Airline Disruption Management with Aircraft Swapping and Reinforcement Learning

Gabriel Hondet, Luis Delgado, Gérald Gurtner

Airline disruption management with aircraft swapping and reinforcement learning

Gabriel Hondet, Luis Delgado, Gerald Gurtner

December 7, 2018

Abstract

Managing fleet disruption is essential for an airline to control delay costs. Delays emerging from these disruptions can be manipulated through fleet operations like aircraft swapping. This paper applies machine learning techniques to the disruption problem. While airlines might do this process manually or using basic predefined rules, the complexity of the problem makes it well suited for a computed approach. The paper describes the principles of reinforced learning and the model used for testing them. Two representations of decision states are considered and applied to a set of historical schedules for an airline. The performance obtained by swapping aircraft using the reinforced learning is finally compared to the idle option, i.e., do not swap any flight. The comparison evinces that while the algorithm is far from being optimal, the agent takes relevant decisions as it performs better than the idle behaviour in heavily disrupted simulations.

1 Introduction

The design of a schedule is strategic as it happens months in advance of its implementation. Since schedules try to maximise profit, they are subject to an efficiency-resilience trade off. A schedule is here said to be resilient if it is not too sensitive to unexpected events, i.e., the probability that the carried out operations deviate in time from what was scheduled is low. The lower the buffer between operations, the higher the cost efficiency but the lower the resilience. In contrast, a schedule allowing more time between operations is resilient but not efficient. To make profit, airlines will therefore try to minimise delays between operations, and thus expose themselves to disruption problems [10, 28].

Costs of disruption problems often rise due to the reactionary delays triggered. To lower these costs, airlines can modify the schedule in real time. However, the number of re-configurations makes the problem highly combinatorial and thus better suited for computed solutions than human found ones. Airlines could modify the trajectory of flights, e.g., selecting a different route or modifying their operating cost index [13]; actively wait for passengers so that they

don't miss their connections [14]; swap aircraft [31]; cancel flights and in some cases ferry aircraft [6]. Commercial tools can be used to create and adjust the required flight plans such as Lufthansa Lido or airline recovery management tools by Sabre.

In this paper, the reconfiguration problem that arises from disruptions in the schedule is solved via reinforcement learning. Once a schedule has been built, it is given to a simulator and an agent. The agent will try to lower a cost function by swapping aircraft to cope with disruptions created by the simulator.

The goal of this paper is to present how reinforcement learning techniques could be used in the disruption management problem. The agent will be trained by a basic reinforcement learning algorithm, the Q learning algorithm.

The article is organised as follows. Section 2 presents a literature review on models of disrupted management. It also presents an overview of the reinforcement learning paradigm. Section 3 describes the model used in this paper including a description of the learning mechanism. Section 4 explains in detail the experimental setup used to test the model. Section 5 shows the results obtained with the model on the experimental setup. Finally, Section 6 draws some conclusions and looks at future work.

2 Literature review

2.1 Disruption management

[31] and [21] provide a survey of different actions that might be considered by airlines to manage disruptions. They include among others: swap aircraft, cancel flight, delay flights, use stand-by aircraft. Studies to minimise the impact of disruptions often only handle a subset of all possible actions. Most articles use aircraft swapping, flight cancelling, flight re-timing and ferrying. Some works do not consider ferrying, as [19], stating that ferrying is very seldom used. Some models would also allow to use stand by aircraft such as in [22].

It is well understood that cost of delay is not linear in function of delay [10, 15]. The immediate consequence is that cost can be lowered by dividing delay among more flights rather than having it concentrated, i.e., in average it is less costly two flights delayed 20 minutes than one flight delayed 40 minutes.

To analyse the different techniques that try to minimise disruption, an air traffic model is required. This is relevant as knock-on effects, i.e., reactionary delay, needs to be considered. Several graph representations have been proposed. For a exhaustive review see [7]:

- Connection network, in which each node is a flight leg and two nodes u, v are connected if leg v can be flown directly after leg u with the same aircraft.
- Time line network, where nodes are events, i.e., arrivals or departures specified by a time and an airport. An edge represents an activity performed by the aircraft.

- Time band network, where each node is either a set of activities performed by an aircraft within a time interval (or time band) or the end of a recovery period, at a certain time and in a specific airport. In this case the flights are modelled as edges.

Passengers can be more extensively considered, as in [14, 19]. Passengers of cancelled flights are often reassigned to a new flight, where possible, or returned to their destination according to the rules of Regulation 261.

In most of the works presented in the review, solutions are found via graph algorithms, such as maximising the flow in [1] or integer programming as in [2]. It has nevertheless also been stated as a repartitioning problem in [20]. [22], uses a heuristic on a graph. In this model, a node is either an aircraft or a departure. A departure is connected to an aircraft in an other airport which may be itself connected to a departure. The network is then used here with a local search meta heuristic, the steepest ascent local search. Another heuristic, the "grasp" heuristic, is used in [3].

