To hold or not to hold? - Reducing Passenger Missed Connections in Airlines using Reinforcement Learning

Tejasvi Malladi
TCS Research
Chennai, India
tejasvi.malladi@tcs.com

Karpagam Murugappan
TCS Research
Chennai, India
karpagam.murugappan@tcs.com

Depak Sudarsanam
TCS Research
Chennai, India
depak.sudarsanam@tcs.com

Ramasubramanian Suriyanarayanan
TCS Research
Chennai, India
ramasubramanian.suriyanarayanan@tcs.com

Arunchandar Vasan
TCS Research
Chennai, India
arun.vasan@tcs.com

ABSTRACT

Missed connections at transit airports are a source of both poor customer experience and reduced airline operational efficiency. Airlines typically handle missed connections by rebooking customers. Recently, airlines have started *holding* departing flights for some time in a rule-based manner to avoid missed connections. However, rule-based heuristics typically use information local to a flight and do not learn in a globally informed way across the entire network.

We complement existing approaches by learning a policy for holding a flight to avoid misconnections, using reinforcement learning (RL). The state presented to the RL agent uses forecasted flight-specific context; and measured network-wide context. The reward uses components that trade off the decrease in on-time performance due to the hold decisions, for a decrease in missed connections. We attribute the global rewards to individual local hold actions through a novel delay tree that approximates the network interactions. Multiple flights are handled through the same instance of the agent handling them in sequence with varying state information.

We evaluate our approach for two different airlines with training and testing over a microsimulator that uses real-world data for calibration. Across different algorithms (DQN, AC, A2C, DDPG), we find that the best performing RL-based agent is able to reduce significantly more (up to 50%) missed connections for a minimal decrease ($\tilde{5}$%) in on-time performance; when compared with a current rule-based heuristic. Further, the approach is tunable and able to transfer learn across different airlines.

KEYWORDS

Airline Operations; Missed Connections; Reinforcement Learning

ACM Reference Format:

Tejasvi Malladi, Karpagam Murugappan, Depak Sudarsanam, Ramasubramanian Suriyanarayanan, and Arunchandar Vasan. 2021. To hold or not to hold? - Reducing Passenger Missed Connections in Airlines using Reinforcement Learning. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

1 INTRODUCTION

Missed connections at transit airports cause both poor customer experience and poor efficiency. Passengers (PAX) aboard a delayed incoming flight who miss connections at a transit airport experience a non-linear jump in the delay to the destination airport. The PAX disutility is even higher if future connecting flights to the final destination are infrequent. As per the Bureau of Transportation and Statistics [19], out of 51 million US PAX itineraries in 2007, there were 44.5 million connections and about a million connections were missed. Typically, PAX who miss connections are rebooked on alternate flights at high actual cost (if on other airlines); or inventory/opportunity costs (if on same airline); resulting in poor airline operating efficiency [6]. Preventing missed connections proactively is thus a key business ask for airlines due to improving PAX experience and efficiency.

Existing approaches: A recent industry practice [18] to reduce missed connections is to hold a departing flight f_2 so that PAX on delayed incoming flight f_1 can make the connection. The Hold-No-Hold (HNH) decision for a departing flight is typically rule-based and made locally. Flight f_2 can be held if its estimated time of arrival (ETA) doesn't exceed the scheduled arrival time by more than the on-time performance (OTP) buffer (typically, 15 minutes). This approach aims to avoid propagation of delay in the network. However, it does not take into account the availability or the cost of future alternate flights for rebooking; the specific flight's context; and the general state of operations of the entire network. Indeed, there is no guarantee of even an on-time arrival due to the inherent uncertainties in airline operations. A rule-based approach, while simple to execute, can be sub-optimal in practice as we show in our evaluation. To the best of our knowledge, this form of the HNH problem has not received much attention (details in Section 2).

Problem statement: We complement existing approaches to HNH by *learning* a *globally informed* policy as opposed to a fixed locally-decided rule. The learned policy may still preclude delays from propagating in the network like in the rule-based approach; but the decision would be a globally informed trade-off. Given the operating context of the airline, PAX connections, and the plans for the flights, we learn a policy that decides whether to hold a departing flight and if yes, by what duration. The objectives of the decisions are to reduce missed connections without significant increase in the

