Schedule optimization and staff allocation for Airport security checkpoints using Guided Simulated Annealing and Integer Linear Programming

Alexis Brun, Eric Feron, Sameer Alam, Daniel Delahaye

▶ **To cite this version:**

Alexis Brun, Eric Feron, Sameer Alam, Daniel Delahaye. Schedule optimization and staff allocation for Airport security checkpoints using Guided Simulated Annealing and Integer Linear Programming. 2024. hal-04619976

HAL Id: hal-04619976

https://enac.hal.science/hal-04619976

Preprint submitted on 21 Jun 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Schedule optimization and staff allocation for Airport security checkpoints using Guided Simulated Annealing and Integer Linear Programming

Alexis Brun, Eric Feron, Sameer Alam, Daniel Delahaye

June 21, 2024

Abstract

Security checkpoints are an important matter of concern for airport operators. When functioning effectively, they not only maintain the airport overall safety at a high level, but also provide passengers with a positive airport experience. The perceived quality of service at the airport greatly depends on the time spent by passengers at the security lines. To ensure optimal safety performance, screening lines have a limited capacity of passengers they can handle. Thus, to prevent extended waiting times for passengers, airports can only adjust the number of simultaneously open check lines. The airport operator must establish optimal schedules for opening security checkpoints and allocating necessary staff. Building upon a prior study focused on predicting the flow of passengers through the security checkpoints, this paper explores simulated annealing algorithm in conjunction with a queue simulator and an integer programming algorithm to establish the most effective opening schedule for security checkpoints based on the prediction given by this previous study. The presented approach also determines the best allocation for dedicated staff based on the forecasted passenger flow. This approach limits the number of open security lines and ensures a waiting time below the maximum limit of 45 minutes set by the airport. It also complies with the work regulations that security agents are subject to.

1 Introduction

In the wake of the Covid crisis, global air traffic is on the up again [1]. With air traffic back at full capacity, large numbers of passengers are returning to airports. Airport operators are once again having to adapt their procedures to handle the influx of travelers. The quality of service perceived by passengers is directly correlated with the time spent in queues at key points around the airport. One of the major key points that the airport operator can influence are the security checkpoints. In fact, the airport decides how many security agents to hire and how many search lines to open for each security checkpoint throughout the day. In the case of Paris Charles de Gaulle airport, decisions on security agent requirements and scheduling are taken 45 days in advance. Minor adjustments can be made to the schedule on D-7 and D-1, but these are limited and very costly. At Paris Charles de Gaulle airport (CDG), and for the Aéroport de Paris (ADP) group in general, security agents represent a major expense. For the ADP group, according to our interviews with their security teams, total yearly expenditure on security agents is around $120 million. Therefore, allocating the optimal number of security agents is necessary to guarantee low waiting times at security checkpoints for passengers, while minimizing the number of security agents required. This paper investigates the use of guided simulated annealing to create an optimal opening schedule of the security checkpoints of Paris Charles de Gaulle airport (Figure 1) based on the passenger flow prediction determined with our previous method [2]. In addition we propose an optimal staff allocation on the opening schedule determined through annealing. Section 2 presents a brief state of the art, Section 3 summarizes how a security checkpoint is modeled and presents an implementation of a dedicated simulated annealing algorithm. Section 4 presents the method used to allocate the security agents to the checkpoints and Section 5 concludes and suggests future improvements.

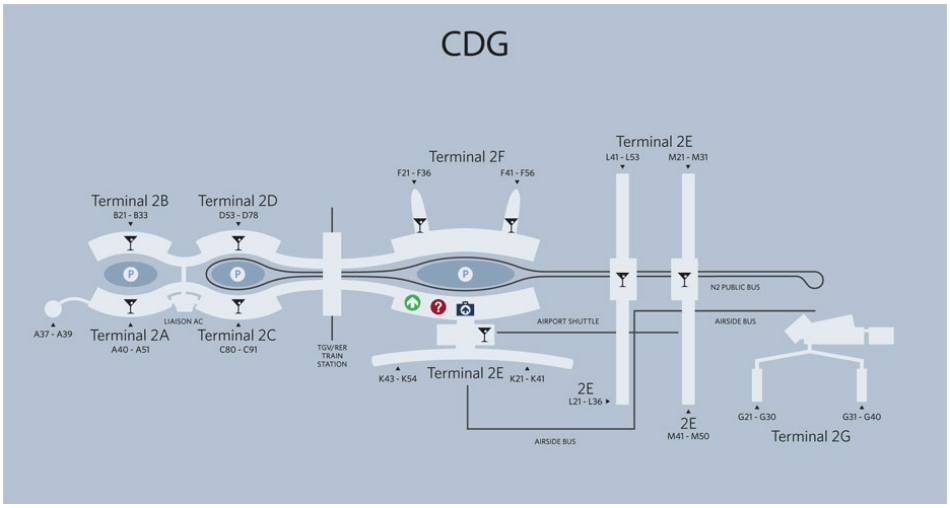

Figure 1: Overview of Paris CDG Airport. (A full detailed map of the airport including the different security checkpoints can be found in [3])

2 State of the Art

2.1 Modeling and simulation of security checkpoints

Security checkpoints play a key role in ensuring an airport functions efficiently. The effective functioning of security checkpoints is vital to provide adequate security measures while also ensuring quick processing of passengers [4]. To manage the efficiency and safety of passengers passing through a security checkpoint, it is necessary to construct a model of the checkpoint. In 2011 Leone and Liu proposed a model for the screening of passengers in the United States [5]. In their work, they modeled the time spent on each step like X-Ray scan, manual searching, explosive trace detectors etc. They measured the performance of the security checkpoint using passenger waiting time as a metric. Their objective was to reduce this time to ten minutes or less. The proposed model tends to maximize the "primary inspection" (X-Ray etc) which are faster and reduce the number of secondary inspection (explosive detector, manual search) which take longer time, without reducing the overall level of security. As shown in [5], the overall screening procedure at the security checkpoints in airport can be split into several elements. The checkpoint's operation can be altered to enhance performance. Objective evaluations are necessary since security screening is different among passengers. Business travelers and large families require different levels of attention and services at checkpoint. It is possible to implement different security policies for distinct passenger types if the airport's configuration allows for it. Simulations done by Mota et al. [6] indicate that the operational efficiency of Mexico City airport can be improved by testing a passenger categorization model. Moreover the structure of a checkpoint affects its processing capacity. Security checkpoints at airports have varying queue layouts. Li et al. [7] describe and analyze the different queue layouts and their effects on checkpoint capacity. Their findings enable the identification of the most effective checkpoint configurations for a given airport. The large number of components of a security checkpoint makes its analysis and modeling complex. This problem of simulating the behavior of a security checkpoint can be addressed using queuing theory. Zhang and al. [8] published a method based on this principle, for which, authors modeled queues of passengers between the US-Canadian border. They created a two stage queuing model, modeling the primary inspection and the secondary inspections as in [5]. They calculated the optimal proportion of secondary inspections needed to get an optimal waiting time for passengers and a checkpoint compliant with the security requirement. However, the limitations of this kind of queuing model have been shown by Stollez on different airport queues [9]. He showed that, given the very specific profile of demand in airports, standard queuing models quickly reach limitations. These limitations were also mentioned by Zhang et al [8], who indicated that if the distribution of passenger flow differs too much from a Poisson distribution, standard queuing models became less efficient. To go beyond the estimates that may be obtained using standard, but restricted, queueing models and associated "closed-form" results, it is profitable to create a full simulation of a security checkpoint to perform on it any type of optimization

