Multi-objective Multiplexer Decision Making Benchmark Problem

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

ABSTRACT

This paper proposes a novel multi-objective decision making benchmark problem. The problem addresses the need in the multi-objective decision making realm for an easy to construct, scalable benchmark problem in the vain of the DTLZ, ZTD, and WFG problems. The problem is inspired by a real-world decision making problem that pilots face in the cockpit. The new problem is an amalgamation of two well-established problems within the literature, the DTLZ and multiplexer problems. The problem additionally makes use of the main concepts and ideas from Robust Decision Making and Multi-scenario Multi-objective Robust Decision Making, especially as these problems enable decision making problems to be somewhat converted into an optimization task. The problem is showcased here and is solved initially using a modified multi-objective optimization variant of a Learning Classifier System, which shows superior results when compared to a random agent.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning approaches**.

KEYWORDS

multi-objective optimization, multi-objective decision making, multi-objective benchmark problem, multi-objective decision making benchmark problem

ACM Reference Format:
Boris Djartov[1,2], Sanaz Mostaghim[2], [1]Institute for flight guidance, German Aerospace center (DLR), Braunchweig, Germany, boris.djartov@dlr.de, [2] Faculty of Computer Science, Otto-von-Guericke-University, Magdeburg, Germany, sanaz.mostaghim@ovgu.de . 2023. Multi-objective Multiplexer Decision Making Benchmark Problem. In *Genetic and Evolutionary Computation Conference Companion (GECCO '23 Companion), July 15–19, 2023, Lisbon, Portugal*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3583133.3596360

1 INTRODUCTION

Dynamic alternate airport selection (DAAS) while flying is a decision making task that pilots may find themselves facing. The task requires decision makers and pilots to change their planned mission and choose a new destination airport while possibly under stressful and unforeseen conditions. The decision makers must take into account multiple factors, such as the location of the airport, their fuel levels, weather conditions, possible logistical problems for the passengers, etc [1]. The pilots need to consider the interdependence of all these factors as well as keep in mind the many trade-offs when making their decision. For example, a pilot may need to consider if the airport with a strong crosswind and a wet runway of is more favorable than an airport that is further away and has strong tailwind. Upon examining the literature, testing problems or similar problems of this nature were difficult to find. The closest type of problem were those related to multi-objective sequential decision-making problems, as outlined in Cassimon's 2012 survey [2]. However, these problems have a predetermined number of objectives and are not easily scalable, Additionally, they are often based on certain simplistic games as is very common in the field of Reinforcement learning. Inspired by the lack of test problems, this paper presents a new benchmark problem that is created by fusing two well-known problems within the literature, the multiplexer problem [3] and the DTLZ [4] [5] problem. The multiplexer was chosen because it is characterized by the interactivity between its features as well as its heterogeneous nature. This mimics the need for pilots to prioritize and pay attention to certain airport characteristics, which can vary based on the circumstances. DTLZ was included to represent the trade-offs that come with the pilot's choice and also because the Pareto front is known and can easily be scaled. Thus, this test problem aims to capture the characteristics posed by DAAS; however, it also aims to be a conceptually simple, scalable, and difficult test problem. The paper organization borrows from the structure of [4] and [5] and is organized in the following manner: The 1 section gives a brief theoretical overview. The following section then gives the requirements for a new multi-objective decision-making benchmark problem. The new multi-objective multiplexer problem (MOMP) is described next. In section 5 the description of the modified version of a Learning Classifier System (LCS) is given. The following section outlines the results from the test problem's debut appearance, along with why approaches like the LCS are important and should be considered in similar decision making problems. Finally, a summary of the work and the conclusions drawn are given.

2 THEORETICAL BACKGROUND

2.1 Dynamic Alternate Airport Selection

Given that the inspiration for this benchmark problem was the DAAS problem, it seems only prudent to give a short primer on it.

GECCO '23 Companion, July 15–19, 2023, Lisbon, Portugal

ACM ISBN 979-8-4007-0120-7/23/07.
https://doi.org/10.1145/3583133.3596360

The problem in a multi-objective optimization context was initially proposed by [1]. DAAS is a decision-making problem where the decision maker, in this case the pilot, needs to abruptly select a new destination airport midflight. Thus, the problem consists of selecting the best airport or solution from a list of possible alternatives, defined by multiple characteristics. In [1], the authors determined the following characteristics for each airport or option:

- Longest runway length available $[m]$
- Wind speed at the location of the airport $[kn]$
- Wind direction at the location of the airport $[^\circ]$
- Status of the breaking runway in terms of how dry or wet it is, expressed as a value from 0 to 6
- Distance from current aircraft position to planned destination $[nm]$
- Distance from current aircraft position to appropriate alternate airport $[nm]$

Additionally, DAAS problem also has additional factors that define the state or the decision making context. The aircraft position, flight level, fuel level and flight path define the circumstances in which the decision needs to be made.