network delay. Our key insight is that *marginal* increase in average airline network delay can reduce *significant* fraction of missed connections. This is made possible by opportunistically exploiting the slack in the schedules for operations without causing disruption.
Challenges: The problem of learning a globally informed policy for HNH is non-trivial for several reasons. First, the propagation of delay through the network because of the physical aircraft (or tail) movement along the logical route plan is generally non-linear and stochastic. Second, a delay induced by HNH can also propagate due to PAX itineraries. For instance, holding flight f_3 while saving delayed incoming PAX, may cause on-time outgoing PAX p on f_3 to be delayed; triggering another HNH decision to hold a flight f_4 to which p connects. Third, airline tail plans that map tails to flights are not stationary for reasons such as rotation of aircraft across sectors; constraints of scheduling maintenance only at hub airports; and last-minute maintenance problems in aircrafts. While constrained optimization is routinely used by airlines [12] for longer-term (6 months to the day-of-departure typically) planning, it is likely infeasible for real-time operations, where it would be better to have a rule-based or learned policy to execute in real-time.
Solution approach: Given the goal of online sequential HNH decision making in an environment with unknown stochastic dynamics, Reinforcement Learning (RL) stands as a suitable candidate [4, 25]. Because exploration for learning is not easy in live operations, we train the agent on a simulated airline environment to make HNH decisions. The state presented to the RL is a pre-processed context of both a specific flight; and current global operating conditions. The agent decides the duration to hold a flight f. The next time the HNH for the same flight is decided, the agent gets a reward for the previous action and the new context for the flight. The reward is engineered to capture how PAX of f benefitted; the delay of f; and how globally PAX and other flights were benefitted/affected due to this decision for flight f. Tunable parameters in the reward function control two trade-offs: First, between the missed connections reduced on flight f and its delay. Second, between flight f's benefits and the global network level average delay and missed connections reduced. Operating constraints such as crew legality; curfew violations; and incoming delays can be implemented as negative rewards or filters imposed on the feasible action space. The agent makes decisions for all flights in the network in sequence essentially simulating multiple agents learning on the same operating environment. The effects of the decisions of the agent across multiple flights are tied together through the global reward and the global component of the state.
Reward engineering: A key challenge in the reward design is the attribution of a particular flight's delay and/or missed connections across multiple hold decisions made in the past. To do this, we build a delay tree that approximately backtraces the origin of a reward event. Specifically, delays and missed connections in a flight are attributed to hold decisions that are the reward's descendants in the propagation tree (shown in Figure 2) using influence indices defined over edges of the delay tree (details in Section 5.1).
Contributions: We train an RL agent that learns an HNH policy to reduce missed connections while respecting the network delay. The state presented to the agent has components that are both flight-specific and network-wide. Similarly, the reward has both local and global components reflecting the trade-offs between missed connections reduced and network delay. To attribute the global component of the reward correctly to past hold decisions, we use a delay tree. We build and validate a microsimulator that models the tail-plan details of an airline network; and PAX connections across flights in airports. We calibrate this model using tail-plans obtained for two different airlines from [22] that we refer to as Air-East (around 460 flights/day) and Air-West (around 1600 flights/day) for confidentiality reasons. For the PAX data for Air-West (Air-East), we obtained the aggregate PAX travel data from [19] (the airline itself) and use this to generate representative PAX itineraries. For a training period of 5 months and a testing period of 1 month, we implement the RL agent using the Deep Q-Network (DQN) [17], Advantage Actor Critic (A2C) [16, 23], Actor Critic (AC) [2], and Deep Deterministic Policy Gradient (DDPG) [13] algorithms; and for varying hyper-parameter values that capture the trade-off between missed connections reduced and network delay. As a baseline, we use variants of a rule-based heuristic that reflect the current industry practice.
Key findings: The RL agent learns to avoid missed connections. For Air-East (Air-West), the best-performing RL algorithm reduces missed connections by 50% (50%) when compared to the rule-based heuristic baseline with a decrease in on-time performance of about 5% (8%) respectively. Unlike a fixed rule-based approach, the RL agent can learn *business-tunable* policies that allow airlines to save PAX for the network delay they can tolerate. We find there is a knee-point corresponding to about a 47% decrease in missed connections (for Air-East) corresponding to a decrease in OTP of 5%; beyond which marginal decrease in missed connections diminishes with marginal increase in network delay. The agent is also able to transfer learn across different airlines with limited retraining.

Our approach is potentially applicable to few other domains: 1) Cross docking in road transportation networks consolidates multiple parcels intended to a common destination. This requires synchrony in the driver schedules of the incoming trucks and the outgoing trucks, even while respecting on-time parcel delivery deadlines. Near-misses of consolidation opportunities can be avoided by holding outgoing truck; and 2) Multi-modal networks (e.g., metro to bus) with integrated payment systems can possibly hold departing buses in response to incoming delays based on user profiles.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents an overview of the solution. Section 4 presents the details of the state used for RL; and Section 5, the reward engineering for RL. Section 6 presents the experimental setup. Section 7 presents the results of our evaluation and discusses the limitations of our approach. Section 8 concludes.

2 RELATED WORK

Existing works for avoiding airline misconnections can be broadly categorized as follows: 1) planning/schedule optimization; 2) gate reassignment; and 3) rule-based holding. Of these, rule-based holding is the closest to us. Most of these focus on solving the problem locally without taking into account the global context and impact.
Planning/schedule optimization: These works aim to optimize flight schedules and routes to minimize misconnections at *a planning stage* [5, 9, 10, 12]. The idea here is to identify the gaps between

planned and actual operations and suggest changes to help minimize misconnects. These include modifying flight timings, flight routes, minimum connection times (MCT) and fleet assignment. Some of them utilize simulations to validate the findings. Our approach focuses on the *operational* stage where the options are fewer.
Gate reassignment: The main idea in these works is to change either the incoming/outbound flight's gate so as to help the connecting passengers make the connection without causing any additional delays [1, 8, 14, 21]. The effectiveness, however, depends on the airport infrastructure and whether the airline or airport controls the assignment; while holding flights is under the airline's control.
Rule-based holding: In this approach, the idea is to identify the available schedule-based slack in the network based on flight schedules and exploit them when needed to help improve passenger connections using deterministic rules [11, 18]. These rules may cap the maximum hold duration; specify minimum activation misconnects; and include other constraints. A rule-based approach is easy to implement, but typically uses local information; and does not learn or adapt from the end to end global impact of the decision.

Patent [3] mentions the possibility of choosing the landing order of flights in the air to minimize misconnections. While this approach is promising, it is unclear if an airline can influence an airport's traffic controller. Further, the impact on the entire network is unclear. [15] tries to proactively reaccommodate passengers on alternate flights without trying to avoid misconnections.
Our contribution: Holding a flight to minimize misconnects and maintaining the global OTP are competing objectives. To the best of our knowledge, the problem of identifying flexible and globally informed policies to resolve this trade-off has not received much attention, more so, using a learning-based approach. In this work, we address this gap using a RL based approach.

