A Learning Classifier System Approach to Time-Critical Decision-Making in Dynamic Alternate Airport Selection

Boris Djartov[1,2], Sanaz Mostaghim[2], Anne Papenfuß[1], Matthias Wies[1]
[1]Institute for flight guidance, German Aerospace center (DLR), Braunchweig, Germany
firstname.name@dlr.de
[2] Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany
sanaz.mostaghim@ovgu.de

***Abstract*—The goal of the paper is to address the need for methods to handle time-sensitive, human-centered, multi-criteria decision-making problems. In the current literature, prevalent methods rely on expressing decision-maker/stakeholder preferences through weights, ideal points, and trade-off matrices. However, these conventional approaches prove unsuitable for time-constrained, atypical, and stressful situations, such as emergencies. In such scenarios, where both time and additional factors significantly affect decision-making abilities, the effective utilization of advanced decision-making techniques becomes challenging. Therefore, this paper explores the possibility of how an intelligent agent might be used to provide possible courses of action to human decision-makers/stakeholders. The agent will be put to the test to tackle the dynamic alternate airport selection problem. In emergency and time-critical situations, like an engine fire or a medical emergency, there is often a need to select an alternate airport destination dynamically midflight. During such emergencies, a lot of information must be collected and evaluated by the pilots as a basis for the decision-making process. The pilots need to compare multiple characteristics of the available airports and weigh the pros and cons of each. Given the need for clear and interpretable retroactive analysis in decision-making in general and in the aviation field in particular, the focus was placed on more interpretable and explainable models from the field of AI. Due to this, the Learning Classifier System (LCS) is to be the primary model explored. The LCS is trained on a custom dataset composed of various decision-making scenarios. The approach shows promising results and appears to merit further investigation.**

***Index Terms*—multi-criteria decision-making, multi-attribute decision-making, learning classifier system**

I. Introduction

Making decisions involving multiple criteria is a challenging task even under normal circumstances. The addition of dynamic and time pressure elements can further complicate an already challenging endeavor. One such endeavour is alternate airport selection during a flight. Given the dynamic nature of flying it may occur that pilots change their initial plans in regards to their landing destination. This would often entail comparing different airports in terms of multiple characteristics such as: available landing distance, weather conditions, runway condition and other factors. Although structured schemes such as FORDEC (Facts, Options, Risks, Decision, Execution, Control) or TDODAR (Time, Diagnosis, Options, Decide, Assign Task, Review) are used by pilots [2] [3], a desire for additional decision-support has been expressed by pilots. Hence, a decision support system capable of analysing all the relevant information and suggesting a course of action seems like a prudent addition in the cockpit [1].

At the DLR (Deutsches Zentrum für Luft- und Raumfahrt e.V., engl. German Aerospace Center), the Intelligent Pilot Advisory System (IPAS) [4] is under development to explore and gain insights into how an AI system might be integrated into the cockpit to assist with pilot decision-making. The IPAS is designed to process all relevant data and provide suggested courses of action to pilots. This paper focuses on envisioning the AI model for such a system. Currently, the goal for the AI model is to offer pilots a valid suggestion regarding which airport to select in the context of the alternate selection problem. The paper's emphasis is not on the system as a whole but rather on the development of an AI model capable of addressing the underlying problem, which is essentially a multi-criteria decision-making challenge. Due to the nature of the problem and the conservative "safety first" approach inherent in the aviation field, emphasis was placed on a more interpretable model. Hence, the Learning Classifier System (LCS) was selected. This choice also aligns with the European Union Aviation Safety Agency (EASA) published roadmap for AI development and its concept of "Trustworthy AI." [5].

The structure of this paper is as follows: The next section provides a brief overview of the theoretical background relevant to the literature discussed in this paper. The following section covers the creation of the dataset. This is followed by a showcase of the employed Learning Classifier System (LCS). The paper concludes with the analysis of experimental results and the drawn conclusions, along with suggestions for possible future research avenues.

II. Theoretical background

The formulation of the Decision Advisory for Airport Selection (DAAS) problem as a multi-criteria decision-making problem was initially proposed in [1]. Multi-criteria decision-making (MCDM) is a systematic process utilized to evaluate and select among competing alternatives based on multiple,

often conflicting, criteria or factors [15]. MCDM is broadly divided further into multi-objective decision-making (MODM) and multi-attribute decision-making (MADM). In MADM, the number of alternatives is constrained, while in MODM, the number of alternatives is not predetermined, and alternatives are restricted by a set of optimal objective constraints [7].

The DAAS problem in [1] and here is construed as a MADM problem, as the number of alternatives (i.e., airports to choose from) is predetermined. Typically, a solution for a MADM problem involves the following steps:

1) Identify Decision Criteria: Determine the relevant criteria or attributes that will be used to evaluate the alternatives. These criteria should be essential in making the decision and reflect the objectives or goals of the problem.
2) Evaluate Alternatives: Assess each alternative against the identified decision criteria. Assign appropriate scores or weights to each alternative for every criterion based on its performance or suitability.
3) Weighting and Aggregation: Assign relative weights to the criteria to represent their importance in the decision-making process. Then, aggregate the scores of each alternative across all criteria, considering their assigned weights, to obtain an overall score or ranking for each alternative.
4) Select the Best Alternative: Analyze the results of the evaluation and ranking to identify the most favorable alternative based on the overall scores. The alternative with the highest overall score is usually chosen as the best solution [14].

