Runway Configuration Assistance: An Offline Reinforcement Learning Method for Air Traffic Management

Milad Memarzadeh* and Krishna Kalyanam†
NASA Ames Research Center, Moffett Field, CA 94035, USA

Runway configuration management deals with the optimal selection of runways and their direction of operation for aircraft arrivals and departures. The configurations are chosen based on the traffic, surface winds, and other meteorological conditions that are complex to model and difficult to predict. In this paper, we develop the runway configuration assistance (RCA) tool, an automated approach based on offline model-free reinforcement learning (RL) that provides decision support for air traffic controllers (ATCos). The proposed tool processes historical data of interest, including decisions made regarding the runway configuration, and their subsequent outcome, to identify a policy that encourages good decisions. The policy search is guided by an appropriately chosen weighted multi-objective utility function (e.g., based on maximizing traffic throughput, minimizing transit times on the surface of the airport, and mitigating safety issues such as go-arounds). The proposed tool is validated using data from two major US airports based on performance metrics developed in collaboration with subject matter experts and is compared against several baseline approaches such as the most frequent configuration chosen by ATCos, supervised learning, and other RL-based approaches.

I. Introduction

Every airport, depending on the geometry of the runways and different directions they can operate on, have a set of unique runway configurations that can be used for arriving and departing aircraft. The basic idea is to have a headwind, since it helps with both the lift for take-off and braking for landing. The optimal setting of the runway configuration depends on several factors such as the traffic load, meteorological conditions (e.g., wind direction and speed, cloud ceiling, visibility), safety concerns (e.g., possibility of go-arounds due to excessive tail and/or cross winds), and noise abatement procedures, controller staffing or other operational issues, runway surface conditions (e.g., wet runway), and runway closures/construction. Once an airport is in an active runway configuration, choosing the optimal time-window to switch the configuration to a different one is a crucial yet challenging task that the Air Traffic Controllers (ATCos) and front-line managers (FLMs) deal with daily. Moreover, switching a configuration takes considerable time

*AI Technical Lead, Aviation Systems Division, NASA Ames Research Center. Corresponding Author: milad.memarzadeh@nasa.gov.
†Senior Aerospace Engineer, Aviation Systems Division, NASA Ames Research Center
A shorter version of this paper was presented at the AIAA Scitech 2024 Forum in Orlando, FL, 8-12 January 2024, with paper number AIAA-2024-0533.

and ATCos need to determine the best time-window for changing the configuration with minimal disruptions based on the traffic and weather forecast. This makes the decision-making more challenging as the forecast of traffic and wind conditions are usually uncertain and change often.

A sub-optimal selection of the runway configuration, or poor timing of configuration changes, can result in significant increase in taxi times for aircraft on the surface of the airport, unnecessary aircraft holding (in the air or on the ground, causing significant delays), and undesired go-arounds for arriving traffic in the air. For example, Federal Aviation Administration (FAA) reports that a maximum allowed crosswind and tailwind components are 25/15 and 10/10 knots for dry/wet runways [1] (although these numbers can be different for different airports/runways or aircraft). Currently, ATCos set the runway configurations based on the current and forecast data relating to traffic and meteorological condition available to them. Due to different preferences and biases of human decision-making, this process may result in subjective decisions and yield sub-optimal results, especially if the predicted outcomes are not realized. Moreover, due to the high amount of uncertainty in the (weather) forecast, the stochastic decision-making requires an extensive combinatorial search among all possible predicted outcomes to identify the optimal policy, which is difficult to perform by human reasoning alone or by using a rule-based system. Furthermore, if an automated rule-based system can be defined, it would be difficult to scale across airports in the National Airspace System (NAS), because each airport would require a complete definition of such rule-based decision-making mechanism from scratch, and it would be subject to change in operational settings (e.g., runway closures or construction).

As a result, there is a need for a data driven approach based on machine intelligence that can alleviate the complexity of the decision-making and facilitates informed decision-making based on the abundance of available historical data and decisions. This would also decrease the possibility of induced bias in the human decision-making and center the rationale based on observed data in the past and forecast data in the near future. Furthermore, such automated and intelligent approach would require minimal work to generalize to other airports across the NAS. A successful development and deployment of such technology will have significant implications for the efficiency, productivity, and safety of the operations at airports across the NAS and hence is the focus of this paper. It would decrease the workload of the ATCo significantly and provide a decision-support for their complex tasks on an hourly basis; this is crucial given the rise in the magnitude and complexity of airspace operations. Furthermore, it improves the efficiency of the commercial traffic at airports, resulting in less delays, shorter taxi-times on the surface of the airports, fewer number of aircraft performing go-arounds, fuel savings, and emissions. Lastly, it can improve the coordination between the FAA and airlines.

II. Related Work

Recent studies have attempted to develop an automated and intelligent decision support tool for runway configuration management (RCM). We categorize this literature into three main categories of (1) model-based control, (2) model-free

control, and (3) data-driven supervised learning. It should be noted that both model-free control and data-driven supervised learning can be categorized as model-free approaches, as they do not characterize or learn any model for the dynamics of the system's state [2].

The first category is the development of model-based control approaches, where a specific mathematical model is built to represent the dynamics of the traffic and meteorological conditions and their effects on the choice of runway configuration (modeling all variables affecting the system). Once such model is defined, different techniques can be used to identify the optimal policy for selecting the runway configuration at each time interval. Examples of such techniques are heuristic search [3, 4], discrete choice modeling [5], mixed integer programming [6], dynamic programming [7], and queuing theory [8–10]. Model-based approaches are interpretable, since they learn a specific model for changes in the traffic, meteorological conditions, and other influential factors in the decision-making. Moreover, they provide guarantees on the near-optimality of obtained policy for the runway configuration. However, the performance of model-based approaches and the associated guarantees are dependent on the accuracy of the learned model. Any modeling error or simplification that are made to alleviate the complexity of the problem (e.g., assuming simplified weather dynamics model [5]), can result in a poor performance in the operational setting, especially when the operational data diverge from the model's expectation. Furthermore, since the model characterization varies from airport to airport, it is expensive to generalize this family of methods to other airports in the NAS. Due to these drawbacks, the model-based control is not an appropriate method for the RCM problem posed in this paper, due to the complexity of characterizing an accurate and realistic model for system dynamics.

The above-mentioned drawbacks of model-based control give rise to the second category of methods called model-free control, mainly based on Reinforcement Learning (RL) techniques [11], a key framework for sequential decision-making problems. Model-free RL has been widely adopted and deployed in the aviation domain [12]. These approaches usually learn a good policy by interacting with a simulation/operational environment and learn from the feedback they receive as a consequence of their decisions. Popular methodologies in this category are Monte Carlo Tree Search (MCTS) [13] and Q-learning [14, 15] that have been widely applied in different domains. Model-free RL is generally easy to implement and efficient to scale. However, the online implementation of model-free RL has one major drawback, which limits its applicability to safety critical systems. It needs a significant number of interactions (episodes of making decisions and learning based on the feedback from the simulation/operational environment) to learn a good policy. Moreover, to learn a good policy, such interactions need to balance between exploitation (making decisions that optimizes the received long-term expected rewards), and exploration (making new and often risky decisions for the sake of learning a better policy). As a result, these interactions can be costly, especially in application to safety critical systems such as Air Traffic Management (ATM), since the algorithm tends to explore poor decisions when the interactions are limited. Hence, majority of the online model-free RL techniques have been applied to domains such as gaming [16] or aviation-specific applications where accurate simulators are available [12].

With the significant rise in the size and quality of available data in ATM in recent years, and to address the major drawback of online model-free RL, the third category of methods have focused on development of data-driven supervised learning techniques. The main goal of methods in this category is to use the vast amount of available historical data and learn to imitate the ATCo decision-making process with the least amount of error [17–20]. Since such approaches only rely on historical data, it is easy to generalize them to airports across the NAS. However, these techniques suffer from one fundamental drawback: since the policy optimization is designed to mimic the ATCo with the least amount of error, they cannot identify and correct mistakes and/or inefficiencies in the historical decisions. In other words, their predictions are not supported by any evidence of better outcomes (such as more efficient surface transit times or a smaller number of go-arounds). Simply put, they learn to mimic the historical decisions, both good ones and bad ones.

A. Our Contribution

In this paper, we develop a solution to the RCM problem based on the family of offline model-free RL [21]. The proposed solution combines the power of model-free RL with data-driven supervised learning and attempts to learn a good policy by only relying on the historical data/decisions, while addressing the shortcomings of the above-mentioned literature: (1) it addresses the shortcoming of model-based control (first category in the literature), as it does not rely on learning a model of the environment and learns the policy directly from data; (2) the offline nature of the method does not require further interactions with the simulation/operational environment and learns a policy solely based on historical decisions, hence addressing the drawback of the second category in the literature (i.e., model-free control); and (3) learning of the policy is supported by a well-defined utility function (also sometimes referred to as reward function), that can differentiate good and bad decisions in the historical data, hence addressing the limitation of data-driven supervised learning techniques (third category in the literature).

Although offline RL addresses majority of the drawbacks in the mentioned literature, there is a fundamental challenge in their deployment, called distributional shift, that needs careful consideration. It happens when the policy that the algorithm learns from historical data is significantly different with respect to the policy that was used (by the ATCos in an operational setting) to collect the data (referred to as behavioral policy). This would result in the algorithm being overly optimistic (and probably wrong) when exposed to Out-of-Distribution (OOD) data, a setting that is rare and not well represented in the historical data, as well as situations that might be created by new operational procedures and are not observed in historical data. For example, a runway that is historically solely utilized during specific wind conditions is not available for a long period of time due to constructions. This would result in an OOD situation in practice. Although classic offline RL approaches fail to address distributional shift, the recent developments in the field show promise. A majority of the state-of-the-art offline RL algorithms deploy a mechanism to address this challenge, such as constraining the policy optimization problem [22], deploying an ensemble approach [23], applying regularization to avoid the excessive distribution shift from the behavioral policy [24], or learning a risk-averse policy [25]. These recent

endeavors have allowed offline RL to be successfully deployed in real-world applications such as robotics and healthcare.

