Airport Runway Configuration Management with Offline Model-free Reinforcement Learning

Milad Memarzadeh*, Tejas Puranik†, Krishna Kalyanam‡, and Wes Ryan§
NASA Ames Research Center, Moffett Field, CA 94035, USA

Runway configuration management (RCM) deals with the optimal selection of runways to operate on (for arrivals and departures) based on traffic, surface wind speed, wind direction and other environmental variables. RCM is one of the most challenging tasks in air traffic management, as it relies on operational and environmental variables (e.g., weather forecast) that are highly uncertain and complex to model. In this paper, an innovative and automated approach is deployed using *offline model-free reinforcement learning* to provide decision-support for RCM. The proposed technology processes historical data about variables of interest, decisions made regarding RCM, and their subsequent outcome, to identify a policy that would encourage good decisions and avoid the poor ones. The policy search is guided by an appropriately chosen weighted utility function (e.g., based on minimizing delays and go-arounds). Finally, the performance of the proposed tool is validated using Charlotte Douglas International Airport as the case study, which shows that the proposed method is superior to other conventional rule-based approaches.

I. Introduction and Related Work

Runway configuration management (RCM) is a challenging task, and it affects the efficiency of the National Airspace System (NAS) and airport surface operations significantly. Every airport, depending on the surface geometry, capacity, local weather patterns, and noise abatement procedures has multiple configurations for the runway usage for arriving and departing flights. Many factors, including the incoming/outgoing traffic load, wind direction and speed, convective weather, cloud ceiling and other environmental factors, affect the choice of runway configuration at any point in time. In addition, other factors such as safety measures and regulations, noise abatement procedures, capacity of each configuration, and preference of the air traffic controllers (ATCs) can also play a role in determining the optimal configuration. 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. In some circumstances, it can

*Universities Space Research Association, AIAA Member, Corresponding Author: milad.memarzadeh@nasa.gov
†Universities Space Research Association, AIAA Senior Member
‡Aviation Systems Division, NASA Ames Research Center, AIAA Associate Fellow
§NASA Aeronautics Research Institute
This paper will be presented at the AIAA Scitech 2023 Forum in National Harbor, MD, 23–27 January 2023.

also lead to safety concerns, such as an aircraft performing multiple go-arounds before being able to land (e.g., due to excessive tail winds). All these factors make RCM an extremely important yet challenging decision-making process for the ATCs and Front-Line Managers (FLMs).

The ATCs/FLMs set the runway configuration based on relevant information and forecasting available at the time, including weather, traffic, noise abatement procedures, safety bounds, etc. This makes the decision-making process subjective based on the accuracy of the available information, weather forecast models and any potential bias in human decision-making. This manual process can sometimes yield sub-optimal results (e.g., significant delays) if the predicted outcomes are not realized and/or their relative impact is not well understood. This is especially evident when model uncertainty is high, leading to an explosion in the size of possible predicted outcomes, all of which cannot be evaluated by human reasoning alone. On the other hand, an automated approach based on machine intelligence can make use of the abundance of available historical data/decisions and search through many more possible scenarios under uncertainty and make better-informed decisions. Such an automated decision-making process is expected to facilitate better RCM decisions by controllers.

Recently, Artificial Intelligence (AI) and Machine Learning (ML) methods have gained traction as an alternative to enhance the RCM decision-making. One popular approach uses different variants of model-based Reinforcement Learning (RL), such as discrete choice modeling [1], dynamic programming [2], and its combination with queuing theory [3, 4] to model the dynamics of the surface operations at an airport and use simulation to learn a near-optimal policy for the runway configuration selection. This approach is interpretable (due to learning of the dynamics model for both traffic and weather) and is guaranteed to provide a near-optimal solution (provided the model is accurate). However, its performance is dependent on characterizing and building an accurate simulation (i.e., dynamics model and utility function) that is used for learning the policy. Any modeling errors in the simulation that are caused by noise in the system or over-simplification (such as assuming simplified weather dynamics models [1]) can result in a poor performance in the operational setting and create safety concerns. Furthermore, since the dynamics model varies from airport to airport, generalization of this framework to different airports across the nation is challenging.

A potential remedy to address the shortcoming of model-based approaches is to use model-free RL. Online model-free RL approaches such as Monte Carlo Tree Search (MCTS) [5] have been widely used in different domains (e.g., sequential games) for learning a good policy without relying on the availability of the underlying dynamics. However, such online approaches still require an accurate simulation environment to interact with and receive feedback. This interaction is impractical in safety-critical systems such as Air Traffic Management (ATM) because data collection is expensive and accurate simulation environments are either not available or too costly to build. For example, a bad decision in the RCM prediction can result in a safety incident involving landing or take-off of a commercial aircraft. As a result, online model-free RL is not usually applied to safety critical systems (e.g., autonomous driving), and the majority of the literature is focused on other applications such as gaming [6].

Due to the shortcomings mentioned above, recent literature is focused on data-driven approaches to tackle the RCM prediction. One area of work uses supervised machine learning to predict the best choice of runway configuration given all the factors such as weather and future traffic [7, 8]. Such supervised learning methods use a vast amount of historical data to imitate the decision-making made by the ATCs/FLMs in the past with least amount of error. However, they cannot identify and correct mistakes or inefficiencies in the historical decisions since there is no mechanism for the model to receive feedback and correct its policy. Moreover, although these approaches are accurate in predicting the runway configuration, the prediction is not supported by any evidence of better outcomes such as decreased transit times or alleviation of safety concerns. The reason is that in supervised learning, the policy is not based on a utility function and therefore does not predict the runway configuration that optimizes some operational utility metric. Instead, it learns a mapping from the available traffic/weather information to the decisions made by the ATCs with the least amount of error.