Based on [15] some of the most common and noteworthy methods to tackle MADM problems are:

- Analytical Hierarchy Process (AHP), which decomposes a decision making problem into a system of hierarchies of objectives, attributes (or criteria), and alternatives [16], [17].
- Technique of Order Preference by Similarity of Ideal Solution (TOPSIS), which selects an alternative based on minimizing the Euclidean distance from most ideal and worst point [18].
- *VIšekriterijumsko KOmpromisno Rangiranje* [Multi-criteria compromise ranking] (VIKOR), which introduces the multi-criteria ranking index based on the particular measure of "closeness" to the "ideal" solution [19], [20]
- Preference Ranking Organization METHods for Enrichment Evaluation (PROMETHEE), a method which is based on mutual comparison of each alternative pair with respect to each of the selected criteria [21].
- Elimination Et Choice Translating REality (ELECTRE), a method consisting of multiple variations for a variety of MADM problems [22].

The aforementioned methods rely on several fairly complicated mechanisms to either gather the decision-makers' preferences or systematize and rank the alternatives. These mechanisms, although useful, cannot be reliably employed in time-critical emergency situations. Therefore in order to overcome this problem the idea to employ an intelligent agent that would suggest a option was proposed. As proposed in [1] the intelligent agent would learn from a multitude of possible scenarios so that it may be equipped to deal with the dynamic nature of flying and DAAS problem.

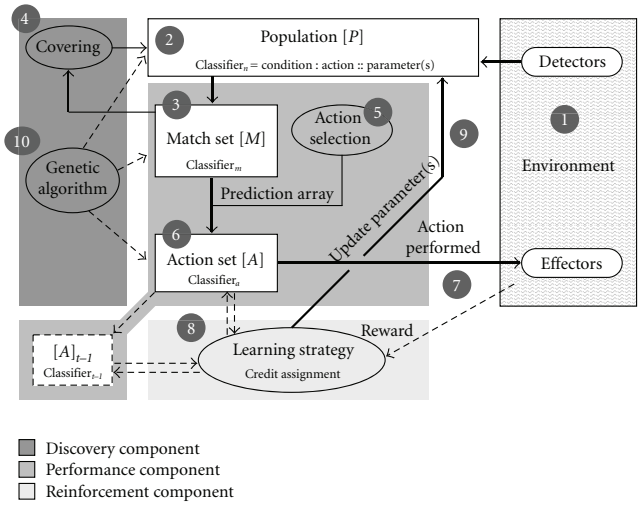

Fig. 1. Visual representation of a LCS, taken from [26]

The model/intelligent agent that was selected was the LCS. An LCS consists of a set of rules known as the population of classifiers. The primary goal is for these classifiers to collectively emulate an intelligent decision maker. This emulation is achieved through two key biological metaphors: evolution and learning. The genetic algorithm and a problem-specific learning mechanism embody these concepts, guiding the system toward better rules. Both mechanisms rely on the system's "environment," which, in the context of LCS literature, refers to the source of input data for the algorithm. The information received from the environment depends on the problem being addressed. For instance, in a maze navigation scenario for a robot, sensory information describing the physical environment serves as input. In a medical diagnosis context, the environment is a training set with preclassified subjects characterized by multiple attributes like genetic polymorphisms. Interacting with the environment, LCSs receive feedback in the form of numerical rewards, driving the learning process. While various LCS algorithm implementations exist, a basic framework includes a finite population of classifiers representing system knowledge, a performance component regulating interaction with the environment, a reinforcement component distributing rewards to classifiers, and a discovery component improving rules through different operators. These components serve as the foundation for numerous variations in LCS algorithms. Figure 1 illustrates how specific mechanisms of LCS interact within the context of these major components [25] [26].

Aside from the interpretability and explainability of the model, LCS has also been extensively used for problems

characterized by both epistasis and heterogeneity, such as the multiplexer problem [23] [25]. Given that these characteristics are also present in the DAAS problem and that the DAAS problem was used as inspiration for the construction of a multi-objective multiplexer benchmark problem [24], using an LCS seems like a suitable approach.

III. Dataset creation

The dataset utilized to train the model will be a refined version of the one used in [1]. It encompasses various specific scenarios, each serving as a depiction of instances where pilots encounter specific predicaments requiring immediate decision-making. Each unique scenario is denoted by an $n \times m$ matrix S_i, where n corresponds to a list of the largest commercial airports currently operating within Germany. Twelve airports, strategically dispersed across the country, were selected for inclusion in the dataset. One of these airports is designated as the planned airport and is consequently inaccessible [28]. The parameter m signifies the characteristics of the airports at the precise moment when a decision must be made. Each airport is characterised by the following attributes:

- Longest runway length available $[m]$
- Crosswind speed at the location of the airport $[kn]$ 2
- Tailwind speed at the location of the airport $[kn]$ 2
- Fuel needed to reach the airport $[kg]$
- Status of the runway, expressed as a value from 0 to 6 II
- Distance from current aircraft position to planned destination $[nm]$
- Distance from current aircraft position to appropriate alternate airport $[nm]$