In [19], the problem is then modelled with a time band network and solved with Cplex using the financial cost as objective function. In addition to the time band network, a "passenger transition relationship" is introduced. This relationship enables the passengers to be transferred on other flights when a flight is cancelled.

When optimising the tactical network operations, several objective functions might be considered. The review [7] mentions counting the total delay, counting the number of operations carried out (e.g., swaps, reassignments, ferryings) and evaluating the real cost. The latter is the most used, however costs are complicated to model and thus several levels of refinements can be seen. For example, two sub-costs are considered in [22]. One is associated with delays and the other with cancellations. Costs associated with passengers might also be included [9,18,19]. Further analysis of costs can be found in [13], [12] and [11].

The test setups presented in [7] span from 3 aircraft and 8 flights to 332 aircraft and 2921 flights, solved in 24 minutes using Cplex. In [22], the method using heuristics has been tested on an instance of 80 aircraft, 44 airports and 340 flights. The time needed to solve such an instance is below 10 seconds.

2.2 Reinforcement learning

Reinforcement learning is a method of machine learning based on the interaction between an agent and its environment. It has been extensively used in games where the agent embodies the player and the environment is the world the player evolves in. It has been proven successful in small games such as Tetris, outperforming the human player with the most basic algorithm known as Q learning using lookup tables [23]. Many more sophisticated learning methods have been developed since, such as Deep Q Networks (DQN), developed by Google using convolutional deep neural networks [25].

The disruption problem in a fleet can be easily compared to a game making reinforcement learning suitable to solve it. The player would be the airline

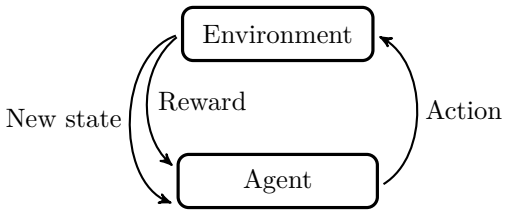


Figure 1: Reinforcement learning principle

operator who can swap aircraft and the environment is the real world and more specifically the fleet being disrupted.

Simple problems may be solved using simple Q learning algorithms using matrices. Real world problems however generally involve a large amount of states or of actions, making Q matrices unusable. To overcome this limitation, multi layer perceptrons can replace the Q matrix as described in [26]. Multi layer perceptrons are able to handle continuous inputs. If the problem involves many actions, restricted Boltzmann machines can be used as presented in [27]. As the problem involves a fleet of aircraft, it might be seen as a multi agent behaviour. While research has been lead on multi agent reinforcement learning, it is stated in [5] that it is even more subject to the curse of dimensionality than single agent reinforcement learning and thus less prone to manage an important amount of actions or states.

3 Model

3.1 Reinforcement learning model

The idea is to make an agent interact with an environment in order to learn the best possible behaviour. This behaviour is modelled by a *policy* which maps *states* of the environment to *actions* the agent should perform. To learn how to behave, each action the agent carries out is rewarded, and each action modifies the environment. The agent will thus adapt its behaviour in function of the rewards it receives (see Figure 1). Every time the agent performs an action, the environment is modified and shifts into a new state. The following notations will be used:

- $\mathcal{S}$ the set of states,
- $\mathcal{A}$ the set of actions,
- $(s, s') \in \mathcal{S}^2$ states of the environment,
- $a \in \mathcal{A}$ an action,
- $\gamma \in [0, 1)$ the vision of the agent or the discount factor,

- $T_f \in \mathbb{N}$ the end of the lifetime of the agent, being, in our case, a day of operation.

Mathematically, the best behaviour is the behaviour maximising a criterion. This criterion is usually the expectation of the cumulative sum of rewards, i.e., the total reward the agent can expect during its lifetime, with r_t the reward at time step t, $\gamma \in [0, 1)$ the discount factor,

$$\mathbb{E}\left(\sum_{t=0}^{T_f} \gamma^t r_t\right) \tag{1}$$

In our case, the reward is the cost savings obtained by performing a given action

Valuation functions are used to discriminate policies. Given a policy $\pi : \mathcal{S} \to \mathcal{A}$, the associate valuation function will be $V^\pi : \mathcal{S} \to \mathbb{R}$. Finding the best policy falls back to computing $arg\,max_\pi V^\pi$.

Given an initial state $s_0 \in \mathcal{S}$, the valuation function can be defined as, with $\mathbb{E}_\pi$ the expectation following policy π, $\forall s \in \mathcal{S}$,

$$V_\gamma^\pi(s) = \mathbb{E}_\pi\left[\sum_{t=0}^{T_f} \gamma^t r_t \middle| s_0 = s\right] \tag{2}$$

The optimal valuation function V^* satisfies the Bellman equation, which can be written as, with $s \in \mathcal{S}$,

$$V^*(s) = \max_{a \in \mathcal{A}} \left\{ r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^*(s') \right\}. \tag{3}$$

Once found, V^* gives the optimal policy π^* such that $V^* = V^{\pi^*}$.

