Trustable Decision Support for Dynamic Applications

by

Dominic J. Duxbury

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

Manchester University

October, 2021

Prof. John Keane

Prof. Norman Paton

Contents

List of Figures

List of Tables

Abstract

Data stream management systems exist to support dynamic analysis of streaming data, often to inform decision-making. Decision Support Systems (DSSs) exist to enable decisions to be made that take into account user priorities. However, although these categories of system are now quite mature, there has been little work investigating their use together. Bringing these technologies together in a way that enables trustable decision support for dynamic applications is a difficult problem with particular impact in the military and medical domains. A framework has been proposed, comprising eight desiderata for trusted dynamic decision support. These desiderata aim to inform architects of dynamic DSSs on the implications of different capacities for decision support. An approach to dynamic decision support employing Genetic Algorithms (GAs) has been proposed. Two case studies have been utilised to show how this approach can be leveraged to provision DSSs with our desiderata. Weighted Product Model (WPM), Analytic Hierarchy Process (AHP), Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) and Preference Ranking Organization METHod for Enrichment Evaluation (PROMETHEE) have been assessed on the stability of results and the consistency of trade-offs, two of our desiderata. This assessment determined TOPSIS to be the method that is the most suitable for dynamic decision support. The problem of evaluating the effect of DSS features on trust has also been addressed and a theoretical framework modelling trust and its antecedents in a real-time DSS has been proposed. This model has then been used to carry out an assessment of the impact of explanation, preferences and dynamic updates as components of dynamic decision support, giving designers of DSSs an indication of which of these features are likely to have a positive impact on decision making in a dynamic environment. Finally, the research has concluded with the identification and discussion of potential areas for future investigation.

Declaration

- No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Copyright

Acknowledgements

First, I would like to thank my supervisors John Keane and Norman Paton for their support, guidance, and most importantly patience during my studies. They have both inspired me, and helped me to develop as a researcher during my time at the University of Manchester. If it was not for John, I would never have decided to start a PhD.

My friends have always been a major motivator, keeping me going through hard times. I'd like to thank Joe, John, Andy and Natalie for helping me to hold onto a vague sense of sanity throughout the last four years. I'd also like to thank Evan, Hardeep and Alessandro for keeping me company at Heald Place, for listening to me talk about boring and complicated topics, and for sitting quietly with me when there was nothing left to ramble about.

I'd also like to thank Charlotte, Benedict and my parents, Helen and Gerard, as I know without them, ceteris paribus, I would not have made it this far. I wholeheartedly believe that completing a PhD during the coronavirus pandemic would have been impossible without the incredible support of my friends, family and Nathaniel Kelly.

I would like to thank Simon Mettrick, Julian Winters and everyone at BAE Systems for their assistance over the course of the PhD and for their warm welcome during my internship in New Malden. Additionally, I would also like to thank BAE Systems for their funding and support as part of my ICASE studentship. Finally, I wish to acknowledge the support of the Engineering and Physical Science Research Council (EPSRC) for their part in funding my research.

1 Introduction

1.1 Motivation

Often, the problem of selecting the correct solution to a problem can be specific to a particular decision maker. An example of this is selecting the correct train journey to get to a destination. One train may be cheaper, whilst another is quicker. Which of these options is better depends on the priorities of the decision maker. Therefore, a DSS with the ability to elicit user preferences is required. Even for a task as simple as train journey planning, the list of requirements for a DSS can become challenging. Preferences must be elicited, synthesised to create a recommendation or ranking, and then presented in a way that is trusted and understood by the decision maker. For train journey planning, this is required in a context where there is uncertainty relating to criteria e.g. train arrival times. As a result, presenting a recommended journey to a user can become a difficult task.

Although DSSs can be built in any knowledge domain, it is an expensive operation to distil information from a system to assist a decision maker. Therefore, in low-stakes decision making, it is generally preferable to remove the human from the loop. As a result of this the majority of DSSs are employed in high-stakes domains, such as medical or military decision making.

Clinical DSSs are often employed in medical practice to help doctors perform accurate diagnosis. These systems can analyse multi-modal data to highlight risks and recommend treatments, whilst ultimately leaving the final decision to the doctor. In the military domain, DSSs can be

used to support command and control. Command and control is a set of organisational and technical attributes and processes employing human, physical, and information resources to solve problems and accomplish missions [140].

