The search phase consists of two main steps, shaking and local search, lines 10–12 of Algorithm 1. In the shaking step, we randomly generate a solution from the kth neighbourhood of x. We denote the perturbed solution by x'. Note that k is initialized to 1, line 7. We apply the local search algorithm to the perturbed solution and save the best-found solution to x''. If x'' is better than the incumbent solution x then

Algorithm 1: BVNS algorithm

```
    Data: Problem Instance, $k_{max}$, $t_{max}$
    Result: Incumbent solution $x$
1   begin
2       Initialization phase:
3           1- Select the set of neighbourhood structures
             $\mathcal{N}_k, k = 1, 2, ..., k_{max}$ that will be used in the search ;
4           2- Find an initial solution $x$ ;
5           3- Choose a stopping condition ;
6       Search phase:
7           time ← 0 ;
8           while $time \leq CpuTime$ do
9               $k \leftarrow 1$ ;
10              while $k \leq k_{max}$ do
11                  1- Shaking. Generate a point $x'$ at random from
                     the $kth$ neighbourhood of x($x' \in \mathcal{N}_k(x)$) ;
12                  2-Local Search. Apply local search algorithm to
                     $x'$; denote with $x''$ the best found solution ;
13                  3-Updating $x$. If $x''$ is better than $x$, then $x \leftarrow x''$
                     and $k \leftarrow 1$; otherwise, $k \leftarrow k + 1$ ;
14              time ← CpuTime ;
15          End
```

we update $x \leftarrow x''$ and set $k \leftarrow 1$; otherwise, we further perturb the current incumbent solution x using the $kth + 1$ neighbourhood.

In addition to the BVNS algorithm, VNS framework has many other variants. For example, suppose we skip the local search step in Algorithm 1. In that case, we obtain a reduced VNS (RVNS) algorithm that is akin to the Monte-Carlo method but is more systematic and leads to better solutions (Hansen & Mladenović, 2001; Mladenović, Petrović, Kovačević-Vujčić, & Čangalović, 2003). In the General VNS (GVNS) algorithm, we sequentially apply more than one search technique in the local search step, line 11 of algorithm 1, Brimberg, Mladenović, Todosijević, and Urošević (2017), and Soylu (2015). If the local search is confined to a subset of the whole search space, then we have the variable neighbourhood decomposition search (VNDS) (Lazić, Hanafi, Mladenović, & Urošević, 2010). BVNS, RVNS, GVNS, and VNDS are just examples of the different VNS variants, interested reader are referred to Hansen and Mladenovic (2003), Hansen and Mladenović (2014), Hansen, Mladenović, and Pérez (2008) and Hansen et al. (2010) for more comprehensive covering of the VNS variants.

VNS algorithms have been applied to both discrete and continuous optimization problems (Mladenović, Dražić, Kovačevic-Vujčić, & Čangalović, 2008). VNS was used to solve a large number of problems such as bin packing (Santos et al., 2019), vehicle routing problem (Simeonova, Wassan, Salhi, & Nagy, 2018), nurse rostering (Rahimian, Akartunalı, & Levine, 2017), time tabling (Tan, Goh, Kendall, & Sabar, 2021), job shop scheduling (Yazdani, Amiri, & Zandieh, 2010) and location (Fleszar & Hindi, 2008). Other interesting, non-classical VNS applications can be found in Mladenović, Delot, Laporte, and Wilbaut (2020), Pardo, Mladenović, Pantrigo, and Duarte (2013) and Todosijević, Mladenović, Hanafi, and Crévits (2012).

3.2. BVNS to RSP

To explain the BVNS algorithm that we use to solve the RSP, we start by describing how we represent the solution. This description has enabled us to create a set of nested neighbourhoods for the shaking step; moreover, this representation has also allowed us to apply a simple but efficient local search to improve perturbed solutions. After describing the solution representation, we explain how we generate the initial solution, followed by a description of the BVNS algorithm to solve the RSP.

Table 4
Solution representation in the BVNS algorithm.

Airplane	A	B	C	D	E	F	G
City	5	2	1	1	3	None	4

3.2.1. Solution representation

In our implementation, we represent the solution as a vector, as shown in Table 4 for a hypothetical case. The solution shows the (airplane,city) assignments. For example, airplane A is assigned to city 5, while airplane F is not assigned to any city. This solution can be represented as $x = \{(A, 5), (B, 2), (C, 5), (D, 1), (E, None), (F, 1), (G, 4)\}$. If the sum of airplane capacities is greater than the quarantine locations, then it means that some airplanes might not be used.

3.2.2. Initial solution generation

To find an initial solution, we follow the following steps:

1. rank airplanes in a non-decreasing order based on their capacity,
2. choose the city that will generate the highest sum of priority values of returned citizens, as shown in Eq. (19). If more than one city result in the same value, we randomly choose one of these cities with equal probabilities.

$$\underset{i}{\text{argmax}} \sum_{j \in N} \omega_j P_{ij} \tag{19}$$

Intuitively, choosing the small airplanes first allows decision-makers to divide repatriated citizens into smaller groups. Thus, decision-makers have more batching options. This increase in batching options is expected to increase the OFVs. Another practical reason for choosing to fill smaller airplanes first has to do with trained pilots to fly these airplanes. In general, more pilots are trained to fly small airplanes that big ones.