In this paper, we propose an offline model-free RL methodology to address the RCM prediction. Offline RL combines reinforcement learning (a key framework for any sequential decision-making problem) with data-driven machine learning. More importantly, it removes the need for an online interaction with the system for data collection, which limits the applicability of online RL methods to safety-critical problems [9]. Although offline RL has been the focus of study in many fields (e.g., autonomous driving, healthcare, robotics), there are fundamental challenges to their deployment that classical RL methods [10] fail to address. One such challenge is the offline nature of the algorithm, which means that there is no exploration involved in learning the policy. Exploration alongside with exploitation are the two pillars of learning a good policy in reinforcement learning, and a vast amount of research is focused on how to properly balance the two. However, in the offline RL framework, there is no room for exploration since the AI agent does not have access to a simulation (or operational) environment to interact with and receive feedback. Another major challenge is distributional shift. This happens when the behavioral policy (an unknown policy that is manifest in the historical data) is different from the policy that is learned by the algorithm using that data. For example, if a state (a specific scenario involving traffic and weather) is not well represented in the historical data, then the policy learned for that state is overly optimistic and most probably wrong. In online RL, it is easy to correct such errors by interacting with the system and receiving feedback. However, in offline RL, there is no such mechanism for error correction. As a result, a majority of the state-of-the-art offline RL algorithms address this issue by either constraining the optimization problem (learning the policy) [11], deploying an ensemble approach [12], or performing conservative updates in the learning scheme to avoid excessive distribution shift from the behavioral policy [13].

To evaluate the effectiveness of offline RL in addressing the RCM prediction challenges, we adopt two recently developed offline RL methods, Conservative Q-Learning (CQL) [13] and Deep Q-Network (DQN) [6], in an offline setting. To comprehensively validate these two approaches, we have developed a "simulated RCM scenario" based on historical data from Charlotte Douglas International Airport (CLT). For this simulated system, we have the complete

state-space model encompassing all the variables of interest and can therefore compute the optimal policy. We compare the performance of our offline RL methods to several policies including the optimal policy, a rule-based policy based on wind direction alone (wind direction is known to be the most dominant factor in determining runway selection), and other simple policies such as sticking to a specific (e.g., preferred or dominant) configuration.

II. Method

The RCM prediction can be viewed as a sequential decision-making problem, where the runway selection choice is the control/decision variable and decisions are made at discrete epochs as the system evolves in a Markovian fashion from one decision epoch to the next subject to the control and other external variables/factors (e.g., traffic, wind conditions). So, it is natural to pose the problem as a Markov Decision Process (MDP) [10].

MDP is defined by a tuple $(S, A, \mathbf{T}, \mathbf{U})$. $s \in S$ is the state of the problem, which contains all the necessary information for the decision-maker (or *agent*). $a \in A$ are the actions available to the agent. $\mathbf{T} : S \times A \rightarrow S$ is a transition function, which defines the dynamics of the state as a result of actions taken by the agent. Lastly, $\mathbf{U} : S \times A \rightarrow \mathbb{R}$ is the utility function that provides feedback to the agent based on the action that she takes in a specific state. Once different components of the MDP are defined, the goal is to find a policy, $\pi : S \rightarrow A$, that maximizes the long-term expected utility, V^{π} (sometimes also called the value function), for the agent managing the system. The value function under a specific policy π is then defined as [14],

$$V^{\pi}(s) = u(s, \pi(s)) + \gamma \sum_{s' \in S} p(s' \mid s, \pi(s)) V^{\pi}(s') \tag{1}$$

Where, $\gamma \in [0, 1)$ is a discount factor, discounting future utilities to their net present value, and $p(s' \mid s, \pi(s))$ is the probability of starting from state s, taking action $\pi(s)$, and ending up in the state s' according to the transition function $\mathbf{T}$. As mentioned above, the goal for the agent is to find a policy that maximizes the value function noted in Eq. (1), i.e., the optimal policy π^*. If an AI agent has a full knowledge of all the components of the MDP including the transition and utility functions, then it can use dynamic programming [10] to find the optimal policy. However, in many real-world problems, such as Air Traffic Management (ATM), this is not the case. As mentioned in the previous section, in the RCM prediction, it is challenging to characterize the dynamics model and utility functions accurately. This is due to the fact that the RCM prediction relies on air traffic load and weather conditions in the airport region, and accurately forecasting these variables and characterizing their uncertain dynamics is hard. As a result, most of the literature that have relied on model-based approaches use simplified models to approximate the dynamics of the traffic and weather, which limits their applicability to the operational setting.