3.2 The Q learning model

3.2.1 Principle

Q learning is based on the use of a function $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ as an equivalent of the valuation function, with $(s, a) \in \mathcal{S} \times \mathcal{A}$,

$$Q^\pi(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) V^\pi(s') \tag{4}$$

The Bellman equation is then with Q, $\forall (s, a) \in \mathcal{S} \times \mathcal{A}$,

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s' \in \mathcal{S}} p(s'|s, a) \max_b Q^*(s', b) \tag{5}$$

The idea is then to update incrementally the values of Q for each transition (s_t, a_t, r_t, s_{t+1}), with s_t, a_t being the state of the environment and the selected action at time step t, r_t the reward associated to s_t and a_t, and s_{t+1} the state

Require: $Q, \gamma \in [0,1], \alpha$
  initialise state s
  repeat
    $a \leftarrow$ choose action from an action set
    play a, observe reward r and new state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma max_{a'} Q(s',a') - Q(s,a))$
    $s \leftarrow s'$
  until training finished

Figure 2: General Q learning algorithm [29]

resulting from the application of a_t on s_t. The update is done using the following formula, which comes from equation 5, with $\alpha(s_t, a_t) \in [0,1)$ the learning rate;

$$\begin{aligned} Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha_t(s_t, a_t) \\ \Big(r_t + \gamma \max_b Q_t(s_{t+1}, b) - Q_t(s_t, a_t)\Big) \end{aligned} \tag{6}$$

3.2.2 Parameters

Looking at equation 6, one can denote two parameters, α and γ

The *discount factor* γ can be interpreted as the sight length of the agent. Seeing equation 2, the higher γ is, the more delayed reward are taken into account. On the opposite, a γ of zero creates a myopic agent which considers only immediate rewards.

If transitions between states are stochastic, the *learning parameter* α must be a sequence verifying the properties (see [32]) $\sum_{k=0}^{\infty} \alpha_k = \infty$ and $\sum_{k=0}^{\infty} \alpha_k^2 \in \mathbb{R}$.

3.2.3 General algorithm

The training algorithm is given in Figure 2. The training is finished when the state becomes terminal. Whether a state is terminal is decided by the underlying environment. Typically, the training algorithm will be run many times to have a Q function close enough to the optimal one.

Action selection Choosing actions consists in solving an exploitation-exploration trade-off. The exploration behaviour consists in exploring the state space, taking decisions that are not necessarily considered the best ones. The exploration is required to avoid local minima. Indeed, while the exploitation behaviour relies on the data accumulated to make the best decisions, it might carry out actions that are wrongly considered best because other ones have not been tried yet. The exploration carries out actions to see whether, by chance, they would be better than the ones known so far.

Different methods can be used to deal with this exploitation exploration trade off while selecting actions, see [29] for a detailed review. In this paper, we use bandit methods. Bandits methods come from an analogy with slot machines,

where a bandit with k arms is able to activate k slot machines sequentially. The goal of the bandit is to select each time the machine yielding the highest reward, maximising profit and minimising the regret of not activating the best machine. The UCB method (for upper confidence bound, see [29]) bandit algorithm can be implemented by using action a defined by, with t the time step, $N_t(s,a)$ the number of times action a has been performed in state s, $c \in \mathbb{R}^+$ a constant controlling exploration,

$$a = \underset{a' \in \mathcal{A}}{arg\,max} \left(Q_t(s,a') + c\sqrt{\frac{ln\, t}{N_t(s,a')}} \right). \tag{7}$$

The first term motivates exploitation. The second one, being higher as the number of visits is low, motivates exploration.

3.2.4 Implementation

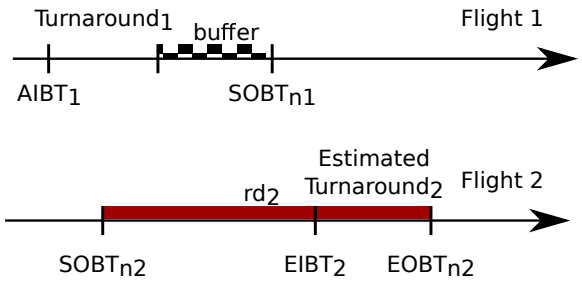
In this paper, a simple implementation approach has been selected for computational reasons. In this form, the agent is a lookup table mapping state-action tuples to their reward expectation. This way, the operation $Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma\, max_{a'}\, Q(s',a') - Q(s,a))$ is straightforward and consists in the replacement of the old value by a new one. While this method has several advantages among which its simplicity, it cannot be scaled to high dimensional states or continuous states. More advanced methods can be used, like neural networks, see [29].