The longest runway length for each airport was obtained from the global navigation database within the BlueSky, ATC Simulator Project [29]. The wind speed, direction and runway status for each of the twelve airports within the scenario were generated using Latin-hypercube sampling [30], where the ranges for all of the variables were:

- Wind speed, 5 to 45 $[kn]$
- Wind direction 0 to 359 $[°]$
- Runway status from 0 to 6, see Table II

Opting for higher wind speeds was preferred to introduce complexity into the decision-making process. The wind direction departs from conventional aviation guidelines and is structured on a relative system, as illustrated in Figure 2. Furthermore, the wind direction and speed that were sampled were transformed into crosswind and tailwind components. This was done in order to both simplify the learning process but is also one of the primary pieces of information relevant to pilots that concerns the wind at an airport. The breakdown of a wind vector into crosswind and tailwind components is also depicted in Figure 2. Additionally, the runway condition was chosen uniformly amongst the possible runway conditions depicted in table II. The proclivity for wet conditions significantly complicate aircraft landings, as runway condition heavily influences the braking performance of an aircraft.

To determine the distance from the planned airport and identify all potential alternate airports, the aircraft's position had to be established for each scenario. To closely emulate realistic situations encountered by pilots, flight plans were created. These flight plans outline how the aircraft should approach its destination, detailing altitudes and the sequence of waypoints to follow. In aviation, a waypoint refers to a specific geographical location used to define a route for area navigation or an aircraft's flight path. A visual representation of a flight plan illustrating the route from Berlin to Frankfurt airport is provided in Figure 3.

TABLE I
Airport Estimated Runway Condition Assessment

Code	Runway Condition Description	Reported Breaking Action
6	Dry	Dry
5	Wet • Water up to 1/8" (3mm) • Damp 1/8" (3mm) or less of • Slush • Dry Snow • Wet Snow	Good
4	Frost Compacted Snow (OAT at or below -15ºC)	Good to Medium
3	Slippery when wet More than 1/8"(3mm) • Dry Snow – max 5" (130mm) • Wet Snow – max 1 1/8" (30mm) Compacted Snow (OAT above -15ºC)	Medium
2	More than 1/8" (3mm) • Water – max 1/2" (12.7mm) • Slush – max 1/2" (12.7mm)	Medium To Poor
1	Ice (cold & dry)	Poor
0	• Wet Ice • Water on top of Compacted Snow • Dry Snow or Wet Snow over Ice	Nil

The flight plans were sourced from an online flight planner primarily designed to assist amateur pilots in crafting flight plans for simulation software [31]. A total of 22 flight plans were utilized in constructing the dataset, representing flights between the previously identified twelve airports. To identify the most common destinations within Germany for each airport, FlightRadar24 [32] was employed. Subsequently, flight plans were generated based on the most likely routes between these airports. For each flight plan, a waypoint was chosen in a pseudo-randomized manner to represent the aircraft's position at decision points. Only waypoints aligning with the cruise phase of a flight, as illustrated in Figure 4, were taken into account.

The fuel needed was calculated by simulating the flights, starting from take-off all the way to landing at each of the possible available options. The custom-made simulation only encompasses the Airbus A320 aircraft and makes use of information from the Airbus A318/A319/A320/A321 Flight Crew Operating Manual [27], the Eurocontrol A320 performance database [35], and consultation with a pilot employed at the DLR.

It is important to highlight that the dataset was crafted to encompass a broad spectrum of scenario difficulties, with a bias toward the higher end of the difficulty scale. Difficulty, in this context, encompasses both the severity of the situation

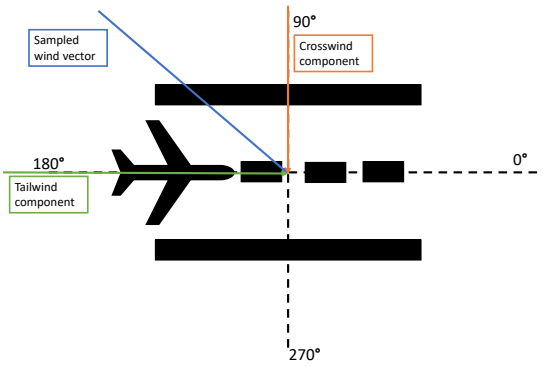

Fig. 2. Relative wind direction representation

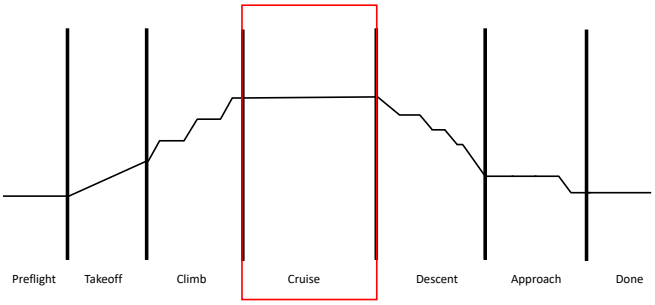

Fig. 4. Phases of a flight, adpted from [33]