Reinforcement learning (RL) is an approach for sequential decision making when the models describing the dynamics ($\mathbf{T}$) and the utilities ($\mathbf{U}$) are either uncertain (model-based RL) or unknown (model-free RL). Q-learning

[15] is an example of a popular and widely adopted model-free RL that uses a temporal difference control algorithm to learn the state-action value functions (also known as Q-values) directly from the interactions of the agent with the environment. In such a setting, the agent usually interacts with the operational/simulation environment by taking actions, without the knowledge of the underlying dynamics, and learns a policy from the feedback that she receives as a consequence of her actions. Finding a good policy with Q-learning requires a significant number of interactions with the operational/simulation environment, which limits its applicability for safety-critical systems such as ATM. As mentioned before, in the RCM prediction, building an accurate simulation environment is costly and impractical. On the other hand, the AI agent won't be allowed in an operational environment to learn a good policy simply due to the fact that the process of learning a policy involves making mistakes that are not tolerable in safety-critical systems. For example, a bad decision in the RCM prediction can result in a safety incident involving landing or take-off of a commercial aircraft.

To address this challenge, in this paper, we propose an offline model-free RL methodology to address the RCM prediction. Offline RL is an active area of research that focuses on learning a good policy based only on previously collected data, without the need for additional interactions of the agent with the environment [9]. In many real-world scenarios involving safety-critical systems such as air traffic management, there are years of historical operational data available that can be leveraged to build a powerful decision-making or decision-support engine. However, due to a fundamental challenge in offline RL, its application to real-world problems has been limited, especially in the era of classic RL. This fundamental challenge is known as *distributional shift*, i.e., where an agent's policy learned from historical data may significantly differ from the policy that was used to collect the data (also referred to as the behavior policy). Moreover, the policy might be overly optimistic about Out-Of-Distribution (OOD) transitions (scenarios that were not present in the historical data) and result in an unsafe policy in those cases. Most successful offline RL algorithms address this challenge with some type of constraints on the learned policy, or they perform conservative updates to the Q-values to avoid excessive distribution shift [11–13, 16].

In this paper, we deploy a state-of-the-art offline RL algorithm, called Conservative Q-Learning (CQL) [13], to provide a solution to the RCM prediction. CQL uses a simple mechanism to regularize the Q-value estimates for the OOD actions and assigns a lower value to them. This prevents the policy from being optimistic and taking those actions in the areas of the state space that are not well represented in the historical data. We compare the performance of CQL to a few baselines such as (1) the optimal policy (assuming the AI agent has full knowledge of the underlying dynamics and utility functions), (2) the offline version of the popular DQN algorithm [6], and (3) a few rule-based policies. In the next section, we will first discuss the RCM case study and data, and then discuss validation of the algorithms.

III. Results and Discussion

To validate the performance of the offline RL algorithms for the RCM prediction, we first design a *simulated RCM scenario* specifically tailored for CLT. The reason for development of this scenario is to comprehensively validate the

performance of the proposed methodology compared to the optimal performance. In an ideal setting, we can validate these models in an operational/simulation environment, however, as mentioned before, this is not practical. Let us discuss how we designed this simulated problem based on the actual data.

A. Simulated RCM prediction for CLT

Charlotte Douglas International Airport (CLT) has only two major configurations for the arrival and departure traffic, namely "North flow" and "South flow". This simplifying characteristic makes it a good candidate for this initial test of the performance of our proposed RCM method. For this work, we use hourly data from 2019. We obtained and fused the data from three main sources: NASA's Sherlock Data Warehouse (`https://sherlock.opendata.arc.nasa.gov/`), FAA's System Wide Information Management (SWIM) database (`https://www.faa.gov/air_traffic/technology/swim/`), and METeorological Aerodrome Reports (METAR) database (`https://www.aviationweather.gov/metar`). Table 1 shows the features in the RCM data product. The two configurations for CLT and their strong correlation with wind direction and speed is illustrated in the wind rose diagrams in Figure 1. Each data point represents a specific observed wind direction and speed, where the wind speed is illustrated as the distance from the center of the circle in knots. As can be seen, when the wind intensity is greater from the North, "North flow" configuration is preferred (almost always), so that the aircraft would take-off and land into a headwind.

Table 1 Features in the fused RCM data product.

Feature	Type	Source
Scheduled/Actual traffic	Continuous	SWIM
Traffic weight class	Categorical Distribution	Sherlock
Throughput capacity	Continuous	SWIM
Runway configuration	Categorical	SWIM
Transit times	Continuous	SWIM
Meteorological conditions	Categorical	SWIM
Temperature	Continuous	SWIM/METAR
Wind direction & speed	Continuous	SWIM/METAR
Cloud ceiling	Continuous	SWIM/METAR
Visibility	Continuous	SWIM/METAR
Weather conditions	Categorical	SWIM/METAR
Go-arounds	Continuous	Sherlock