3.2.3. Neighbourhood definition

Like any other combinatorial optimization problem, the RSP is about finding an optimal solution $x_{opt} \in X \in S$ where X and S represent the sets of feasible solutions and solution space, respectively. A generic representation of a maximization, combinatorial optimization problem is shown in Eq. (20).

$$\max\{f(x) : x \in X, X \in S\} \tag{20}$$

$\mathcal{N}_k$, $k = (1, 2, \ldots, k_{max})$, is a finite set of pre-selected neighbourhood structures, whereas $\mathcal{N}_k(x)$ is the set of solutions in the kth neighbourhood of x. Neighbourhood $\mathcal{N}_k(x)$ of solution x is defined by the Hamming distance between $x' \in \mathcal{N}_k(x)$ and x. The Hamming distance between two RSP solutions represent the sum of different (airplane, city) assignments. For example, solution $\{(A, 5), (B, 2), (C, 5), (D, 1), (E, None), (F, 1), (G, 4)\}$ has a Hamming distance of 3 from the solution show in Table 4. Henceforth, we use ρ to represent the distance between solution $x' \in \mathcal{N}_k(x)$ and x, as shown in Eq. (21). Consequently, we can define Neighbourhood $\mathcal{N}_k(x)$ as shown in Eq. (22).

$$\rho(x', x) = |x' \setminus x| = |x \setminus x'| \tag{21}$$

$$\rho(x' \in \mathcal{N}_k(x), x) = k \tag{22}$$

3.2.4. Shaking and local search

The shaking and local search steps are the two main components for any BVNS algorithm. For the shaking step, we use $\mathcal{N}_k(x)$, $k = 1, 2, \ldots, k_{max}$, neighbourhoods, where the value of k_{max} is experimentally chosen as we show in the next section. Assume a small problem having two airplanes and two cities. Assuming an incumbent solution $x^{incumb} = \{(A, 1), (B, 2)\}$, and $k_{max} = 2$, then we have the following two shaking neighbourhoods:

1. $\mathcal{N}_1(x) = \{\{(A,1),(B,1)\},\{(A,1),(B,None)\},\{(A,2),(B,2)\},$
$\{(A,None),(B,2)\}\}$.
2. $\mathcal{N}_2(x) = \{\{(A,2),(B,1)\},\{(A,2),(B,None)\},\{(A,None),(B,1)\},$
$\{(A,None),(B,None)\}\}$.

It is clear that $\rho = 1$ and $\rho = 2$ for $\mathcal{N}_1(x)$ and $\mathcal{N}_2(x)$, respectively. In the shaking step and depending on the k value, one of the possible neighbouring solutions is randomly selected. In the local search step, we only use the $\mathcal{N}_1(x)$ neighbourhood; moreover, we choose a steepest-ascent strategy to move to a neighbouring solution, i.e., we compare the different neighbouring solutions and choose the one that leads to the best OFVs. .

In the local search step, we change the assignment of one airplane at a time and choose the best reassignment, whereas in the shaking operation, we randomly change the assignments of k airplanes as defined by neighbourhood $\mathcal{N}_k(x)$.

Algorithm 2 shows the main steps of the BVNS algorithm as we apply it to solve the RSP. Inputs to the algorithm include the initial solution, number of neighbourhoods used in the shaking operation, and maximum computation time. Solution x is initialized by copying the initial solution. Moreover, we start with the least shaking value, $k = 1$. The main loop of Algorithm 2 consists of two steps, shaking and local search. In the shaking step, k assignments are randomly changed with respect to x to generate solution x'. The local search algorithm tries to find the best solution in $\mathcal{N}_1(x')$. If the solution found in $\mathcal{N}_1(x')$ is better than x, then we update x and have $k = 1$; otherwise, we increment k. Lastly, we end the algorithm if we find that the computation time exceeds t_{max} after the local search step.

Algorithm 2: BVNS for the RSP

```
Data: $x$, $k_{max}$, $t_{max}$
Result: $x, t^*$
1  begin
2      $x \leftarrow$ initial solution ;
3      $t \leftarrow 0$ ;
4      while $t \le t_{max}$ do
5          $k \leftarrow 1$;
6          while $k \le k_{max}$ do
7              $x' \leftarrow$ Shake($x, k$);
8              $x' \leftarrow$ LocalSearch($x'$);
9              $k \leftarrow k + 1$;
10             if $f(x') > f(x)$ then
11                 $x \leftarrow x'$;
12                 $k \leftarrow 1$;
13                 $t^* \leftarrow t$
14         $t \leftarrow$ CpuTime() ;
```

4. Computational study

In this section, we compare the efficiency and effectiveness of the BVNS algorithm with the B&B algorithm of *CPLEX12.10*. For comparison, we generate a set of 108 instances of different configurations. We start this section by describing the generated instances. We then tune the BVNS parameters before running our comparison experiment and analysing its results.

4.1. Instances

To generate a set of representative instances, we consider the following parameters:

1. Number of cities. We use the following number of cities: 150, 200, and 250.
2. Number of trips is 600, 800, and 1000.
3. Airplanes' capacities. We randomly select one of the following values of the airplane capacities 100, 150, 200, 250, 300, and 350.
4. Number of priority groups. We use n equal to 3, 6 and 9.
5. Assigning priority values to priority groups. We use two techniques to generate priority values for the priority groups:
    (a) Deterministic: Priority value equals the group number. For example, for the three-group case, the priority values are 1, 2 and 3.
    (b) Random: Randomly choose n values from a normal distribution having a mean of n and standard deviation of $\sqrt{n}$.
6. Number of individuals per group. Using the average capacity of airplanes, we uniformly generate the P_{ij} values to be between 0.5 to 1.5 of this average capacity.
7. Fleet capacity to quarantine capacity ratio, FQ. We choose the quarantine capacity to be either 80% or 120% of the fleet capacity. Thus, the number of repatriated citizens would be limited by either the fleet capacity or the number of quarantine locations.