3.3 Disruption management problem description

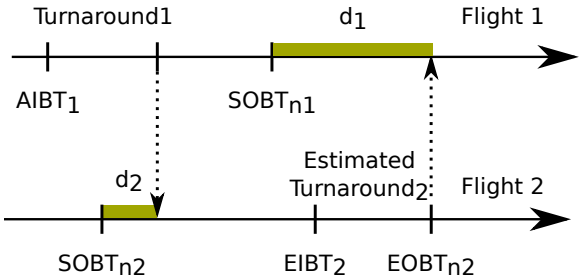
This paper focuses on the ability to swap aircraft when airlines are faced with disrupted operations. Swapping aircraft consists in exchanging aircraft between two flights. Intuitively, swapping can be cost effective for at least two reasons. The first is illustrated in Figure 3 where the amount of delay that is expected to be propagated by a given flight (rd_2) is divided between two flights producing d_1 and d_2 with $d_1 + d_2 \leq rd_2$. In this examples, two flights are delayed instead of one but part of the buffer for flight 1 is used to recover delay from flight 2, so not only the delay is divided between more flights, but also reduced. This is cost effective because the cost of delay is non-linear and the cost of the two lower delays is lower than the cost of a single higher one. Note that aircraft swapping then helps to manage reactionary delay which account for 44% of all the delay experienced [15].

Another reason to perform aircraft swaps is that, in some cases, an aircraft with delay might have more legs in its schedule than another flight the airline can swap the flight with. By doing this swap, the number of legs where delay can be propagated can be reduced.

Finally, swapping are also often used in practice to save costs in other ways that are out of the scope of this paper. For instance the airline can concentrate all delay into one flight and then cancel it; modelling this would require



(a) No swapping: Flight 1 on time, Flight 2 expected to propagate reactionary delay (rd_2)



(b) Swapping: Flight 2 experiencing d_2 delay and Flight 1 expecting d_1 delay. $d_1 + d_2 \leq rd_2$

Figure 3: Example of benefit of swapping

modelling passengers flows as load factors will have to be modelled and the consideration of cancellations.

More parameters such as additional passenger information (e.g. type of passengers, and their connections) can be used by airlines to better estimate the cost of the different options.

Some constraints need to be considered when performing the swaps. An aircraft can be swapped with another only if the flight of one ends where the flight of the other starts and the aircraft type is the same. Or in other words, two aircraft can be swapped if they have an airport in common in their flights itineraries. On real operations, further constrains need to be considered such as crew availability, number of seats per specific aircraft, etc.

An aircraft becomes available for a flight after the turnaround operations are preformed. For example, in Figure 3b, the first aircraft can be used for the next rotation of the second flight after its turnaround. These ground operations are modelled as a function of the type of aircraft and airport.

3.4 Scope of model

The main objective of the air traffic model is to capture the propagation of delay through the network, as this will allow us to capture situations as the

ones described in Figure 3. The simulator focuses on the fleet of an airline and on one day of operations. The simulator is driven by the landings of aircraft. When an aircraft lands, it can either follow its scheduled flight plan sequence or swap the remaining flights of the day with another aircraft of the same type. As mentioned, other disruption management strategies (e.g., ferrying, flight cancellation or trajectory modification) are not considered. There are no stand-by aircraft available to swap with and crew management is not considered.

A total cost is computed at the end of the day based on flight delay and on the ending position of the aircraft.

3.5 Delays and uncertainty

Delays have been capped to five hours. Above this threshold, it is generally more cost effective to cancel the flight which is out of scope of this paper.

3.5.1 Sources of delay

There are four sources of delay and uncertainty in the system:

- reactionary delay,
- taxiing in and out of the runway,
- air traffic flow management delay,
- any other sources of delay (e.g., ground operations)

The distributions used to generate the delays have been calibrated and validated in previous projects [8, 30].

Taxiing The taxiing in and out time distributions are Gaussians. The mean and the standard deviation depend on the airport and type of operation [8].

Reactionary Reactionary delays, i.e., delays caused by delays on previous flights, are explicitly modelled and thus not generated from a distribution.

Air traffic flow management (ATFM) A probability and distribution of ATFM delay is built on historical data analysing the Demand Data Repository (DDR2) [16].

Miscellaneous Encompasses all other sources of primary delay not explicitly modelled (e.g., late arrival of passengers, mechanical failure, &c.). The distribution of miscellaneous delays is an exponential one with parameter 15 capped at 90 minutes [8].

Table 1: Statistics of the simulator calibration (average among all flights and quantiles among delayed flights, in minutes).