In this paper, we specifically implement a state-of-the-art offline RL algorithm called Conservative Q-Learning (CQL) [24], to develop the Runway Configuration Assistance (RCA) tool. CQL uses a simple regularization technique (explained in Section III.B) to alleviate the distributional shift in the policy. To validate the RCA tool, we use two years of real-world data from two major airports in the US: Charlotte Douglas International Airport (CLT), as an example of "less complex" airport, and Denver International Airport (DEN), as an example of "more comple" airport in the NAS. As a point of comparison, we compare performance of the RCA tool with several baselines such as: (1) ATCo preference: a simple policy that uses the most common decisions made by the ATCos in each state in the past; (2) supervised learning: based on random forest that learns to mimic the ATCo with least amount of error [26] and serve as a representative of data-driven approaches developed in the literature; and (3) an offline version of a popular Deep Q-Network (DQN) algorithm [16], that would represent a brute force application of online RL method in an offline setting without any underlying mechanism to alleviate the distributional shift in the learned policy.

In the remainder of the paper, we first share details of the implemented RCA tool, then we elaborate details of the runway configuration operations at CLT and DEN, and finish with the findings and performance comparisons between the RCA tool and the baseline methodologies.

III. Methodology

Runway configuration management by the ATCos can be viewed as a sequential decision-making process at specific time intervals, t (e.g., each lasting for a quarter of an hour). We model this problem as a Markov Decision Process (MDP) [11], which is defined by a tuple (S, A, f_T, f_U). The controller (also referred to as agent throughout the paper) observes the state of the environment at each time step, $s_t \in S$, and selects what action (i.e., choice of runway configuration) is appropriate for the current time interval, $a_t \in A$. The state space includes all available data that affects the runway configuration decision-making, such as arrival/departure traffic, wind conditions, meteorological conditions, and time of the day. Once the action is implemented, the operational environment moves into the next time interval and a new state, $s_{t+1} \sim f_T(s_t, a_t)$, according to a model called transition function (also referred to as dynamics model), with domain $f_T : S \times A \rightarrow S$. Moreover, the agent will observe a feedback related to the effect of her action in the operational environment, also referred to as utility throughout the paper $u_t \sim f_U(s_t, a_t)$, according to a utility function with domain $f_U : S \times A \rightarrow \mathbb{R}$. The feedback or utility function can include traffic throughput, average transit times on the surface of the airport, length of the queues forming on the taxiways, and number of go-arounds as a result of high winds. Although the decision-making process for runway configurations is a continuous event, due to the fidelity of the available data, we adopt a 15 minute time interval between t and $t + 1$. It should be noted that this might induce noise and/or error in the deployment of the tool in real-time, however, it should not cause any disruptions as a decision-support tool for planning purposes.

Given all of the MDP definitions, the goal the agent (controller) is to find the optimal policy, $\pi^* : S \rightarrow A$, that maximizes the accrued utilities over the management time horizon, also referred to as the optimal value function:

$$V^*(s_t) = \max_{\pi} f_U(s_t, \pi(s_t)) + \gamma \sum_{s_{t+1} \in S} p(s_{t+1} \mid s_t, \pi(s_t)) V^{\pi}(s_{t+1}), \tag{1}$$

where, $\gamma \in [0, 1)$ is a discount factor, discounting future utilities to their net present value, and $p(s_{t+1} \mid s_t, \pi(s_t))$ is the probability of starting in state s_t, taking action $\pi(s_t)$, and ending up in the state s_{t+1} according to the transition function f_T. The optimal policy represents the runway configuration for each state (that can correspond to unique traffic and wind conditions). On the other hand, the optimal value function measures the aggregate metrics (such as average transit times, throughput, and go-arounds) of implementing the optimal policy over a certain period. If the agent has full knowledge of all components of the MDP, then one can use dynamic programming [11] to find the optimal value function, V^* and associated optimal policy, π^*. This is one of the reasons that model-based approaches (described in Section II) learn a representation of such transition and utility models. However, in many real-world applications such as air traffic management, the transition (or dynamics) model is complex, and as a result not feasible to learn from limited historical observations. Furthermore, any error introduced in learning the model, would affect the quality of the policy obtained and its applicability in practice. As a result, our solution relies on a family of model-free control, specifically the popular Q-learning algorithm [14, 15].

A. Q-learning: a model-free RL solution

Model-free RL algorithms such as Q-learning plan to learn the optimal policy, π^*, by only relying on the interactions of the agent with the environment, i.e., observing state of the environment at time interval s_t and deploying an action a_t, and her observations from the environment, i.e., the next state s_{t+1} and the utility u_t. They do so by gradually learning the optimal state-action value functions (also known as Q-values, hence the name of the algorithm) via an off-policy temporal difference algorithm [11]:

$$Q(s_t, a_t) = (1 - \alpha) Q(s_t, a_t) + \alpha [u_{t+1} + \gamma \max_{a \in A} Q(s_{t+1}, a)], \tag{2}$$

where α is the learning rate. It is called an off-policy algorithm because it directly approximates the optimal Q-value, Q^*, independent of the policy being followed by the agent. As evident in Eq. (2), the Q-learning algorithm learns the Q-values gradually by interacting with the environment and observing the outcome of the interactions. As a result, the agent requires to interact with the environment for significant amount of time until the algorithm can approximate a near-optimal policy. The exact number of required interactions to guarantee a near-optimal policy is hard to quantify and is subjective depending on the complexity of the decision-making problem. In theory, in the case where each state-action pair is observed an infinite amount of time, the Q-learning algorithm is guaranteed to converge to the

optimal policy [11]. In practice, the higher the complexity of the runway configuration decision-making, the higher number of interactions needed to learn a good policy. For example, as we will see later in the two case studies of this paper, decision-making at DEN is much more complex than CLT, and as a result, would require significantly more interactions. As a result, in the early period, where the number of interactions/observations are limited, the policy obtained by the algorithm is usually far away from the optimal one and an agent using such a policy might make a lot of mistakes. This attribute limits the direct application of Q-learning algorithm to safety-critical systems such as ATM. Furthermore, the classical Q-learning algorithm stores the learned Q-values in a tabular form, which does not scale to many real-world applications with large and unstructured state/action spaces.

Mnih et al. [16] proposed Deep Q-Network (DQN) that uses a non-linear function approximator such as neural networks to estimate the Q-values instead of storing them in a tabular form. This innovation allowed the application of the Q-learning algorithm to complex real-world problems with high-dimensional and unstructured state/action spaces. In the DQN algorithm, gradient-based optimization methods are used to learn the parameters (weights and biases) of the neural network, i.e., θ, by minimizing the Mean Squared Error (MSE) between the Q-value estimates of the neural network, $Q_\theta(s, a)$, and the target Q-values, y, calculated based on the feedback from the environment:

$$Q_\theta = \arg\min_\theta \mathbb{E}_{s,a,u,s' \sim D}\Big[\left(Q_\theta\left(s, a\right) - y\right)^2 \Big], \tag{3}$$

where, $\mathbb{E}$ is the expectation operator, D represent the observed instances of one time-interval transitions (s, a, u, s'), and y is the target Q-value estimates that can be calculated as follows:

$$y = u + \gamma \max_{a \in A} \hat{Q}_{\hat{\theta}}(s', a), \tag{4}$$

where s' is the next observed state of the environment. As noted in the above definitions, two separate networks are used to estimate the Q-values, Q_θ, and the target values, $\hat{Q}_{\hat{\theta}}$. This has been shown to improve the convergence of the gradient-based optimization and faster learning of the Q-values [16]. Usually, the parameters of the Q-network, θ, is copied to the target network after each few iterations of the training or gradually through a soft-update.

Although DQN addresses one drawback of the classical Q-learning, they both only work well in online fashion, where the agent can interact with the environment (operational or simulation) for a significant amount of time and learn a good policy by receiving feedback on her interactions. However, in many real-world applications, building an accurate simulator of the real operational environment is expensive or impractical and experimenting in the actual operational environment is impossible. As a result, the agent can only rely on the historical data (historical decisions made and feedback received) to learn a good policy, i.e., offline RL. However, as described in Section II.A, direct application of these approaches in an offline setting have not been successful [21]. In this paper, we deploy a recently developed offline RL algorithm, Conservative Q-Learning, that has shown significant success in real-world deployments [24].

B. Conservative Q-learning: an offline RL solution

Conservative Q-learning (CQL) [24] addresses the main challenge of offline RL (i.e., distributional shift, discussed in Section II.A) by regularizing the estimates of the Q-values by the neural network. This additional regularization technique keeps the estimates of the Q-values for the unlikely actions (based on historical data) low, hence it lower-bounds the optimal Q-function. Due to learning the lower-bound for the optimal Q-function, Q^*, a policy chosen based on the learned function will be conservative and take less risky actions. The optimization objective of the CQL algorithm is defined as follows,

$$\begin{aligned} Q_\theta = \arg\min_\theta \alpha \mathbb{E}_{s\sim D}\left[\log \sum_a \exp(Q_\theta(s,a)) - \mathbb{E}_{a\sim\hat{\pi}_\beta(a|s)}[Q_\theta(s,a)]\right] + \\ \frac{1}{2}\mathbb{E}_{s,a,u,s'\sim D}\left[(Q_\theta(s,a)-y)^2\right], \end{aligned} \tag{5}$$

where D represent the historical data, each instance corresponding to a one time-interval transition (s, a, u, s'), generated by an unknown behavior policy $\pi_\beta(a \mid s)$ used to collect the data (e.g., the policy used by ATCo in setting the runway configurations). Also, $\hat{\pi}_\beta$ represents the estimated behavior policy based on observed historical data, and y is the target Q-values (similar to Eq. (4)) defined as $y = u + \gamma \max_{a\in A} \hat{Q}_{\hat{\theta}}(s', a)$. There are three terms in Eq. (5), the first two terms serve as the regularization terms with α governing weights associated with them: the first term (log-exponential of Q-values) minimizes the Q-values in general, while the second term maximizes the Q-values under data distribution based on estimated behavior policy $\hat{\pi}_\beta$ (this is used to obtain a tight lower-bound on the optimal Q-function, Q^*). The last term is the standard MSE error between the learned Q-values and the target Q-values similar to the Eq. (3).