Based on the parameters mentioned above, we have 108 configurations, i.e., three values for $|M|$, three values for $|U|$, three values for the number of groups, two ratios of $\frac{\sum_{k\in U} C_k}{QC}$, and two methods to choose priority values. We use the following format $A - B - C - D - E$ to name the generated instances where A, B, C, D, and E show the number of cities, number of airplanes, number of priority groups, priority values' selection method, and the fleet to quarantine capacity, respectively. We use either D or R for the deterministic and random priority values' generation techniques, respectively. For example, instance 150-600-3-D-0.8 means that we have 600 airplanes, via which we need to repatriate citizens from 150 cities. The citizens belong to three priority groups, for which we choose the priority values using the deterministic technique. The 0.8 value shows that the number of quarantine locations is 80% of the seats' sum in all airplanes.

We describe the instances' generation using Algorithm 3. In Algorithm 3, the P matrix represents the number of individuals in each city and group. The P matrix components are compatible with the aforementioned P_{ij} definition. Vectors C and Ω represent the airplanes' capacities and priority values per group, respectively. The number of quarantine locations, QC, is the last output needed to create an instance.

Algorithm 3 starts by initializing counter airplane to 1 and variable *sumCapacity* to 0. We use the variable *sumCapacity* to calculate the sum of seats in the randomly chosen airplanes. We use this value in addition to FQ to find QC. Algorithm 3 consists of three loops. In the first loop, line 4–8, we randomly choose the airplanes capacities, line 5, we add these capacities to *sumCapacity*, line 6, and update the airplane counter, line 7. The first loop ends once the airplane counter is equal to U. Line 9 shows how we use *sumCapacity* to find QC. Before starting the second loop, lines 11–15, we calculate the airplanes' average capacity, *avgCap*, as shown in line 10. In the second loop, we create the P matrix. Finally, the third loop, lines 16–18, shows how we assign priority values to the different groups, i.e., how to generate Ω.

4.2. Experiments

We solve the 108 instances using both the BVNS algorithm and the default B&B algorithm of *CPLEX12.10* to check the quality of the BVNS solutions. However, before solving these instances, we run a tuning experiment to choose k_{max}. In all our experiments, we report the OFVs, as shown in Eq. (1). The processor used to run all experiments is Core (TM) i7-1065G7, 1.30 GHz. Lastly, the BVNS algorithm was coded in C language.

Table 5
Average solutions found for selected instances using different values for k_{max}.

Instance	k_{max}				
	5	10	15	20	25
150-600-9-D-1.2	817,658.40	817,656.60	817,575.60	817,583.60	817,562.50
150-600-9-R-1.2	1,319,837.90	1,319,839.60	1,319,828.30	1,319,820.20	1,319,820.10
150-800-9-D-1.2	1,080,136.90	1,080,111.40	1,080,144.80	1,080,207.30	1,080,244.50
150-800-9-R-1.2	1,820,399.90	1,820,478.30	1,820,408.90	1,820,394.20	1,820,131.80
150-1000-9-D-1.2	1,335,187.40	1,335,148.70	1,335,137.10	1,335,049.30	1,334,997.90
150-1000-9-R-1.2	2,137,139.40	2,137,103.80	2,137,009.40	2,137,106.60	2,137,104.70
200-600-6-D-0.8	499,114.70	499,109.60	499,117.40	499,113.60	499,112.30
200-600-6-D-1.2	564,792.90	564,793.30	564,789.30	564,794.00	564,789.70
200-800-3-D-0.8	362,473.70	362,493.20	362,505.00	362,486.00	362,481.60
200-800-9-D-1.2	1,086,538.90	1,086,646.60	1,086,641.00	1,086,707.50	1,086,776.90
200-1000-9-D-1.2	1,373,005.50	1,373,076.10	1,373,048.50	1,373,041.20	1,373,037.90
200-1000-9-R-0.8	1,585,900.50	1,585,902.70	1,585,901.90	1,585,901.40	1,585,902.30
250-600-9-R-0.8	997,707.90	997,705.70	997,696.80	997,697.60	997,694.30
250-600-9-R-1.2	1,539,459.20	1,539,404.70	1,539,379.30	1,539,423.60	1,538,316.70
250-800-9-R-0.8	1,323,761.80	1,323,767.50	1,323,761.30	1,323,768.30	1,323,740.90
250-800-9-R-1.2	1,819,138.60	1,819,194.50	1,819,075.40	1,819,082.30	1,819,062.40
250-1000-6-R-0.8	1,541,729.60	1,541,733.	1,541,731.20	1,541,729.90	1,541,728.30
250-1000-6-R-1.2	2,195,729.10	2,195,721.40	2 195 762.3	2,195,742.40	2,195,712.40
Number of Bests	4	6	3	3	2

Table 6
P-values as obtained by the Wilcoxon signed-rank test for different pairs of k_{max} values.

		k_{max}				
		5	10	15	20	25
	5	–	0.647	0.580	0.670	0.098
	10	0.647	–	0.066	0.327	0.026
k_{max}	15	0.580	0.066	–	0.799	0.081
	20	0.670	0.327	0.799	–	0.018
	25	0.098	0.026	0.081	0.018	–

4.2.1. *Tuning experiment*

Users need to choose two parameters only to run the BVNS algorithm: t_{max} and k_{max}. The more time the algorithm is given, the better the results are. Having $t_{max} = 20$ s was sufficient to obtain competitive results; thus, we do not tune the t_{max} parameter. However, we test five values for the k_{max} parameter: 5, 10, 15, 20, and 25, to choose the best k_{max} value. We solve 18 instances using the five k_{max} values, as shown in Table 5. The chosen instances were selected to cover a wide range of cases. Column 1 of Table 5 shows the instance name while the other five columns show the average OFVs that we find by solving each instance 25 times.