Variable	*Avg.*	*Q10*	*Q50*	*Q70*	*Q90*
Reactionary delay	3.97	0	5	20	46
Departure delay	11.04	18	32	45	70

3.6 Exceptional delays

To stress the environment, exceptional delays are randomly created. Each simulation is started with a probability of encountering an exceptional delay. Each departing flight might have an exceptional delay, based on this probability. A delay is said to be exceptional if it is superior to three hours as airline must compensate clients from three hours of delay [17].

3.7 Calibration and validation

The behaviour of the simulator has been checked against the values given in [15]. Particularly, have been checked

- the average departing delay,
- the average reactionary delay,
- the proportion of reactionary delay among all delays.

As an overview, 21.21% of the flights were delayed and some additional data has been gathered in table 1.

3.8 Costs

Costs depend on the aircraft model and the delay. The cost function is increasing non linearly with the delay: the more a flight is delayed, the higher the ratio cost/delay is (i.e., the second derivative of the cost w.r.t the delay is positive). These costs are modelled from [10].

Additionally, if an aircraft ends in an airport which is not the final airport indicated on the initial schedule, a cost is added to reflect the rerouting of crews or aircraft. This cost has been set to the maximum cost of delay.

3.9 Simulator as reinforcement learning environment

The successive steps of the simulator fit the succession of states needed in reinforcement learning.

Feeding the agent with all the data computed in the simulator would be inefficient. As an example, states used by the algorithm need to be visited as much as possible along an episode, which requires them to be, ideally, time

independent. It would therefore be counter productive to embed the time of the simulation in the state.

To ease the work of the algorithm, states are converted to *observations*. An observation is the reduction of a state, gathering the essential information to allow the agent to take the appropriate action and estimate the cost associated to the action.

Choosing which information to include in the observation is a difficult problem. They need to be precise enough, to improve final performance; but not too detailed to avoid the explosion of the number of states, leading to a slower convergence. Fields that can be considered to be included in the observations are: delay, number of extra flights in the day remaining for a given aircraft or type of airport.

3.10 Actions

In each state, the agent performs an action on the environment. Here, an action is either swapping the landed aircraft with another one or doing nothing. Swapping two aircraft means that the aircraft exchange the remaining of their respective flight paths. We thus consider swapping aircraft equivalent to swapping flights or, to be more precise, flight paths.

One must then decide which flights can be swapped with. The first way would be to allow to swap with any not flown flight departing from the current airport. However, this have two drawbacks. First, too many actions would be made available, slowing the convergence. Secondly, actions wouldn't be clearly recognisable by the agent. The agent needs to associate an expectation of the cost following an action on a state. But swappable flights might be very different from one state to another which could increase the variance of the final reward given the current state.

3.11 Reward

In reinforcement learning, the reward allows the agent to decide which action is the most suited to the state the environment is in. Furthermore, by the estimation of reward expectations, the agent is also able to determine the best action for not only the current state but also the probable sequence of all future states.

In our model, the reward is the final cost of delays. Since the algorithm maximises the expectation of reward, if the value on which the reward is based should be minimised, the reward will be the opposite of this value. We will therefore use the opposite of the cost as reward.

Table 2: First observation type example.

Field description	Example instance
aircraft → flight path	$\{(a_1, f_2), (a_2, f_1), ...\}$
aircraft → delay	$\{(a_1, \text{low}), (a_2, \text{high}), ...\}$
aircraft → sobt before ready time	$\{(a_1, t), (a_2, f), ...\}$
aircraft → more remaining	$\{(a_1, f), (a_2, f), ...\}$
landed aircraft id	2
landed aircraft delay	low

4 Experimental setup

4.1 Schedules

The algorithm uses one day of operation of one airline. It has been tested on a subset of the traffic of Vueling on October 12, 2014. The initial schedule has a fleet of 90 aircraft. In order to speed up experiments, only 6 aircraft have been kept. The aircraft has been chosen such that there is a fair amount of swapping possibilities, i.e., each aircraft has at least one swapping possibility along its flight path. Flight data has been obtained from [16].

The cost of reallocation at the end of the day has not been taken into account. As reallocation costs only happen at the end of an episode, they are the most delayed reward possible and are thus hard to learn. Given that this paper is a proof of concept and not a real life application, applicability is reduced in favour of results.

4.2 Observations and actions

4.2.1 Observation

Two type of observations have been tried. The first one is based on flight paths and contains:

- a mapping from aircraft to flight paths,
- a mapping from aircraft to their delay,
- the identifier of the landed aircraft,
- the delay of the landed aircraft,
- a mapping from aircraft to whether, if there is a swappable flight on its flight path, the SOBT (scheduled off block time) of this swappable flight is before the ready time of the landed aircraft,
- a mapping from aircraft to whether it has more remaining flight in its flight path than the landed aircraft.

Table 3: Second observation type example.