2.2 Multi-objective decision making

The DAAS problem was transformed into a bi-objective optimization problem in [1], with risk and cost as the two objectives to consider. The risk objective represented the possibility of physical danger, while the cost objective represented the financial implication of each decision. These two objectives showcased the trade-offs that pilots needed to consider when making their decision. However, in the DAAS problem, due to the dynamic nature of flying, many of the characteristics are bound to change. Thus, it is not a static decision-making context and depends on the changing factors on the ground, such as the wind speed and weather phenomena, as well as the status of the aircraft, its technical status, any medical emergencies, etc.

Keeping in mind the variability in the circumstances that needed to be taken into account before making the decision, the problem borrowed certain elements from the Robust Decision Making (RDM) [6] [7] framework, more specifically from Multi-Objective Robust Optimization (MORO) [8] [9] [10]. Robust Decision Making (RDM) is an approach to decision making that helps decision makers navigate uncertainty and account for the potential for unexpected events or changes in the future. RDM involves exploring multiple possible scenarios or futures, analyzing the potential outcomes of different decisions under each scenario, and selecting a decision that is robust across a wide range of potential futures. RDM is often used in complex decision-making contexts where there is high uncertainty and a need for long-term planning [6]. Namely, the authors from [1] presented the problem as a RDM problem where a synthetic data set was created corresponding to difficult scenarios which could be encountered and needed a suggested course of action.

This dynamic nature of DAAS problem, coupled with the fact that the decision makers could be in unforeseen and stressful situations, necessitated the creation of a multi-objective decision support system that was not reliant on the typical multi-objective decision-making methods from the literature. Methods such as the AHP [11] [12], VIKOR [13], PROMETHEE [14], ELECTRE [15] proved to be insufficient as the decision makers could not be reliably asked to give their preference in every situation when a decision needed to be made, and it was unreasonable to give their preference before hand for every possible situation that might occur. The methods from RDM also do not provide a sufficiently good way to tackle the problem. As previously stated they are geared more towards analyzing and selecting decisions in terms of long term planning. The need to analyze each of the solutions necessitates greater attention from the decision maker, which may not be possible in a stressful situation. Thus, although the current literature helped shape this problem, an added element was needed to cope with difficulties that have arisen due to time and focus constraints. An intelligent decision support system capable of giving good suggestions across a multitude of varying scenarios would help the decision makers achieve their goals even while burdened by atypical situations.

Problems from the literature similar to DAAS problem were those found in the multi-objective reinforcement learning realm. Decision-making problems with multiple objectives include Deeps Sea Treasure [16], MO-Puddleworld [17], MO-Mountain-Car [18], and Resource Gathering [19]. However, because they have a fixed number of objectives, these problems are not scalable and they primarily focus on sequential decision making. A visual representation of the difference between multi-objective reinforcement learning and the DAAS problem can be seen in figure 1. The figure aims to highlight that the DAAS problem is a single decision needs to be made with the actions available being very dynamic, i.e. the characteristics of both the airport and the status of the airplane, which is not usually encountered in the realm of reinforcement learning, where the possible set of actions are often known and determined. In terms of Robust decision making a similar and well known problem was the lake problem, presented in initially in [20] where the goal was to managing eutrophication in lakes subject to potentially irreversible change. The problem involved identifying and evaluating different management strategies that aim to balance multiple objectives, such as improving water quality, preserving biodiversity, and ensuring the sustainability of the lake ecosystem. Although, greater focused seems to be placed on multi-objective optimization where the goal is to generate new Pareto optimal solutions or to find the Pareto front, once identified it seems that decision makers and stakeholders need to select a subset, often one solution, from the found solutions. To this end, decision making methods, where an alternative needs to be chosen from a set of possible alternatives, are relevant and vital. The ideas from RDM and MORO of examining multiple probable possible circumstances in order to have a robust and optimal decision together with the added need to be used in a dynamical changing environment warrants a more established benchmark problems that can be used to test and compare the intelligent decision support systems.

2.3 Component problems

In order to better understand the proposed problems that comprise the Multi-objective Multiplexer Problem, a short primer will be given on both the multiplexer and DTLZ problems.