The best average results are underlined in Table 5, whereas the number of best answers found for each k_{max} value are shown in the last row of Table 5. We use the Wilcoxon signed-rank test (Woolson, 2007) to check if there is a significant difference between the results obtained by the different k_{max} values. We conduct ten tests to compare all the different paired results. Table 6 shows the $P - values$ obtained for each comparative test. Assuming a 0.05 level of significance, we can ignore $k_{max} = 25$ from any further consideration because its associated P-values are less than 0.05, as can be seen by in the last column and row of Table 6. The results, however, show no significant difference among the other four values of k_{max}. Since we cannot statistically select the best k_{max} value, we choose $k_{max} = 10$ because six best answers were found using this value.

4.2.2. *Comparative study*

We solve the MILP models of the 108 instances using an exact solver and using the BVNS algorithm to check the efficiency and effectiveness of the BVNS algorithm. Table 7 shows the results of this experiment where column 1 shows the instance names. The first part of Table 7, columns 2–4, show the LB, UB, and gap between the two bounds, respectively, by running the default B&B algorithm of *CPLEX12.10* for 8 h, 28,800 s, or until an optimal solution is found, i.e. $UB = LB$. The formula used to calculate the gap values in column 4 of Table 7 is $\frac{UB-LB}{LB} \times 100\%$. The second part of Table 7, columns 5 and 7, reports the OFVs obtained using the BVNS algorithm where each instance is solved 10 times using $t_{max} = 20$. Columns 5 and 6 of Table 7 show the best and average solutions, respectively. The last part of Table 7, columns 7, shows the percent deviation between the best BVNS solution compared to the LB found by the exact method , $\frac{Best-LB}{LB} \times 100.$. We choose the LB solutions in calculating deviations between the two algorithms because these solutions are feasible. Note that for deviations less than 0.01%, we use + sign to indicate a better performance by the BVNS algorithm.

Algorithm 3: Instance generation algorithm

Data: $|M|$,$|U|$,$|N|$, priority value selection technique, and FQ

Result: $\mathbf{P} = \begin{array}{c} \\ 1 \\ 2 \\ \dots \\ m \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \dots & n \end{array} \\ \begin{pmatrix} P_{11} & P_{12} & \dots & P_{1n} \\ P_{21} & P_{22} & \dots & P_{21} \\ \dots & \dots & \dots & \dots \\ P_{m1} & P_{m2} & \dots & P_{mn} \end{pmatrix} \end{array}$, $C = [C_1, C_2, ..., C_u]$, $\Omega = [\omega_1, \omega_2, ..., \omega_n]$, and QC

```
begin
    $airplane = 1$ ;
    $sumCapacity = 0$ ;
    while $airplane \leq |U|$ do
        $C_{airplane} \xleftarrow{random} discrete(200, 250, 300, 350)$ ;
        $sumCapacity = sumCapacity + C_{airplane}$ ;
        $airplane = airplane + 1$ ;
        End;
    $QC = sumCapacity \times FQ$ ;
    $avgCap = \frac{sumCapacity}{|U|}$ ;
    for $i = 1 : |M|$ do
        for $j = 1 : |N|$ do
            $P_{ij} \xleftarrow{random} uniform(0.5 \times avgCap,\ 1.5 \times avgCap)/$ ;
            End ;
        End ;
    for $j = 1 : |N|$ do
        $\omega_{airplane} \leftarrow$ priority value selection technique ;
        End ;
    End ;
```

Table 7
Comparison between the exact and the VNS solutions.