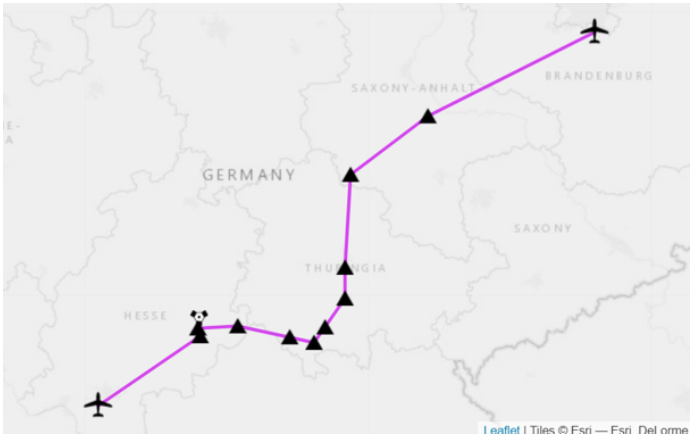

Fig. 3. Example flight plan Berlin airport to Frankfurt airport, taken from [31]

in terms of physical danger (e.g., high winds and wet runways) and the prevalence of airports characterized by these extreme weather phenomena. This intentional skew towards higher difficulty levels creates challenging scenarios where most, if not all, options are unfavorable. This design ensures that the system can also tackle seldom encountered edge cases when the pilots would need the most support.

In essence, the scenarios capture crucial decision-making moments. Each scenario corresponds to one of twelve airports, with the originally planned destination being unavailable. These airports are characterized by specific attributes such as runway length, wind direction, wind speed, runway condition, and distance from the aircraft. By sampling 100 combinations of wind speed, wind direction, and runway condition, and considering 22 flight plans, a dataset of 2200 scenarios is created. The normalized dataset can be found here: https://github.com/borisdjartov/Dataset.

IV. LCS STRUCTURE

The DAAS problem can be redefined as a single step multi-class classification problem, similar to that of the real-multiplexer problem. The employed LCS uses hyperrectangle conditions, linear approximation, and integer actions. The majority of the structure is based on [36] and makes use of the enhanced Michigan-style learning classifier - often termed XCSF [37] [38]. The LCS is comprised of:

1) The condition C which is a hyperellipsoidal condition and is formally represented as:
$$C = (\vec{m}, \Sigma) = ((m_1, m_2, ..., m_n)^T, (\sigma_1, \sigma_2, ..., \sigma_n)), \quad (1)$$
where $\vec{m}$ specifies the center of the ellipsoid, the notation T denotes the transpose of a (column) vector or matrix, and Σ defines the fully weighted Euclidean distance metric of the ellipsoid, also termed Mahalanobis distance [39]. This transformation matrix determines the stretch of the ellipsoid and the rotation in the n-dimensional problem space.
2) The possible action set A, that is, $A \in A$ where $A = \{a1, ..., a_m\}$ represents the set of all possible actions.
3) The linear function prediction R which is specified by the weight vector
$$R = \vec{w} = (w_0, w_1, ..., w_n)^T, \quad (2)$$
where w_o is the offset weight
4) The prediction error ε which estimates the mean absolute deviation of the reward predictions.
5) F which is the fitness of the classifier.

The values are iteratively modified and evolved by the means of reinforcement learning (RL) techniques and a GA. The LCS is initialized with an empty population. Each learning iteration t, the LCS receives a problem instance x_t with the corresponding actual function value y_t. The LCS forms a match set $[M]$ of all classifiers whose conditions are active. Classifier activity is determined by
$$\text{cl.ac} = \exp\left(-\frac{(\mathbf{x}-\mathbf{m})^T\Sigma^{-1}(\mathbf{x}-\mathbf{m})}{2}\right),$$
which essentially determines the distance from the current input and then applies the Gaussian kernel on the distance. A classifier is active (that is, it matches) if its current activity is bellow some threshold θ_m. Initial classifiers are generated by a covering mechanism that creates intervals controlled by

parameter $r0$. The covering mechanism is triggered until there are classifiers in $[M]$ such that all of the possible actions are represented in $[M]$. The covering mechanism generates matching condition given a problem instance $\vec{x}$ and adds them to the population $[P]$. The center of the hyperellipsoid ($\mathbf{m}$) is set to the current input ($\mathbf{x}$). Only the diagonal entries in the transformation matrix (Σ) are initialized to the squared inverse of the uniformly randomly chosen number between zero and the parameter r_0. All other matrix entries are set to zero. In this way, covering creates axis-parallel hyperellipsoidal condition parts. Afterwards $[M]$ can be used to form a prediction about the class. The LCS forms a prediction array $P(\mathcal{A})$ of expected payoff values for all possible actions. The prediction is then determined by the inner product $(\vec{x} - \vec{m})^T * \vec{w}$, where vectors $\vec{x}$ and $\vec{m}$ are enhanced by a trailing one and zero, respectively. Afterwards, the environment provides rewards in the form of a scalar value of the actual value of the expected function. The learning happens iterative from the problem instances. Given that the problem set consists of single step instances after every instance, using the feedback from the environment it updates the reward prediction, prediction error,fitness estimate and set size estimate of all classifiers in $[M]$. The weight vector is updates using the delta update rule:

$$w_i \leftarrow w_i + \eta \left(y_t - (\vec{x}^* - \vec{m}^*)^T \vec{w}\right) (x_i^* - m_i^*),$$

where η denotes the learning rate. Afterwards the prediction error is updated according to:

$$\varepsilon \leftarrow \varepsilon + \eta \left(|r - P(A)| - \varepsilon\right)$$

where r is the scalar reward and $P(A)$ is the prediction for the specific action constant. Fitness is updated by:

$$\kappa = \begin{cases} 1 & \text{if } \varepsilon_0 \leq \alpha \left| \frac{\varepsilon}{\varepsilon_0} - \nu \right| \\ 0 & \text{otherwise} \end{cases}$$

$$\kappa' = \frac{\kappa \cdot num}{\sum_{\text{cl} \in [M]} cl.k \cdot cl.num} \tag{3}$$

$$F \leftarrow F + \eta(\kappa' - F) \tag{4}$$

Finally the set size estimate is updated:

$$a_j \leftarrow a_j + \eta(|[M]| - a_j) \tag{5}$$

The LCS employs a steady-state, niche genetic algorithm (GA) for the evolution of rules. The GA employs set-size relative tournament selection [40] based on the fitness estimates of classifiers to choose two parental classifiers from the current match set $[M]$. Subsequently, two offspring are produced through crossover and mutation operations. Uniform crossover is implemented, wherein corresponding values in the two classifier conditions are exchanged with a probability of 0.5. Mutation, with a probability of μ, modifies each entry in the condition part, randomly shifting the center of the hyperellipsoid within its current interval. During mutation, a matrix entry undergoes a maximal decrease (increase) of the value by 50%. If the value is initially set to zero, it is initialized to a randomly selected value, akin to the covering process for diagonal matrix entries, while considering the parameter μ_0. The implemented LCS also incorporates a deletion operator in order to maintain the population size as specified and provide additional pressure to better the classifiers. The process begins by calculating the average fitness of the classifiers in the population. Next, for each classifier in the population, a deletion vote is calculated. The deletion vote is initially set to the classifier's set size estimate. If the classifier's number of matches exceeds a certain threshold θ_{del} and its fitness is less (for maximization problems) or greater (for minimization problems) than a fraction δ of the average fitness, the deletion vote is increased. This is done by multiplying the deletion vote by the classifier's number of matches and the ratio of the fraction of the average fitness to the classifier's fitness. This means that classifiers with a high number of matches and low fitness (for maximization problems) or high fitness (for minimization problems) are more likely to be selected for deletion. The the deletion votes are normalized by dividing each vote by the sum of all votes. This creates a probability distribution over the classifiers, where classifiers with higher deletion votes have a higher probability of being selected for deletion. A classifier is then randomly selected for deletion according to this probability distribution, and removed from the population. This process is repeated until the population size is reduced to the maximum allowed size.

V. Experimental design and results analysis

A. Reward Function

In order to train the LCS described previously a reward function is needed that would provide the scalar reward r. In this case the DAAS problem has been transformed into a multi-class reinforcement learning classification problem. The reward for which is defined as:

$$R(\vec{x}) = \frac{1}{\sum_{i=1}^{n} \mathbf{b}_i * c + \sum_{j=1}^{m} \mathbf{u}_j} \tag{6}$$

where $\vec{x}$ is the input vector, while b_i and u_j represent the ith and jth elements of the respective vectors. The vector $\vec{b}$ represents all the bounded attributes, that is it represents the constraints. Should certain constraints be broken, the difference between the allowed value and the current value are noted in $\vec{b}$, the cardinality of $\vec{b}$ is thus dependent on the number of constraint violations. Additionally, the factor $c = 1 + |\vec{b}|$ is equal to the number of constraint violations or cardinality of $\vec{b}$ + 1. The vector $\vec{u}$ represents attributes with out constraints and that should be optimized as much as possible. The reward function was generated to overcome some of the limitations proposed in the initial paper [1]. The revised reward function continues to prioritize the incorporation of considerations for both physical risk and economic implications. The risk aspect is effectively represented using the bound attributes, while the unbound attributes aim to minimize financial losses. This updated method for calculating the reward provides greater flexibility for different airlines or groups to choose the factors they wish to consider. Moreover, the reward function is designed to cover a wide range of

possible scenarios. For instance, in a situation where most or all airports are deemed unsuitable for the aircraft from a safety perspective, the bound term will dominate the equation for most options. This compels the Learning Classifier System (LCS) to minimize the impact of physical risk as much as possible. The inclusion of the factor c serves two purposes. Firstly, it enhances the importance of constraint violations, particularly in terms of safety considerations. Secondly, it aligns with the "Swiss Cheese" model of accidents [41], which states that while a large deviation in one factor can be a cause of an accident, smaller deviations in multiple factors can also lead to unwanted consequences . Therefore, with the factor c, the system penalizes choices with many smaller deviations, aiming to address the potential issues outlined by the "Swiss Cheese" model . The objective then is to mitigate the impact of a failed mission by upholding safety measures and considering the financial implications of decisions. In essence, the problem demands the minimization of potential causes of trouble for pilots. However, it was converted into a maximization problem by taking the inverse of the original minimization problem. This adjustment was made to tailor the reward function more closely to the LCS and its description in the literature. In the current implementation there were four constraints:

- Required landing distance $<=$ Available landing distance.
- Fuel needed to reach airport $<=$ Fuel available.
- Tailwind at selected airport did not exceed 15 $[kn]$.
- Crosswind at selected airport did not exceed 30 $[kn]$

The distance from the final airport to the chosen airport as well as the fuel usage to reach it should both be minimized. Additionally, the wind components as well as the runway status were needed to calculate the required landing distance.