The Boolean n-bit multiplexer problem, first presented by Koza [3] is based on the electronic multiplexer (MUX), a device that takes

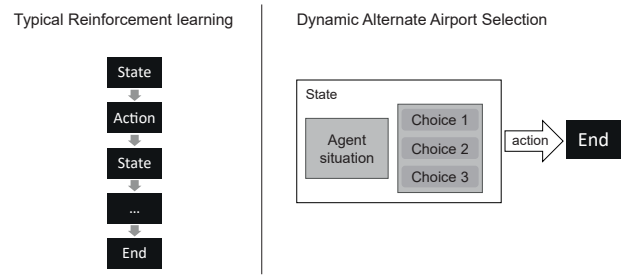

Figure 1: DAAS problem state to action relationship

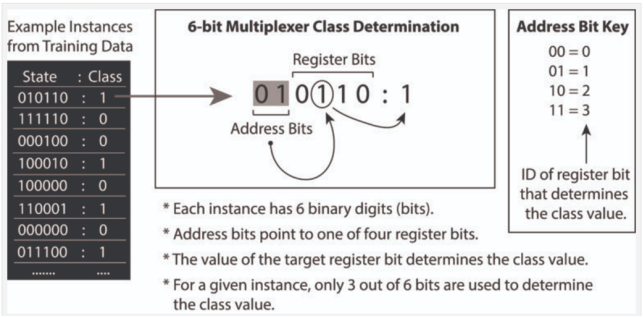

Figure 2: 6-bit Multiplexer problem [21]

multiple analog or digital input signals and switches them into a single output [21]. An example of a 6-bit multiplexer problem can be seen on figure 2, courtesy of Urbanowicz and Borwne's book Introduction to Learning Classifier Systems [21]. The multiplexer problem is a supervised learning classification problem where the goal is to predict the class of a sequence of bits. The multiplexer data set is generated by randomly creating strings of bits and assigning a class to the sequence by examining the address bits and how they relate to all the others elements in the sequence. The goal would then be to train an algorithm to try and learn the pattern and be able to successfully predict the outcome of a presented sequence of bits. The multiplexer problem is characterised by its interactions between the features/ bits and also it's heterogeneity, where for different sets of instances, a distinct subset of features will determine the class value. Although, easy to construct the multiplexer problem is non-trivial and is predicated on the existence of a somewhat complicated pattern to determine the class. The problem can also be easily scaled by simply expanding the number of address bits, so its complexity can be modified with minor changes [21] [22]. The presence of varying importance between the factors depending on the situations is what makes the multiplexer problem and DAAS problem so similar. Namely in DAAS problem the characteristics of the actual aircraft i.e. its position, fuel level, altitude and planned mission represent the address bits. They dictate what of the characteristics that represent the airports should be considered more and by how much more. This is one of the reasons that the multiplexer problem was chosen.

The DTLZ benchmark problems were presented in [5] and [4]. Within the papers the writers specify a bottom up approach for the

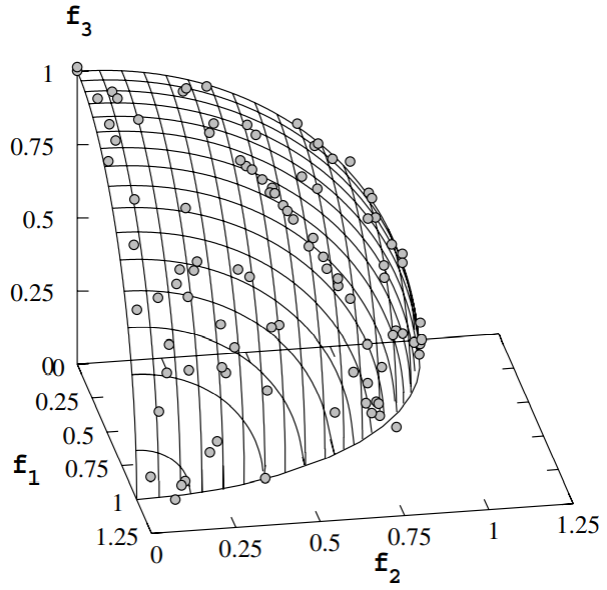

Figure 3: The NSGA-II Population on Test Problem DTLZ2 [5] [4]

design of the test problem. Specifically, the mathematical function that defines the Pareto-optimal front is assumed beforehand and the objective space is constructed based on this front. The testing suite contains multiple problems numbered from one to nine, spread across the two papers. Here the DTLZ2 problem will only be showcased. DTLZ2 mathematically is described as follows:

$$min(f_1(\vec{x})) = (1 + g(x_m))(\cos(x_1\pi/2)... \cos(x_m 2\pi/2) \cos(x_m\pi/2),$$
$$min(f_2(\vec{x})) = (1 + g(x_m))(\cos(x_1\pi/2)...$$
$$... \cos(x_{m-2}2\pi/2) \sin(x_{m-1}\pi/2),$$
$$min(f_3(\vec{x})) = (1 + g(x_m))(\cos(x_1\pi/2)... \cos(x_{m-2}2\pi/2),$$
$$...$$
$$min(f_m(\vec{x})) = (1 + g(x_m))(\sin(x_1\pi/2),$$

with

$$g(x_m) = \sum_{x_i \in x_m} (x_i - 0.5)^2,$$
$$0 \leq x_i \leq 1, \text{ for } i = 1, 2, ..., n$$

Where $\vec{x}$ is a vector constructed with $k = n - m + 1$ variables. The Pareto-optimal solutions correspond to $x_i = 0.5$ for all $x_i \in x_m$ and all objective function values must satisfy $\sum_{i=1}^{m} f_i^2 = 1$. The DTLZ2 as tackled by the NSGA-II algorithm, for 3 objectives is represented visually on figure 3. The inclusion of a multi-objective optimiztion component is in order to mimic the trade-offs that pilots need to consider and make when deciding on an airport.