Instance	CPLEX			BVNS		Dev.
	UB	LB	Gap (%)	Best	Average	%
150-600-3-D-0.8	274,649.0	274,362	0.10	274,589	274,589.0	0.08
150-600-3-D-1.2	332,411.0	305,801	8.70	305,801	305,801.0	0.00
150-600-3-R-0.8	259,605.0	259,605.0	0.00	259,605.0	259,605.0	0.00
150-600-3-R-1.2	516,129.7	462,562.9	11.58	462,562.9	462,562.9	0.00
150-600-6-D-0.8	513,711.0	513,210	0.10	513,231	513,231.0	+
150-600-6-D-1.2	611,439.0	573,022	6.70	573,144	573,144.0	0.02
150-600-6-R-0.8	896,424.1	895,305.7	0.12	895,016.1	895,016.1	−0.03
150-600-6-R-1.2	1,008,133.4	896,555.5	12.45	896,555.5	896,555.5	0.00
150-600-9-D-0.8	728,773.0	728,522	0.03	728,413	728,413.0	−0.01
150-600-9-D-1.2	868,625.0	817,245	6.29	817,606	817,433.0	0.04
150-600-9-R-0.8	1,237,482.0	1,237,175.0	0.02	1,234,900.0	1,234,900.0	−0.21
150-600-9-R-1.2	1,516,494.8	1,319,573.4	14.92	1,319,807.0	1,319,800.0	0.02
150-800-3-D-0.8	355,287.0	354,989	0.08	354,827	354,827.0	−0.05
150-800-3-D-1.2	456,725.0	420,585	8.59	420,585	420,585.0	0.00
150-800-3-R-0.8	491,164.3	490,738.5	0.09	490,558.0	490,558.0	−0.04
150-800-3-R-1.2	756,939.9	741,249.0	2.12	741,258.0	741,258.0	+
150-800-6-D-0.8	662,415.0	662,232	0.03	661,485	661,485.0	−0.11
150-800-6-D-1.2	801,581.0	750,717	6.78	750,745	750,745.0	+
150-800-6-R-0.8	1,157,575.9	1,156,809.4	0.07	1,157,572.0	1,157,572.0	0.07
150-800-6-R-1.2	1,207,829.9	1,126,380.4	7.23	1,126,594.0	1,126,594.0	0.02
150-800-9-D-0.8	964,186.1	961,972	0.23	962,186	962,186.0	0.02
150-800-9-D-1.2	1,081,769.4	1,075,909	0.54	1,080,156	1,079,973.0	0.39
150-800-9-R-0.8	1,857,913.0	1,857,511.0	0.02	1,856,769.0	1,856,769.0	−0.04
150-800-9-R-1.2	2,027,362.8	1,816,872.4	11.59	1,820,540.0	1,820,434.0	0.2
150-1000-3-D-0.8	452,911.0	452,499	0.09	452,671	452,671.0	0.04
150-1000-3-D-1.2	560,545.0	515,765	8.68	515,675	515,675.0	−0.02
150-1000-3-R-0.8	709,157.4	708,966.0	0.03	708,749.5	708,749.5	−0.03
150-1000-3-R-1.2	1,058,693.0	944,926.6	12.04	944,926.6	944,926.6	0.00
150-1000-6-D-0.8	852,014.0	851,725	0.03	851,894	851,894.0	0.02
150-1000-6-D-1.2	1,015,194.0	950,360	6.82	950,595	950,595.0	0.02
150-1000-6-R-0.8	1,557,130.4	1,547,935.2	0.59	1,547,019.2	1,547,019.2	−0.06
150-1000-6-R-1.2	1,603,409.8	1,476,669.2	8.58	1,476,958.3	1,476,958.3	0.02
150-1000-9-D-0.8	1,196,263.0	1,196,151	0.01	1,196,223	1,196,223.0	0.01
150-1000-9-D-1.2	1,414,288.0	1,333,348	6.07	1,335,162	1,335,083.0	0.14
150-1000-9-R-0.8	2,167,499.6	2,166,777.4	0.03	2,166,365.9	2,166,365.9	−0.02
150-1000-9-R-1.2	2,392,003.2	2,136,325.8	11.97	2,137,172.4	2,137,116.9	0.04
200-600-3-D-0.8	276,975.0	276,301	0.24	276,595	276,595.0	0.11
200-600-3-D-1.2	342,120.0	341,660	0.13	341,660	341,660.0	0.00
200-600-3-R-0.8	597,284.2	596,728.7	0.09	596,366.3	596,366.3	−0.06
200-600-3-R-1.2	617,690.2	613,441.9	0.69	611,902.7	610,790.4	−0.25
200-600-6-D-0.8	502,175.7	499,736	0.49	499,114	499,094.6	−0.12
200-600-6-D-1.2	601,987.0	562,703	6.98	564,794	564,768.5	0.37
200-600-6-R-0.8	576,671.3	576,461.0	0.04	575,501.2	575,474.2	−0.17
200-600-6-R-1.2	1,404,501.3	1,218,466.0	15.27	1,219,142.5	1,219,128.8	0.06
200-600-9-D-0.8	705,717.2	702,318	0.48	701,766	701,742.2	−0.08
200-600-9-D-1.2	847,577.0	793,139	6.86	797,947	797,592.2	0.60
200-600-9-R-0.8	1,217,347.7	1,217,077.2	0.02	1,214,903.2	1,214,856.2	−0.18
200-600-9-R-1.2	1,428,400.5	1,312,056.2	8.87	1,312,923.2	1,312,548.2	0.07
200-800-3-D-0.8	362,985.0	362,408	0.16	362,505	362,504.6	0.03
200-800-3-D-1.2	449,925.0	413,955	8.69	413,955	413,955.0	0.00
200-800-3-R-0.8	470,532.8	470,282.8	0.05	470,459.2	470,459.2	0.04
200-800-3-R-1.2	388,894.6	359,598.6	8.15	359,621.2	359,621.2	0.01
200-800-6-D-0.8	669,089.0	669,089	0.00	668,994	668,993.0	−0.01
200-800-6-D-1.2	798,355.0	746,382	6.96	747,194	747,194.0	0.11
200-800-6-R-0.8	1,068,196.9	1,067,819.0	0.04	1,066,688.9	1,066,688.9	−0.11
200-800-6-R-1.2	1,342,415.6	1,190,603.4	12.75	1,191,052.3	1,191,052.3	0.04
200-800-9-D-0.8	976,304.7	969,400	0.71	969,144	969,144.0	−0.03
200-800-9-D-1.2	1,152,237.0	1,079,197	6.77	1,086,859	1,086,665.5	0.70
200-800-9-R-0.8	1,200,959.2	1,198,900.9	0.17	1,199,214.8	1,199,213.3	0.03
200-800-9-R-1.2	2,160,708.5	1,935,397.3	11.64	1,935,698.2	1,935,652.7	0.02
200-1000-3-D-0.8	453,616.0	453,084	0.12	453,216	453,216.0	0.03
200-1000-3-D-1.2	567,577.0	522,347	8.66	522,357	522,357.0	+
200-1000-3-R-0.8	713,515.9	712,840.8	0.09	712,731.0	712,731.0	−0.02
200-1000-3-R-1.2	576,930.8	507,454.3	13.69	507,454.3	507,454.3	0.00
200-1000-6-D-0.8	831,907.0	831,714	0.02	831,367	831,367.0	−0.04
200-1000-6-D-1.2	975,633.0	915,639	6.55	916,193	916,193.0	0.06
200-1000-6-R-0.8	1,327,015.9	1,327,015.9	0.00	1,326,567.9	1,326,567.9	−0.03
200-1000-6-R-1.2	2,002,829.0	1,817,682.8	10.19	1,821,404.0	1,821,404.0	0.2
200-1000-9-D-0.8	1,187,698.0	1,187,698	0.00	1,187,165	1,187,165.0	−0.04
200-1000-9-D-1.2	1,457,644.0	1,369,533	6.43	1,373,130	1,372,798.7	0.26
200-1000-9-R-0.8	1,586,042.6	1,585,888.0	0.01	1,585,903.1	1,585,903.1	+
200-1000-9-R-1.2	2,613,334.8	2,326,629.1	12.32	2,331,644.6	2,331,341.9	0.22
250-600-3-D-0.8	281,004.0	280,694	0.11	280,717	280,676.3	0.01