The regularization technique designed in the CQL algorithm (the first two terms in Eq. (5)) guides the Q-network to learn a lower-bound on the optimal Q-values, which prevents the over-estimation of Q-values that is common in offline RL due to exposure to OOD data and function approximation error. For further details on the theoretical details of the CQL algorithm, refer to [24]. Now that we have reviewed details of the proposed method, in the next section, we design the runway configuration management problem as an MDP and specify its different components, e.g., state and action spaces, and the utility function.

IV. Runway Configuration Management

The overall flowchart for developing the runway configuration assistance (RCA) tool is illustrated in Figure 1. Among the many features that affect the runway configuration, we solicited Subject Matter Expert (SME) - retired ATCos opinion to select the most influential features to include in the state and action spaces. Additionally, we used their feedback to define the utility function. We use two main sources of data: FAA's Aviation System Performance

Metrics (ASPM) reports and NASA's Sherlock Data Warehouse to extract the relevant features. Most of traffic related data is collected from ASPM, while the weather and meteorological data is collected from METeorological Aerodrome Reports (METARs) processed by Sherlock, and safety related data (such as go-arounds) are collected from Sherlock data warehouse.

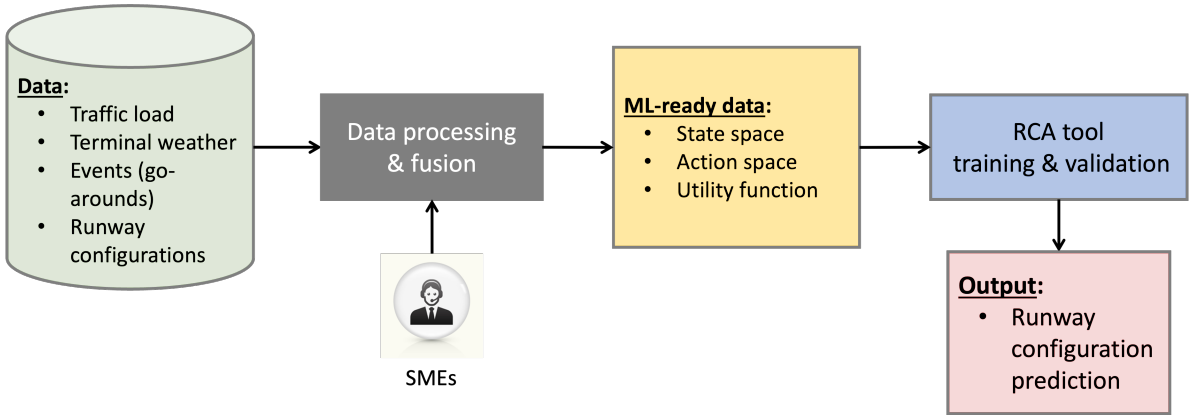

Fig. 1 The overall flowchart of development of the RCA tool.

A. State space

We define the state space, S, based on four features: (1) hour of the day, (2) wind direction, which we discretize into eight unique states as depicted in Figure 2 (left panel), (3) wind speed, discretized into six unique states depending on the intensity of the wind, as depicted in Figure 2 (right panel), and (4) the meteorological conditions which is binarized into two unique states of Visual flight rules (noted as V) and Instrument flight rules (noted as I) depending on the visibility and the cloud ceiling. State V represent good/normal conditions while state I represent low-visibility and non-optimal conditions.

https://aspm.faa.gov/
https://sherlock.opendata.arc.nasa.gov/sherlock$_$open/
https://www.aviationweather.gov/metar

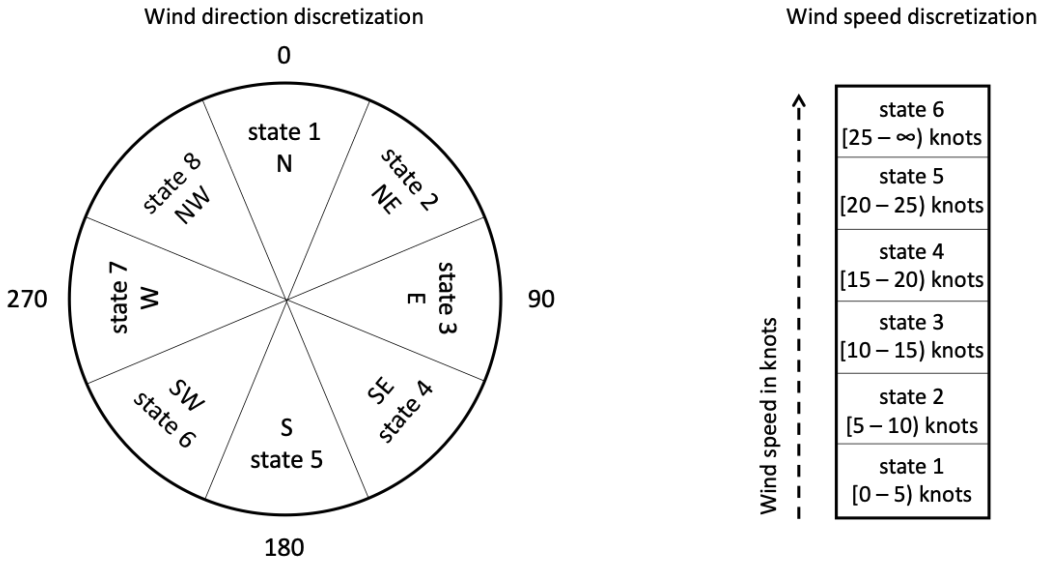

Fig. 2 This figure shows the descretization of wind direction into eight unique states (left) and the wind speed into six unique states (right).

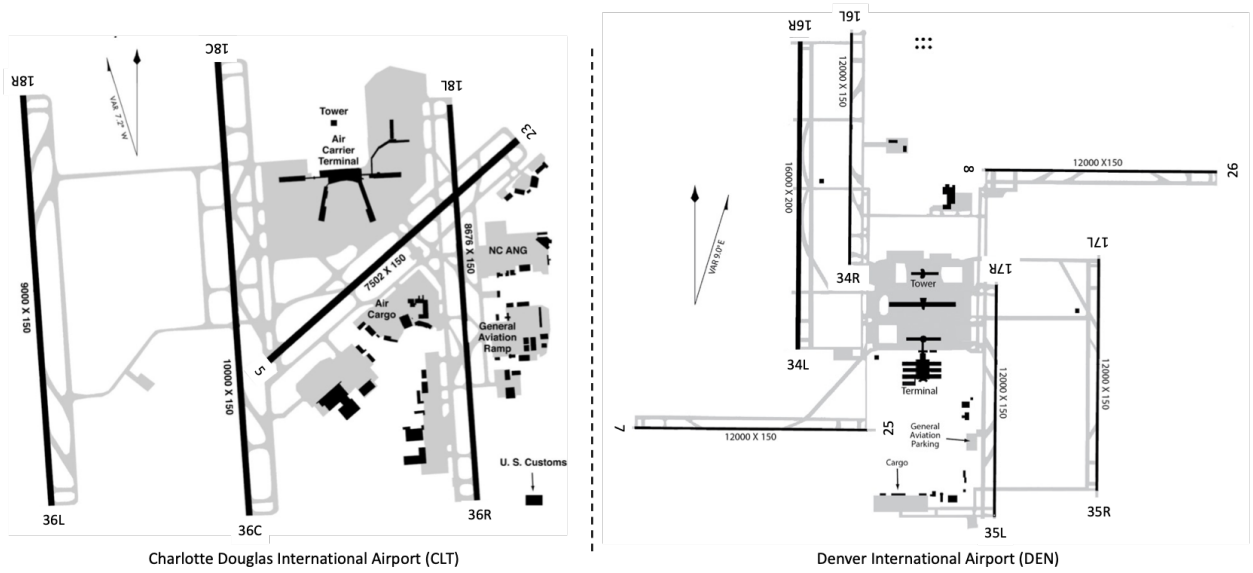

Fig. 3 Airport runway diagrams for CLT (left) and DEN (right).

B. Action space

The set of actions, A, available to the CQL agent and/or the ATCo are defined based on the used runway configurations at two airports that are focus of this study, i.e., Charlotte Douglas International Airport (CLT) and Denver International Airport (DEN). Figure 3 shows the runway diagrams for CLT (left panel) and DEN (right panel).

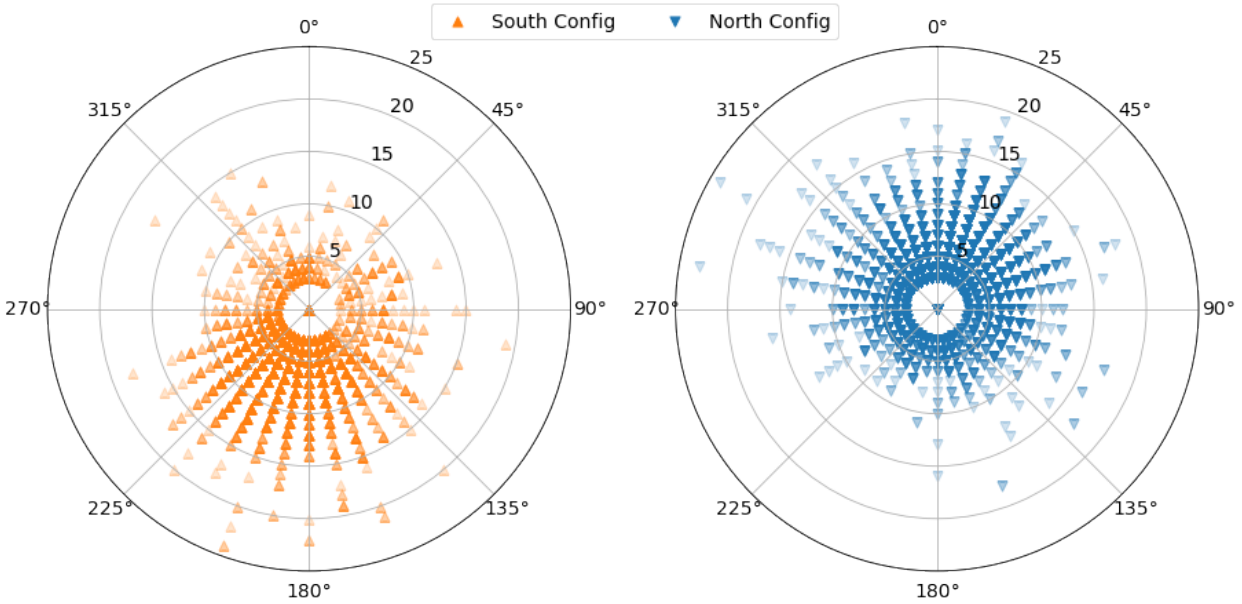

Fig. 4 This figure shows correlation of selected configuration with wind direction and wind speed at CLT for year 2019.