2.4 Learning Classifier systems

A Learning Classifier System (LCS) is a type of machine learning algorithm that combines reinforcement learning with genetic algorithms to learn and improve upon a set of rules or classifiers over

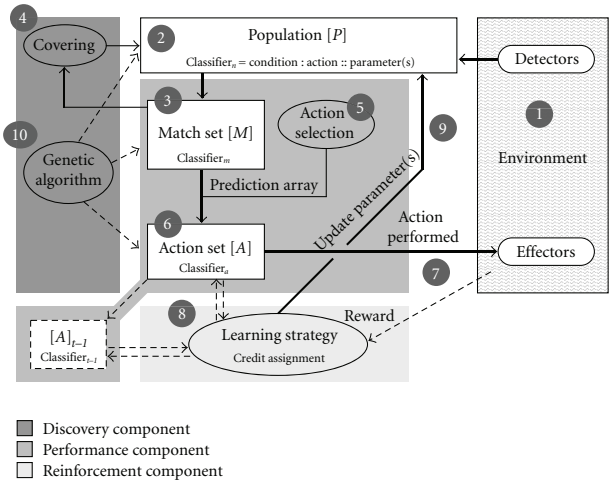

Figure 4: Visual representation of a LCS [22]

time. The general idea behind LCS was introduced by John Holland in [23], as a way to model how natural selection and evolution work in biological systems. The LCS approach involves creating a population of potential classifiers, each of which represents a set of rules that can be applied to input data. The system then evaluates the fitness of each classifier, based on how well it performs on a given task. The fittest classifiers are then selected to reproduce, with their genetic material passed on to the next generation of classifiers. Over time, the system converges on a set of high-performing classifiers that can be used to make accurate predictions or decisions. One of the strengths of the LCS approach is that it is highly interpretable. Because the system generates a set of rules or classifiers that are designed to capture the underlying patterns in the data, it is easier for humans to understand how the system is making its choices. This can be particularly useful in domains where transparency and interpretability are important, such as healthcare, finance, or legal settings. Another strength of the LCS approach is its ability to handle noisy or complex data. Because the system uses a genetic algorithm to evolve its classifiers over time, it is able to adapt to changing environments and identify patterns in data that might be difficult for other machine learning algorithms to detect. In addition to its strengths in interpretability and handling complex data, the Learning Classifier System (LCS) is also particularly well-suited for tackling a specific problem known as the multiplexer problem. This problem involves generating a set of rules or classifiers that can accurately predict the output of a circuit with multiple inputs and outputs, where the output depends on a specific combination of input values. Overall, the Learning Classifier System is a powerful machine learning algorithm that combines the benefits of reinforcement learning and genetic algorithms to learn and improve upon a set of rules over time. Its strengths in interpretability and ability to handle complex data make it a promising approach for a wide range of applications. A visual representation of a generic LCS can be seen on figure 4

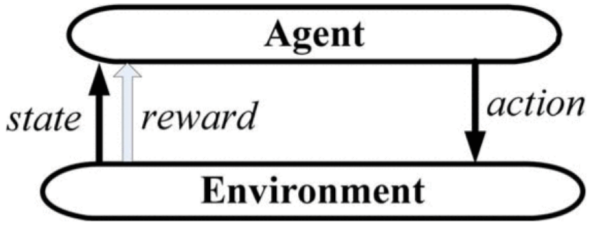

Figure 5: Typical framing of a reinforcement learning approach

3 BENCHMARK PROBLEM REQUIREMENTS AND CHALLENGES

3.1 Requirements

It is important to determine the requirements that the benchmark problem ought to have for a multi-objective decision making problem. Hence, it seams like a prudent step to take the already established requirements presented in [5] and [4]. Thus the requirements for this benchmark problem are:

- The benchmark problem should be easy to construct.
- The benchmark problem should be scalable and adjustable in terms of its difficulty, especially in terms of the number of objectives.
- The Pareto front should be known and determined for easier examination of the results as they relate to an optimal solution.
- The benchmark problem should exhibit similarities to some real world problems.