(*continued on next page*)

Table 7 (*continued*).

Instance	CPLEX			BVNS		Dev.
	UB	LB	Gap (%)	Best	Average	%
250-600-3-D-1.2	337,822.0	310,692	8.73	310,702	310,702.0	+
250-600-3-R-0.8	412,523.7	412,272.2	0.06	412,473.8	412,440.8	0.05
250-600-3-R-1.2	695,346.3	590,141.4	17.83	590,141.4	590,141.4	0.00
250-600-6-D-0.8	492,414.0	492,310	0.02	490,790	490,582.2	−0.31
250-600-6-D-1.2	601,015.0	562,233	6.90	562,934	562,741.0	0.12
250-600-6-R-0.8	878,103.8	877,958.1	0.02	876,497.6	875,677.0	−0.17
250-600-6-R-1.2	720,199.4	675,802.8	6.57	675,936.3	675,904.3	0.02
250-600-9-D-0.8	708,174.0	707,965	0.03	706,458	706,004.0	−0.21
250-600-9-D-1.2	834,751.0	783,561	6.53	786,359	785,986.3	0.36
250-600-9-R-0.8	998,385.8	998,169.2	0.02	997,704.6	997,628.3	−0.05
250-600-9-R-1.2	1,711,258.3	1,530,472.8	11.81	1,538,797.6	1,537,769.2	0.54
250-800-3-D-0.8	362,583.0	362,441	0.04	362,343	362,276.6	−0.03
250-800-3-D-1.2	450,379.0	414,239	8.72	414,239	414,239.0	0.00
250-800-3-R-0.8	606,342.6	604,678.8	0.28	605,170.2	602,717.9	0.08
250-800-3-R-1.2	1,139,993.6	998,606.4	14.16	998,606.4	998,606.4	0.00
250-800-6-D-0.8	668,624.0	668,454	0.03	668,348	668,231.8	−0.02
250-800-6-D-1.2	786,515.0	735,488	6.94	736,644	736,642.3	0.16
250-800-6-R-0.8	1,124,947.5	1,124,731.3	0.02	1,123,179.0	1,123,169.3	−0.14
250-800-6-R-1.2	1,762,496.4	1,561,454.2	12.88	1,564,705.6	1,564,705.5	0.21
250-800-9-D-0.8	957,210.0	956,795	0.04	956,987	956,846.8	0.02
250-800-9-D-1.2	1,174,051.0	1,103,920	6.35	1,104,828	1,104,486.6	0.08
250-800-9-R-0.8	1,324,840.0	1,324,840.0	0.00	1,323,762.1	1,323,752.6	−0.08
250-800-9-R-1.2	2,053,715.2	1,818,806.9	12.92	1,819,571.1	1,819,396.2	0.04
250-1000-3-D-0.8	460,302.0	460,045	0.06	459,842	459,842.0	−0.04
250-1000-3-D-1.2	561,064.0	516,027	8.73	516,094	516,094.0	0.01
250-1000-3-R-0.8	715,737.4	711,874.6	0.54	715,035.4	715,035.4	0.44
250-1000-3-R-1.2	816,331.9	733,049.4	11.36	733,091.1	733,091.1	0.01
250-1000-6-D-0.8	845,762.8	838,194	0.90	840,902	840,901.4	0.32
250-1000-6-D-1.2	979,761.0	908,464	7.85	919,047	919,047.0	1.15
250-1000-6-R-0.8	1,542,044.5	1,541,351.1	0.04	1,541,738.6	1,541,723.0	0.03
250-1000-6-R-1.2	1,466,381.2	1,324,902.2	10.68	1,336,454.9	1,336,454.6	0.86
250-1000-9-D-0.8	1,216,568.6	1,195,626	1.75	1,205,052	1,205,034.2	0.78
250-1000-9-D-1.2	1,432,947.0	1,333,103	7.49	1,350,841	1,350,628.5	1.31
250-1000-9-R-0.8	2,012,399.7	1,926,784.8	4.44	1,926,624.2	1,926,618.3	−0.01
250-1000-9-R-1.2	2,436,395.7	2,174,451.7	12.05	2,195,616.3	2,195,478.3	0.96

When comparing the LB and UB of the exact solver, as shown in columns 2–4 of Table 7, we find that the optimal solutions of only five instances were found in 8 h. Moreover, the average gap between the two bounds is 4.6%, while the maximum gap between the two bounds is 17.8 for instance 250-600-3-R-1.2. Analysing the BVNS solutions, as shown in columns 5 and 6 of Table 7, we find that the average BVNS solutions were equal to the best BVNS solutions for 59 instances, i.e., the BVNS algorithm always converges to the same solution, which can either be a local or a global optimum. Out of these 59 instances, the BVNS algorithm converges to 28 better, 12 equal and 19 worse solutions than the LBs found by the exact solver, respectively. Moreover, the average deviation between the best and average BVNS solutions for the 108 instances, $\frac{Best-Average}{Best}\times 100$, did not exceed 0.02%. This small deviation shows the robustness of the BVNS algorithm in solving the RSP.