CLT is an example of a major airport with less complex runway configuration options in the NAS. It has three parallel runways and a short diagonal one that is rarely used. There are two major configurations at CLT, the North-bound flow and the South-bound flow. In the North flow, runways 36L, 36C, and 36R are the main used runways for arrival and departure, while in the South flow, the opposite directions of these runways, i.e., runways 18R, 18C, and 18L, are used by the air traffic. As a result, the main decision in setting the runway configuration is to use one of these flows depending on the traffic load, weather, and meteorological conditions as well as noise abatement procedures. It should be noted that noise abatement procedures are not explicitly encoded in the model currently, however, the model can learn the preferred configurations as they are more utilized in normal operating conditions. Figure 4 shows that the runway configuration decision at CLT is heavily influenced by the wind direction and speed, when wind is blowing from North, the North configuration is preferred and vice versa. Each data point in the figure shows the direction with respect to the North that the wind is blowing from, and the speed is noted as the distance from the center of the diagram. The intensity of the color for each point represents the amount of historical data available for the specific wind condition, the higher the intensity, the more represented in the historical data. It should be noted that when the wind is calm (typically less than 10 knots), the North and South configurations are used interchangeably to maintain stability of the airport operations (by simply keeping the previous configuration) and to respect airline preferences due to proximity of terminal gates to the used runways, etc. Overall, the historical data has shown that the North configuration is the preferred one at CLT by the ATCo (used on average 61% of times).

On the other hand, DEN is an example of an airport with (complex) multiple runway configuration settings. As

depicted in Figure 3 (right panel), it has six runways, among which four of them (34L/16R, 34R/16L, 35L/17R, and 35R/17L) are North/South bound while the other two runways (7/25 and 8/26) are East/West bound. As a result, there are more combinations of runways that can be used for arrival/departure by the ATCo. Based on our comprehensive analysis of the data for years 2018 and 2019, and feedback from SMEs, eleven major configurations are identified, as reported in Table 1. For example, for the configuration named N/NEW, North-bound runways (34R/L and 35R/L) are used for both arrival and departure, while runways 8 (East-bound) and 25 (West-bound) are used only for departure.

Table 1 Major runway configurations for DEN.

Configuration [Arr/Dep]	Arrival Runways	Departure Runways	Usage Frequency [%]
SE/SE	16R/L, 17R/L, 7, 8	16R/L, 17R/L, 7, 8	18.8
S/S	16R/L, 17R/L	16R/L, 17R/L	15
N/NEW	34R/L, 35R/L	34R/L, 35R/L, 8, 25	14.5
S/SEW	16R/L, 17R/L	16R/L, 17R/L, 8, 25	12.6
N/N	34R/L, 35R/L	34R/L, 35R/L	12.3
NE/NE	34R/L, 35R/L, 7, 8	34R/L, 35R/L, 7, 8	11.7
NW/NW	34R/L, 35R/L, 25, 26	34R/L, 35R/L, 25, 26	8.6
SW/SW	16R/L, 17R/L, 25, 26	16R/L, 17R/L, 25, 26	3.4
E/E	7, 8	7, 8	1.6
NS/EW	34R/L, 35R/L, 16R/L, 17R/L	8, 25	1.2
W/W	25, 26	25, 26	0.3

Figure 5 shows the heatmap of changes in the runway configurations at DEN by the ATCo in one year. The rows show the flow (configuration) at each time interval and the columns show the flow at the next time interval. The diagonals of this matrix represent no changes in the configuration and as they represent the majority of the data (cases where the configuration does not change from one time interval to the next), we have masked them in this figure for better illustration of the flow changes and its complexity. A majority of the flow changes are intuitive, for example the configuration SE/SE transits to either S/S or S/SEW a majority of the time, depending on the changes in the operational conditions. However, we also observe sudden major changes in the configuration often at DEN, for example switches between NE/NE to SE/SE or N/N to S/S. This sudden shift could be due to major shifts in the wind conditions and/or changes in operational procedures.

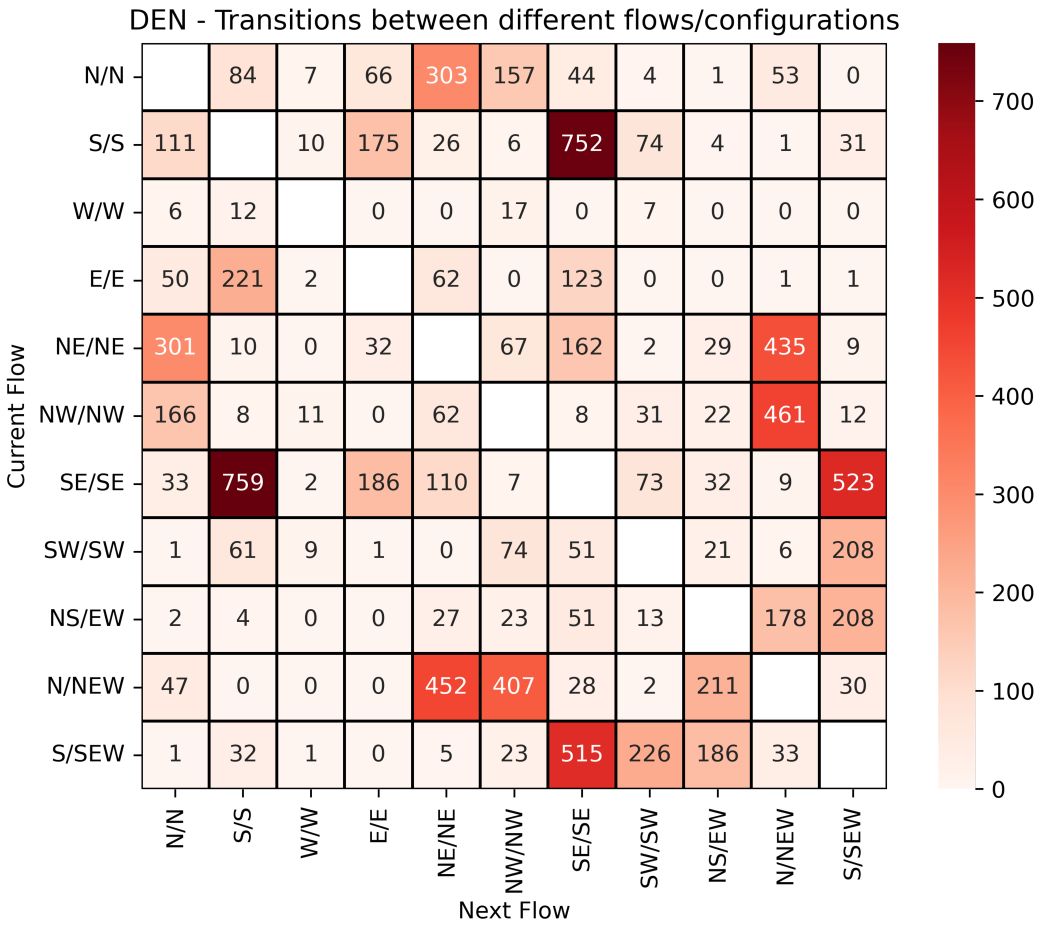

Fig. 5 This heatmap illustrates the frequency of configuration changes at DEN made by the ATCo and the complexity of the decision-making process.

C. Utility function

We define the utility function based on the combination of factors that are affected by the decisions made by ATCo about the runway configurations, as well as the invaluable feedback from the SMEs. It is defined as follows,

$$u_t = \lambda v_t - \mu \tau_t - \beta c_{t,ga} - \eta c_{t,mga} - \zeta \mathbb{I}[a_t \neq a_{t-1}] \tag{6}$$

Where, v_t is the traffic throughput (incoming and outgoing) at time interval t, τ_t is the average transit times on the surface of the airport, $c_{t,ga}$ is the number of aircraft performing a single go-around, $c_{t,mga}$ is the number of aircraft performing multiple go-arounds, and $\mathbb{I}[a_t \neq a_{t-1}]$ is the indicator function which is equal to 1 if the runway configuration switches from time interval $t-1$ to t and 0 otherwise. The last term models the inertia of the agent in changing the configurations too often to improve the stability of the decisions made by the RCA tool. λ, μ, β, η, and ζ are the weights associated with each term. We perform hyper-parameter tuning (as described in Section V.B) to find

the best combination of weights for the utility function, but depending on the airport and the operational procedures, the weights can be tuned according to the experts knowledge as well. The factors included in Eq. (6) were identified by SMEs as being the most important factors that quantify the efficiency of a runway configuration selection policy. A good policy usually results in higher traffic throughput and lower excessive transit times on the surface, while also reducing number of go-arounds for arriving aircraft.

In the next section, we will first define the performance metrics to validate the RCA tool against baseline methods, then discuss the main findings of the study, and finish with pointing out the challenges of the developed RCA tool as potential directions for future research.