B. Hyperparameter Tuning

The experiments involved hyperparameter tuning using Bayesian optimization and leveraged [42]. This method constructs a posterior distribution of functions (Gaussian process) that best describes the function you aim to optimize. As the number of observations grows, the posterior distribution improves, and the algorithm becomes more certain about which regions in the parameter space are worth exploring and which are not [43]. For each combination of parameters, the model was run 31 times due to its stochastic nature. The average reward across the entire DAAS problem dataset was used as the output from one of these runs and subsequently as the estimate of the performance for the combination of parameters. The best-found hyperparameters are illustrated in Table II.

C. Model Training Results

Figures 5 and 6 illustrate the results from the LCS model trained using the best-found hyperparameters. These figures depict the average cumulative reward across 31 runs of the LCS model. This means that for each trial, representing a single instance of the dataset (i.e., a scenario), the average reward for each of the 31 runs was calculated and then plotted.

Hyperparameter	Lower bound	Upper Bound	Optimal Value
Population size	100	220	196
r	0.01	0.1	0.097
θ_m	0.1	1	0.382
α	0.01	0.1	0.028
v	1	10	7.23
min error	0.01	0.1	0.022
η	0.01	0.1	0.06
init. pred. error	0.01	0.1	0.047
mutation probaility	0.1	1	0.278
crossover probability	0.1	1	0.89
δ	not tuned	not tuned	0.1
θ_{del}	not tuned	not tuned	20

TABLE II
HYPERPARAMETER VALUES

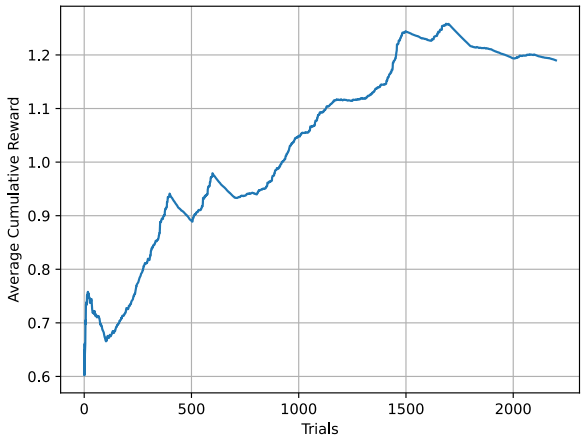

Fig. 5. Average cumulative reward

Upon observing Figure 5, we notice a consistent increase in the LCS's performance with only slight peaks, reaching a plateau near the end of the trials, i.e., near the end of the dataset. Interestingly, when examining the constraint violations, we observe an initial spike in the number of violations before a decline. It's noteworthy that the average number of constraint violations at the end stabilizes around 1.2. Given the dataset's nature, this suggests that the algorithm prioritized safety, as expected.

D. Comparison with a Neural Network

To gain a better understanding of the model's performance, it was compared to a Deep Q Network (DQN) implemented in Stable-Baselines3 [44]. Given the emphasis on interpretability, the neural network's structure was kept smaller and simpler. To determine an appropriate structure, the Optuna framework was utilized [45]. The search was limited to two to six layers with 64, 128, 256, 512, or 1024 nodes per layer. The optimal configuration identified was a four-layer network with 1024 nodes per layer. Both the LCS and DQN were trained and then