3.2 Difficulties

When it comes to decision making problems, what can be observed from the literature, is that it is very subjective and dynamic process. More accurately decision making is based on "context". One could argue that the very idea of trying to capture and represent the preferences of the decision maker as an ideal point, a weight vector, or a type of trade-off matrix is to try and establish the context in which the decision making is taking place. Examining the field of reinforcement learning, through interactions with an uncertain environment, a learning agent seeks to learn an ideal action policy. Every step along the way, the learning agent is not informed clearly what action to do, and instead it must choose and carry out the appropriate course of action to maximize long-term benefits. A scalar reward signal that assesses the impact of this state transition is then sent to the agent once the chosen action causes the environment's current state to change into its subsequent one [24] [25]. An example of a reinforcement signal is presented in figure 5.

Thus, it appears that in reinforcement learning, the state serves as a context for a decision. What is needed is to somehow provide a context component that can be easily employed in a benchmark problem. A context component can be defined as a subset of attributes that interact with some other subset of attributes to influence the final decision in a significant manner. Thus, context

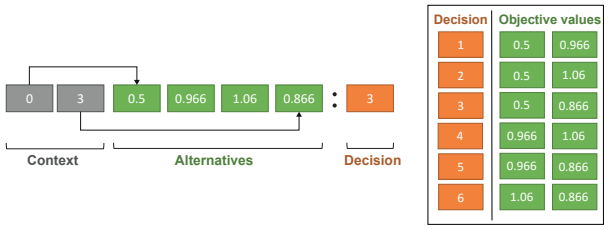

Figure 6: General structure of the MOMP

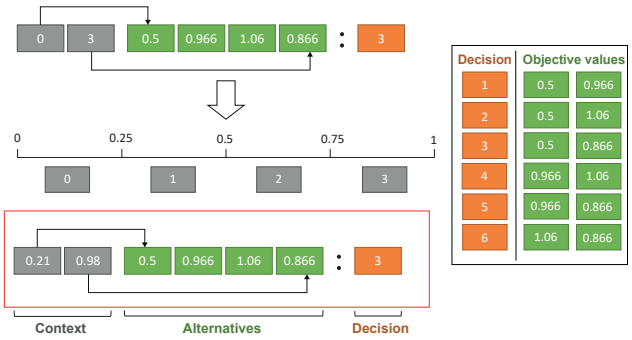

Figure 7: Final MOMP

can be defined as the state with regard to reinforcement-learning problems and as the presence of epistasis and heterogeneity for multi-objective non-sequential decision making problems. The fact that these two characteristics are present in the multiplexer problem makes it a good starting point for a benchmark decision making problem.

4 MULTI-OBJECTIVE MULTIPLEXER PROBLEM

4.1 Description

Having defined the multiplexer and DTLZ problems, established the requirements, and gone over the real-world problem that inspired the multiplexer problem, we can now proceed to defining the multi-objective multiplexer problem.

The multi-objective multiplexer problem (MOMP) can be defined as a multi-objective multi-class classification problem, where the goal is to select an appropriate class, i.e., make a choice from a set of possible options. The problem is envisioned to be in the form of a synthetic, user-generated data set, where each instance is a sequence of positive real numbers. The sequence consists of two main parts, similar to the multiplexer. The context part, which is akin to the address bits, and the alternatives part, which is similar to the register bits found in the multiplexer problem A visual representation of the general structure of the MOMP can be seen in figure 6. The string of numbers in the figure represents an instance from a data set that can be easily generated. The correct context values specify the location of the objective values that lie on the Pareto front and are determined, in this case, via the objective values from the DTLZ problem, $\sum_{i=1}^{m} f_i^2 = 1$. The number of possible options to choose from represents the possible number of combinations that the alternative values can be arranged in. The number of classes or options is $\binom{n}{k}$, where n represents the number of desired alternative values and k represents the number of objectives. The number of context variables is dependent on the number of objectives, which is determined by the end user. Combinations were chosen instead of permutations so as to avoid having more than one Pareto optimal decision. An additional specification of the problem that should be noted is that the objective values are always assigned from left to right as they appear in the sequence of values. The possible decisions and classes that can be assigned are also based on combinations of the remaining alternative values. The way to generate the decisions and combinations is arbitrary; what matters is that the same method be used across all of the instances in the data set.

For example, decision one should always refer to selecting the first and second values of the sequence in that particular order.

To make the problem even more challenging and to blur the line between the context and alternative values, the context values are swapped out for values within a certain range. This is also done so that it is harder for the algorithm to distinguish subsets of values, as is the case in a real world problem like DAAS problem. This is achieved by taking an interval, typically from 0 to 1, and dividing it into a number of smaller intervals, with equal size. The number of smaller intervals corresponds to the number of alternatives available. For example, in figure 6, there are four alternatives available, and thus the number range can be divided in increments of 0.25. Thus, a context value of 0 can be represented by any value from the interval [0, 0.25]. The more accurate and complete version of the problem is visually shown in figure 7.