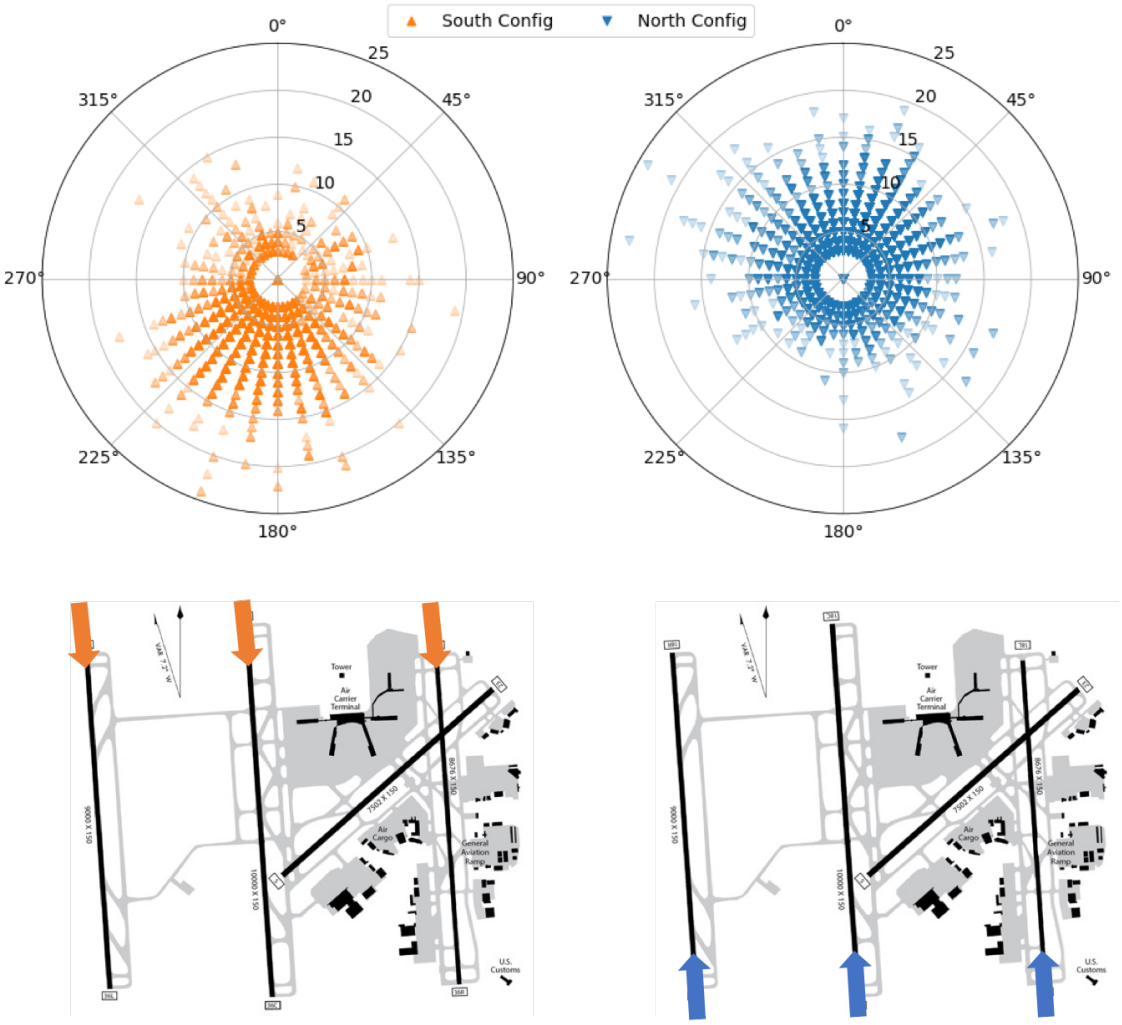

Fig. 1 This figure shows: (top) correlation of selected configuration with wind direction and wind speed at CLT for 2019; and (bottom) the runway configurations for the North and South setting, where the arrows indicate the direction of flight for landing and take-off for that configuration. The runway diagram on the bottom is taken from [17].

Based on discussions with Subject Matter Experts (SMEs) and preliminary data analysis, we define the different components of the MDP model for the simulated RCM prediction. We discretize the state space aligned with the literature in discrete state-action RL problems. The state space is characterized by three variables: wind direction (8 possible states partitioning the total 360° equally, with 0°, 45°, 90°, ..., 315° being the center of each partition), wind speed (4 possible states by binning the wind speed into $[0-5)$, $[5-10)$, $[10-15)$, $[15,\infty)$ intervals), and hour of the day (24 possible states). This defines a 768-dimensional state space. The actions available to the agent are the two possible configurations, i.e., "North" and "South" flow. We have used historical data to estimate the transition probabilities that would define the dynamics of the environment. We have used empirical estimates based on the 2019 data for CLT to estimate the transition probabilities. Finally, the utilities are defined as follows,

$$u_t = \lambda v_t - \mu \tau_t - \beta c_{t,\text{ga}} - \eta c_{t,\text{mga}} - \zeta \mathbb{I}[a_t \neq a_{t-1}] \tag{2}$$

where, v_t is the traffic throughput, τ_t is the average transit time on the surface, $c_{t,\text{ga}}$ is the number of aircraft performing a go-around at time interval $[t-1, t]$, $c_{t,\text{mga}}$ is the number of aircraft performing multiple go-arounds, and $\mathbb{I}[a_t \neq a_{t-1}]$ is the indicator function, which is 1 if the configuration changes from time $t-1$ to t, or 0 otherwise. This last term is designed to mimic the resistance of controllers to switching the configuration too often and avoids the high variance of configuration selection by the AI agent. $\lambda, \mu, \beta, \eta$, and ζ are all hyper-parameters that characterize the weight of each term in the overall utility function. They can be fixed based on the SMEs domain knowledge or can be tuned using a proper cross validation and hyper-parameter tuning. In the simulated example in this paper, we have fixed the weights to the following based on our preliminary analysis: $\lambda = \mu = 5, \beta = \zeta = 10$, and $\eta = 100$. It should be noted that, depending on the airport or the tolerance/preference of the ATCs/FLMs, different hyper-parameters might be preferred to the ones presented here. For example, a higher value of λ would encourage a higher traffic throughput, while a higher value of μ would emphasize the importance of decreasing the transit times.