Field description	Example instance
aircraft → sobt before ready time	$\{(a_1, t), (a_2, f), ...\}$
aircraft → toward stn.	$\{(a_1, f), (a_2, t), ...\}$
aircraft → more remaining	$\{(a_1, t), (a_2, f), ...\}$
aircraft → on orig. flight path	$\{(a_1, f), (a_2, t), ...\}$
landed id.	4
total delay	high

Table 2 presents an example of the fields included in this observation description.

The second observation type does not contain the flight path nor the delay. The fields considered are:

- a mapping from aircraft to whether the SOBT of its next swappable flight is before the ready time of the landed aircraft,
- a mapping from aircraft to whether it is currently flying toward the current airport,
- a mapping from aircraft to whether it has more remaining flights than the landed one,
- a mapping from aircraft to whether it is on its original flight path (if reallocation costs are considered),
- the identifier of the landed aircraft.
- the accumulated delay on the fleet

Table 3 contains an example of the fields considered in the second type of observation. To avoid the explosion of the number of states, delay amounts are further discretised from the set of integers to a finite number of variables (e.g. $\{\text{low}, \text{medium}, \text{high}\}$). With d the number of values used to describe delay and n the cardinal of the fleet, there are $nd \cdot n! \, 2^{2n} d^n$ observations for the first model and $dn2^{4n}$ for the second (with generally $d \geq n$).

4.2.2 Action

A flight is considered swappable if:

- the origin airport is the same as the current one,
- it is the first flight (according to SOBT) in the flight path of the aircraft to match the above requirement,
- the previous flight has not landed yet.

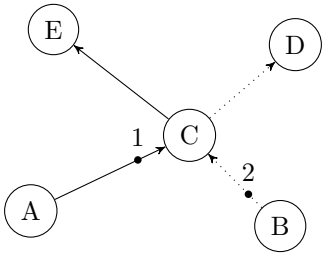


Figure 4: Swappability example: airports and flight for two aircraft

In short, for an aircraft, a flight is swappable if it is the first one to depart from the current airport and the aircraft is not on the ground going ready to do it. Figure 4 shows an example of two flights that can be swapped: aircraft 1 is supposed to fly the plain legs while aircraft 2 flies the dotted legs. Aircraft 1 is arriving at airport C and aircraft 2 just departed from B, at landing of aircraft 1, it can swap the remaining of its flights taking the C to D leg. Aircraft 2 will therefore operate the C to E leg.

5 Results

5.1 Parameters

All the parameters tuned in the model are:

- γ the discount factor, see 3.2.2.
- c the exploration control parameter when using UCB bandits, see Equation 7.
- p_d the probability of introducing a severe delay on each flight, a severe delay is 3h.
- q_i the initial value of the Q matrix.

For the experiments, the parameters $(\gamma, c, p_d, q_i) = (0.95, 10, 0.06, -9 \cdot 10^4)$ have been used.

5.2 Learning process

To verify that the agent learns and performs better, a metric from [25] will be used. At the beginning, random states are taken by following a random policy along an episode. Then during the training, at each time step, with S the set of selected states and $\mathcal{A}$ the set of available actions, is computed

$$\frac{1}{card\, S} \sum_{s \in S} \underset{a' \in \mathcal{A}}{max}\, Q(s, a'). \tag{8}$$

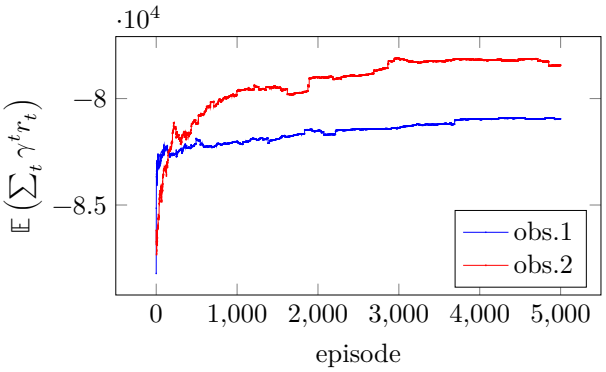


Figure 5: Average maximum Q values over 5000 episodes with $p_d = 0.06$, $c = 10$, $\gamma = 0.95$, $q_i = -90,000$.

This metric thus gives an indication of the evolution of the discounted reward expectation through training.

Figure 5 shows this metric used with both observations. The second observation type seems to perform better since it reaches the $-8 \cdot 10^4$ of expected discounted reward in approximately 1,000 trainings whereas the first observation type does not even reach the $-8 \cdot 10^{-4}$ in 5,000 trainings. The overall increase asserts that the agent learns how to perform better, since its expectations of rewards are getting higher. The sudden falls may be interpreted as the agent realising that delays might happen.