MOMP is a multi-objective optimization problem where the goal is to minimize the objectives. Specifically, the goals is to

$$min(F_1, F_2, F_3, ..., F_m)$$

where F_m is calculated as an average out of all the values for objective m, across the entire data set of length n

$$F_m = \frac{1}{n} \sum_{i=1}^{n} f_m$$

In essence the goal is for any algorithmic approach to understand the relationship and select the decision that corresponds to the objective values that lie on the Pareto front. Put simply, every classification, i.e., decision, is evaluated by multiple objective values. The goal is to simply select all the decisions such that, over the entire data set, the objectives are minimized. This, however, is based on the presented form of the problem which makes use of the DTLZ test problem for a scalable objective function. However, the problem is flexible, and the minimization or maximization of the objectives can be entirely left up to the discretion of the end user.

4.2 Problem construction

Given that one of the requirements of a benchmark problem is ease of construction, here the generic procedure for constructing a bi-objective minimization MOMP problem of length six is outlined:

(1) Select the Pareto optimal front. The MOMP problem is envisioned to make use of the DTLZ problems, and for future

steps, it is assumed that the DTLZ2 problem was chosen for this step.
(2) To serve as the initial value for the Pareto front value, generate a random value between (0,1).
(3) Generate the additional corresponding value based on DTLZ2's formulation so that these two values together represent a point on the Pareto front.
(4) Choose values for "padding numbers". The padding numbers are added to the larger of the previously generated random numbers (in a minimization context) The "padding numbers" should be chosen such that these values together with any other value must produce a point that is sub Pareto optimal. The number of padding numbers should be $p = c - m$, with m representing the number of objectives and c representing the number of alternative values. An example for padding numbers is 0.1 and 0.15.
(5) Create all possible permutations of the alternative sequence of numbers, making note of the position of the optimal values and changing accordingly. This means that for each set of optimal numbers and "padded numbers" generated they can appear in all positions in the alternatives sub array. Thus the context values should be updated accordingly.
(6) Divide up the interval [0, 1] into smaller, equal intervals. The number of intervals should be equal to the number of alternative values.
(7) Assign the previously noted positions of a Pareto optimal value to a specific interval.
(8) Generate the values for the context based on the previously determined intervals.
(9) Repeat the previous steps until you have a data set of the desired size.
(10) Shuffle the instances in the created data set so that the position swapped instances are not next to one another.

4.3 Requirements satisfaction

This subsection serves as a quick checklist to determine if the MOMP problems satisfy the previously stated requirements.

- Given the laid out steps, the problem is easy to construct. The difficulty associated with its construction is the selection of the function that represents the Pareto front. The following steps are nothing more than random number generation, given some constraints.
- Since the problem is based on the multiplexer and DTLZ problems, it inherits their ease of scaling. Namely, to make the problem more difficult, one can increase the length of the sequence. The MOMP also allows users to adjust how difficult the problem is, as the user can have a long sequence but only two objectives. What is important to note is that the number of alternatives should be at least $2 + c$, where c is the number of context values. This is done so that the problem can have sufficient amount of non-dominated options.
- The problem has a well-defined Pareto front, and like in the multiplexer problem, the pattern to determine the proper choice is known and easy for a human to check.

Algorithm 1 Multi-objective LCS

```
for iteration = 1.. to number of data set iterations (number of times the LCS goes over the data set do
    if no rule matches an instance then
        create a new rule using covering operator

    else
        evaluate rules -> calculate the average for each objective based on the instances they matched and the decision that they recommended
        select the next population based on non-dominated sorting and a predetermined population limit = number of instances/4
        apply uniform crossover
        apply mutation
        add the offspring to the initial population

```

- The benchmark problem is based on the DAAS problem and aims to mimic the nuances that were not accurately represented by a benchmark problem in the literature.

5 MULTI-OBJECTIVE LEARNING CLASSIFIER SYSTEM

This section serves to give a short description of the structure of the LCS used to solve the proposed MOMP. The general structure of the LCS remains mostly the same as the Michigan-Style Learning Classifier System used to solve single-objective problem. The difference is in the inclusion of the Non-dominated sorting from the NSGA-II algorithm [26]. The mutation operator was custom made to suit the interval encoding that was chosen for the rules, i.e. the rules were comprised of a set of intervals, lower and upper bounds for all of the provided features of the data set. This interval based encoding method was also chosen as it seams to be the easiest to analyze later on and draw conclusions from. The general psudocode implementation for the LCS is given in 1.

The presented LCS also makes use of covering operator. This inclusion, commonly found in more recent LCS, ensures that there is at least one rule that can handle the current instance. Additionally, it enables the construction of a more specific initial population, which is not initialized at random but rather offers a good starting off point. The crossover operator used is a standard uniform crossover operator, while the mutation operator involved widening, shrinking or converting the intervals to a "do not care" value. The pseudocode for the mutation is given in algorithm 2.