3 SOLUTION OVERVIEW

Figure 1 shows an overview of our solution approach. The environment for the RL agent consists of an airline network simulator, a context engine and a reward engine. The context engine generates the abstracted state s_t (described in Section 4) that captures both flight-specific and global context from the raw data generated by the airline network simulator. Given the state s_t, the agent decides to hold the flight f by the action a_t. Because it is impractical to train the agent on a live airline system, we use a microsimulator for the airline operations. The simulator is described in Section 6 and validated in Section 7.1. Post training, the agent can be deployed on a live real-world system. The action a_t is implemented in the system and the effects are observed over the next 24 hours. Note that multiple flights on the same day between two airports will have different flight identifiers; and so, the RL agent will make decisions separately. For the case of multiple-hops of the same aircraft with the same flight identifier, we use the origin airport to disambiguate. The observed effect of the hold decision over the next 24 hours are fed back to the RL agent at the next instance of the same flight f at the next epoch t+1, through the rewards described in Section 5.
Handling multiple flights: Ideally, each flight would have an HNH agent and all these agents would learn over a common simulation timeline where their actions affect each others' operations. However, this approach is simply not scalable for the hundreds of

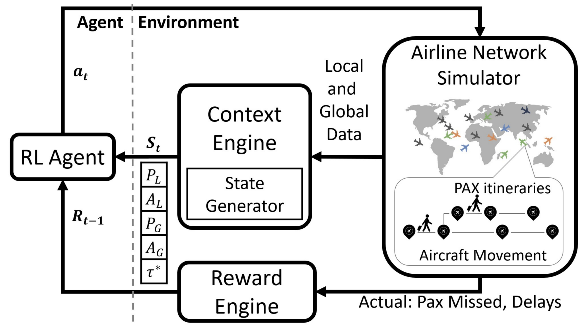


Figure 1: Solution overview

flights we consider in our evaluation. Further, using multi-agent RL directly is known to have several limitations [7, 20, 24]. Instead, we adopt an approach where one RL agent makes the decision of multiple flights in sequence (with each flight's specific state). This reasonably approximates multiple instances training in parallel.

4 STATE REPRESENTATION

The state presented to the RL agent is shown in Table 1. We distinguish between two components of the state - forecasted and actual. The effects of the HNH action *local* to the flight f can be *forecasted* reasonably well and thus any (offline non-RL) machine learning can also be exploited in the context engine. However, the *global* effect of an HNH decision of flight f can only be *observed* and learned as part of the RL training. This is because the global effect jointly depends upon how future flights are held; and the overall system evolves stochastically. In line with our intuition, the first two components of the state, vectors $\mathbf{P_L}$ and $\mathbf{A_L}$ represent the forecasted local PAX utility (PU) $P_L(\tau)$ and the airline utility (AU) $A_L(\tau)$ of that flight f for various choices of the hold time τ. The next two components are P_G and A_G represent the global PAX and airline utilities measured across all flights. The last component τ^* is a helper variable derived from vectors $\mathbf{P_L}$ and $\mathbf{A_L}$.
Local PU vector $\mathbf{P_L}$: For a given hold time τ (which could be zero), the context engine estimates if a PAX p_i on a delayed incoming flight misses or makes the connection. A basic approach that we use is to deterministically decide that p_i makes the connection if their connection window after the hold τ is above an airport-specific minimum connection time (MCT). In general, the context engine can also use offline machine learning (ML) models if available from historical data and make more informed estimates. For example, a wheel-chair PAX may require more time to make the connection than the MCT. The offline ML can then generate a probability that a particular PAX will make the connection for a connection window.

For a given hold τ, after estimating if PAX p_i makes the connection either deterministically or probabilistically, the context engine estimates the expected delay of p_i to the final destination $\delta_i(\tau)$ depending upon availability of alternate flights, etc. The disutility σ_i to PAX p_i due to the PAX delay $\delta_i(\tau)$ is defined thus:

$$\sigma_i(\tau) = \begin{cases} 0 & \text{if } \delta_i(\tau) \leq 15 \text{ minutes (an on-time arrival)} \\ \frac{\min(\delta_i(\tau), \Delta_P)}{\Delta_P} & \text{if } \delta_i(\tau) > 15 \text{ minutes (delayed arrival)} \end{cases}$$

Table 1: Notation for state and reward

Symbol	*Meaning*	*Type*	*Time*
s_t	**State presented to agent at time** t		
$\mathbf{P_L}$	Local PU	Forecasted	$[t, t{+}1]$
$\mathbf{A_L}$	Local AU	Forecasted	$[t, t{+}1]$
P_G	Global PU	Actual	$[t\text{-}1, t]$
A_G	Global AU	Actual	$[t\text{-}1, t]$
τ^*	Helper variable	Derived	$[t, t{+}1]$
R_T^f	Total Reward for flight f	Derived	$[t, t{+}1]$
R_L^f	Local Reward for flight f	Derived	$[t, t{+}1]$
R_G^f	Global Reward for flight f	Derived	$[t, t{+}1]$
P_L^f	Local PU for flight f	Actual	$[t, t{+}1]$
A_L^f	Local AU for flight f	Actual	$[t, t{+}1]$
P_G^f	Global PU attributed to f	Actual	$[t, t{+}1]$
A_G^f	Global AU attributed to f	Actual	$[t, t{+}1]$
α	Knob for PU-AU trade-off		
β	Knob for local-global trade-off		

Here, Δ_P is normalizing constant for PU. The utility for a PAX p_i for a hold τ is thus $1-\sigma_i(\tau)$; and the local PU $P_L(\tau)$ across all PAX p_i is the average of $(1-\sigma_i(\tau))$ across all PAX p_i. We experimented with various choices for the utility function but have presented only the final version. Section 7.3 summarizes few of the other approaches we tried and how they fared. While we use a thresholded linear function of δ_i for PU, PU can include costs incurred to the airline due to regulatory compensations for the delay, missed connection, etc. Such costs would generally increase with increasing δ_i.

Local AU vector $\mathbf{A_L}$: For each hold τ, the context engine estimates the arrival delay $\delta_f(\tau)$ of f at its destination. The AU of a hold is defined as $1-\frac{\delta_f(\tau)}{\Delta_F}$, where Δ_F is a normalizing constant for AU. Note that this does not include any higher order effects of the delay of f, which are estimated by the global rewards. Any non-linear change in AU (e.g., a curfew at the arrival airport will be violated) can also be captured in the vector $\mathbf{A_L}$ for those values of τ.

Global PU P_G and AU A_G: For the measured global utilities, we use the average PU and AU across all PAX and all flights of the airline over a historical window $[t\text{-}W, t]$ as a proxy for the global system state. We choose W=24 hours. A low global PU indicates that many PAX may be missing connections, and so flight f could be more aggressive in holding, while a low AU indicates f should be more conservative in holding as the system already has significant delays. The effect of this flight f's action would be fed back into the next epoch of HNH decision making; and the RL agent thus learns about the global effects of the local action.