5.3 Comparing with the idle behaviour

The usability of the algorithm is determined by asserting whether it performs better than doing nothing. Figure 6 shows the boxplots of the distribution of rewards (i.e., negative cost of delay at the end of the day) for the agents and the idle behaviour. The assumption that the agent with observation type 2 is better than the other is verified, since all displayed quantiles are higher for the former. Comparing with the idle agent, our trained agent seem to perform worse in general. However, the agent 2 (with observation type 2) has a shorter left whisker than the idle, showing that the agent is able to recover perturbed episodes.

The fact that the agent does not perform as well as the idle might be explained by local maxima, i.e., the agent believes that, given a certain observation, swapping is the best idea although it is not. Indeed, even after 20,000 episodes, it can be seen, inspecting the details of the simulations, that some observations have been visited only a dozen times. This problem might be solved by training more the agent or providing other options rather than just swapping aircraft. The average number of swaps per simulated day of operations, computed on 1,000 episodes ran with trained agents, are 8.7 for agent 1 and 7.3 for agent 2.

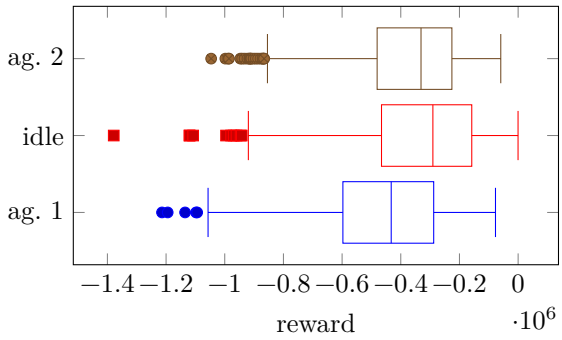


Figure 6: Comparing the idle behaviour with the agent. Agent 1 uses the first observation type while agent 2 uses the second. Both agent have been trained over 20,000 episodes.

The performance observed can also be explained due to the limited options given to the agents (swapping or not), in some cases swapping can be a risky operation as downstream effects can be generated. The limited set of options explains the relatively high number of swaps performed but further validation is also needed. The next section discusses some future work that can be done to improve the overall final performance of the algorithm.

6 Conclusion

This paper presents a method based on machine learning to deal with air traffic disruptions by doing aircraft swapping. The model used for the algorithm is able to manage a whole fleet of aircraft and is primarily modelling the delay propagated through flights, as well as defining the actions that can be performed on the fleet. The step by step dynamic of the simulator, coupled with the cost associated with an action is well suited for reinforcement learning.

A basic reinforcement learning algorithm, the tabular Q learning, is presented here, along with a non trivial method concerning action selection. The results evince some interesting features, namely the ability to recover from heavily disrupted traffic and future potential lines of research.

Further work

Concerning the algorithm, several options are possible. The tabular method might be improved by choosing more adequately the states. Indeed, it has been seen in Figure 5 and Figure 6 that the design of the observation impacts the performances of the agent. These features might be chosen either on expert knowledge, or based on analysis of the relations between the variables of the simulator and the resulting cost. This analysis might be done using supervised learning and decision trees, where the tree should predict the cost based on the

variables.

More advanced reinforcement learning algorithms can be used, such as replacing the look-up table by a neural network such as in [26], more advanced recurrent neural networks as in [4] or using the latest A3C (Advantage Actor Critic) algorithm by Deepmind presented in [24].

The model can also be extended, by e.g., adding the possibility to cancel or ferry flights. Passengers can also be included in the model, giving the opportunity to choose to wait for an arriving flight having delay if there are passengers from this flight who connect with the departing flight. Aircraft performance could also be modelled, giving the ability to choose whether to speed up the next flight, as could the opportunity to choose between different routes. However, adding all these possibilities will increase the number of possible actions, making the above methods inefficient. To deal with large action space, a restricted Boltzmann machine could be used as a substitute for the look-up table, as explained in [27] where an agent has been proved able to manage 2^{40} actions.

More complex cost estimations could be used, for example by incorporating explicit passenger costs.

Finally, further validation of the algorithm outcome should be needed. However, information on aircraft swap can be hard to obtain since available data set do not have information about if flights were swapped or not.

References

[1] Jarrah A, Yu G, Krishnamurthy N, and Rakshit A. A decision support framework for airline flight cancellations and delays. *Transportation Science*, 1993.

[2] T. Andersson and P. Värbrand. The flight perturbation problem. *Transportation Planning and Technology*, 27(2):91–117, 2004.

[3] Michael F. Argüello, Jonathan F. Bard, and Gang Yu. *Models and Methods for Managing Airline Irregular Operations*, pages 1–45. Springer US, Boston, MA, 1998.

[4] Bram Bakker. Reinforcement learning with long short-term memory. In *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, NIPS'01, pages 1475–1482, Cambridge, MA, USA, 2001. MIT Press.