work needed. Perez and al. [10] used a simulation approach to optimize the resources allocation at the different security checkpoints. In their paper the authors indicates that one of the major issues is the uncertainties of the passenger flow at security checkpoints. In the absence of a satisfactory passenger flow prediction model, they incorporate uncertainty directly into their simulation. In that way all the decisions about staff allocation are made taking into account the uncertainties in the passenger arrival model. Thanks to their method, authors achieved a great reduction of the passenger waiting time during screening compared to benchmark scenarios. In fact, a simple change in one of the components of a checkpoint can result in a very different operation of the checkpoint. To address such a change, Wilson and al. created the Security Checkpoint Optimizer (SCO) [11]. In their paper, the authors present a methodology capable of simulating a change in the screening process at a security checkpoint. With this method, airport operators can optimize their process without the need for testing it each time a change is introduced. To model the complete checkpoint, Ruiz and Cheu [12] build a model of each component, allowing the simulator to be adapted to any configuration. With their approach, they can simulate diverse scenarios and determine the performance of each checkpoint configuration such as the size of the queue or the mean waiting time. Control and estimation of queue size can be achieved iteratively through methods that include Markov chain models. These processes also take into account the incorporation of probabilities associated with the number of passengers processed over time [13]. Kierzkowski and Kisiel [14] developed a security control system model for Wroclaw Airport, determining key parameters for passengers. Their algorithm computes the average time spent by passengers at the checkpoint and simulates security agents requirements to limit queue size. Depending on the specific criteria (such as queue size or queue time), their method can be utilized to establish varying security agent schedules. They implemented their model into the management system of the security checkpoints at the airport and, according to their computational model, they were able to increase security checkpoint capacity from 750 passengers per hour to 1025 passengers per hour.

2.2 Security Staff Allocation

Once the checkpoint's operation has been properly modeled, the necessary staff must be identified to ensure its efficient operation. The allocation of security personnel to various checkpoints can be viewed as an optimization problem. To achieve cost-effectiveness, the airport management aims at minimizing its personnel. Simultaneously, the airport aims to assign adequate personnel to manage passenger traffic while ensuring that the queue size and waiting time remain reasonable and security is warranted. To efficiently distribute staff across checkpoints, accurate passenger flow predictions are necessary. Hanumantha et al. [15] utilized flight schedules and a learning model to enhance the estimate provided by the schedule. A mixed integer programming model was employed by the authors to determine the optimal configuration for operating security checkpoints and assigning security agents. Linear Integer optimization models are attractive because they can, in theory, run the optimization process to optimality through branch-and-bound methods, for example. However, their magnitude and complexity rise quickly with the number of variables. If problems become excessively large, alternate methods may be preferred. Metaheuristics can generate high-quality solutions for large problems in a reasonable amount of time. Valeva et al. [16] explain how metaheuristics are utilized to optimize workforce planning for a manufacturer. Their model enables the fast computation of quality outcomes, even for large instances that would be unmanageable with an ILP model even with the best solver available. More recently, Scozzaro and al. propose a method to optimize the staff allocation at security checkpoints in case of disruptions [17]. In this article, the authors consider the situation of an access mode disruption at Paris Charles de Gaulle airport and describe an optimization method to re-allocate the staff at the security checkpoints. Their numerical models indicate that, in case of an access mode disruption, re-allocating the staff using their approach can reduce the passenger wait time by more than 70%.

3 Security checkpoints schedule

This section presents the modeling of the operation of a security checkpoint at Paris Charles de Gaulle airport. It then presents the method used to generate security checkpoint schedules, the associated results and a comparison with the method used at the airport.

3.1 Model

At Paris Charles de Gaulle Airport, each security checkpoint is divided into screening lines, whose number depend on the size of the corresponding boarding areas. The airport operator is free to determine the desired number of lines to open for each checkpoint. The number of lines opened simultaneously during a checkpoint is dependent of the checkpoint's size and the availability of security agents at that time. At Paris CDG airport, the number of security agents to be hired for a particular day is determined 45 days in advance. This constraint is managed by forecasting passenger flow utilizing historical data. The airport currently employs statistical analyses based on their past years of operation. Several models have been developed to forecast passenger flow. Monmousseau et al. [18] introduced a method that employs recurrent neural networks. The network consists of 200 Long Short Term Memory (LSTM) cells, which are designed specifically for handling time series predictions. However, one drawback of these networks is the large amount of data required for training and the high computational power needed. In this paper, we use the predictions generated by a dense neural network model that we proposed to address this particular issue [2]. One of the primary advantages of the straightforward architecture of this compact, dense neural network is that it requires only minimal computational power to train effectively. This low computational power enables experimentation with multiple input and output scenarios to align with the requirements of the staff allocation problem. Using this network, it is possible to adjust the time step of the time series prediction to match the precision required for simulating the security checkpoint. To model passenger flow behavior at the security checkpoint, a straightforward model has been implemented. The checkpoint is modeled as a First in First Out (FIFO) queuing model and fed with a general FIFO queue. This model, adapted from the generic queuing model of Neufville et al. [19], is illustrated by Figure 2. Passengers arrive at the security checkpoint and join the general queue. They are then directed to available screening positions and wait to be processed by agents (Queue in). The next step, "Process," is the activity of agents screening the passengers. At present, processing is represented by a constant flow, or by the number of passengers that can be handled per unit of time. According to CDG operators, one screening line can process up to 120 passengers per hour. For future studies, modifications could be made to the "Process" section to account for the potentially more complex behavior of the screening process such as differentiating processing time according to passenger type, or adding uncertainty to processing time, or introducing probabilities of secondary inspections [5]. The global structure of the checkpoint model can be extended to accommodate more complex airport checkpoint configurations, including passenger differentiation approaches like those proposed by Mota et al. [6], priority or dedicated queues for first and business class passengers, and dedicated queues for connecting or delayed passengers.

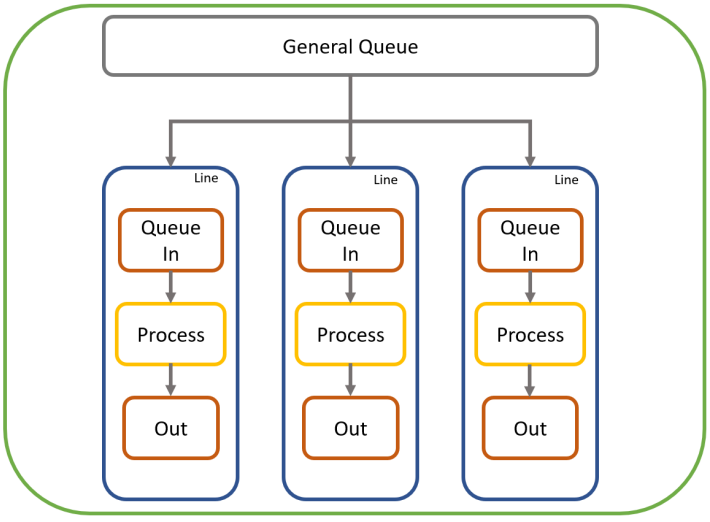

Figure 2: Model of a security checkpoint at Paris CDG airport.