V. Results and Discussion

A. Performance metrics and baseline methods

We compare the performances of different methods according to two metrics: (1) agreement with historical decisions (noted as Agreement in the figures): for this metric, we show how often each method's prediction, $\hat{a}$, agrees with the historical decisions made by the ATCo, $\bar{a}$, i.e., $\frac{\sum_{j=1,\dots,N} \mathbb{I}[\hat{a}_j=\bar{a}_j]}{N}$, where N is the number of data in the testing set, and $\mathbb{I}[\hat{a}_j = \bar{a}_j]$ is the indicator function which is equal to 1 if $\hat{a}_j$ and $\bar{a}_j$ are identical and 0 otherwise. We quantify this as both average agreement across the different configurations as well as the confusion matrix that shows the level of agreement for each configuration separately; and (2) violation of obvious decisions (noted as Violation in the figures): for this metric, we identify landing and take-off scenarios that exceeds the tailwind component of 15 knots as a violation, based on the feedback from the SMEs and estimate what percentage of times each method violates such cases, i.e., $\frac{\sum_{j=1,\dots,N} \mathbb{I}[W_{j,\text{tail}} \geq 15 | \hat{a}_j]}{N}$, where $W_{j,\text{tail}}$ is the tailwind component according to the predicted runway configuration by each model, $\hat{a}_j$, N is the number of data in the testing set, and $\mathbb{I}[W_{j,\text{tail}} \geq 15 \mid \hat{a}_j]$ is the indicator function which is 1 if the tailwind component is equal to or greater than 15 knots, and 0 otherwise. For example, in the case study of CLT, if the wind is blowing strongly (more than 15 knots) from North, if an algorithm suggests the South configuration, it is considered a violation.

In order to quantify the performance of the CQL algorithm developed as the RCA tool in this paper, we compare its performance against three baseline methods: (1) ATCo preference (noted as ATCo in figures): a simple policy that uses the most common decisions made in each state by the ATCo in the past; (2) supervised learning (noted as supervised in figures): based on random forest that learns to mimic the ATCo with least amount of error [26] and serve as a representative of data-driven approaches developed in the literature; and (3) an offline version of a popular Deep Q-Network (DQN) algorithm [11], that would represent a brute use of online RL approaches in the offline setting without any underlying mechanism to alleviate the challenges of offline RL. We obtained ATCo preference policy by selecting the most common action taken by the controllers in the past for each unique state (unique combination of wind

direction, wind speed, hour of the day, and meteorological condition).

B. Implementation details

We obtained and processed 2018 and 2019 calendar year data for both CLT and DEN and divided them randomly into training (60%), validation (20%), and testing (20%) sets. We used the above-mentioned metrics (agreement and violation) to perform hyper-parameter tuning using training and validation sets to identify values of hyper-parameters (reported in Table 2) that result in optimum performance. For each of the hyper-parameters listed in the table, we performed a grid search between the minimum and maximum values reported.

Table 2 The minimum, maximum, and optimal values of hyper-parameters.

Hyper-parameter	Min	Max	CLT	DEN
α in Eq. (5)	5	1000	500	500
Mini-batch size	10	100	20	100
Discount factor γ in Eq. (2)	0.9	0.99	0.9	0.9
Optimizer's learning rate (Adam)	10^{-4}	10^{-3}	10^{-4}	10^{-4}
λ in Eq. (6)	1	10	1	1
μ in Eq. (6)	1	10	5	5
β in Eq. (6)	1	100	10	10
η in Eq. (6)	1	100	10	5
ζ in Eq. (6)	1	10	1	1

Once the final hyper-parameters are selected, we have trained the best model on the combination of the training and validation sets. The results presented in this section are based on the application of the best model out of the hyper-parameter tuning on the testing set (an unseen set of data during training and hyper-parameter tuning). Both Q-network and target Q-network are fully-connected feed-forward neural network with input shape equal to the number of features in the state space, two hidden layers each with 100 neurons and ReLU activation, and output layer with shape equal to the number of actions. This architecture was chosen aligned with most of the literature in offline RL, and we did not perform any hyper-parameter tuning for the architecture of the neural network. All of the training are done with 50,000 episodes each containing one random sample mini batch from the historical data. Each training (with 50,000 episodes, each containing one random mini batch of size 100 transitions) takes about 43 seconds for CLT and 70 seconds for DEN on an Apple MacBook Pro (with M2 Max Chip and 96GB of memory) and the use of the trained

model in a testing is real-time. The pseudo-code for implementation of the CQL algorithm is depicted in the Figure 6.

Conservative Q-learning algorithm

1 Initialize replay memory D

2 Initialize action-value function Q with random weights θ

3 Initialize target action-value function $\hat{Q}$ with random weights $\hat{\theta}$

4 Fix the number of episodes M, mini-batch size N, discount factor γ, and CQL loss weight α

5 **For** episode $e = 1,\ M$ **do**

6 Sample random mini-batch size N of transitions $\left(s_j, a_j, u_j, s_{j+1}\right)_{j=1}^{N}$ from D

7 Set $y_j = u_j + \gamma \max_{a \in A} \hat{Q}_{\hat{\theta}}\left(s_{j+1}, a\right)$

8 Calculate bellman loss $L_{\text{bellman}} = \mathbb{E}_{(s_j, a_j, u_j, s_{j+1}) \sim N}\left[\left(Q_\theta\left(s_j, a_j\right) - y_j\right)^2\right]$

9 Calculate CQL loss $L_{\text{CQL}} = \mathbb{E}_{s_j \sim N}\left[\log \sum_a \exp\left(Q_\theta\left(s_j, a\right)\right) - \mathbb{E}_{a \sim \hat{\pi}\left(a|s_j\right)}\left[Q_\theta\left(s_j, a\right)\right]\right]$

10 Calculate total loss $L = \alpha\, L_{\text{CQL}} + \frac{1}{2} L_{\text{bellman}}$ Eq. (5)

11 Perform a gradient step on L with respect to the Q-network parameters θ

12 Perform a soft update on target network weights $\hat{\theta} = \tau\theta + (1 - \tau)\hat{\theta}$

13 **End For**

Fig. 6 Pseudo-code for implementation of the CQL algorithm.

Lines 1-4 in the pseudo-code are initializing the memory for saving samples of the historical data, weights of the Q-network, as well as the hyper-parameters of the model (e.g., mini-batch size, discount factor, etc). Then, in line 5, we start a loop that for each episode of training, we sample a random mini-batch of the transitions from historical data (line 6) and (1) calculate the target Q-values, explained in Eq. (4) (line 7), (2) calculate the bellman loss, explained in Eq. (3) (line 8), (3) calculate the CQL loss, explained in Eq. (5) (line 9), and (4) calculate the total loss as a weighted average of the bellman loss and the CQL loss (line 10). Finally, we perform a gradient step on the weights of the Q-network (lines 11-12).

C. Performance evaluation

Figure 7 illustrates the achieved performance by each of the methods according to the two performance metrics (i.e., agreement and violation) discussed in Section V.A. Let us first start with the ATCo preference (noted as ATCo in the figure). The average agreement percentage for this policy is 81.6% for CLT and 53.7% for DEN, which shows the great variations in the controller's decision-making. If the controllers always made the same decision in the same specific wind conditions in the historical data, these numbers would be 100%. Lower level of agreement for DEN shows the significantly higher complexity of runway configuration management compared to a simpler airport such as CLT. Supervised learning (noted as supervised in the figure) performs better than just relying on the most common action (i.e., ATCo preference policy) and achieves the average agreement percentage of 88.2% and 60.7% for CLT and DEN, respectively. Although supervised learning achieves great performance, specially at CLT, there are multiple reasons

why it cannot achieve complete agreement with historical decisions. Some of the reasons are: (1) complexity of the decision-making process and variability among historical (human) decisions, and (2) other factors affecting the decision on the runway configuration (such as operational procedures or airline preferences) that are not present in the dataset. The latter reason is more significant under calm wind scenarios.

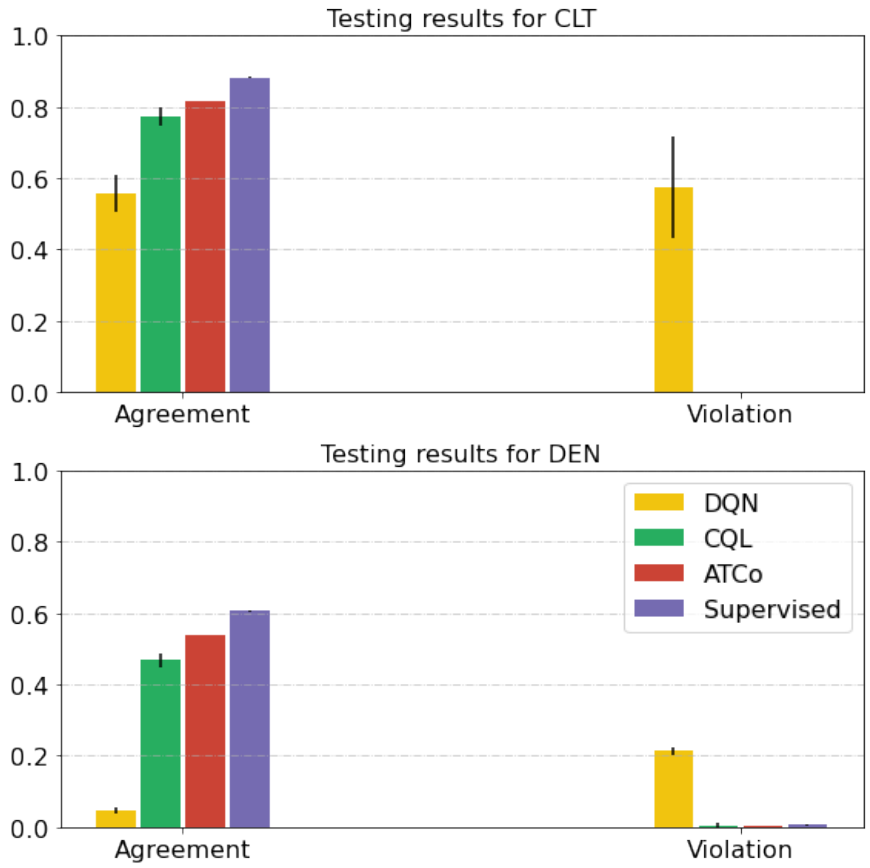

Fig. 7 This figure compares the performance of CQL (green) with the DQN (yellow), ATCo preference (red), and supervised learning (purple) for both CLT and DEN.