[5] L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. *Studies in Computational Intelligence*, 310:183–221, 2010.

[6] Jia-Ming Cao and Adib Kanafani. The value of runway time slots for airlines. *European Journal of Operational Research*, 126(3):491 – 500, 2000.

[7] Jens Clausen, Allan Larsen, Jesper Larsen, and Natalia J. Rezanova. Disruption management in the airline industry – concepts, models and methods. *Computers & Operations Research*, pages 809–821, 2009.

[8] ComplexityCost project. D4.5 – Final Technical Report. Technical report, SESAR JU, 2016.

[9] Andrew Cook, Luis Delgado, Graham Tanner, and Samuel Cristóbal. Measuring the cost of resilience. *Journal of Air Transport Management*, 56:38 – 47, 2016. Long-term and Innovative Research in ATM.

[10] Andrew Cook and Graham Tanner. European airline delay cost reference values. Technical report, University of Westminster, 2005.

[11] Andrew Cook, Graham Tanner, and S. Anderson. Evaluating the true cost to airlines of one minute of airborne or ground delay. Technical report, EUROCONTROL and University of Westminster, 2004.

[12] Andrew Cook, Graham Tanner, and A. Lawes. The hidden cost of airline unpunctuality. *Journal of Transport Economics and Policy*, 46:157–173, 2012.

[13] Andrew Cook, Graham Tanner, V. Williams, and G. Meise. Dynamic cost indexing – managing airline delay costs. *Journal of Air Transport Management*, 15:26–35, 2009.

[14] Luis Delgado, Jorge Martín, Alberto Blanch, and Samuel Cristóbal. Hub operations delay recovery based on cost optimisation – dynamic cost indexing and waiting for passengers. In *Proceedings of the 6th SESAR Innovation Days*, 2016.

[15] EUROCONTROL. Coda digest 2017, all-causes delay and cancellations to air transport in europe. Technical report, EUROCONTROL, 2017.

[16] EUROCONTROL. DDR2 reference manual for generic users. Technical Report V. 2.9.4, EUROCONTROL, 2018.

[17] European Commission. Regulation (EC) No 261/2004 of the European Parliament and of the Council of 11 February 2004 establishing common rules on compensation and assistance to passengers in the event of denied boarding and of cancellation or long delay of flights, and repealing Regulation (EEC) No 295/91. Technical report, European Commission, 2004.

[18] Yuzhen Hu, Yan Song, Kang Zhao, and Baoguang Xu. Integrated recovery of aircraft and passengers after airline operation disruption based on a grasp algorithm. *Transportation Research Part E: Logistics and Transportation Review*, 87:97 – 112, 2016.

[19] Yuzhen Hu, Baoguang Xu, Jonathan F. Bard, Hong Chi, and Min'gang Gao. Optimization of multi-fleet aircraft routing considering passenger transiting under airline disruption. *Computers & Industrial Engineering*, 80:132 – 144, 2015.

[20] Rosenberger J, Johnson E, and Nemhauser G. Rerouting aircraft for airline recovery. *Transportation Science*, 2003.

[21] Niklas Kohl, Allan Larsen, Jesper Larsen, Alex Ross, and Sergey Tiourine. Airline disruption management—perspectives, experiences and outlook. *Journal of Air Transport Management*, 13(3):149 – 162, 2007.

[22] M. Løve, K. R. Sørensen, J. Larsen, and J. Clausen. Using heuristics to solve the dedicated aircraft recovery problem. *Central European Journal of Operations Research*, 13(2):189–207, jun 2005.

[23] S. Melax. `https://melax.github.io/tetris/tetris.html`.

[24] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, David Silver, Tim Harley, Timothy P. Lillicrap, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, 2016.

[25] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Dann Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*, 2013.

[26] G. A. Rummery and M. Niranjan. On-line q-learning using connectionist systems. Technical report, Cambridge University Engineering Department, September 1994.

[27] Brian Sallans and Geoffrey E. Hinton. Reinforcement learning with factored states and actions. *Journal of Machine Learning Research*, 5:1063–1088, 08 2004.

[28] Milind G. Sohoni and Sanjiv Erat. Can time buffers lead to delays? the role of operational flexibility. *SSRN*, April 2015.

[29] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2017.

[30] Vista project. D5.2 – final assessment report. Technical report, SESAR JU, 2018.

[31] Hans-Wieger M. Vos, Bruno F. Santos, and Thomas Omondi. Aircraft schedule recovery problem – a dynamic modeling framework for daily operations. *Transportation Research Procedia*, 10:931 – 940, 2015. 18th Euro Working Group on Transportation, EWGT 2015, 14-16 July 2015, Delft, The Netherlands.

[32] Christopher J.C.H. Watkins and Peter Dayan. Q-learning, machine learning. *Machine Learning*, pages 279–292, 1992.