The operator's decision process can be easily modeled with this security checkpoint model. At the strategic level, the operator determines the number of lines to be opened simultaneously at each security checkpoint during each hour of the day and each day of the week. Reassigning security agents

every 15, 10, or 5 minutes, for example, is not feasible. Asking an agent to change positions every 10 minutes is not only an uncomfortable working environment, but the travel time between security checkpoints makes this type of reassignment impossible. For each security checkpoint, the schedule for opening lines can be modeled as a list of 24 positive integers, with each integer representing the number of screening lines to be opened during every hour of the day. But to account for all passengers going through security on a given day a list of 24 integers is not enough. In fact, the airport's data indicates that some passengers, due to very early or very late flight hours, may go through the security checkpoint on the day before or after their flight in case it is delayed. Although most passengers arrive two hours before their flight, some arrive as early as six hours ahead or three hours late (Figure 3). Our list of 24 integers must therefore be extended to include passengers arriving between 6 p.m. on the previous day and 3 a.m. on the following day, which increases its size to 33 positive integers. Once the problem is modeled, assuming that no more than 10 screening lines are available at each checkpoint, there are 10^{33} staffing configurations. While using a very efficient FIFO simulator, the large number of possible lists prevents us from employing an exhaustive approach (Brut Force) to find the best configuration. To efficiently explore this large state space, an efficient approach can be found in metaheuristics. Metaheuristics yield favorable results on combinatorial and nonlinear optimization problems while maintaining a reasonable computational time, even on large-scale instances [20][21]. In the following, we discuss the simulated annealing method, which we will use because of its successful implementation to solve many optimization problems. [22][23][24].

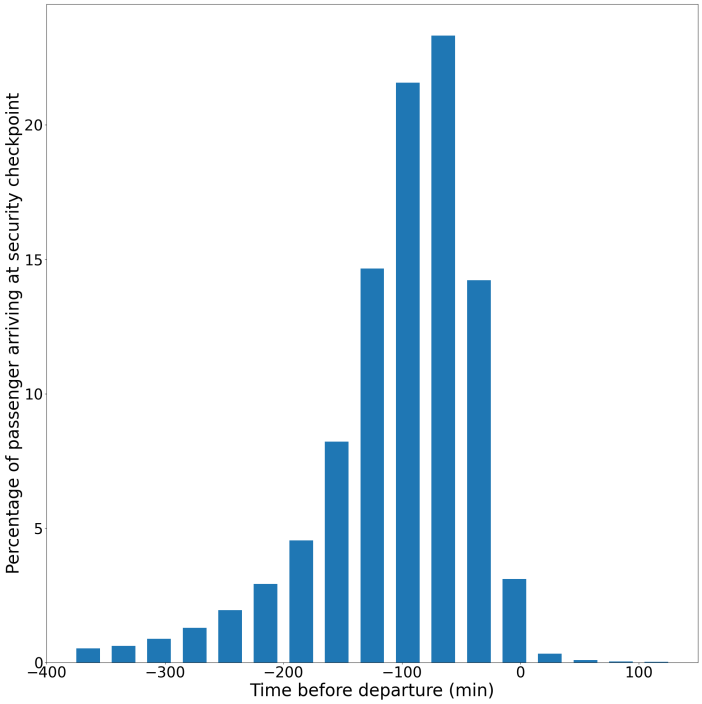

Figure 3: Percentage of passenger arriving at security checkpoint in function of the time before the departure time of their flight.

3.2 Simulated Annealing

Simulated annealing is an optimization method that draws inspiration from the process used in metallurgy, where a substance is heated and then gradually cooled to reduce its internal energy and increase its strength. (Figure 4).

This industrial process was transposed into an optimization technique, enabling the identification of extrema for a given function. Kirkpatrick et al. [26] completed this transposition in 1983. The analogy with the physical annealing method requires the minimized function to represent the material's internal energy (E) at a particular temperature (T). The methodology initiates the process from a starting point within the system and makes slight modifications to produce a new state. Once the

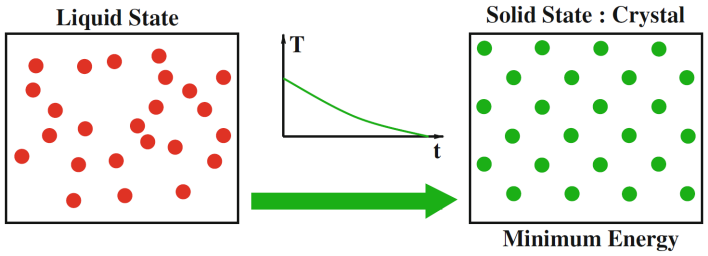

Figure 4: Diagram of a material reaching a state of minimum energy (crystalline) by slow cooling [25].

state improves the criterion, it is accepted and becomes the new current state of the system. Simulated annealing is valuable as it permits the acceptance of points in a space that deteriorate the criterion with a set probability. Metropolis, et al. [27] described the probability of accepting a deteriorated state (Equation 1). Given an initial state i of energy E_i and a new state j of energy E_j, if $E_j \leq E_i$ then j improves the criterion and it is accepted. Otherwise j is accepted with probability

$$P(j \text{ accepted}) = e^{(\frac{E_i - E_j}{k_b T})} \tag{1}$$

Where T is the temperature of the system in Kelvin and k_b is Boltzmann's constant ($k_b = 1.38 \times 10^{-23} J.K^{-1}$).

Accepting a state that degrades the criterion can prevent the optimization process to be trapped in a local minimum and allows for exploration of a larger area of the state space. The probability of accepting such a state increases with higher temperature, similar to how higher temperature leads to increased atom mobility in metals. And just like annealed metal, the algorithm's temperature T will decrease with every iteration, resulting in lower and lower acceptance probabilities. At each temperature stage, numerous neighboring operations and evaluations may be conducted. The greater the number of operations, the more extensive the exploration of the state space. Nevertheless, the higher the number, the longer the algorithm's execution time. This number of evaluations at each temperature step will be denoted as K. Under these conditions, simulated annealing is asymptotically optimal, i.e. with sufficient iteration successive solutions converge to the optimal solution [28].

3.2.1 Procedure

We are considering a state space S, an initial state $s_c \in S$, and a function f defined on S that have to be minimized. We are also considering a neighborhood function N which associates $N(s)$ with a neighbor in S of the element s. In addition, we are considering the cooling parameter λ and the number of states explored at each temperature step K.

Under these conditions the simulated annealing cooling procedure proceeds as in Algorithm 1 [25]. The ending condition of this method is a low enough temperature to stop the annealing process when reached by the cooling process. This final temperature is denoted T_f.

3.2.2 Definition of the Annealing Parameters

As described in Section 3.1, each element of the state space for security checkpoints consists of a list of 33 integers ranging from 0 to the maximum number of simultaneous screening lines available (assumed to be 10).

The neighborhood operator N is defined in Algorithm 2 and displayed on Figure 5:

Similarly, the cooling process is controlled by a geometric law with the parameter λ. This parameter can be adjusted between 0 and 1, with a smaller value of λ resulting in a faster decrease in temperature, leading to a quicker program termination. On the other hand, if it approaches 1, the program explores more states, leading to an increase in both execution time and the quality of the provided solution. By recording the final annealing temperature chosen by the user as T_f, we can calculate the total number of states explored during the process. This number is represented as n_{tot}. According to the geometric law of cooling :

$$T_{n+1} = \lambda T_n \tag{2}$$

Algorithm 1 Cooling Procedure of the Simulated Annealing

Require: s_c, T_0, T_f, N, f, λ, K
  while $T > T_f$ **do**
    for $k \leftarrow 1$ to K **do**
      $s' \leftarrow N(s_c)$
      if $f(s') < f(s_c)$ **then**
        $s_c \leftarrow s'$
      else
        $r \leftarrow random(0, 1)$
        if $e^{(\frac{f(s_c)-f(s')}{T})} \geq r$ **then**
          $s_c \leftarrow s'$
        end if
      end if
    end for
    $T \leftarrow \lambda T$
  end while
  return s_c

Algorithm 2 Neighborhood Operator

Require: s, $MaxLine$
  $index \leftarrow$ Random integer $\in [0, length(s)]$
  $s[index] \leftarrow$ Random integer $\in [0, MaxLine]$
  return s

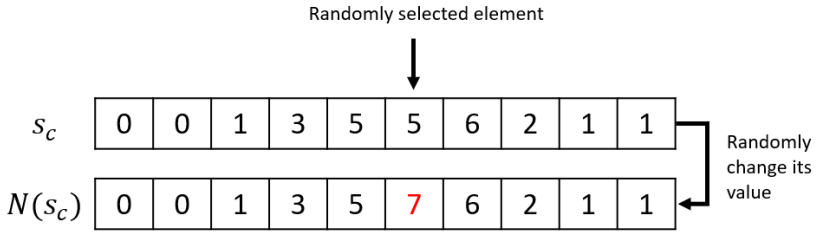

Figure 5: Neighborhood Operator application example.