Helper variable τ^*: To help the agent learn faster, we include $\tau^* = \arg\max_\tau(\alpha P_L(\tau){+}(1\text{-}\alpha)A_L(\tau))$ as part of the state to pick the hold time τ that maximizes the local weighted PU and AU. Because τ^* maximizes only the locally forecasted values, it can be computed along with the local forecasts for PU and AU; and as we show later, helps convergence. The agent learns to refine τ^* by either accepting it or modifying it.

Table 2: Notation used for delay tree (DT)

Symbol	*Meaning*
f_i	Flight i
D_i	Departure delay of f_i
A_i	Arrival delay of f_i
H_i	Hold duration of f_i
G_i^D	Departure Ground-time delay of f_i
G_i^A	Arrival Ground-time delay of f_i
T_i	Air-time delay of f_i
$\rho(X, Y)$	Influence index of a variable X on variable Y

5 REWARD ENGINEERING

The rewards are chosen to encourage hold decisions that 1) reduce PAX missed connections; and 2) do not increase network delay significantly. With this view, the total reward of the agent for a flight f is a weighted combination of the measured: 1) local PU and AU; and 2) global PU and AU across all flights attributable to f. The total reward R_T^f for a flight f is defined as $\beta R_L^f + (1-\beta)R_G^f$. Using the notation in Table 1, the local component R_L^f is given by $R_L^f = \alpha P_L^f + (1-\alpha)A_L^f$ and the global component R_G^f is given by $R_G^f = \alpha P_G^f + (1-\alpha)A_G^f$. For a flight f, P_L^f and A_L^f are the PU and AU *realized* or *measured* at time $t+1$ respectively for the hold action taken at t by the RL agent for flight f.

5.1 Global reward attribution

Delays and PAX misses globally are a joint effect of multiple hold decisions in the system. To apportion the current global PU and AU across past hold decisions, we use a delay tree (DT) that works back in time from a specific delayed flight to hold decisions in the past that contributed to this delay. Missed connections avoided can be similarly handled and we omit discussing it separately for the sake of brevity. The DT approximates the complex propagation of delays by capturing major factors and abstracting away minor ones. We summarize the DT generation and usage using notation in Table 2.

Generation of the DT: Figure 2 traces the DT of arrival delay A_i of flight f_i at the destination airport. **Rule 1:** A_i mainly depends upon the departure delay D_i; the air-time delay T_i; and the ground time delay at arrival G_i^A. **Rule 2:** Departure delay D_i is usually due to three main factors: 1) the arrival delay of the previous flight A_j of the same tail; 2) the hold duration H_i; and 3) the ground delay in departure G_i^D. The arrival delay A_j is in turn either due to the delayed departure D_j of the previous flight f_j of the same tail; or the delay in the air-time due to head winds, etc. Similarly, D_j can be recursively expanded into a sub-tree. **Rule 3:** A hold delay H_i of flight f_i depends upon the arrival delays A_{i_k} of the flights f_{i_k} with incoming connecting PAX for flight f_i. Note that a flight that starts from rest at the beginning of its tail plan will not have an arrival delay; but could have a ground delay or hold delay. By repeatedly applying rules 1, 2 and 3, the DT for the arrival delay of a flight can be generated tracing back into some window in the past that includes hold decisions of other flights. Any hold in the DT of A_i can be rewarded for its contribution to A_i.

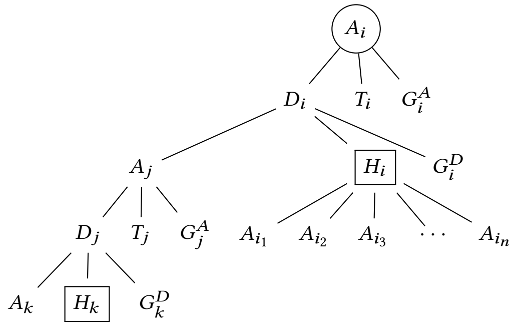


Figure 2: Delay tree attributing the arrival delay reward A_i to past hold actions $\boxed{H_i}$ and $\boxed{H_k}$

Using the DT to attribute rewards: Consider Figure 2. Our goal is to split up PU pu_i and AU au_i of the delayed flight A_i across the hold decisions (e.g., H_i and H_k) in the delay tree of A_i that result in this outcome. To do this, we define an influence index $\rho(X, Y)$ for each edge $X - Y$ as a measure of the influence of child node X on parent node Y. Once ρ is defined for all edges of the DT, the influence index of any hold H *in the past* on an outcome O (resultant delay or reduced missed connections) *in the future* is the product of the influence indices of the path leading to outcome O.

For arrival/departure delays: If f_i is on-time, defined as $A_i \leq 15$ minutes, then D_i, T_i, and G_i^A do not influence A_i and beyond as delay propagation is arrested at A_i. If the flight arrival is not on-time, then the influences of the components are split as a ratio to the total. Specifically, if X_i denotes one of (D_i,T_i,G_i^A), then we have:

$$\rho(X_i, A_i) = \begin{cases} 0 & \text{if } A_i \leq 15 \text{ minutes} \\ \dfrac{\max(X_i, 0)}{\Sigma_j \max(X_j, 0)} & \text{otherwise} \end{cases}$$

The intuition here is that only the positive components among the X_i effectively increase the non-zero overall delay A_i; and the extent to which they contribute is simply their fraction of the sum of all positive components. By construction, at least one of the X_i will be non-zero when $A_i > 15$ minutes and so $0 \leq \rho \leq 1$ always. The influence of the children of a departure delay D_i on their parent D_i is defined in an identical manner.