Supervised learning is designed to mimic the controller's historical decisions with least amount of error. As a result, it can serve as an upper bound for the agreement metric that any of the methods can achieve. The reason for this is that this approach learns to repeat the decisions of the controllers, both good ones and bad ones. However, the RL-based approaches (i.e., DQN and CQL), rely on the feedback (utility) to estimate which decisions were good and which ones were bad. As a result, they learn to only repeat the good ones and replace the bad decisions with the ones that show a higher utility. As depicted in the figure, CQL performs well compared to the ATCo preference and supervised learning at average agreement percentage of 77.4% and 46.9% for CLT and DEN, respectively. This performance is superior to the offline version of popular DQN approach that only achieves 55.8% and 4.6% for CLT and DEN, respectively. All algorithms except DQN achieve low percentages of violation metric, and CQL achieves lower violation percentages

compared to the supervised learning approach. CQL does not violate any cases for CLT, and only 0.38% of cases for DEN, compared to 0.07% (CLT) and 0.7% (DEN) for the supervised approach. DQN fails at finding a good policy that will not violate obvious decisions for the runway configuration, resulting in a significant number of violations. This is one of the important take-aways that has been emphasized by previous literature in the RL community as well [21, 24], which states that popular and high-performing online RL algorithms are not guaranteed to (and most often fail at) performing well in an offline setting.

Figure 8 shows the confusion matrices for the performance of CQL algorithm on the testing data for both CLT and DEN. The actual configuration (class) represent the historical decisions made by the controllers and is shown on the y-axis, while the predicted configuration by CQL is shown on the x-axis. Numbers on the diagonal elements of the matrix shows correct predictions (agreement between CQL and historical decisions), while the off-diagonal elements depict the number of confusions instances where CQL and historical decisions disagreed upon. In the case of DEN, a majority of the disagreements are among very similar configurations. For example, in the case of S/S configuration, a majority of the times that CQL disagrees with the historical decisions is in prediction of SE/SE (63.5%) instead of S/S, which are very similar configurations. Another example is the confusion of the algorithm between NE/NE and N/NEW configurations. This is partly due to the present class imbalance in the data as noted in Table 1 and can be improved by additional data collection and training of the model with larger amounts of data. Moreover, the use of class balancing techniques such as over-sampling and/or under-sampling can potentially improve performance of the model, however, it is out of the scope of this paper and will be investigated in the future.

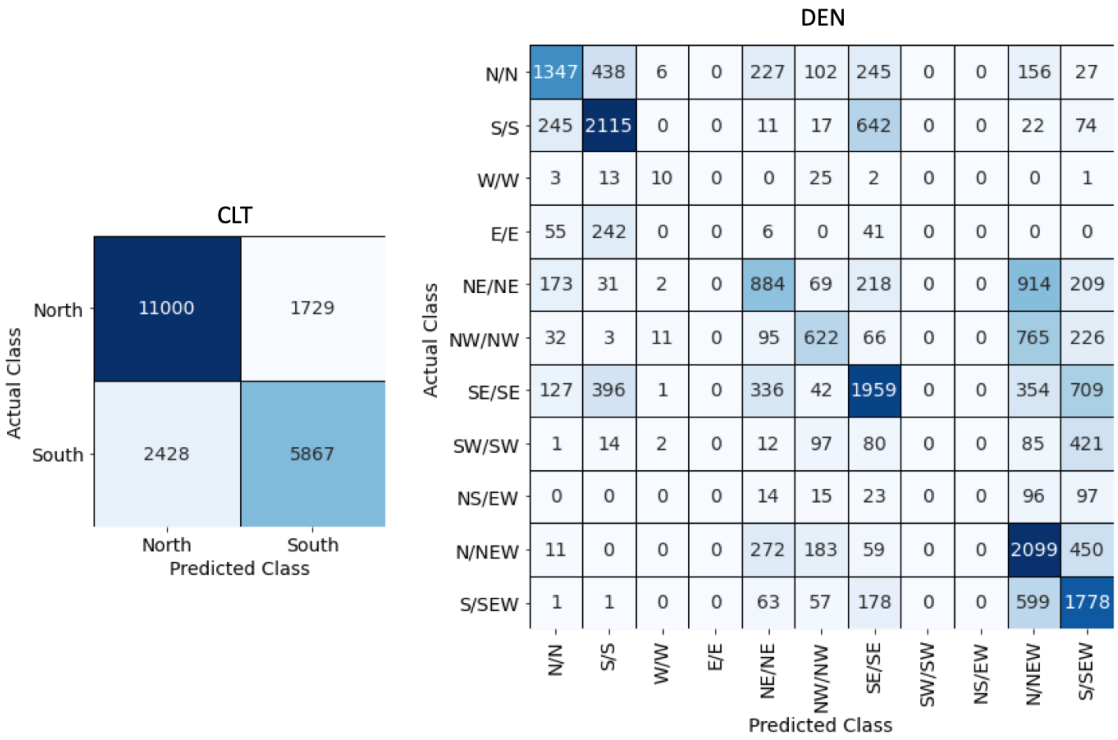

Fig. 8 Confusion matrices for the performance of CQL for CLT (left) and DEN (right).

It should be noted that the disagreements between the ATCo and the RCA tool cannot always be categorized as a mistake by the tool. For example, in investigating some of these disagreements in the case of CLT (the off-diagonal elements in Figure 8), one major pattern was identified by the help of SMEs as the quick wind change scenario. In such cases, the wind direction and/or speed changes abruptly and drastically for a short period of time, and then returns to the original values after few minutes. Figure 9 shows an example of such case for CLT. It shows the runway configuration chosen by the ATCo (Config), the meteorological conditions (MC), cloud ceiling in 1000 ft (Ceiling), visibility in miles (Vis), wind direction based on degrees from North, and wind speed in knots.

As it can be seen from the figure, the wind changes from a calm wind from the North to a strong wind from the South for a short period of time (30 to 45 minutes) before switching back to North. The ATCos do not react to this quick and transient change in the wind conditions and keep the configuration at North due to inefficiency of quick switches in the runway configuration. However, the RCA tool selects South configuration for both time slots of 17:00 and 17:15. Looking more closely at the reasons for the RCA tool switching the configuration showed that in all cases in the training data where the conditions (both wind and meteorological conditions) were similar to those two time slots, the ATCo selected South configuration. As a result, the RCA tool was biased to select the South configuration.

A potential remedy to this problem, and a direction of our future research is to include features based on the forecast of meteorological and wind conditions in the state space of the RCA tool, so that the algorithm is able to identify such

scenarios of quick changes in the wind conditions and make more intuitive decisions aligned with the ATCo.

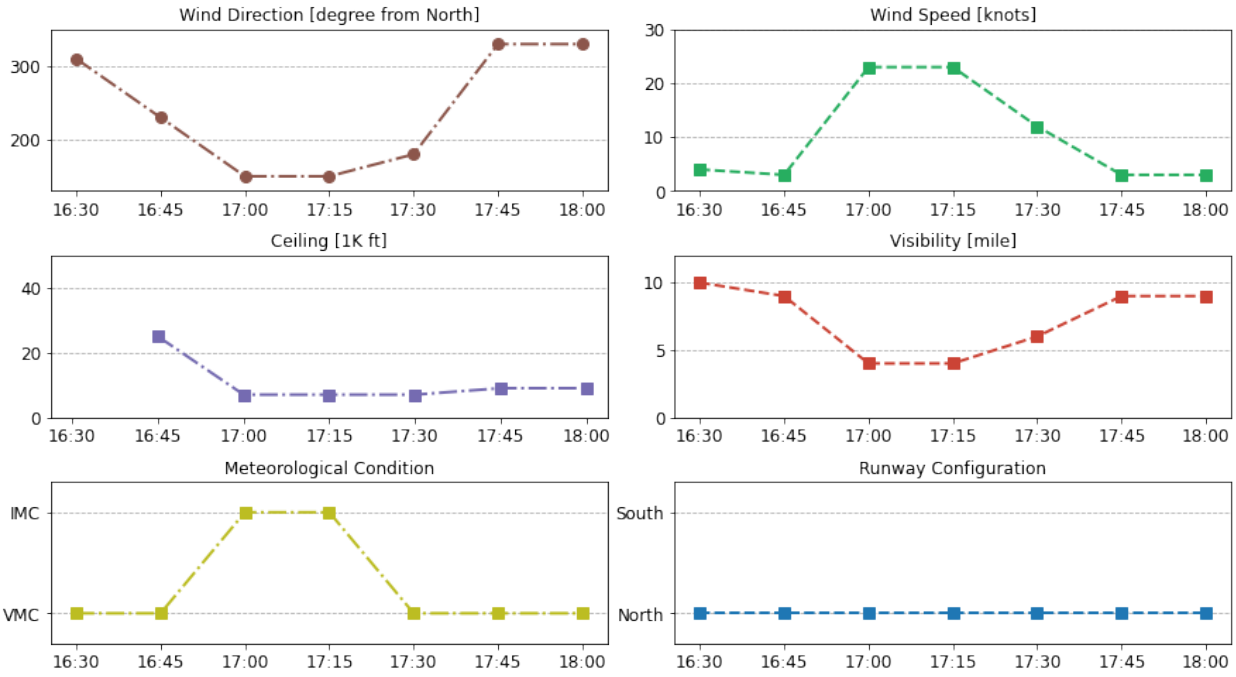

Fig. 9 An example of quick changes in wind conditions at CLT.