With T_0 the initial temperature, T_f the final temperature and n_T the total number of temperature steps :

$$T_f = \lambda^{n_T} T_0 \tag{3}$$

And the total number of states explored is then given by :

$$n_{tot} = \frac{K(\ln(T_f) - \ln(T_0))}{\ln(\lambda)} \tag{4}$$

Knowing the machine's performance during program execution, one can choose the parameters K and λ to explore the maximum number of states while maintaining a reasonable execution time. The final temperature is set for testing at $T_f = \frac{T_0}{10000}$. With these parameters, and keeping an execution time of around 30 minutes, we end up with :

- $\lambda = 0.95$
- $K = 500$
- $n_{tot} = \frac{500(\ln(\frac{T_0}{10000}) - \ln(T_0))}{\ln(0.95)} \approx 89781$

The initial temperature T_0 should be sufficiently high to allow for broad state space exploration during annealing. Nevertheless, as the initial temperature increases, so does the final temperature. Acceptance of states that violate the criterion may occur at a high final temperature when it is no longer needed. To identify the initial temperature, a deliberately low starting temperature should gradually increase until a sufficiently high state acceptance rate is attained. This procedure for "heating" is outlined in Algorithm 3 and illustrated in Figure 6

Algorithm 3 Heating process

Require: s, T, N, f, K, $AcceptRate$
  $r \leftarrow 0$
  while $r < AcceptRate$ **do**
    $count \leftarrow 0$
    for $k \leftarrow 1$ to K **do**
      $s' \leftarrow N(s)$
      if $f(s') < f(s)$ **then**
        $count \leftarrow count + 1$
      else
        if $e^{(\frac{f(s)-f(s')}{T})} \geq 0.5$ **then**
          $r \leftarrow r + 1$
        end if
      end if
    end for
    $r \leftarrow \frac{count}{K}$
    $T \leftarrow 1.5 \times T$
  end while
  return T

The objective function utilized in annealing aims to simulate solution quality from the perspective of airport operations. A quality solution from the operator's standpoint involves a schedule that reduces passenger wait times while also limiting the number of simultaneously opened lines. The maximum time that passengers spend in the queue can be determined using the First in First Out (FIFO) simulator, as described in section 3.1. The objective function, which represents the quality of the solution, can be calculated in a naive manner as

$$f(s) = \sum_{i \in s} i + FIFO(s) \tag{5}$$

where s denotes the state proposed by the annealing algorithm, in our case a list of 33 integers described in Section 3.1. i represents the number of simultaneously opened lines at a time step of s, and $FIFO(s)$

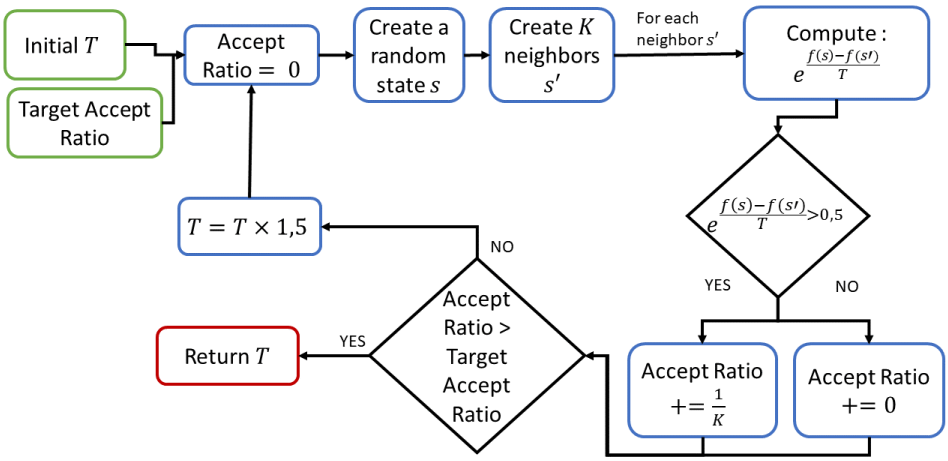

Figure 6: Diagram of the simulated annealing heating procedure

stands for the maximum time that passengers spend in the security checkpoint queue, calculated by the simulator using an opening schedule described by s. However, the current model assigns equal weight to both an open security line and an extra minute of passenger waiting time, but assigning different weights to each part of the objective function is possible. Subsequent discussions with airport operators revealed a better model that emphasizes passengers' waiting times within a specified range of acceptability, rather than solely minimizing them. The maximum allowable waiting time for a passenger is a parameter determined by the airport based on various factors. These factors include the average connecting time, airport layout, and number of passengers. For modeling purposes, we will refer to this maximum allowable waiting time as $T_{\max}$. Taking into account this new parameter, the objective function for the annealing process is

$$f(s) = \sum_{i \in s} i + \alpha \max(0, FIFO(s) - T_{\max}). \tag{6}$$

An extra parameter, α, has been incorporated into the objective function. This parameter allows adjustment for penalizing overtime $T_{\max}$ to a greater or lesser extent. For the remainder of the study, the value of the α parameter will be set to 100. In this form, the function enables the annealing process to initially decrease the waiting time below the $T_{\max}$ threshold before restricting the number of simultaneously opened security lines.

To begin the simulated annealing process, the initial state s_c must be defined for the simulated annealing algorithm to generate neighboring states and continue the optimization process. While initial states are typically generated randomly, a different approach can yield better results. Each security checkpoint has 10^{33} possible states, but with the parameters specified in Section 3.2.2, equation 4 gives us that only $n_{tot} \approx 89781$ states will be explored. To limit the exploration of trivially bad states and ultimately obtain a better-quality solution, it is advisable to start the simulated annealing from a good starting point rather than a randomly generated one. To be of interest, however, the starting point needs to be determined very quickly. A greedy heuristic has been used to efficiently determine the starting point for the annealing. The heuristic starts by opening all security lines at every time step, and then gradually closes them until the $T_{\max}$ limit is exceeded. Algorithm 4 provides a detailed description of this heuristic.

3.3 Results

For the numerical experiments, the simulated annealing algorithm was executed based on the parameters described in the preceding section. The chosen day is derived from our prior work's neural network test dataset [2]. The initial point of the annealing process achieved through Algorithm 4 is depicted in Figure 7. As simulated annealing is a non-deterministic algorithm, the outcomes may vary from one execution to another. Table 1 presents the results acquired from ten trials compared with the greedy heuristic, which, being deterministic, gives the same result from one run to the next. Table 2 displays the mean and standard deviation of results and execution time. The schedule for opening the security

Algorithm 4 Greedy Heuristic for Annealing Start Point Creation

Require: $MaxLine$, $MaxTimeStep$, $FIFO$, $T_{\max}$
  $s \leftarrow []$
  for $k \leftarrow 1$ to $MaxTimeStep$ **do**
    $s+ = MaxLine$
  end for
  $s' \leftarrow s$
  while $FIFO(s') < T_{\max}$ **do**
    $index \leftarrow IndexMax(s')$
    $s'[index] \leftarrow s'[index] - 1$
  end while
  $s'[index] \leftarrow s'[index] + 1$
  $s \leftarrow s'$
  return s

18:00	19:00	20:00	21:00	22:00	23:00	0:00	1:00	2:00	3:00	4:00	5:00	6:00	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	0:00	1:00	2:00
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
4	4	4	4	4	4	4	4	4	4	4	4	4	4	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
3	3	3	3	3	3	3	3	3	3	3	3	3	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3

Figure 7: Example of Start Schedule for the Annealing created with the Greedy Heuristic.

checkpoint, obtained through simulated annealing, is displayed in Figure 8. The open lines at a given time step are shown as green boxes, the others are shown as red boxes. Each box also contains the number of lines to be open at that time step.