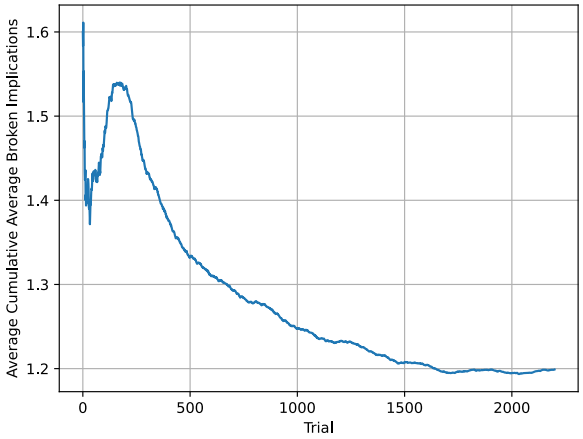

Fig. 6. Average cumulative number of constraint violations

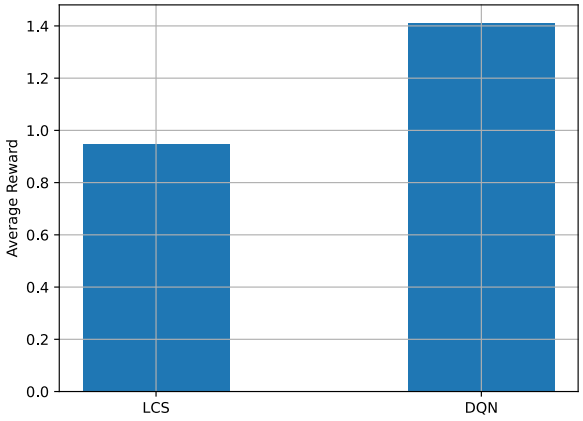

Fig. 7. Average reward comparison between LCS and DQN

applied to predict on the training dataset. This process was repeated 31 times due to the LCS's more stochastic nature. The comparison is visually depicted in Figure 7. From the figure, it's evident that the neural network outperformed the LCS. However, the focus shouldn't solely be on performance; rather, attention should also be on the complexity required to surpass the LCS. Despite being outperformed by the neural network, the LCS remains the more interpretable and explainable option, even though a neural network of this size is still more challenging to understand. Therefore, despite its inferior performance, the LCS's high interpretability and explainability warrant further research.

VI. Conclusion and Future Work

This paper presents an approach to tackle a time-constrained multi-criteria decision-making problem by transforming it into a reinforcement learning problem, using a learning classifier system (LCS) tailored for real-valued problems. The paper demonstrates this approach by solving the dynamic alternate airport selection problem (DAAS), which involves pilots selecting a suitable airport mid-flight based on multiple characteristics. Although not as effective as a Deep Q Network, the LCS proved capable of handling the task and is a promising candidate for future research, particularly due to its higher interpretability and explainability compared to neural networks. Further investigation should explore acceptable trade-offs between interpretability and model performance. Additionally, efforts are underway to explore additional LCS configurations and conduct more hyperparameter tuning, along with expanding the dataset to include technical failures and enhancing the simulation. It's important to note that while the dataset includes many technical details and weather data, it's not sourced from real-world data. However, despite the challenge of gathering a sufficient number of edge cases, it still provides significant value and aids in creating intelligent decision support systems in aviation. A finalized model is intended for use in a prototype decision support system with real pilots, aiming ultimately to enhance aviation safety.

References

[1] B. Djartov, S. Mostaghim, A. Papenfuß and M. Wies, "Description and First Evaluation of an Approach for a Pilot Decision Support System Based on Multi-attribute Decision Making," 2022 IEEE Symposium Series on Computational Intelligence (SSCI), Singapore, Singapore, 2022, pp. 141-147, doi: 10.1109/SSCI51031.2022.10022076.

[2] Walters, A.J., 2002. Crew resource management is no accident. Aries.

[3] Hörmann, H.J., 1994. FOR-DEC-A prescriptive model for aeronautical decision making.

[4] Würfel, J., Djartov, B., Papenfuß, A. and Wies, M., 2023. Intelligent pilot advisory system: The journey from ideation to an early system design of an AI-based decision support system for airline flight decks. AHFE 2023.

[5] European Union Aviation Safety Agency (2023) "EASA-AI-Roadmap 2.0: A human-centric approach to AI in aviation", Cologne.

[6] Majumder, M. and Majumder, M., 2015. Multi criteria decision making. Impact of urbanization on water shortage in face of climatic aberrations, pp.35-47.

[7] Mi, X., Tang, M., Liao, H., Shen, W. and Lev, B., 2019. The state-of-the-art survey on integrations and applications of the best worst method in decision making: Why, what, what for and what's next?. Omega, 87, pp.205-225.

[8] Lempert, R.J., Groves, D.G., Popper, S.W. and Bankes, S.C., 2006. A general, analytic method for generating robust strategies and narrative scenarios. Management science, 52(4), pp.514-528.

[9] Groves, D.G. and Lempert, R.J., 2007. A new analytic method for finding policy-relevant scenarios. Global Environmental Change, 17(1), pp.73-85.

[10] Kwakkel, J.H., Haasnoot, M. and Walker, W.E., 2015. Developing dynamic adaptive policy pathways: a computer-assisted approach for developing adaptive strategies for a deeply uncertain world. Climatic Change, 132, pp.373-386.

[11] Hamarat, C., Kwakkel, J.H., Pruyt, E. and Loonen, E.T., 2014. An exploratory approach for adaptive policymaking by using multi-objective robust optimization. Simulation Modelling Practice and Theory, 46, pp.25-39.

[12] Trindade, B.C., Reed, P.M., Herman, J.D., Zeff, H.B. and Characklis, G.W., 2017. Reducing regional drought vulnerabilities and multi-city robustness conflicts using many-objective optimization under deep uncertainty. Advances in Water Resources, 104, pp.195-209.

[13] Watson, A.A. and Kasprzyk, J.R., 2017. Incorporating deeply uncertain factors into the many objective search process. Environmental Modelling and Software, 89, pp.159-171.

[14] Bozorg-Haddad, O., Zolghadr-Asli, B. and Loaiciga, H.A., 2021. A handbook on multi-attribute decision-making methods. John Wiley and Sons.