A positive deviation in column 7 of Table 7 means that the best BVNS solution is better than the LB solution. The BVNS algorithm found better solutions for 62 instances, more than half of the instances. Conversely, the LB solutions were better for 29 instances, and the same solutions were found by both algorithms for 17 instances. We statistically check the significance of the BVNS results using the Wilcoxon signed-rank test. We compare the best BVNS solutions to the LB solutions found by *CPLEX12.10*. The P-value of the test is 0.009, which shows that there is a significant difference between the BVNS solutions and the LBs found by the exact solver. Given the solution quality and computation time, only 20 s, in addition to its simplicity in terms of explaining it for non-professional decision-makers make the BVNS algorithm a trusted algorithm to solve the RSP.

5. Conclusion and future work

In this paper, we introduce a real-life new problem caused by COVID-19 that we call RSP. This problem pertains to scheduling flights to repatriate a country's citizens when they are stranded in other countries. Decision-makers in countries that had repatriation programs must rely on their flag-carrying airlines to repatriate their citizens. Moreover, repatriated citizens must stay in quarantine locations that have limited capacity. Decision-makers must prioritize the return of citizens who could not afford to live abroad or have high necessities to return.

We model the RSP from the decision-makers perspective through a MILP. We also suggest a BVNS algorithm to solve this problem. We generate a set of 108 instances and solve these instances using an exact solver and the BVNS algorithm. When comparing the BVNS algorithm to the exact MILP, it is evident that the BVNS algorithm shows competitive results concerning time and solution quality. The BVNS algorithm was allowed 20 s to execute, compared to eight hours for the exact algorithm. The model assumes a static RSP; however, the RSP can be dynamic, and quick solutions are needed; thus, decision-makers can rely on the BVNS for high-quality solutions in a short time.

We hope that researchers may extend the proposed model to other real-life repatriation problems that countries might face. For the mathematical models, several real-life constraints were not considered—for example, countries allowing a certain number of flights per day. Another example is the scheduling of pilots and flight crews. If pilots need to stay in quarantine as do passengers, flight crews may be a significant constraint. Moreover, researchers can improve the MILP formulation or suggest new meta-heuristics to solve this problem. Last, we hope that governments can use these models to address other repatriation or evacuation problems.

CRediT authorship contribution statement

Sameh Al-Shihabi: Conceptualization, Methodology, Software, Formal analysis, Writing - original draft. **Nenad Mladenović:** Methodology, Software, Formal analysis, Writing - review and auditing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Arab News (2020). Low-skilled expat workers in Middle East worst hit as hiring drops 50% over coronavirus. https://www.arabnews.com/node/1704501/business-economy. Accessed: 2020-11-15.

Brimberg, J., Mladenović, N., Todosijević, R., & Urošević, D. (2017). General variable neighborhood search for the uncapacitated single allocation p-hub center problem. *Optimization Letters*, *11*(2), 377–388.

Chinazzi, M., Davis, J. T., Ajelli, M., Gioannini, C., Litvinova, M., Merler, S., et al. (2020). The effect of travel restrictions on the spread of the 2019 novel coronavirus (COVID-19) outbreak. *Science*, *368*(6489), 395–400.

Finaccord (2017). Global expatriates: Size, segmentation and forecast for the worldwide market. https://www.finaccord.com/Home/About-Us/Press-Releases/Global-Expatriates-Size,-Segmentation-and-Forecas. Accessed: 2020-11-15.

Fleszar, K., & Hindi, K. S. (2008). An effective VNS for the capacitated p-median problem. *European Journal of Operational Research*, *191*(3), 612–622.

Hansen, P., & Mladenović, N. (1997). Variable neighborhood search for the p-median. *Location Science*, *5*(4), 207–226.

Hansen, P., & Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-Heuristics* (pp. 433–458). Springer.

Hansen, P., & Mladenović, N. (2001). J-means: a new local search heuristic for minimum sum of squares clustering. *Pattern Recognition*, *34*(2), 405–413.

Hansen, P., & Mladenovic, N. (2003). A tutorial on variable neighborhood search. *Les Cahiers du GERAD*, ISSN 711, 2440.

Hansen, P., & Mladenović, N. (2014). Variable neighborhood search. In *Search methodologies* (pp. 313–337). Springer.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2008). Variable neighbourhood search: methods and applications. *4OR*, *6*(4), 319–360.

Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, *175*(1), 367–407.

Hashmi, S. (2020). Coronavirus threatening expat exodus from the UAE. https://www.bbc.com/news/world-middle-east-54418336. Accessed: 2020-11-15.

Haydar, N. (2020). Federal government to add 20 repatriation flights to bring Australians stranded abroad during coronavirus home. https://www.abc.net.au/news/2021-01-16/repatriation-flights-stranded-australians-coronavirus-government/13064006. Accessed: 2020-11-15.

Içduygu, A. (2020). Stranded irregular migrant workers during the COVID-19 crisis: The question of repatriation. In *COVID-19 and the transformation of migration and mobility globally. August*.

Karim, S. S. A., Tahir, F. A. M., Mohamad, U. K., Bakar, M. A., Mohamad, K. N., Suleiman, M., et al. (2020). Experience repatriation of citizens from epicentre using commercial flights during COVID-19 pandemic. *International Journal of Emergency Medicine*, *13*(1), 1–7.