3.3.1 Creation of security line opening schedules

18:00	19:00	20:00	21:00	22:00	23:00	0:00	1:00	2:00	3:00	4:00	5:00	6:00	7:00	8:00	9:00	10:00	11:00	12:00	13:00	14:00	15:00	16:00	17:00	18:00	19:00	20:00	21:00	22:00	23:00	0:00	1:00	2:00
0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	1	1	1	1	2	1	1	1	0	1	2	2	2	1	0	0	0	1
0	0	0	0	1	0	0	0	0	1	2	1	0	1	1	3	2	2	1	1	2	2	1	2	1	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	1	2	2	1	2	2	1	2	1	3	1	3	1	2	1	1	2	0	0	0	0	2	1
0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1
0	0	0	0	0	0	1	0	0	0	1	1	1	1	1	2	1	1	1	0	1	1	1	1	0	0	1	0	0	0	0	1	0
0	0	0	1	0	0	0	0	1	1	1	1	2	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
1	0	2	2	0	0	4	0	0	2	1	1	1	3	7	7	7	6	7	4	2	5	3	3	2	5	4	1	5	2	0	0	1
0	2	0	1	1	0	0	1	1	1	0	4	5	4	6	6	6	6	6	6	3	1	1	0	3	1	2	4	1	1	0	1	0
1	0	0	3	0	0	0	0	2	1	1	6	6	6	6	5	5	2	4	3	2	4	3	3	4	4	5	2	0	3	0	0	0
0	0	0	0	0	1	1	0	0	1	1	5	4	5	5	5	3	4	4	3	2	3	2	2	3	5	2	1	1	1	1	0	0
0	1	1	0	0	0	0	1	1	1	1	0	1	2	2	3	3	1	1	3	1	0	1	1	2	2	1	1	0	0	0	1	0
0	0	0	0	0	0	1	0	1	1	1	1	2	1	2	1	3	1	1	1	2	1	0	1	1	1	1	0	0	0	0	0	1
1	1	0	0	0	0	0	0	0	1	1	1	1	4	4	2	2	3	3	1	3	1	1	3	1	3	1	0	0	0	1	0	1

Figure 8: Example of Security Checkpoint Opening Schedule Created via Simulated Annealing.

Table 2 indicates that the standard deviations produced by simulated annealing are consistently low. This observation suggests that the optima obtained via simulated annealing are consistent across experiments. Additionally, this method enables a quick determination of the number of lines that need to be opened at various airport security checkpoints. Moreover, we observe that enforcing a strong penalty for exceeding $T_{\max}$ in the cost function has allowed us to uphold the corresponding constraint in a systematic manner. Overall, we believe that simulated annealing produces a functional timetable for the security checkpoints at the airport. This timetable can subsequently serve as a constraint in our study to establish specific schedules for each security team.

3.3.2 Comparison of simulated annealing results with airport operational planning practices

At Paris Charles de Gaulle Airport, the opening hours for security checkpoints are scheduled according to the following procedure. Airport operators create a prediction with a one-hour time step for the

	Simulated Annealing			Greedy Heuristic		
Execution Time (s)	**Max Waiting Time (min)**	**Total Open**	**MaxSimult**	**Max Waiting Time (min)**	**Total Open**	**MaxSimult**
976	45	507	31	45	887	32
970	45	495	36			
961	45	494	32			
944	45	494	32			
943	45	501	34			
933	45	510	34			
963	45	491	33			
954	45	496	33			
1015	45	512	34			
983	45	494	30			

Table 1: Outputs of the Simulated Annealing and the greedy Heuristic for 10 runs. Total Open corresponds to the total number of security lines open during the day and MaxSimult corresponds to the maximum number of lines open simultaneously.

	Execution Time (s)	**Max Wait Time (min)**	**Total Open**	**MaxSimult**
Mean	964,2	45	499,4	32,9
Standard Deviation	23,70	0,00	7,60	1,73

Table 2: Mean and Standard Deviation of the outputs of the simulated annealing for 10 runs.

passenger flow, and consider a fixed handling capacity of 120 passengers per hour for each security line. The security line opening schedule is then created to meet the demand for passenger flow. Generally, the flow of passengers over an hour is not evenly divisible by 120. Therefore, the number of open lines is always rounded up to adequately meet the demand. The application of this method for one security checkpoint is illustrated on Figure 9.

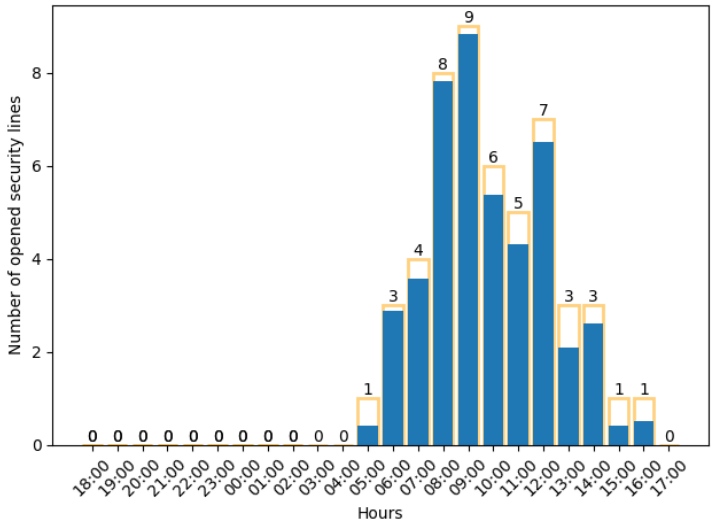

Figure 9: Illustration of CDG method to determine the number of security lines to open for a checkpoint. The number of lines needed with the fixed capacity of 120 pax/hour is displayed in blue and the number of opened lines rounded up is displayed in orange and written above the bar

The airport's announced security line handling capacity appears overly optimistic when compared to observed reality. A processing rate of 120 passengers per hour corresponds to an ideal situation, but in reality, this capacity frequently experiences disruptions. For instance, passengers who are unfamiliar with the screening process may have kept a metallic item with them. Additionally, even when operations run smoothly, random additional checks are conducted. The secondary inspections, such as palpation and detection of traces of explosives, described by Leone and Liu [5], require longer processing time and monopolize the security agents' time, thereby reducing their capacity to manage the flow of passengers. Consequently, the actual processing capacity of a security line falls short of the ideal limit of 120 passengers per hour currently used to plan the opening of security check lines. For the purposes of this study, we utilize a processing capacity of eight passengers per five-minute interval.

Consequently, a security line has the potential to process a total of 96 passengers within the span of one hour. This method, processing 96 passengers per hour, was employed on the same test day as that of the simulated annealing approach. An example of a timetable obtained by employing the prevailing methodology at the airport is depicted in Figure 10. A comparison of the opening schedules generated by the straightforward method of the airport and the simulated annealing method is presented in Table 3. The data was obtained by using the security checkpoint simulator (FIFO) on both schedules. One can see from Figure 10 that the rise in the number of lines opened is reflected in the maximum number of open lines at the same time. By applying the queue simulation to the airport schedule, it becomes obvious that the maximum waiting time constraint is not met by the schedule. Specifically, one passenger waited for over 2 hours, and such an extended waiting period can result in passengers missing their flights. Table 3 also shows that the simulated annealing gives a schedule that keeps the maximum waiting time below 45 minutes. This reduction is achieved by slightly increasing the total number of lines to be opened during the day by 6%.