D. Policy comparisons

Figure 10 illustrates the learned policy by the CQL algorithm compared to the ATCo preference for CLT (top panel) and DEN (bottom panel). The location of each point on the circle shows the wind direction (degrees from North), while the distance from the center of the circle indicates the wind speed (in knots). Since the hour of the day and meteorological conditions are not specifically structured in the graph, you can see multiple action choices for each unique combination of wind direction and speed. We can clearly see that in the case of CLT, both policies result in North/South configuration when the wind is blowing from North/South respectively. This is expected based on our understanding of favorable wind conditions and the simplicity of the airport surface at CLT. There are also several overlap areas that the North and the South configurations are used interchangeably. Most of these cases correspond to the calm wind conditions (typically less than 10 knots), as well as cases where the wind is blowing from East and/or West, where the choice of configuration is influenced by operational preferences (stability of the configuration, airline preferences, noise abatement procedures, etc.). On the other hand, the visualized policy at DEN reveals greater complexity. Although the trends of major actions used in each unique state is similar among the two policies, their differences are more significant than the simpler case of CLT. For example, we can visually see that most of disagreement between the two policies at DEN are when the wind direction is around South to Southwest, where the ATCo preference is to mostly use the S/S configuration, while CQL alternates between S/S and two other counter-intuitive configurations of NE/NE and N/N.

One reason for this could be that there are less representative historical data with this state/action combination and further data collection could improve the CQL policy.

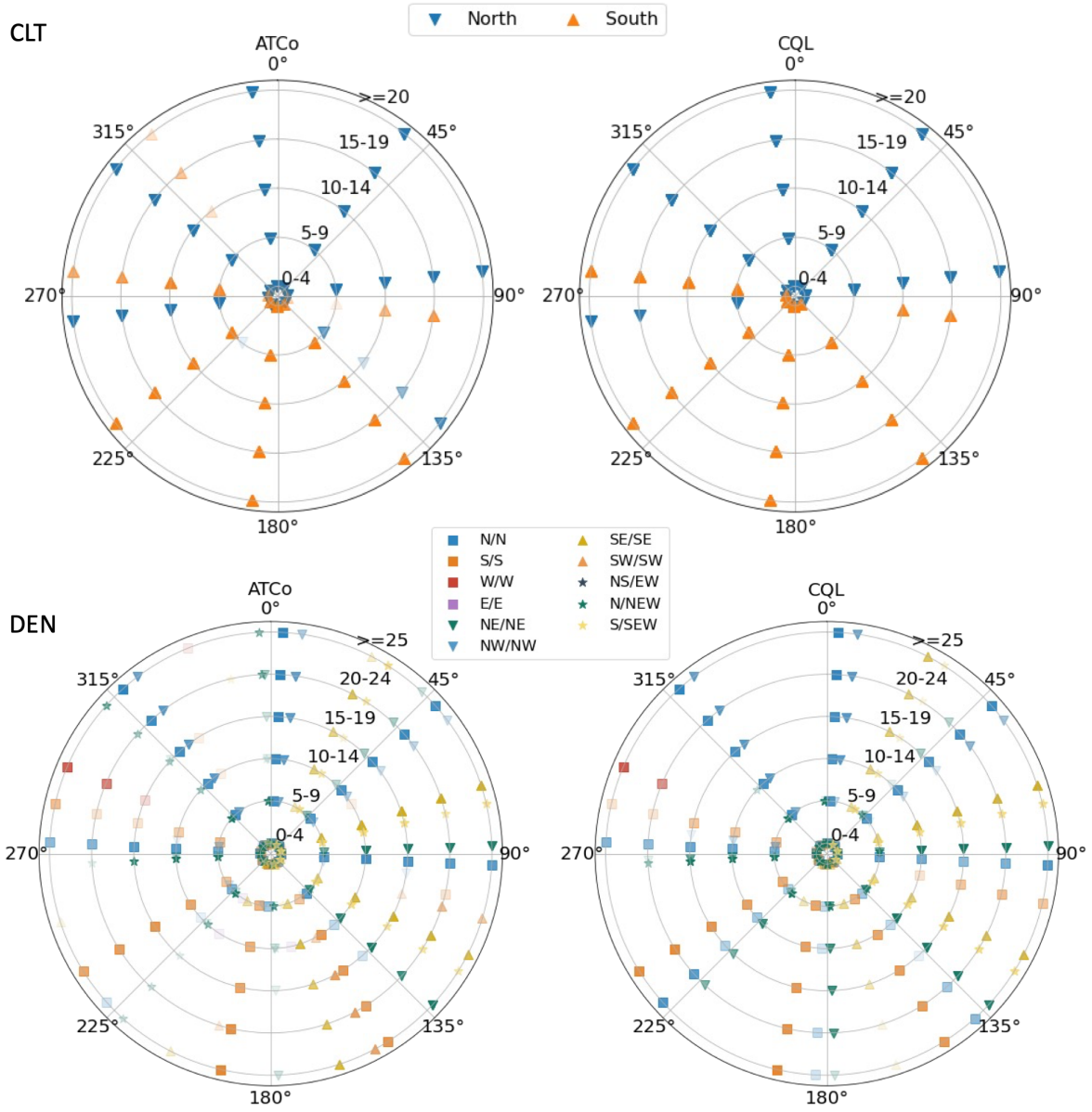

Fig. 10 This figure visually compares the ATCo preferred policy and CQL learnt policy for CLT (top) and DEN (bottom).

E. Forecast scenarios

Figure 11 illustrates two examples of how the CQL approach can be deployed in the operational settings to help ATCo, compared to the baseline methods. In each scenario, the first data instance is representing the current state at DEN (current weather is reported from METAR and current traffic is from ASPM), while remaining instances of data are based on the 12-hour forecast data, where the forecast of weather is reported from NOAA's Localized Aviation MOS

Program (LAMP) and the schedule of arriving and departing traffic is extracted from ASPM. We selected these two scenarios as examples of simple (scenario 1) and complex (scenario 2) operational setting based on the number of times the configurations need to be changed in the next 12-hour period.

Let us first focus on scenario 1: this scenario happens on February 13th, 2019, at 11AM, where the current wind is blowing from the East, with a medium intensity (10-15 knots), and the current meteorological condition is visual, meaning it is clear. The rest of the data instances show how wind direction, wind speed and the meteorological conditions (elements included in the state space) are forecasted for the next 12 hours. On the top panel, the recommendation of each algorithm for the optimal runway configuration to be used is illustrated. As you can see, CQL algorithm agrees with both supervised learning and ATCo preference policy most of the time, with slight disagreements in similar configurations. As depicted in the forecast data, the wind direction is switching from E/NE/N directions to the S/SW direction at around 7PM and picks up intensity around 9PM. We can observe that all the algorithms (except DQN, which completely fails at finding a reasonable policy) suggest the runway configuration to be switched to S/S and other similar configurations of SE/SE and S/SEW. The bottom panel in the same graph shows the expected volume of incoming and outgoing traffic at DEN. This visualization panel can be used in real-time by the controllers to identify the optimal time-window to switch the configurations given the upcoming change in the wind conditions. For example, given that the volume of traffic peaks at around 6-8PM, it might be wise to switch the configuration at a window before 6PM, that would allow a smoother transition of the operations and less amount of delays and/or excessive taxi times. It should also be noted that this visualization panel will be updated as time evolves and more accurate data about the changes in the weather are obtained.

https://vlab.noaa.gov/web/mdl/lamp

Fig. 11 This figure shows how the recommendation of the CQL algorithm can be deployed in the operational setting and visualized for ATCo.

Scenario 2 represents a more complex setting, where the wind conditions change three times in a short span of time. This scenario represents February 5th, 2019, at 6AM at DEN. As can be seen, the current wind is North bound with a very low intensity, but the meteorological condition is instrument flight rule (I), meaning that the visibility and/or cloud ceiling are low. However, the wind is forecasted to switch to SE, S, and then SW direction in the next few hours while picking up intensity at around 10AM. It then again is switching to NW/W direction at round noon

with increasing intensity and then it switches back to SE/S around 5PM. All of algorithms (except DQN that fails) are suggesting the use of N/NE-bound configurations for the first few hours despite the forecasted change in the wind. This is because the intensity of the wind is low for the first few hours and changing the configuration suddenly might result in inefficient operations. Another reason for staying in the same configuration could be due to the non-ideal meteorological conditions. However, as the conditions are changing to visual and the wind intensity is forecasted to increase, all of the approaches suggest a switch to S/SEW configuration around 9-11am time-interval and then another switch to NW-bound configuration around noon, with a final switch to SE-bound configurations around 5PM. Controllers can identify optimal time-windows for the switches based on the projected volume of the traffic to minimize potential delays in such chaotic days.

These two scenarios illustrate just two of many examples of how the RCA tool can be utilized in the operational setting to enhance the decision-making of controllers about runway configurations at airports across the NAS.

F. Limitations of the RCA tool

The RCA tool, i.e., implementation of the CQL algorithm for runway configuration management with SMEs in the loop, addresses a majority of the existing gaps in the literature (detailed in Section II), however, it comes with its own challenges that can be addressed in the future direction of research. One challenge, as mentioned in Section II.A, is the exposure to the Out-Of-Distribution (OOD) data, which is an active area of research in offline RL. An integration of a simple rule-based mechanism (with the help of SMEs) to deal with such scenarios that would assure the safety and validity of the model's prediction is an example remedy to this challenge. It also emphasizes the importance of having SMEs in the loop in any use cases of AI/ML in safety critical systems, such as air traffic management.

Another challenge for all data-driven methods (e.g., model-free control, supervised learning, etc.) is the class imbalance that is usually present in most real-world problems. For example, among the major configurations at DEN (as depicted in Table 1), the top six configurations are used 85% of times and are well-represented in the historical data, while the other five are much less used (although being as important). As a result, a data-driven approach would naturally emphasize the good performance for those configurations and might ignore the minority configurations (for example the RCA tool performs poorly for SW/SW and NS/EW configurations as depicted in Figure 8). This class imbalance might also be different from airport to airport. Although we did not focus on this aspect in this paper, it is an important direction for further improvement of the tool in the future.