[15] Martin Aruldoss, T. Miranda Lakshmi, V. Prasanna Venkatesan, "A Survey on Multi Criteria Decision Making Methods,American Journal of Information Systems, 2013, Vol. 1, No. 1, 31-43 and Its Applications"

[16] Saaty TL (1980) The analytic hierarchy process. McGraw Hill, New York

[17] Saaty TL (2000) Fundamentals of decision making and priority theory with AHP. RWS Publications, Pittsburg

[18] Hwang CL, Yoon K (1981) Multiple attribute decision making: methods and applications: a state-of-the-art survey. Springer, London

[19] Yu PL (1973) A class of solutions for group decision problems. Manage Sci 19:936–946

[20] Zeleny M (1982) Multiple Criteria Decision Making. McGraw-Hill, New York

[21] Rao RV, Patel BK (2010) Decision making in the manufacturing environment using an improved PROMETHEE method. Int J Prod Res 48:4665–4682

[22] Jahan A, Ismail MY, Mustapha F, Sapuan SM (2010) Material selection based on ordinal data. Mater Des 31(7):3180–3187

[23] Alvarez, I.M., Browne, W.N. and Zhang, M., 2016, July. Human-inspired scaling in learning classifier systems: Case study on the n-bit multiplexer problem set. In Proceedings of the Genetic and Evolutionary Computation Conference 2016 (pp. 429-436).

[24] Djartov, B. and Mostaghim, S., 2023, July. Multi-objective Multiplexer Decision Making Benchmark Problem. In Proceedings of the Companion Conference on Genetic and Evolutionary Computation (pp. 1676-1683).

[25] Urbanowicz, R.J. and Browne, W.N., 2017. Introduction to learning classifier systems. Springer.

[26] Urbanowicz, R.J. and Moore, J.H., 2009. Learning classifier systems: a complete introduction, review, and roadmap. Journal of Artificial Evolution and Applications, 2009.

[27] Airbus, "A318/A319/A320/A321 Flight Crew Operating Manual", ISSUE DATE: 11 DEC 13, REFERENCE: LFO A318/A319/A320/A321 FLEET FCOM

[28] "List of the busiest airports in Germany.", Wikipedia, url:https://en.wikipedia.org/wiki/List$_of_the_busiest_airports_in_Germany$

[29] Hoekstra, Jacco and Ellerbroek, Joost. (2016). "BlueSky ATC Simulator Project: an Open Data and Open Source Approach."

[30] Mckay et al., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code." Technometrics, 1979.

[31] Online Flight Planner,url:http://onlineflightplanner.org/

[32] Flightradar24, url:https://www.flightradar24.com/50.13,8.29/11

[33] Yuan Lyu, Rhea P. Liem, "Flight performance analysis with data-driven mission parameterization: mapping flight operational data to aircraft performance analysis", Transportation Engineering,Volume 2,2020,

[34] Olive, X., Tanner, A., Strohmeier, M., Schäfer, M., Feridun, M., Tart, A., Martinovic, I. and Lenders, V., 2020, October. OpenSky Report 2020: Analysing in-flight emergencies using big data. In 2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC) (pp. 1-10). IEEE.

[35] Eurocontrol. (2024). Aircraft Performance Details: Airbus A320. Retrieved from https://contentzone.eurocontrol.int/aircraftperformance/details.aspx?ICAO=A320

[36] Butz, M.V., Lanzi, P.L. and Wilson, S.W., 2006, July. Hyper-ellipsoidal conditions in XCS: rotation, linear approximation, and solution structure. In Proceedings of the 8th annual conference on Genetic and evolutionary computation (pp. 1457-1464).

[37] Wilson, S.W., 1999, July. Get real! XCS with continuous-valued inputs. In International Workshop on Learning Classifier Systems (pp. 209-219). Berlin, Heidelberg: Springer Berlin Heidelberg.

[38] Wilson, S.W., 2002. Classifiers that approximate functions. Natural Computing, 1(2-3), pp.211-234.

[39] Atkeson, C.G., Moore, A.W. and Schaal, S., 1997. Locally weighted learning. Lazy learning, pp.11-73.

[40] Butz, M.V., Goldberg, D.E. and Tharakunnel, K., 2003. Analysis and improvement of fitness exploitation in XCS: Bounding models, tournament selection, and bilateral accuracy. Evolutionary computation, 11(3), pp.239-277.

[41] Eurocontrol Experimental Center, Revisiting the "Swiss Cheese" model of accidents, 2006, url:https://www.eurocontrol.int/publication/revisiting-swiss-cheese-model-accidents

[42] Fernando Nogueira, Bayesian Optimization: Open source constrained global optimization tool for Python, 2014, url = https://github.com/fmfn/BayesianOptimization

[43] Snoek, J., Larochelle, H. and Adams, R.P., 2012. Practical bayesian optimization of machine learning algorithms. Advances in neural information processing systems, 25.

[44] Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M. and Dormann, N., 2021. Stable-baselines3: Reliable reinforcement learning implementations. Journal of Machine Learning Research, 22(268), pp.1-8.

[45] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A Next-generation Hyperparameter Optimization Framework. In KDD.