Command and control can be split into the planning and execution stages. Militaries have spent considerable effort on development of DSSs to support the planning stage. Comparatively, far less has been spent on assisting with execution [19]. The problem of assisting execution can be framed as a dynamic decision making problem and therefore requires a dynamic DSS. For dynamic decision support, trust is even more critical. The decision maker must make decisions under real-time constraints and therefore has less opportunity to verify or second guess the system. Therefore, to support the execution of missions a trusted dynamic DSS is required.

Maritime facilities face challenging demands, including monitoring ocean traffic, port safety, and emergency response. New technology is required to tackle these challenges, under the stresses of higher levels of traffic and an increased need for rigorous safety and prompt emergency response [104, 123].

Drones are a technology that has been identified for this role, and managing these drones for various tasks is one aspect of the expanding role of harbour management. Situational awareness has been highlighted as crucial in domains where the effects of ever-increasing technological and situational complexity on the human decision maker are a concern [95]. Drones provide a means for aerial situational awareness, within a harbour and beyond [48, 88]. One task, which has the potential to improve situational awareness, is the automatic identification of ships approaching a harbour. Drones can be employed to take photos of ships, for the identification of traffic and potential threats. To select an appropriate route, the decision maker must consider multiple conflicting objectives, such as identifying as many ships as possible, identifying ships as early as possible and reducing fuel costs. Navigating this large space of potential routes and making consistent trade-offs between objectives is a difficult task. Therefore, the management of such drones is a complex command and control problem that can potentially be simplified through the use

of MCDM methods, employed within a DSS. A further challenge for such a system is that ocean traffic is constantly moving and quickly changes direction. This necessitates that the problem be solved using a dynamic DSS, supporting the execution stage by updating routes as the scenario unfolds.

1.2 Decision Support

A DSS is a computer system designed to support users when making complex decisions. For a DSS, the choices made by decision makers often affect the state of the system. It is therefore useful to model decision makers as not just users, but as components of a Cyber-Physical-Social System (CPSS). CPSSs span the physical, information, cognitive and social domains. In the CPSS field, human users are considered a component of the system, falling within the cognitive domain [82]. Human components can be a necessary part of a system, such as when making the final judgement for life or death decisions. DSSs are therefore often vital, as they bridge the information and cognitive domains by distilling data to assist decision makers. Furthermore, DSSs also support information moving in the other direction by enabling the elicitation of knowledge from the user.

DSSs are enabled by decision analysis. Decision analysis is the field concerned with the study of complex decisions. Multi-Criteria Decision Making (MCDM) is a sub-discipline of decision analysis comprising techniques for evaluating solutions with multiple conflicting criteria [54]. A common example of this is purchasing a car; the safest car is not usually the cheapest and so these criteria are conflicting. For such problems, the presence of multiple objectives gives rise to a set of optimal solutions (known as Pareto-optimal solutions), rather than a single optimal solution. In the absence of information regarding the priorities of a user, it is impossible to say if any one of these Pareto-optimal solutions is better than any other. As a result, a vital part of the elicitation of knowledge from the user is understanding the user's priorities (or preferences) towards each objective. Pairwise Comparisons (PCs) are a common approach for this, used as

part of many MCDM methods [4, 16, 118–120, 150]. The strategy breaks down the problem of assigning ratio values to a set of objectives into manageable chunks. This is done by asking the user to determine their preference with respect to only two objectives at a time. Once a set of PCs is complete, one for each pair of objectives, a one-dimensional priority weighting vector can be derived. The PC methodology has been shown to outperform constraint-based approaches for preference elicitation [5].

1.3 Decision Support for Dynamic Applications

DSSs exist to support users in navigating a space of Pareto-optimal solutions [54]. Data streams exist as an abstraction to support analysis of dynamic data as it is produced [100]. These seem to be complimentary paradigms, which can be brought together to support *decision making with dynamic data*.