Lastly, a challenge for most of the data-driven approaches is to quantify the optimality and the stability of the solution obtained (such as the ones obtained by the RCA tool and/or supervised learning), without access to the underlying model governing the system's dynamics. Further research needs to focus on how to perform verification and validation of such data-driven tools in application to safety critical systems without the availability of a simulation environment.

VI. Conclusions

This paper outlines the details of the development and validation (using real-world historical data) of a runway configuration assistance (RCA) tool. The specific algorithm deployed in this work, conservative Q-learning (CQL), uses a simple mechanism to conservatively estimate the optimal Q-function, resulting in learning a policy that is safe when exposed to out-of-distribution data. This feature addresses one of the fundamental challenges of offline RL, the distributional shift, and allows its application to real-world safety critical systems.

We specifically implement CQL to enhance the decision-making process of air traffic controllers (ATCo) in runway configuration management. Several highly uncertain and variable factors such as wind, meteorological conditions and air traffic volume affect such decision-making processes. We process and fuse several data sources to obtain all relevant information for the RCA tool including FAA's ASPM data, NASA's Sherlock data warehouse (reduced flight data format and events data), and NOAA's LAMP data. Two airports across the National Airspace System (NAS) are selected for comprehensive validation: Charlotte Douglas International Airport (CLT) as a representative airport with simple runway configurations, and Denver International Airport (DEN) as a representative airport with a complex runway configuration setting (as depicted in Figure 5).

We compare performance of the deployed CQL algorithm against several baselines such as ATCo preferred policy, supervised learning (based on Random Forest), and offline implementation of a popular online RL algorithm, Deep Q-Network (DQN). The deployed RCA tool, i.e., CQL algorithm, performs outstanding in both airports achieving 77.4% (CLT) and 46.9% (DEN) agreement with historical decisions (Figure 7). This is lower than supervised learning by only 10.8 percentage-points (pp) (CLT) and 13.8pp (DEN), considering that the main goal of supervised learning is to mimic the historical decisions with least amount of error. ATCo preferred policy only agrees with the historical decisions 81.6% (CLT) and 53.7% (DEN) of the times which shows how variable the ATCo decision-making is. Furthermore, the confusion matrix (Figure 8) for DEN shows that a majority of the disagreements are among similar runway configurations and further data processing and/or data balancing techniques might improve its performance.

Another important finding illustrates the weakness of brute deployment of popular online RL algorithms, such as DQN, in an offline setting. The DQN algorithm fails to learn a good policy for both airports and achieves poor results according to the performance metrics (Figure 7). This emphasizes the importance of regularization techniques used in the CQL algorithm or other state-of-the-art offline RL methods. On the other hand, CQL is able to learn a policy that is safe, resulting in less than 0.07% (CLT) and 0.7% (DEN) violation of obvious decision scenarios (Figure 7), and agrees with the ATCo preferred policy significantly (Figure 10). This is a desired outcome in offline RL, since the ATCo policy was the policy that was used to collect the historical data (also called the behavior policy in the literature), and the goal of offline RL is to learn a policy that does not deviate from the behavior policy significantly (such deviation is called distributional shift).

Lastly, we illustrate how the RCA tool can be used in a real-time operational environment to enhance the decision-

making of the ATCo (Figure 11), by visualizing the prediction of configuration changes based on the weather forecast and the scheduled air traffic. These visualizations will allow the ATCo to identify the optimal time-interval to change the runway configurations that would result in less disruptions to the operations and less transit times on the surface of the airport (less delays). Furthermore, we summarize the existing challenges of the RCA tool proposed in this paper and identify important avenues for further explorations in this area and as examples of future directions of this work.

Acknowledgments

The authors acknowledge the invaluable support and feedback from collaborators and subject matter experts affiliated with the Federal Aviation Administration's (FAA) Office of NextGen (ANG). Moreover, the authors acknowledge the funding of this research from NASA Ames Research Center under contract NNA16BD14C.

References

[1] "National Safety and Operational Criteria for Runway Selection Plans and Noise Abatement Runway Use Programs," *Federal Aviation Administration*, 2014. URL `https://www.faa.gov/air_traffic/flight_info/aeronav/acf/media/Presentations/14-02_DRAFT_FAA_Order_8400_9_Runway_Selection_and_Use_Plan.pdf`.

[2] Precup, R.-E., Roman, R.-C., and Safaei, A., "Data-Driven Model-Free Controllers," *The CRC Press*, 2022.

[3] Provan, C., and Atkins, S., "Tactical airport configuration management," *2011 Integrated Communications, Navigation, and Surveillance Conference Proceedings, Herndon, VA, USA*, 2011. https://doi.org/10.1109/ICNSURV.2011.5935346.

[4] Oseguera-Lohr, R., Phojanamongkolkij, N., Lohr, G., and Fenber, J., "Benefits Assessment for Tactical Runway Configuration Management Tool," *2013 Aviation Technology, Integration, and Operations Conference, Los Angeles, CA*, 2013. https://doi.org/10.2514/6.2013-4395.

[5] Avery, J., and Balakrishnan, H., "Data-Driven Modeling and Prediction of the Process for Selecting Runway Configurations," *Transportation Research Record*, Vol. 2600, 2016. https://doi.org/10.3141/2600-01.

[6] Bertsimas, D., Frankovich, M., and A., O., "Optimal Selection of Airport Runway Configurations," *Operations Research*, Vol. 59, No. 6, 2011, pp. 1407–1419.

[7] Li, L., Clarke, J., Chien, H., and Melconian, T., "A probabilistic decision-making model for runway configuration planning under stochastic wind conditions," *IEEE/AIAA 28th Digital Avionics Systems Conference*, 2009. https://doi.org/10.1109/DASC.2009.5347528.

[8] Jacquillat, A., Odoni, A., and Webster, M., "Dynamic Control of Runway Configurations and of Arrival and Departure Service Rates at JFK Airport Under Stochastic Queue Conditions," *Transportation Science*, Vol. 51, No. 1, 2016, pp. 155–176. https://doi.org/10.1287/trsc.2015.0644.

[9] Badrinath, S., Li, M., and Balakrishnan, H., "Integrated Surface–Airspace Model of Airport Departures," *Journal of Guidance, Control, and Dynamics*, Vol. 42, No. 5, 2019, pp. 1049–1063.

[10] Bai, X., and Menon, P., "Decision Support for Optimal Runway Reconfiguration," *2013 Aviation Technology, Integration, and Operations Conference, Los Angeles, CA*, 2013. https://doi.org/10.2514/6.2013-4397.

[11] Sutton, R. S., and Barto, A. G., "Introduction to Reinforcement Learning," *MIT Press, Cambridge, MA, 2018*, 2018.

[12] Razzaghi, P., Tabrizian, A., Guo, W., Chen, S., Taye, A., Thompson, E., Bregeon, A., Baheri, A., and Wei, P., "A Survey on Reinforcement Learning in Aviation Applications," *ArXiv*, 2022. https://doi.org/10.48550/arXiv.2211.02147.

[13] Browne, C., Powley, E., Whitehouse, D., Lucas, S., Cowling, P., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., and Colton, S., "A Survey of Monte Carlo Tree Search Methods," *IEEE Transactions on Computational Intelligence and AI in Games*, Vol. 4, No. 1, 2012, p. 10.1109/TCIAIG.2012.2186810.

[14] Watkins, C., "Learning from delayed rewards," *PhD Thesis, University of Cambrdige, England.*, 1989. URL https://www.cs.rhul.ac.uk/~chrisw/thesis.html.

[15] Watkins, C., and Dayan, P., "Q-learning," *Machine Learning*, Vol. 8, 1992, pp. 279–292.

[16] Mnih, V., Kavukcuoglu, K., and Silver, D., "Human-level control through deep reinforcement learning," *Nature*, Vol. 518, 2015, pp. 529–533.

[17] Raju, R., Mital, R., Wilson, B., Shetty, K., and Albert, M., "Predicting Runway Configurations and Arrival and Departure Rates at Airports: Comparing the Accuracy of Multiple Machine Learning Models," *2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), San Antonio, TX, USA*, 2021. https://doi.org/10.1109/DASC52595.2021.9594389.

[18] Ahmed, M., Alam, S., and Barlow, M., "A Multi-Layer Artificial Neural Network Approach for Runway Configuration Prediction," *Proceedings of the ICRAT 2018, 8th International Conference on Research in Air Transportation, Castelldefels, Catalonia, Spain*, 2018.

[19] Khater, S., Rebollo, J., and Coupe, W., "A Recursive Multi-step Machine Learning Approach for Airport Configuration Prediction," *AIAA Aviation Forum*, 2021. https://doi.org/10.2514/6.2021-2406.

[20] Churchill, A., Coupe, W., and Jung, Y., "Predicting Arrival and Departure Runway Assignments with Machine Learning," *AIAA Aviation Forum*, 2021, pp. 10.2514/6.2021–2400.

[21] Levine, S., Kumar, A., Tucker, G., and Fu, J., "Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems," *ArXiv*, 2020. URL https://arxiv.org/abs/2005.01643.

[22] Fujimoto, S., Meger, D., and Precup, D., "Off-Policy Deep Reinforcement Learning without Exploration," *in Proceedings of International Conference on Machine Learning (ICML)*, 2019, pp. 2052–2062.

[23] Agarwal, R., Schuurmans, D., and Norouzi, M., "An Optimistic Perspective on Offline Reinforcement Learning," *in Proceedings of International Conference on Machine Learning (ICML)*, 2020, pp. 104–114.

[24] Kumar, A., Zhou, A., Tucker, G., and Levine, S., "Conservative Q-Learning for Offline Reinforcement Learning," *ArXiv*, 2020. URL https://arxiv.org/abs/2006.04779.

[25] Urpi, N., Curi, S., and A., K., "Risk-Averse Offline Reinforcement Learning," *in Proceedings of International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2102.05371.

[26] Puranik, T. G., Memarzadeh, M., and Kalyanam, K. M., "Predicting Airport Runway Configurations for Decision-Support Using Supervised Learning," *2023 IEEE/AIAA 42nd Digital Avionics Systems Conference (DASC)*, 2023. https://doi.org/10.1109/DASC58513.2023.10311186.