For hold delays: The hold delay H_i depends upon the incoming flight delays of the flights $f_{i_1}, f_{i_2}, \dots f_{i_n}$ which feed PAX into the flight f_i. In addition, it also depends upon the incoming aircraft's delay A_j as higher A_j would reduce the window available to choose H_i. Let S_i be defined as $\{A_{i_k} \mid A_{i_k} < H_i\}$, i.e., the set of incoming flights with delays lesser than the eventual hold time. Clearly, flights not in S_i do not influence H_i because their delay does not flow into the H_i finally. Within S_i, we apportion the influence as $\frac{1}{|S_i|}$.

$$\rho(A_{i_k}, H_i) = \begin{cases} 0 & \text{if } H_i = 0 \text{ or } A_{i_k} \notin S_i \\ \frac{1}{|S_i|} & \text{if } H_i \neq 0 \text{ and } A_{i_k} \in S_i \end{cases}$$

By construction, the influence indices of all children on their parent node add up to 1. Therefore, the reward of a delayed arrival A_i (in terms of PAX and airline utilities) of flight f_i can thus be split among the hold decisions (e.g., H_i directly and H_k indirectly) in the DT rooted at A_i. Once this is done for all flights f_i that arrive in a window $[t, t + W]$, we can estimate the global PU P_G^f and AU A_G^f attributed to the hold decision at flight f at time t. Multiple paths from a hold to a delay are unlikely, but can be handled by calculating the shortest or even the average across the paths.

6 EXPERIMENTAL SETUP

6.1 Simulator

To model the environment of airline network operations, we implemented a discrete event microsimulator in Python. The main events that the simulator models are: 1) arrivals; departures; holds from the airline perspective; and 2) PAX delayed at destination or missing flights. Using these events, the simulator captures PAX movement across flights (without micro-modeling the intra-airport movement) and the propagation of the hold delays across flights through 1) subsequent flights of the same held tail; and 2) PAX delayed due to holds who may trigger subsequent holds.

Delay propagation: We used real-world tail plan data for 2 years from two different airlines (Air-East and Air-West) operating with hubs in different geographies. Air-East has 460+ flights per day using 130+ aircrafts to 130+ destinations with one major hub. Air-West has 1600+ flights per day, using 800+ aircrafts, to 340+ destinations with 8 major hubs. We used actual and planned times of arrival and departure to estimate delay distributions for various flights. When samples were limited for specific flights, we estimate the coefficient of variance around the expected flight time using clustering with other flights of similar characteristics according to short, medium, and long haul flights. Intrinsic delays are generated using the underlying delay distributions and propagated stochastically along with hold delays along the tail plan.

PAX profiles: We obtained the aggregate real-world PAX-metrics for Air-East directly from the airline and for Air-West from [19]. These give the number of missed connections and the total number of connections made at hubs. In addition, we obtained number of connections between pairs of flights (or cities) and used that to synthetically generate PAX itineraries that are statistically representative of the original data. Specifically, we obtain a routing matrix that indicates what fraction ρ_{ij} of a flight f_i contributes to a departing flight f_j. We estimate the mean and deviation of ρ across flight pairs and sample approximate PAX itineraries while respecting correlations induced by 1) the total occupancy of flights; and 2) in-flow and out-flow constraints at airports.

Context engine to generate forecasted state variables: For a given hold τ, we estimate whether a PAX misses the connection or not depending upon whether the connection time exceeds the MCT. For PAX who do not miss, the delays are just the estimated delay of the flight. For estimating the final delays for missed PAX, we assign them to the next scheduled flights to the same destinations, which can accommodate them depending upon their simulated occupancy. We do not distinguish across PAX by ticket class and assign them to next available flights in a first-cum-first-served manner. Once PAX are assigned to future flights, their delays to the final destination is estimated by the context engine as the scheduled ETA of the rebooked flight. This provides the forecasts for the PU. Note that once the PAX has been rebooked, we have sampled

Table 3: Hyper-parameters for the RL learning

Parameter	*Value*
DQN architecture	(17,17,7)
A2C architecture	(17,17,(7,1))
AC architecture	((17,34,17,7),((17,7),24,17,1))
DDPG architecture	((18,18,1),((18,1),19,19,1))
Optimizer	Adam
Learning rate	0.001
Discount factor γ	0.8
Mini-batch size	32
Replay buffer	10% of training epochs

an actual availability from the distribution of availability across multiple flights to the final destination. Similarly, to forecast the AU of a held flight, we use the mean of the delay distributions used to sample delays of the flights.

6.2 RL implementation details

The RL agent is implemented outside the simulation environment using the `TensorFlow` framework. Derived values of the system state are directly captured from the simulator using book-keeping data structures. Rewards are likewise measured and attributed to various hold decisions as described before. We choose $\alpha = 0.75$ and $\beta = 0.75$ in our evaluation; their tunability is explained in Section 7.3. We use the following RL algorithms for learning: DQN, AC, A2C and DDPG. DDPG solves the problem with a continuous hold time, while the other algorithms act in a discrete action space of $[0..30]$ minutes in steps of 5 minutes. This window of 30 minutes is chosen as per business inputs for the maximum hold time permissible. Details about the neural network architectures and the hyperparameters used in the learning and testing are shown in Table 3. The training consisted of 25 episodes(1 episode = 1 week) and testing, 5 episodes. Air-East (Air-West) with about 460 (1613) flights per day takes 20.4 (173) hours to train for a period of 151 days. The testing is faster and takes 4.1 (32) hours for a period of 30 days for Air-East (Air-West). The server utilized is a 16 core machine with 32GB RAM.

Metrics and baseline: The business metrics are: 1) number of missed connections reduced; 2) the on-time performance of the airline; defined as the arrival at destination within 15 minutes of the scheduled time; and 3) the average arrival and departure delays in the system. For RL metrics, we use the average reward of the agents; and the Q-value or the critic's value function; and the loss of the neural approximators. We use the following baselines: 1) No-Hold, which does not allow for any holds. 2) Heuristic-15, the current industry standard heuristic that allows holds up to 15 minutes if the ETA after holding is within 15 minutes of the scheduled time; and 3) Heuristic-30, a 30-minute variant of Heuristic-15, to ensure a fair comparison with the RL action space of 30 minutes.