B. Results and validation

We compare the performance of the proposed offline model-free RL algorithm, CQL, to five other policies: (1) the optimal policy: assuming that the estimated transition and utility functions based on the historical data are the true underlying dynamics of the environment, we can find this policy by maximizing the value function in Eq. (1). This policy would represent the upper bound for all other algorithms here as the best theoretical performance; (2) the offline version of the popular DQN algorithm: this would represent a baseline for CQL, as the offline version of DQN does not explicitly have a mechanism to cope with OOD data; (3) a rule-based policy based on wind direction. This is a simplified policy that uses South configuration if wind is blowing from South, Southeast, or Southwest and North otherwise, due to the fact that the North configuration is a preferred configuration by the ATCs at CLT (visualized in Figure 2); (4) a policy that always selects North configuration; and (5) a policy that always selects South configuration. The training times for both DQN and CQL algorithms are negligible for a year-worth of training data and one airport, and both algorithms can train in near-real time and provide inference in real time.

Figure 2 visualizes the policies obtained by each of these approaches as a function of wind direction and speed. Some scenarios have multiple actions associated with them that capture the variability of the policy as a function of the hour of the day. As it can be seen, the rule-based policy only changes as a function of the wind direction; however, the optimal, DQN, and CQL policies depend on all components of the state space. The DQN and CQL policies are based on training them with 50,000 episodes, each containing a 100 mini-batch of transitions sampled randomly from the replay buffer. Visually, the figure illustrates that the CQL's policy is much closer to the optimal policy compared to the

DQN, however, we will compare their performance in a quantified manner in the next paragraphs.

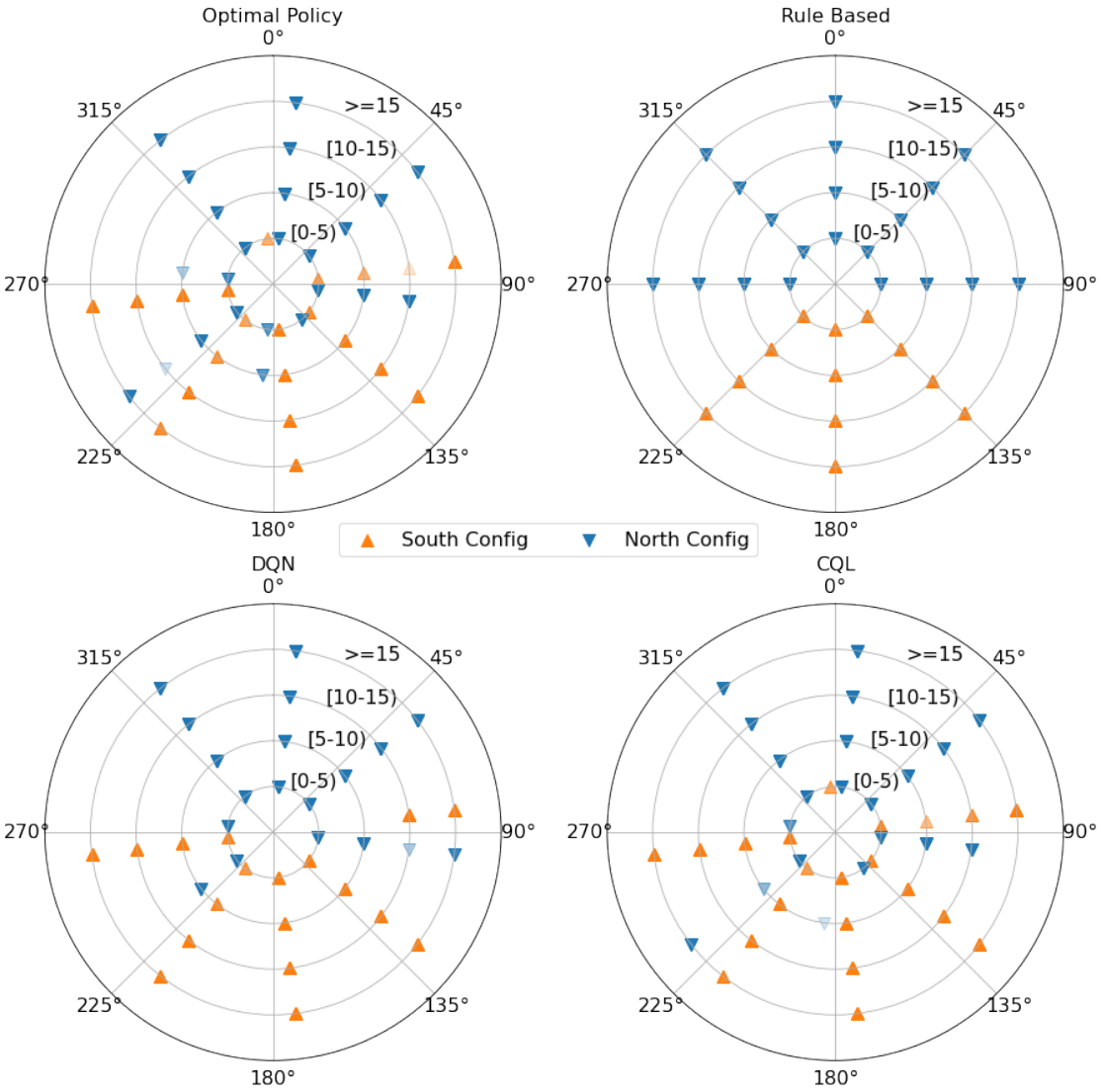

Fig. 2 This figure visualizes policies obtained by each algorithm as a function of wind direction/speed. The variations in the policy for a specific wind direction/speed capture the effect of the hour of the day on the policy.