The mutation operator can be applied on each interval in the rule generated. It's design was also motivated to reduce one of the drawback that LCS are known for, namely their high number of hyperparameters. Hence, although the size of the shrinking and widening of the intervals could be tinkered with, it was chosen to be based on the present intervals so that the algorithm can be flexible in its changes to the mutations while not having to have additional hyperparameter tuning.

Algorithm 2 Interval Mutation operator

```
for iteration = 1.. to number of intervals do
    generate mutation type number from [0,1]
    if interval == "do not care interval" then
        continue
    else if mutation type number <= expand threshold value then
        exapender = upper bound - lower bound
        upper bound += expander
        lower bound += exapander

    else if  expand threshold value < mutation type number < shrink threshold value  then
        shrinker = (upper bound - lower bound) /4
        upper bound = upper bound - shrinker
        lower bound = lower bound + shrinker

    else
        interval = "do not care" interval

```

6 RESULTS ANALYSIS

The LCS was run on a constructed data set of the MOMP with 1200 instances and a length of six values with the DTLZ2 multi-objective optimization problem as the objective function. Given that it is a supervised problem and the optimal decisions are know we can easily check the algorithms performance. It is important to note however, that the algorithm should be given a vector of values as feedback given that the problem is envisioned as a multi-objective reinforcement learning problem. Treating it like a single-objective classification problem defeats the purpose behind its main ideas. Thus, the algorithms performance can be judged based simply on the number of optimal classifications/decisions divided by the number of total prediction, i.e how many correct decisions were made from the total number of predictions.

The experiments were executed using python 3.6.8 and made heavy use of the DEAP python package [27]. The experiments were run 31 times and the results were compared to a random agent. The comparison in performance can be observed in figure 8 and the result was also deemed statistically significant with a Mann–Whitney U test with a p-value of 0.0004901 and $\alpha = 0.05$. Although statistically significant the LCS was only slightly better than the random agent and has a low number of accurate predictions. This should indicate that this problem is not easy to solve especially when incorporating a LCS. The reason for implementing and focusing on LCS and straying away from neural networks, in their current form, is due to the better interoperability of LCS. In decision making in general and the aviation industry in particular being able to understand how and why decisions are made is of vital importance, thus interoperability and techniques for analysis gain as much of an importance as the act of solving the problem. This paper, its algorithms and proposed problems also aims at showcasing this point. When important and difficult decisions need to be made our methods must be more than black boxes and must be as interpretable as possible.

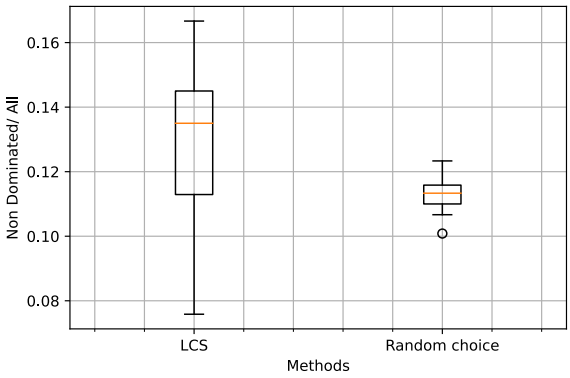

Figure 8: Comparison between a random agent and the Multi-objective LCS on MOMP

7 CONCLUSION AND FUTURE WORK

7.1 Conclusions drawn

This paper presents a new scalable, flexible benchmark problem for multi-objective decision making. The problem was based on the dynamic alternate airport selection problem encountered by commercial airline pilots and aimed to provide more test problems in the multi-objective decision-making literature. The multiplexer and DTLZ problems were combined in order to create a new problem that is able to satisfy the requirements needed by a benchmark problem in the field of multi-criteria decision making and also capture the nuances of the DAAS problem. Based on [21], Learning Classifier Systems (LCS) are uniquely suited to tackle problems such as the multiplexer problem. More specifically, problems that are characterized by heterogeneity and epistasis can be tackled very well with LCS. Being rule-based methods that make use of genetic algorithms, they can easily be adapted to a multi-objective purpose [28] [29]. Furthermore, their rule-based nature makes them more transparent and interpretable than other popular machine learning methods, such as deep neural networks [21]. Thus, to solve the newly presented MOMP, employing and exploring a solution using an LCS seems like a wise choice.

7.2 Avenues for future work

Regarding future endeavours two main ideas are currently being considered and developed. First and foremost is to borrow yet again from RDM and develop analytical tools to better understand and interpret the behaviour of the intelligent decision support system, specifically with regards to the multi-objective LCS. The second is to try and capitalize on the ability of neural networks to generalize by including them as a part of the LCS The aim is to use ideas neuroevolution in order to generate rules which are basically neural networks. With a focus on smaller and more interpretable neural network the LCS would be leaning on its often neglected attribute i.e. that it is an ensemble method as well. The ability for even smaller neural networks to generalize coupled with the LCS ability to segment and break down larger problems seems to merit further

research and would hopefully yield greater success in tackling the MOMP.