	Max Wait Time (min)	**Total Open**	**MaxSimult**
Airport Method	125	470	46
Simulated Annealing	45	499	32
Improvement	64%	-6%	30%

Table 3: Performance comparison of opening schedules created using the airport method and the simulated annealing method.

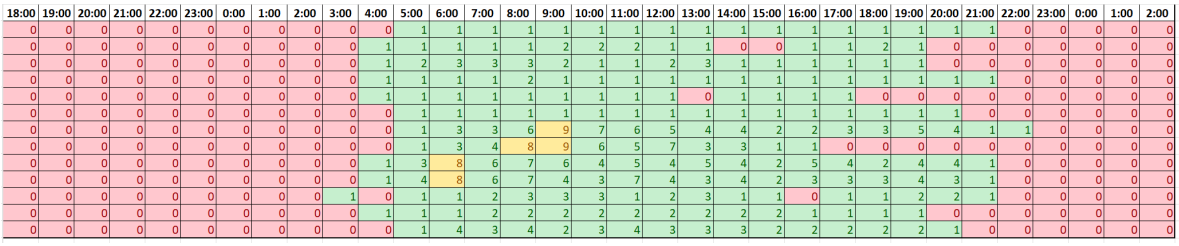

Figure 10: Example of Security Checkpoint Opening Schedule Created with the Method of CDG Airport with a capacity of 96 passenger per hour per line. Red boxes indicate when the checkpoint is closed, while the green boxes indicate when it is open and the number of simultaneous lines allowed. The yellow boxes represent time steps with a higher number of simultaneously open lines than obtained via simulated annealing.

With the schedule for opening security checkpoints now established, the subsequent stage of the research will involve accommodating requests to open the checkpoints while adhering to the constraints associated with the security agents' schedules.

4 Security Agent Optimal Scheduling

The simulated annealing results can give the number of security checkpoint lines that must be opened on a given day. However, security personnel must be assigned to operate these security checkpoints while meeting several additional constraints, including maximum daily working hours and minimum break durations. To address this issue, we formulate and solve an integer linear programming model.

4.1 Model

In France, security agents are limited to a 12-hour workday and must take a break of at least 20 consecutive minutes every 6 hours [29]. They are also limited to 48 hours of working hours per week. There may be additional legal constraints, depending on the way the security company operates and any agreements it has with the unions. Despite being legally mandated, these conditions are not consistently enforced. The detrimental effects of 6 hours of uninterrupted work and 12 hours working day on a security agent's ability to concentrate can have consequences for airport overall security.

Dealing with every constraints individually for every security agent at the airport can quickly increase the model's complexity. To address this issue, a standard shift of 4 hours of work, followed by a 1-hour break and then 4 more hours of work is proposed. The structure of this standard shift is displayed on Figure 11. In addition, following interviews with our partners at Paris Charles de Gaulle airport, the

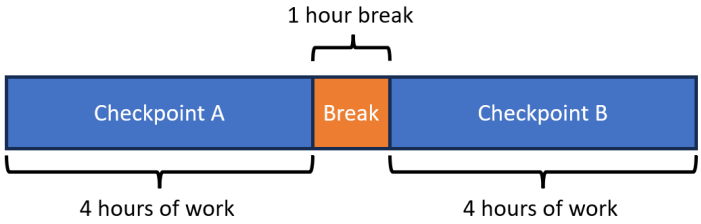

Figure 11: Example of the structure of the shifts used in the model.

suggested shifts largely align with established airport protocols.
For each security checkpoint p, the simulated annealing process provides the minimum number of security teams required to handle passenger flow for each time interval t. This number is denoted as N_{pt}. For shifts meeting the above criteria, we generate a three-dimensional array S such that any element $S(i,p,t) \in S$ equals 1 if shift i assigns a team of agents to position p at time t, and 0 otherwise. Finally, we establish integer decision variables n_s, denoting the number of shifts s necessary to meet the demand N_{pt}. Noting the sets of security checkpoints P, time steps T, and the total number of created shifts N_{Shifts}, the optimization model can be formulated as :

$$\begin{aligned} \min \quad & \sum_{s\in[1,N_{\text{Shifts}}]} n_s \\ \text{s.t.} \quad & \sum_{s\in[1,N_{\text{Shifts}}]} n_s S(s,p,t) \geq N_{pt}, \quad \forall p \in P, \forall t \in T, \\ & n_s \geq 0, \quad s = 1,\dots,N_{\text{Shifts}} \end{aligned} \tag{7}$$

Note that the total number of shifts generated can be computed directly from the number of security checkpoints P, the number of time steps T, and the criteria for an acceptable shift with D the duration of a shift. This value is given by the following expression:

$$N_{\text{Shifts}} = P^2 \times (T - ((D-1)) \tag{8}$$

The solver used to solve this problem is Gurobi version 10.0.2.

4.2 Results

The method is tested on a randomly selected day from the neural network's test dataset used to forecast passenger flow [2]. This dataset contains the complete list of flights at Paris Charles de Gaulle airport for every day of the year. For each flight, the dataset contains its theoretical departure time and the boarding hall from which it is scheduled to depart, allowing us to determine which security checkpoint will be needed to manage its passengers. The dataset also contains information derived from the chosen day, allowing to determine whether the day in question is a holiday, weekend, etc. This dataset provides good predictive performance for passenger flow at the strategic level with our neural network, but cannot help predict tactical changes such as schedule adjustments due to flight delays or connection disruptions. At this strategic level, the airport can only rely on predictive models to determine the number of passengers it will have to handle and therefore the number of lines of security checkpoints to open, which is why we use the prediction of our neural network to test our method. The minimum number of security teams required is obtained through the simulated annealing, described in section 3.2. The day used for our experiment had 13 security checkpoints (P), 33 time steps (T), and used shifts in the format depicted in Figure 11. In total, there were $N_{\text{Shifts}} = 4225$ possible shifts, according to equation 8. As shown in Figure 12, the algorithm can generate a timetable that allocates sufficient personnel to different checkpoints. Nonetheless, the input data coming from predictions generated by the neural network, has limited accuracy and may diverge from the actual passenger flow. To compare

the obtained results with the actual passenger flow at the airport on that day, a second simulation was conducted using the observed flow. Simulated annealing was used to redetermine the required number of security agents based on this data. The corresponding schedule for the security agents was then created using the linear programming algorithm. The additional curve in Figure 12 represents the schedule of security agents created from the second simulation run on the observed flow on that day.