In order to quantify the performance of the different policies compared to the optimal policy, we simulate each policy in a stochastic forward simulation. Figure 3 shows, on the left, the discounted long-term expected utility (i.e., value function) of managing with each policy calculated according to Eq. (1), and on the right, the cumulative immediate utilities for each policy. We performed 100 independent forward simulations, and the figure shows mean +/- standard deviation of the performance according to these simulations. As it can be seen, CQL, DQN, and rule-based policies perform well and close to the optimal policy, with CQL being the best among them. As expected, overly simplified policies of always picking the South or North configuration performed poorly, with North being the better one due to controllers' preference for this configuration over South at CLT, historically.

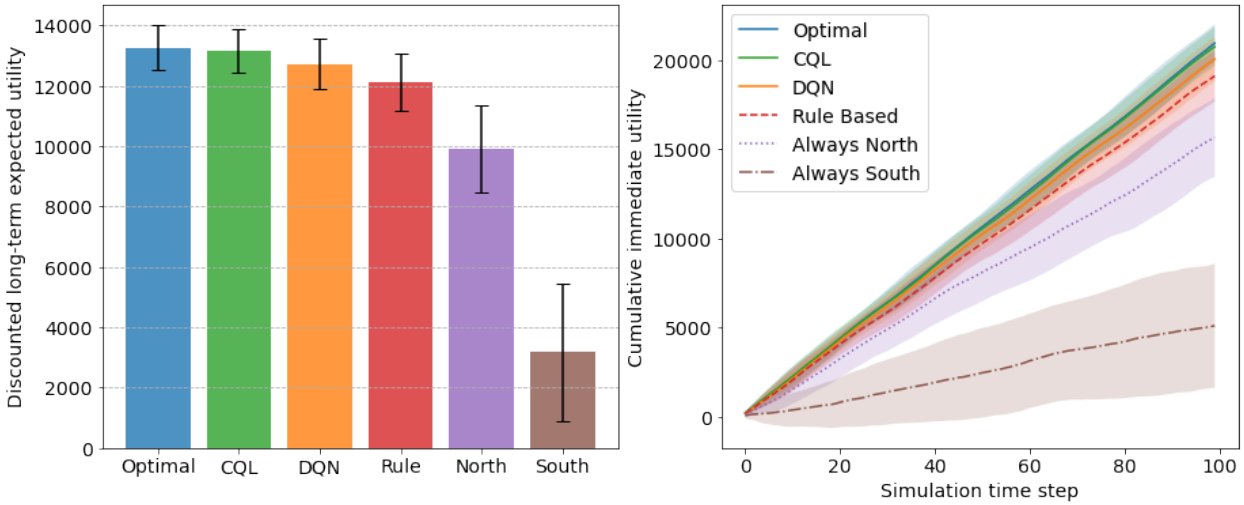

Fig. 3 This figure compares the performance of different policies compared to the optimal policy in terms of (left) discounted long-term expected utility and (right) cumulative immediate utility.

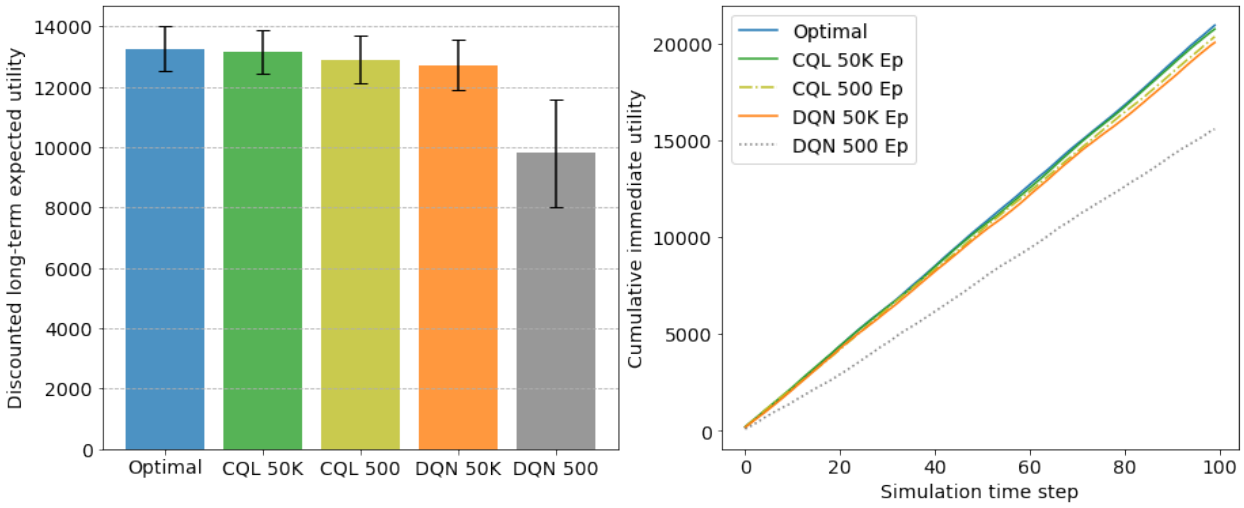

Fig. 4 This figure illustrates the robustness of the policy learned by the DQN and CQL approaches to the size of training data.