7 RESULTS

7.1 Simulator validation

PAX profiles validation: A heat-map of the connectivity matrix across cities is shown in Figure 3 for both the input connectivity

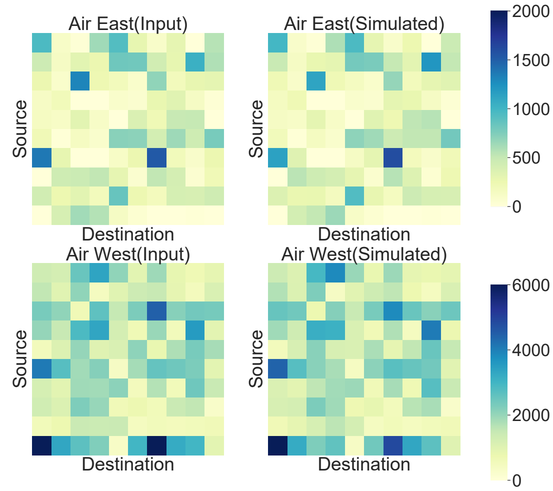
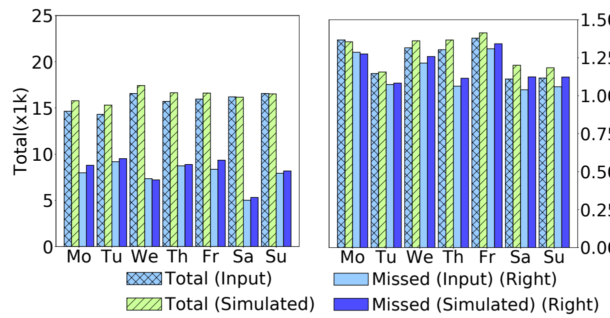


Figure 3: Validation of PAX profiles.

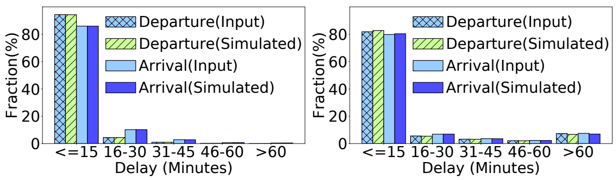


(a) Air-East **(b) Air-West**

Figure 4: Validation of connections.

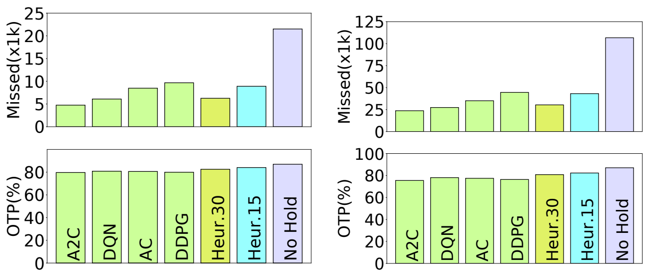
matrix; and the synthetically generated PAX itineraries. The Y-axis (X-axis) shows the source (destination) airports. The map itself shows the intensity of the connections between the source and destination airports. We find that both the input specification and the output generated by the model are similar validating the PAX patterns at an aggregate level. The matrix is shown for the pairs of cities with the highest number of connections. We observe similar patterns for other cities and at a flight-level and omit the visualization due to the sparsity of the connections. The average relative error across the top 10 pairs of cities is 0.65% for both airlines.

Missed connections: Figure 4 shows the baseline number of PAX missing connections over multiple days of the week without holds. The X-axis shows the day of the week. The primary Y-axis shows the total number of PAX connections for that day; and the secondary Y-axis shows the number of PAX missing connections on that day. An average 3% (5.5%) of connecting PAX for Air-East (Air-West) miss their connections during normal operating conditions due to inherent delays in airline operations. The simulation matches the input data well. The average relative error between the input and the simulated number of missed connections is around 4.95% across the entire dataset for Air-East; and 3.42% for Air-West.

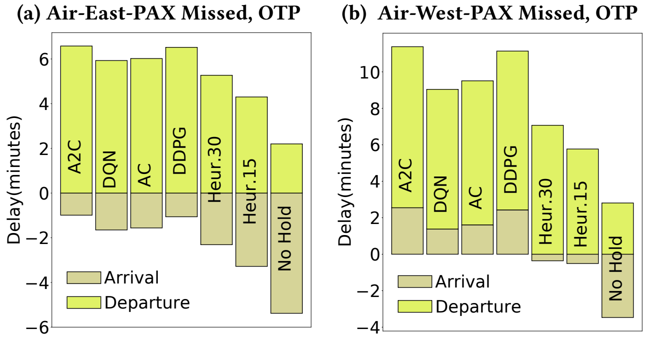


(a) Air-East **(b) Air-West**

Figure 5: Validation of network delays.



(a) Air-East-PAX Missed, OTP **(b) Air-West-PAX Missed, OTP**



(c) Air-East-Delays **(d) Air-West-Delays**

Figure 6: Business metrics for Air-East and Air-West - Missed PAX; OTP; and Delay.

Network delays: Figure 5 shows the histogram of the arrival and departure delays for both Air-East and Air-West. Again, we find that the distributions of the input actual delay data and the simulated data are close to each other. In particular, the simulated data matches well even in the tail of the distribution. For Air-East, the average OTP across aggregate, short, medium, long haul flights is around 88.5% (88.22%), 87.68% (87.69%), and 81.9% (84.08%) for the actual (simulated) data. Similarly, for Air-West, the average OTP across aggregate, short, medium, long haul flights is around 86.06% (86.52%), 87.83% (87.78%), and 87.55% (86.5%) for the actual (simulated) data. These observations confirm that the OTP metrics in the input data are well captured by the simulation. Having validated that the simulator environment is a reasonable representation of the network operations, we now evaluate the RL models.

7.2 Business metrics

Figures 6a and 6b show the PAX missed and on-time performance for Air-East and Air-West respectively. For each business metric, the results for the four RL algorithms, No-Hold, Heuristic-15 and Heuristic-30 are shown for a testing period of 1 month and a training period of 5 months. For Air-East, we find that the best RL algorithm (A2C) reduces 80% missed connections compared to the no-hold baseline; 50% compared to the currently used Heuristic-15; and 24% compared to Heuristic-30. The corresponding decrease in the OTP is about 8%, 5% and 3% compared for no-hold, Heuristic-15, and Heuristic-30 respectively. For Air-West, it reduces missed connections by 78%, 51% and 25%; with 13%, 8%, and 6% decrease in OTP when compared for no-hold, Heuristic-15, and Heuristic-30 respectively for A2C. We note that the learning approach reduces missed connections significantly for a marginal decrease in OTP.