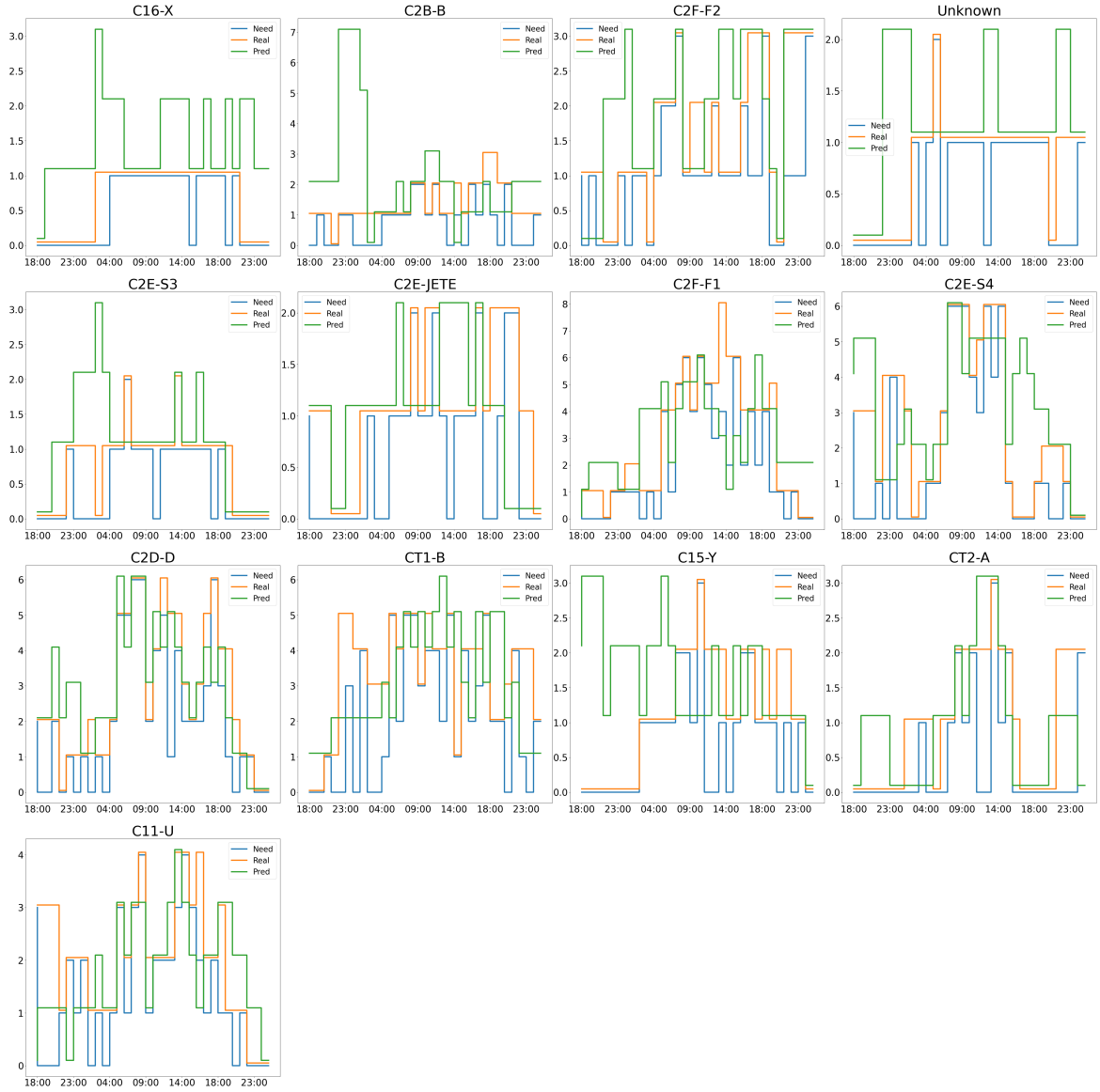

Figure 12: Need of security teams (blue) vs assigned security teams by the schedule created with the predicted flow (green) vs assigned security teams by the schedule created with the observed flow (orange) on a tested day for each of the boarding rooms of CDG airport. (To avoid overlapping, a minor offset is applied to each curve.)

Although we can see directly from Figure 12 whether a sufficient number of security agent teams have been assigned to meet the demand at each checkpoint, it is also necessary to look at the cumulative demand over the day in the entire airport. In fact, this prediction of the number of security agents is made at the strategic level, several weeks in advance. However, on the day of operation itself, since the neural network's prediction cannot be perfect, there is a gap between the need for security agents and our planning. The question is whether overestimating the need for agents at certain checkpoints can compensate for underestimates at other checkpoints. After all, if there is a shortage of agents at

one checkpoint at a given time, but there is an excess of agents elsewhere, it will always be possible to reassign those agents at the tactical level. Therefore, we need to check whether there are enough security teams at the airport at any given time of the day. Figure 13 displays a comparison of the total demand for security agent teams and the actual number of teams present at the airport on the day of the test. Figure 13 shows that the proposed method achieves an adequate number of security teams despite varying constraints such as working time and break schedules.

Nonetheless, our prior research found that the neural network passenger flow prediction tends to be underestimated [2]. Therefore, simulated annealing allows opening security checkpoints to be based on an underestimated passenger flow. This issue is illustrated in Figure 13, which shows two instances of missing agent teams. The discrepancy is minor and distributed across multiple checkpoints, as demonstrated by Figure 12. To validate these observations, several tests were conducted over ten consecutive days in February 2023. The method presented was applied on each of these days to predict passenger flow using our neural network. Security line opening schedules were created based on these predictions, and corresponding security agent schedules were established. Table 4 compares the differences in cumulative demand for security agents over these 10 days. This table displays the differences between the number of security agents determined using our model and the actual number required, as determined by the model. The table illustrates that the program's predicted number of required teams closely matches the actual number. The largest relative gap, at 9%, occurred on 2023/02/09. Thus, it is reasonable to conclude that this predictive error would have only a minor impact on operations. Finally, based on the increase achieved through the simulated annealing compared to the airport's previous method, additional teams can be planned without significantly diminishing this improvement.

	Required Security Team	**Maximum Difference**
01/02/2023	100	-5
02/02/2023	95	-4
03/02/2023	95	-4
04/02/2023	93	-3
05/02/2023	106	-2
06/02/2023	87	-4
07/02/2023	78	-5
08/02/2023	86	-6
09/02/2023	91	-8
10/02/2023	91	-9
Mean	**92,2**	**-5**
Standard Deviation	**7,72**	**2,16**

Table 4: Maximum difference observed between the cumulative demand of security teams and the number provided using the method over ten consecutive days of February 2023.

5 Conclusion and Perspectives

The study concludes that effective scheduling of security agents can alleviate passenger congestion at Charles de Gaulle airport. A process based on predictions of the passenger flow via a neural network generates a schedule for to schedule staffing of security checkpoints. This process relies on simulated annealing. Simulated annealing can generate a schedule for opening security checkpoints, ensuring a waiting time below a predefined threshold. This optimal opening schedule serves as a constraint for an ILP algorithm that assigns security teams to various checkpoints at the tactical level while ensuring compliance with constraints related to legal work time of the agents. The number of security agents required using this method is significantly lower than that obtained by the airport's empirical method. However, since the neural network's predictions cannot be perfect, there may be times when the number of predicted agents is insufficient to handle the passenger flow. This limitation presents an opportunity for future studies. The neural network could be tuned and its parameters tuned to reduce underestimation of passenger flow. Additionally, refining the simulated annealing parameters and enhancing the execution time may ultimately lead to a more optimal security screening line opening schedule. Further research could be conducted to assess the reliability of this method over