In the next step, we intend to test the robustness of the policies learned by these algorithms to the number of training episodes (number of training data needed to reach a good policy). The DQN and CQL results in Figure 3 were obtained by training the policy with 50,000 episodes (each containing 100 random mini-batch samples of transitions). In Figure 4, we compare the performance of DQN and CQL algorithms when we decrease the number of episodes of training from 50,000 to 500. As you can see, performance of CQL stays close to the original CQL policy and the optimal policy, while the performance of the DQN algorithm drops significantly. The reason for this significant drop in the performance of DQN is the fundamental challenge we mentioned about offline RL: the *distributional shift*. When the size of training data is limited, the chance of significant deviation between the behavior policy and the learned policy by the offline RL

algorithm is higher, and there is a higher chance of the AI agent being exposed to the Out-Of-Distribution (OOD) data in the inference. Since DQN has no built-in mechanism to alleviate the effect of OOD data, it performs poorly in those cases. On the other hand, the CQL algorithm, which makes conservative updates to the Q-values and hence has an underlying mechanism to alleviate the effect of OOD data, exhibits greater robustness to the size of training data. As a result, its performance stays close to the optimal policy despite the availability of limited training data.

Lastly, we compare performance of these approaches in two scenarios of inference in the operational setting. In both of these scenarios, we assume that the model has access to an uncertain forecast of the state variables (i.e., wind direction and speed) and a trained policy. Then, we use a forward simulation to estimate the probability that the model will suggest a configuration change at the airport in the next 12 hours. This could be used by air traffic controllers as a decision-support tool to make more informed and timely decisions regarding the runway configuration changes at CLT.

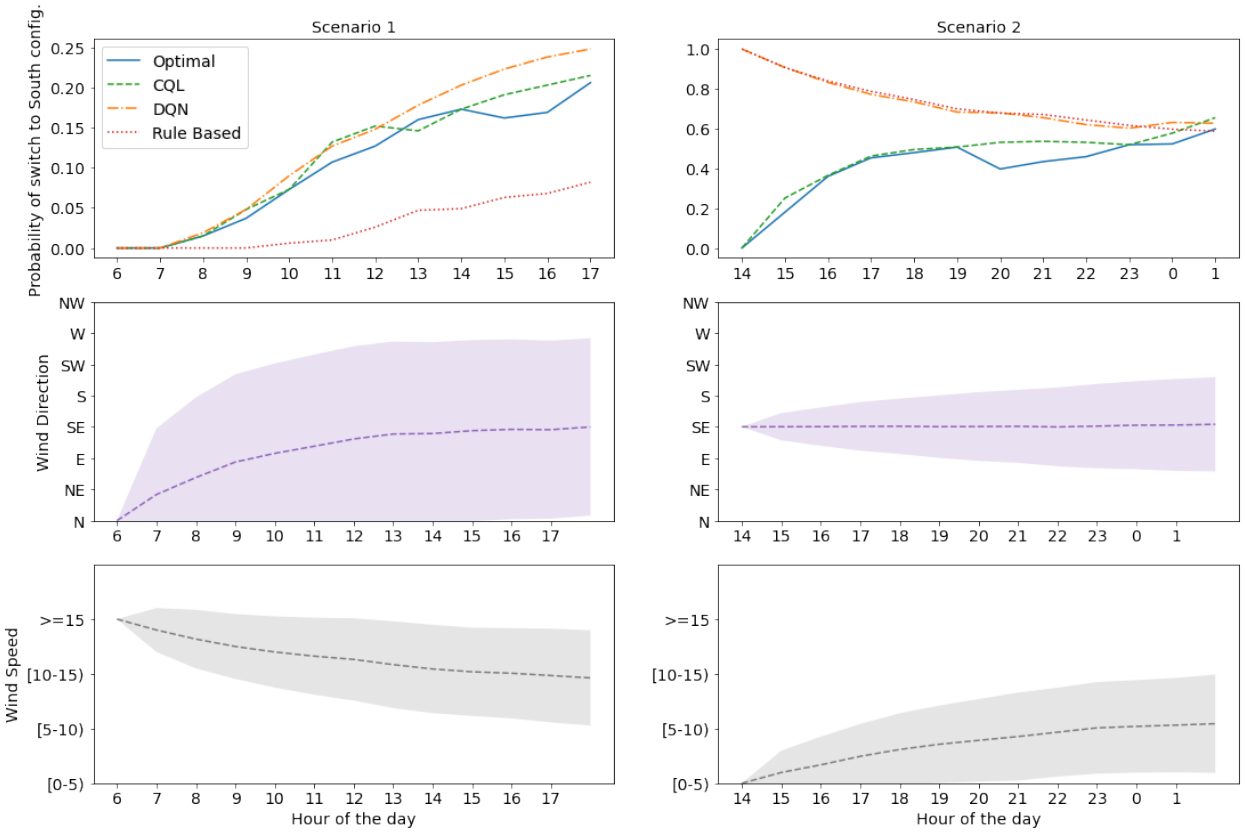

Fig. 5 Two scenarios of deploying the learned policies in an inference mode of operational setting to forecast the next 12 hours.