OTP explained: The extent of decrease in the OTP can be explained using Figure 6c and 6d. The X-axis shows the various algorithms. The Y-axis shows both the arrival (AD) and the departure (DD) delays. We find that with no-hold for Air-East (Air-West), there is an arrival delay of -5.38 (-3.47) minutes; though the departure delay is 2.81 (2.2) minutes. The total *slack* in the airtime available to still arrive on-time is 7.58 (6.28) minutes. As the RL agent starts holding flights, the departure delay increases, consumes the available slack, and results in an increased arrival delay; which in turn decreases the OTP. We observe that even Heuristic-15 causes a fall in the OTP. This is because the ETA used for arrival prediction is subject to randomness in the air-time, etc., and does not guarantee OTP. Among the RL algorithms DDPG shows the worst performance. This is likely because DDPG is unable to handle the jumps in the state-value as a function of the action (hold-time).

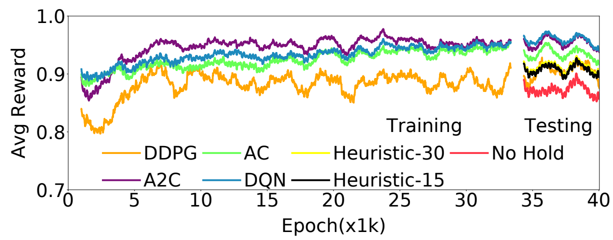
7.3 RL metrics

Figures 7 shows the RL metrics for Air-East. Figure 7a shows the average reward obtained by the algorithm during the training and testing phases. The X-axis shows the epoch. The Y-axis shows the smoothened reward over the last 1000 epochs. We find that for DQN and A2C, the testing performance seems comparable with the rewards obtained over a sliding window of the last 1000 epochs. However, for AC, the performance during testing is lower suggesting that the algorithm has not learned to generalize well. Figure 7b shows the Q-value or the critic value function where applicable. The X-axis shows the epoch in each episode, and the Y-axis shows the smoothened value. We find that all the algorithms learn. The value function increases and then saturates as the learning yields to exploitation. Finally, Figure 7c summarizes the average error in the neural networks used in the various algorithms. The error falls rapidly and then saturates indicating that the networks approximate their learning targets well. Air-West results are similar except that the testing reward, in general, is lower than training due to the higher complexity of Air-West's airline network.

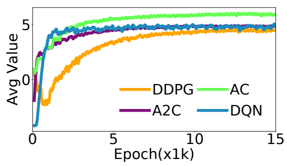
Tunability: In practice, an airline would require a tunable way of trading off OTP for saving PAX. Figure 8 explores the trade-off between the OTP and the reduced missed connections. Each point in the graph shows the output of the RL agent trained for a specific value of the control knobs α (trade-off between PAX missed connections and OTP) and β (trade-off between local and global rewards). The X-axis shows α, while different curves show varying β. The primary (secondary) Y-axis shows the number of PAX connections saved in solid lines (OTP in broken lines). For each

Table 4: Air-west metrics for No Hold, variants of baseline heuristics; and RL state and reward.

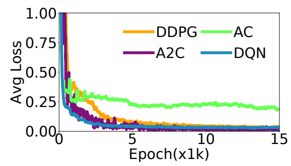
Version	Misconn. (PAX)	DT (%)	OTP (%)	DT (%)	Converge. (epochs)	DT (%)
No Hold	106,698	NA	87.0	NA	NA	NA
Heuristic-15	43,149	NA	82.3	NA	NA	NA
Heuristic-15++	48,201	NA	83.9	NA	NA	NA
Heuristic-30	32,896	NA	80.0	NA	NA	NA
Heuristic-30++	38,854	NA	82.3	NA	NA	NA
RL-V1	83,323	4.3	81.9	2.9	1,969	-6.6
RL-V2	47,137	0.1	79.9	4.2	2,168	0.3
RL-Final w/o τ^*	23,149	5.5	74.4	4.7	8,089	37.6
RL-Final	23,402	5.2	76.2	5.0	5,332	38.7



(a) Average reward



(b) Value



(c) Neural network errors

Figure 7: RL metrics (smoothed)- Air-East

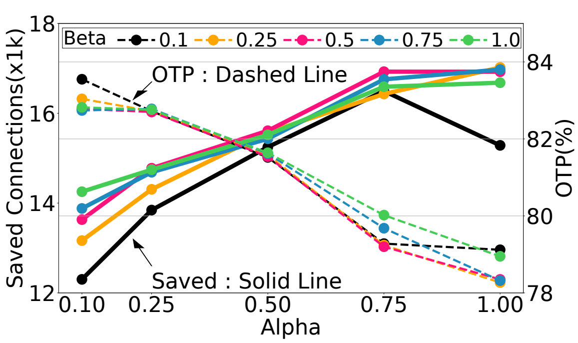


Figure 8: Learned tunable trade-off between OTP and missed connections. Generated for A2C on Air-East.

value of β, α is varied and the number of *saved* connections and OTP are plotted as one solid and broken line respectively. We observe the following. First, as α increases, saved connections increases till it reaches diminishing returns. OTP on the other hand steadily decreases with increasing α. This suggests that a choice of $\alpha^* = 0.75$ results in a pareto-like maximum benefit for PAX saved for a given increase in the OTP. Second, the curves are relatively insensitive to β, except for the corner cases of $\beta = 0.1$ and $\beta = 1$. This suggests that the global local trade-off starts resolving as long as β is non-zero. We conclude that RL with hyper-parameters is a more tunable and flexible approach than the rule-based baseline heuristics.

Transfer learning: We evaluate the ability of the model to transfer learn across airlines by training the RL agent on Air-East and testing it on the Air-West after retraining. When compared to training from scratch, we find that the number of misconnections increases by around 12% while the training time is cut-down by around 50%.