In conclusion, this paper presented a new benchmark problem for multi-objective decision as well as methods and ideas on how to tackle it that would hopefully fulfill a much-needed demand within the field.

REFERENCES

[1] Anne Papenfuss Matthias Wies Boris Djartov, Sanaz Mostaghim. Description and first evaluation of an approach for a pilot decision support system based on multi-attribute decision making. In *Proceedings of the 2022 IEEE Symposium Series on Computational Intelligence (IEEE SSCI)*. IEEE, 2022.

[2] Thomas Cassimon, Reinout Eyckerman, Siegfried Mercelis, Steven Latré, and Peter Hellinckx. A survey on discrete multi-objective reinforcement learning benchmarks. In *Proceedings of the Adaptive and Learning Agents Workshop (ALA 2022)*, 2022.

[3] John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4:87–112, 1994.

[4] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. *Scalable test problems for evolutionary multiobjective optimization*. Springer, 2005.

[5] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 1, pages 825–830 vol.1, 2002.

[6] Robert J Lempert, David G Groves, Steven W Popper, and Steve C Bankes. A general, analytic method for generating robust strategies and narrative scenarios. *Management science*, 52(4):514–528, 2006.

[7] David G Groves and Robert J Lempert. A new analytic method for finding policy-relevant scenarios. *Global Environmental Change*, 17(1):73–85, 2007.

[8] Jan H Kwakkel, Marjolijn Haasnoot, and Warren E Walker. Developing dynamic adaptive policy pathways: a computer-assisted approach for developing adaptive strategies for a deeply uncertain world. *Climatic Change*, 132:373–386, 2015.

[9] Caner Hamarat, Jan H Kwakkel, Erik Pruyt, and Erwin T Loonen. An exploratory approach for adaptive policymaking by using multi-objective robust optimization. *Simulation Modelling Practice and Theory*, 46:25–39, 2014.

[10] BC Trindade, PM Reed, JD Herman, HB Zeff, and GW Characklis. Reducing regional drought vulnerabilities and multi-city robustness conflicts using many-objective optimization under deep uncertainty. *Advances in Water Resources*, 104:195–209, 2017.

[11] Thomas L Saaty. Decision making with the analytic hierarchy process. *International journal of services sciences*, 1(1):83–98, 2008.

[12] Thomas L Saaty. *Fundamentals of decision making and priority theory with the analytic hierarchy process*. RWS publications, 1994.

[13] C-L Hwang and Abu Syed Md Masud. *Multiple objective decision making—methods and applications: a state-of-the-art survey*, volume 164. Springer Science & Business Media, 2012.

[14] R Venkata Rao and Bhisma K Patel. Decision making in the manufacturing environment using an improved promethee method. *International Journal of Production Research*, 48(16):4665–4682, 2010.

[15] Ali Jahan, Md Yusof Ismail, Faizal Mustapha, and Salit Mohd Sapuan. Material selection based on ordinal data. *Materials & Design*, 31(7):3180–3187, 2010.

[16] Peter Vamplew, Richard Dazeley, Adam Berry, Rustam Issabekov, and Evan Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. 2011.

[17] Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, 7, 1994.

[18] Richard S Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. *Advances in neural information processing systems*, 8, 1995.

[19] Alain Dutech, Timothy Edmunds, Jelle Kok, Michail Lagoudakis, Michael Littman, Martin Riedmiller, Bryan Russell, Bruno Scherrer, Richard Sutton, Stephan Timmer, et al. Reinforcement learning benchmarks and bake-offs ii. *Advances in Neural Information Processing Systems (NIPS)*, 17:6, 2005.

[20] Stephen R Carpenter, Donald Ludwig, and William A Brock. Management of eutrophication for lakes subject to potentially irreversible change. *Ecological applications*, 9(3):751–771, 1999.

[21] Ryan J Urbanowicz and Will N Browne. *Introduction to learning classifier systems*. Springer, 2017.

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

[23] John H Holland. Complex adaptive systems. *Daedalus*, 121(1):17–30, 1992.

[24] Chunming Liu, Xin Xu, and Dewen Hu. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398, 2015.

[25] Conor F Hayes, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. A practical guide to multi-objective reinforcement learning and planning. *Autonomous Agents and Multi-Agent Systems*, 36(1):26, 2022.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[27] Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

[28] Ester Bernadó-Mansilla, Xavier Llorà, and Ivan Traus. Multi-objective learning classifier systems. *Multi-Objective Machine Learning*, pages 261–288, 2006.

[29] Ryan J Urbanowicz, Randal S Olson, and Jason H Moore. Pareto inspired multi-objective rule fitness for noise-adaptive rule-based machine learning. In *Parallel Problem Solving from Nature–PPSN XIV: 14th International Conference, Edinburgh, UK, September 17-21, 2016, Proceedings 14*, pages 514–524. Springer, 2016.