Figure 5 illustrates two operational scenarios. In scenario 1 (left column), the current time is 6am and the wind is blowing from the North with a high intensity level ($\geq$ 15 knots). The optimal policy dictates that we should use North configuration. Different rows on the figure shows the mean +/- standard deviation of forecast for the next 12 hours for probability of runway configuration switch (top panel), wind direction (middle panel) and wind speed (bottom panel),

respectively, based on 1000 independent simulations. As illustrated, the wind direction is expected to change in the next few hours with considerable uncertainty, and its intensity is expected to decrease. As a result, the probability of the configuration switching to the South increases, accordingly to our model. It should be noted that in the top panel, the closer a policy is to the optimal policy (blue line), the better it has performed. Both the CQL and DQN approaches estimate this probability with a good accuracy and stay close to the optimal policy, however, the rule-based method underestimates this probability significantly.

Scenario 2 represents a more complex setting, where the current time is 2pm in the afternoon and the wind is blowing from the southeast with a low intensity ($[0-5)$ knots). Despite the wind direction being somewhat from the south, we can see that the optimal policy is to still use North configuration. This is not a surprise, as the North configuration is preferred by controllers at CLT. However, as the intensity of the wind grows in the ensuing hours, the probability of switching to the South configuration grows. In this scenario, only the CQL algorithm is able to mimic the optimal policy well. As can be seen, both the DQN and rule-based policies prescribe changing to the South configuration, which we know is not optimal.

These two example scenarios illustrate the superiority of the proposed offline RL approach, i.e., CQL, to the other alternative policies. As can be seen from Figure 5 and previous ones, CQL has the best performance, as it most closely mimics the optimal policy.

IV. Conclusion

In this paper, a state-of-the-art offline model-free reinforcement learning methodology, called conservative Q-learning (CQL), was deployed to address the runway configuration management decision-making. Offline RL combines reinforcement learning (a key framework for any sequential decision-making problem) with data-driven machine learning. More importantly, it removes the need for an online interaction with the operational/simulation environment for data collection, which limits the applicability of online RL methods to safety-critical systems. To comprehensively validate this approach against other AI-based or rule-based methodologies, a simulated RCM scenario was developed based on data obtained from Charlotte Douglas International Airport for the year 2019.

The experiments specifically showed that the CQL approach performed better than a more traditional offline RL approach such as the DQN, as well as simplified rule-based policies, and performed close to the optimal policy (Figure 3). Furthermore, the validation process quantified the robustness of learned policy by each method as a function of available training data and showed that the state-of-the-art CQL algorithm still performed close to the optimal policy when the amount of training data was limited (Figure 4). However, DQN's performance suffered as the amount of training data was decreased. This is due to the fact that DQN does not employ any underlying mechanism to deal with Out-Of-Distribution data.

Lastly, performance of these approaches were compared in two operational settings where the quality of policies

produced were tested, subject to uncertain forecasts (Figure 5). These operational examples illustrated that CQL performed better than the other approaches in mimicking the optimal policy for up to 12 hours in the future, while DQN and rule-based policies performed worse.

Future work: In this paper, performance of the offline model-free RL was validated in a simulated setup where the optimal policy can be calculated. However, in the real-world scenario, the underlying dynamics and utilities are not available and/or characterized fully. As a result, there is a need to develop quantifiable metrics with feedback from subject matter experts to validate performance of the deployed algorithm in an operational setting. A direction of future work is to validate the CQL algorithm for RCM at CLT with the help of SMEs. Another future research direction is to generalize the developed technology to other airports with more complex configuration options involving multiple runways operated in various combinations for arrivals and departures.

Acknowledgement

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.

References

[1] 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.

[2] Li, L., Clarke, J.-P., Chien, H.-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.

[3] Jacquillat, A., Odoni, A. R., and Webster, M. D., "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.

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

[5] Browne, C. B., Powley, E., Whitehouse, D., Lucas, S. M., Cowling, P. I., 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. https://doi.org/10.1109/TCIAIG.2012.2186810.

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

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

[8] Churchill, A., Coupe, W. J., and Jung, Y. C., "Predicting Arrival and Departure Runway Assignments with Machine Learning," *AIAA Aviation Forum*, 2021. https://doi.org/10.2514/6.2021-2400.

[9] 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.

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

[11] 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.

[12] 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.

[13] 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.

[14] Bellman, R. E., "Dynamic Programming," *Princeton University Press, NJ, USA.*, 1957.

[15] Watkins, C., "Learning from delayed rewards," *Cambridge University Press, Cambridge, UK*, 1989.

[16] Dabney, W., Rowland, M., Bellemare, M. G., and Munos, R., "Distributional Reinforcement Learning with Quantile Regression," *The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18)*, 2018, pp. 2892–2901.

[17] Malik, W., Lee, H., and Jung, Y., "Runway Scheduling for Charlotte Douglas International Airport," *16th AIAA Aviation Technology, Integration, and Operations Conference, Washington, DC, USA.*, 2016. https://doi.org/10.2514/6.2016-4073.