Variations: Table 4 summarizes experiments with No Hold, variants of the baseline heuristic; and the RL state and reward functions. Each metric shown is the version that includes the DT; and the column DT shows the % improvement because of using the DT. The "++" variants of the heuristics denote allow holds if a minimum number of PAX connections are saved. RL still performs better than the improved baseline heuristics. RL-V1 and RL-V2 denote variants of RL where PU reward considers all PAX; while RL-Final allows only misconnecting PAX. Similarly, RL-V1 uses 0-1 binary AU which jumps from 1 to 0 at 30 minutes; while RL-V2 and RL-Final vary the AU from 1 to 0 smoothly from 15 minutes to 30 minutes. In sum, we find that RL-Final works best in saving PAX; and using the DT heuristic for global reward apportioning helps one or more of PAX saved; OTP; and RL convergence time in epochs. Using the helper variable τ^* improves the convergence time significantly by 247.3% with DT and 243.2% without DT while giving results similar ($\leq$ 1.1% difference) to extensive training without τ^*.

8 DISCUSSION AND CONCLUSION

We summarize a few limitations of our approach and possible improvements. First, our approach learns to exploit the slack in the network by learning the optimal behavior on an *average* during *normal* operations. During network-wide delays (e.g., as defined by a change-point algorithm), it is better to fall back to no-holds. Second, we assumed that PAX on a delayed flight make the connection if we hold for at least the extent of the incoming delay. A higher-fidelity simulator could model the stochastics of PAX movement within an airport; depending upon whether the arrival and departing flights are in the same terminal; the gate-to-gate delay; etc. Third, an alternate approach to learning could be to observe an expert flight operations manager and imitate their behaviour. Fourth, a more comprehensive reward function could use differing rewards for different flights in terms of both missed connections reduced and delays. Further, it could use realistic costs from historical data if available for rebooking costs; and delays. Such data is proprietary and has been abstracted as the control knobs in our work. RL can readily use them if available.

We trained and tested an RL agent for HNH on a simulated testbed using a state-space and reward function designed to trade off decreasing missed connections for decreasing on-time performance. We used a delay tree to approximately attribute global rewards to individual actions. We observed that RL agents save significantly more PAX connections, even while the on-time performance decreases marginally. Our approach is tunable and able to transfer learning across airlines.

REFERENCES

[1] Hasnain Ali, Yash Guleria, Sameer Alam, and Michael Schultz. 2019. A passenger-centric model for reducing missed connections at low cost airports with gates reassignment. *IEEE Access* 7 (2019), 179429–179444.

[2] Andrew G Barto, Richard S Sutton, and Charles W Anderson. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE transactions on systems, man, and cybernetics* 5 (1983), 834–846.

[3] James E Bostick, Munish Goyal, Kimberly G Starks, and Michael Wendel. 2019. Optimizing real-time and planned air-traffic. US Patent 10,283,001.

[4] Paul Bressloff. 1995. Stochastic dynamics of reinforcement learning. *Network: Computation in Neural Systems* 6, 2 (1995), 289–307.

[5] Luis Cadarso and Ángel Marín. 2013. Robust passenger oriented timetable and fleet assignment integration in airline planning. *Journal of Air Transport Management* 26 (2013), 44–49.

[6] A. J. Cook, Graham Tanner, and A. Lawes. 2012. The Hidden Cost of Airline Unpunctuality. *Journal of Transport Economics and Policy* 46, 2 (2012), 157–173.

[7] F. L. Da Silva, R. Glatt, and A. H. R. Costa. 2017. MOO-MDP: An Object-Oriented Representation for Cooperative Multiagent Reinforcement Learning. *IEEE Transactions on Cybernetics* 49, 2 (2017), 567–579.

[8] Yu Gu and Christopher A Chung. 1999. Genetic algorithm approach to aircraft gate reassignment problem. *Journal of Transportation Engineering* 125, 5 (1999), 384–389.

[9] Suna Hafizogullari, Prathi Chinnusamy, and Cenk Tunasar. 2002. Simulation reduces airline misconnections: a case study. In *Proceedings of the Winter Simulation Conference*, Vol. 2. IEEE, 1192–1198.

[10] Gabriel Hondet, Luis Delgado, and Gérald Gurtner. 2018. Airline disruption management with aircraft swapping and reinforcement learning.

[11] Ilhan Ince, Lourdmareddy Gumireddy, and Noel Alfonso. 2014. Misconnect management systems and methods. US Patent App. 13/837,462.

[12] Shan Lan, John-Paul Clarke, and Cynthia Barnhart. 2006. Planning for robust airline operations: Optimizing aircraft routings and flight departure times to minimize passenger disruptions. *Transportation science* 40, 1 (2006), 15–28.

[13] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971 (2016).

[14] Binod Maharjan and Timothy I Matis. 2011. An optimization model for gate reassignment in response to flight delays. *Journal of Air Transport Management* 17, 4 (2011), 256–261.

[15] Lindsey A McCarty and Amy EM Cohn. 2018. Preemptive rerouting of airline passengers under uncertain delays. *Computers & Operations Research* 90 (2018), 1–11.

[16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*. PMLR, 1928–1937.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. (2013).

[18] United Newsroom. 2019. United Airlines Makes Connecting the World Easier Than Ever with ConnectionSaver. https://hub.united.com/united--makes-connecting-easier-connectionsaver-2638762086.html

[19] U.S. Department of Transportation. 2019. Bureau of Transportation and Statistic. https://www.transtats.bts.gov/

[20] Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noboru Sean Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. 2019. *The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition.* Technical Report.

[21] Moschoula Pternea and Ali Haghani. 2019. An aircraft-to-gate reassignment framework for dealing with schedule disruptions. *Journal of Air Transport Management* 78 (2019), 116–132.

[22] Inc. RBI. 2019. Flightstats. https://www.flightstats.com/

[23] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. 2016. High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[24] Yoav Shoham, Rob Powers, and Trond Grenager. 2003. Multi-agent reinforcement learning: a critical survey. (2003).

[25] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.