Kharroubi, S., & Saleh, F. (2020). Are lockdown measures effective against COVID-19? *Frontiers in Public Health*, *8*, 610.

Krishna Kumar (2020). Sixth phase of vande bharat mission to repatriate Indian citizens from Gulf begins. https://english.alarabiya.net/News/world/2020/09/01/Sixth-phase-of-Vande-Bharat-Mission-to-repatriate-Indian-citizens-from-Gulf-begins. Accessed: 2020-11-15.

Lau, H., Khosrawipour, V., Kocbach, P., Mikolajczyk, A., Schubert, J., Bania, J., et al. (2020). The positive impact of lockdown in Wuhan on containing the COVID-19 outbreak in China. *Journal of Travel Medicine*, *27*(3), taaa037.

Lazić, J., Hanafi, S., Mladenović, N., & Urošević, D. (2010). Variable neighbourhood decomposition search for 0–1 mixed integer programs. *Computers & Operations Research*, *37*(6), 1055–1067.

Liao, K. A. S. (2020). Operation 'Bring Them Home': learning from the large-scale repatriation of overseas Filipino workers in times of crisis. *Asian Population Studies*, *16*(3), 310–330.

Lytras, T., Dellis, G., Flountzi, A., Hatzianastasiou, S., Nikolopoulou, G., Tsekou, K., et al. (2020). High prevalence of SARS-CoV-2 infection in repatriation flights to Greece from three European countries. *Journal of Travel Medicine*, *27*(3), taaa054.

Meena, S., Chan, J., Phan, T.-V., Butenko, S., Hurley, J., McGowen, P., et al. (2021). Repatriation operation in South Australia during the COVID-19 pandemic: initial planning and preparedness. *Communicable Diseases Intelligence*, *45*(10.33321).

Mladenović, M., Delot, T., Laporte, G., & Wilbaut, C. (2020). The parking allocation problem for connected vehicles. *Journal of Heuristics*, *26*(3), 377–399.

Mladenović, N., Dražić, M., Kovačevic-Vujčić, V., & Čangalović, M. (2008). General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, *191*(3), 753–770.

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097–1100.

Mladenović, N., Petrović, J., Kovačević-Vujčić, V., & Čangalović, M. (2003). Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *European Journal of Operational Research*, *151*(2), 389–399.

New South Wales Government (2020). COVID-19: Information for people returning from overseas or victoria to hotel quarantine. https://www.health.nsw.gov.au/Infectious/factsheets/Pages/hotel-quarantine.aspx. Accessed: 2020-09-30.

Pardo, E. G., Mladenović, N., Pantrigo, J. J., & Duarte, A. (2013). Variable formulation search for the cutwidth minimization problem. *Applied Soft Computing*, *13*(5), 2242–2252.

Rahimian, E., Akartunalı, K., & Levine, J. (2017). A hybrid integer programming and variable neighbourhood search algorithm to solve nurse rostering problems. *European Journal of Operational Research*, *258*(2), 411–423.

Santos, L. F. M., Iwayama, R. S., Cavalcanti, L. B., Turi, L. M., de Souza Morais, F. E., Mormilho, G., et al. (2019). A variable neighborhood search algorithm for the bin packing problem with compatible categories. *Expert Systems with Applications*, *124*, 209–225.

Simeonova, L., Wassan, N., Salhi, S., & Nagy, G. (2018). The heterogeneous fleet vehicle routing problem with light loads and overtime: Formulation and population variable neighbourhood search with adaptive memory. *Expert Systems with Applications*, *114*, 183–195.

Soylu, B. (2015). A general variable neighborhood search heuristic for multiple traveling salesmen problem. *Computers & Industrial Engineering*, *90*, 390–401.

Syal, A. (2020). India set to repatriate 90,000 citizens from the UAE. https://simpleflying.com/india-set-to-repatriate-90000-citizens-from-the-uae/. Accessed: 2020-11-15.

Tan, J. S., Goh, S. L., Kendall, G., & Sabar, N. R. (2021). A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, *165*, Article 113943.

The Jordan Times (2020). Fifth phase of repatriation of Jordanians abroad to commence on august 15. https://www.jordantimes.com/news/local/fifth-phase-repatriation-jordanians-abroad-commence-august-15. Accessed: 2020-11-15.

Todosijević, R., Mladenović, M., Hanafi, S., & Crévits, I. (2012). VNS based heuristic for solving the unit commitment problem. *Electronic Notes in Discrete Mathematics*, *39*, 153–160.

WHO (2020a). Considerations for quarantine of individuals in the context of containment for coronavirus disease (COVID-19). https://apps.who.int/iris/bitstream/handle/10665/331497/WHO-2019-nCoV-IHR_Quarantine-2020.2-eng.pdf. Accessed: 2020-11-15.

WHO (2020b). Listings of WHO's response to COVID-19. https://www.who.int/news/item/29-06-2020-covidtimeline. Accessed: 2020-11-15.

Woolson, R. (2007). Wilcoxon signed-rank test. In *Wiley encyclopedia of clinical trials* (pp. 1–3). Wiley Online Library.

Yazdani, M., Amiri, M., & Zandieh, M. (2010). Flexible job-shop scheduling with parallel variable neighborhood search algorithm. *Expert Systems with Applications*, *37*(1), 678–687.

Youssef, D., Berry, A., Ghosn, N., Zalzali, M., Fadlallah, R., Abou Abbas, L., et al. (2021). Phased repatriation of lebanese expatriates stranded abroad during coronavirus disease 2019 (COVID-19) pandemic. *Research Square*.