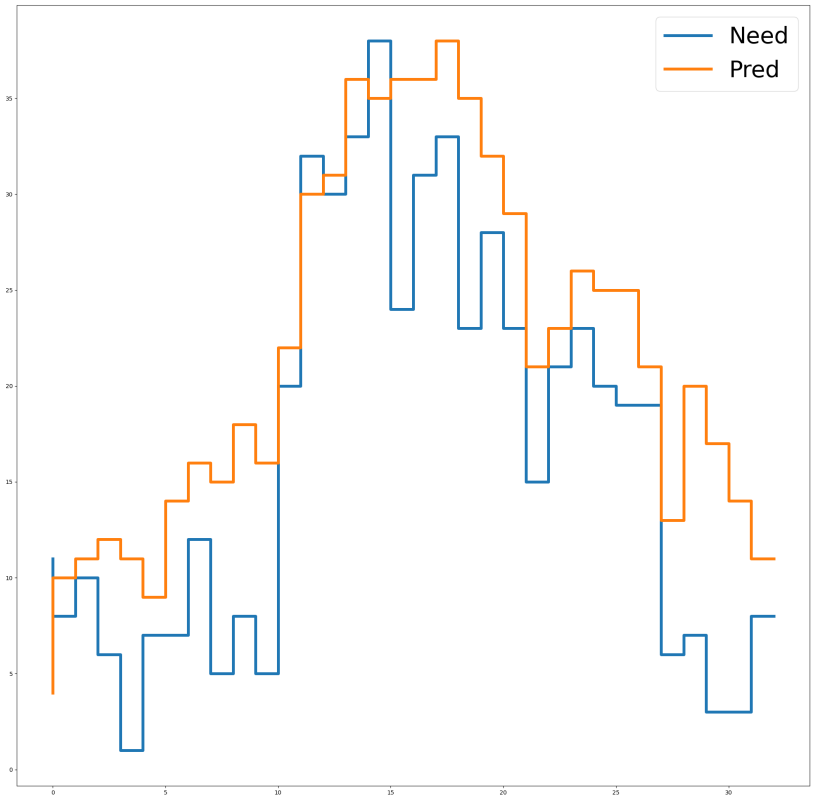

Figure 13: Cumulative demand of security agents teams for the whole airport compared to the number of security agents teams available on a tested day.

an extended period, especially on days affected by disruptions such as strikes or adverse weather conditions. Additional performance metrics may be developed to measure the quality of a security checkpoint schedule. Furthermore, the security checkpoint simulator can be modified to reflect more complex passenger behavior and security procedures. Finally, this approach could be validated at the airport, specifically on designated security checkpoints, and eventually throughout the entire airport.

References

[1] International Air Transport Association. Global outlook for air transport. Technical report, December 2023.

[2] Alexis Brun, Daniel Delahaye, Eric Feron, and Sameer Alam. Predicting passenger flow at Charles de Gaulle airport using dense neural networks. *Proceedings of International Workshop on ATM/CNS*, 1:25–32, 2022.

[3] Groupe ADP. Cdg terminal map, 2018. https://easycdg.com/airport-guide/maps-terminal-paris-cdg-airport-charlesdegaulle/ [Accessed: (04-01-2024)].

[4] Arthur Knol, Alexei Sharpanskykh, and Stef Janssen. Analyzing airport security checkpoint performance using cognitive agent models. *Journal of Air Transport Management*, 75:39–50, mar 2019.

[5] Kelly Leone and Rongfang (Rachel) Liu. Improving airport security screening checkpoint operations in the US via paced system design. *Journal of Air Transport Management*, 17(2):62–67, mar 2011.

[6] Miguel Mujica Mota, Paolo Scala, Alejandro Murrieta-Mendoza, Angel Orozco, and Alejandro Di Bernardi. Analysis of security lines policies for improving capacity in airports: Mexico city case. *Case Studies on Transport Policy*, 9(4):1476–1494, 2021.

[7] Yongli Li, Xin Gao, Zhiwei Xu, and Xuanrui Zhou. Network-based queuing model for simulating passenger throughput at an airport security checkpoint. *Journal of Air Transport Management*, 66:13–24, jan 2018.

[8] Zhe George Zhang, Hsing Paul Luh, and Chia-Hung Wang. Modeling security-check queues. *Management Science*, 57(11):1979–1995, nov 2011.

[9] Raik Stolletz. Analysis of passenger queues at airport terminals. *Research in Transportation Business & Management*, 1(1):144–149, aug 2011.

[10] Eduardo Perez, Logan Taunton, and Jorge A. Sefair. A simulation-optimization approach to improve the allocation of security screening resources in airport terminal checkpoints. In *2021 Winter Simulation Conference (WSC)*. IEEE, dec 2021.

[11] Diane Wilson, Eric Roe, and S. So. Security checkpoint optimizer (SCO): An application for simulating the operations of airport security checkpoints. In *Proceedings of the 2006 Winter Simulation Conference*. IEEE, dec 2006.

[12] Emiliano Ruiz and Ruey Long Cheu. Simulation model to support security screening checkpoint operations in airport terminals. *Transportation Research Record: Journal of the Transportation Research Board*, 2674(2):45–56, jan 2020.

[13] Francesco Chiti, Romano Fantacci, and Andrea Rizzo. An integrated software platform for airport queues prediction with application to resources management. *Journal of Air Transport Management*, 67:11–18, mar 2018.

[14] Artur Kierzkowski and Tomasz Kisiel. Simulation model of security control system functioning: A case study of the wroclaw airport terminal. *Journal of Air Transport Management*, 64:173–185, sep 2017.

[15] Girish Jampani Hanumantha, Berkin T. Arici, Jorge A. Sefair, and Ronald Askin. Demand prediction and dynamic workforce allocation to improve airport screening operations. *IISE Transactions*, 52(12):1324–1342, may 2020.

[16] Silviya Valeva, Mike Hewitt, and Barrett W. Thomas. A matheuristic for workforce planning with employee learning and stochastic demand. *International Journal of Production Research*, 55(24):7380–7397, jul 2017.

[17] Geoffrey Scozzaro, Catherine Mancel, Daniel Delahaye, and Eric Feron. Optimising security screening resources during airport access mode disruptions. In *SESAR Innovation Days*, Budapest, Hungary, December 2022.

[18] Philippe Monmousseau, Gabriel Jarry, Florian Bertosio, Daniel Delahaye, and Marc Houalla. Predicting passenger flow at Charles de Gaulle airport security checkpoints. In *2020 International Conference on Artificial Intelligence and Data Analytics for Air Transportation (AIDA-AT)*. IEEE, feb 2020.

[19] Richard de Neufville, Amedeo R. Odoni, Peter P. Belobaba, and Tom G. Reynolds. *Airport Systems: Planning, Design, and Management*. McGraw-Hill Education, New York, 2nd edition edition, 2013.

[20] Michel Gendreau and Jean-Yves Potvin. *Handbook of Metaheuristics*. Springer Publishing Company, Incorporated, 2nd edition, 2010.

[21] Kashif Hussain, Mohd Najib Mohd Salleh, Shi Cheng, and Yuhui Shi. Metaheuristic research: a comprehensive survey. *Artificial Intelligence Review*, 52(4):2191–2233, jan 2018.

[22] Rosmalina Hanafi and Erhan Kozan. A hybrid constructive heuristic and simulated annealing for railway crew scheduling. *Computers & Industrial Engineering*, 70:11–19, apr 2014.

[23] Wayan Mahmudy. Improved simulated annealing for optimization of vehicle routing problem with time windows (vrptw). *Kursor*, 7:109–116, 01 2014.

[24] Arianit Islami, Supatcha Chaimatanan, and Daniel Delahaye. Large-scale 4D trajectory planning. In *Lecture Notes in Electrical Engineering*, pages 27–47. Springer Japan, 2017.

[25] Daniel Delahaye, Supatcha Chaimatanan, and Marcel Mongeau. Simulated annealing: From basics to applications. In *Handbook of Metaheuristics*, pages 1–35. Springer International Publishing, sep 2018.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, may 1983.

[27] Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, jun 1953.

[28] V. Granville, M. Krivanek, and J.-P. Rasson. Simulated annealing: a proof of convergence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):652–656, June 1994.

[29] Direction de l'information légale et administrative. Convention collective nationale des entreprises de prévention et de sécurité du 15 février 1985. etendue par arrêté du 25 juillet 1985. In *Journal officiel de la République française*. Jul 30 1985. https://www.legifrance.gouv.fr/conv_coll/id/KALISCTA000005725201/?idConteneur=KALICONT000005635405&origin=list.