Current practice in stream data processing makes extensive use of Stream Processing Engines (SPEs) which provide a framework for acting upon elements in a stream. For decision support, an interesting challenge is how to build on these capabilities to support real-time decision support over streams. Dynamic decision support is necessitated by real-time MCDM problems; real-time problems require a response within specific time constraints. An MCDM problem can be described as real-time when it is affected by changing values of criteria, or changing sets of solutions to a problem. For the example of purchasing a car, this could mean the devaluation of a car over time or cars being removed from the marketplace. These problems require a prompt response, as the best solution is likely to change if the decision maker spends too long adjudicating.

1.3.1 Multi-objective Evolutionary Algorithms

As a result of the lack of a single optimal solution, multi-objective problems demand an approach to finding as many Pareto-optimal solutions as possible. Multi-Objective Evolutionary

Algorithms (MOEAs) have been proposed as a solution to this problem [30, 46, 62]. The primary reason for this is their ability to calculate multiple Pareto-optimal solutions in a single run [31]. The iterative nature of MOEAs also lends itself to real-time problems. The population (the intermediate set of solutions) can be updated each generation, in an attempt to maintain a set of Pareto-optimal solutions as the environment evolves. The result is a Dynamic Multi-Objective Evolutionary Algorithm (DMOEA) [148]. Research into DMOEAs is still in the early stages but has recently seen growing attention from the evolutionary computation community [154].

Muruganantham *et al.* [99] called for more benchmark problems, appropriate performance metrics and more efficient algorithms to further the research into DMOEAs. In their paper, they introduced the Kalman filter technique for DMOEAs. This approach uses predictions to help guide the search towards changed optima, as a means of accelerating convergence. To meet the growing trend of DMOEA research, Gee *et al.* [51] put forward a test suite. Their paper proposed a new dynamic test suite that allows researchers to assess the diversity maintenance and tracking ability of DMOEAs. Diversity is an important metric for DMOEAs, with high diversity allowing the algorithm to adapt more efficiently to a changing environment.

1.3.2 Frameworks for dynamic multi-criteria decision making

Whilst providing a means to produce and maintain a set of pareto-optimal solutions, DMOEAs alone do not solve the problem of dynamic decision support. This is due to two problems:

1. DMOEAs maintain a set of solutions rather than a single recommendation;
2. these sets of solutions are often highly unstable.

Together, this causes much difficulty when the decision maker is faced with selecting a single solution. Consequently, multiple frameworks for dynamic decision maker have been created to help guide the decision maker in a dynamic environment. To tackle the problem of selecting a

solution, Campanella *et al.* [22] introduced a framework for improving stability in dynamic decision making. Their approach takes into account the historical criteria values of alternatives to evaluate the appropriateness of a solution at decision-time. Their paper presents a number of aggregation methods for synthesising this historical data into a single criterion. The choice of aggregation depends on how the decision maker values the best-case value and the worst-case over the time period. To improve this approach, Zulueta *et al.* [156, 157] suggested a temporal factor for the selection process. This approach takes into consideration the rate and direction of change in criteria values as part of the aggregation. Another framework, produced by Yan *et al.* [149], proposes an alternative method for handling the differences in temporal behaviour of alternatives using grey numbers. A grey number is an abstraction that represents an indeterminate value that falls within an interval or a set of numbers [81]. These numbers can be "whitenized" to return a crisp value. This framework applies grey numbers as a means to aggregate the criteria values of alternatives across periods of time.

These methodologies aim to improve the stability of the rankings as the problem evolves, highlighting stability of results as a desired feature for dynamic DSS. An outstanding question is *what is the best way to evaluate stability and which methods can be employed to provide high stability of results?*

1.4 Trustable Decision Support

Decision analysis is often utilised to support decisions in medical and military fields. In these high-stakes decision making domains, experts are relied upon to make a final decision, supported by DSSs. Together they form a human-computer team, (ideally) performing better than either the human or computer alone [129, 133].

An important aspect underpinning the effectiveness of human-computer teams is *trust*. If a decision maker does not trust the system they are working with, then useful outputs can be

discarded or ignored. On the other hand, if a user has too much trust in a system this can lead to over-reliance [9, 78]. Therefore, an effective team requires a decision maker to be aware of what the system does and does not know. This highlights the importance of *interpretability* in DSS. Interpretability is the ability to explain or to present in understandable terms to a human [39]. This includes giving decision makers the ability to understand what aspects a system has taken into consideration, and how it has used these factors to arrive at a solution. We refer to this capability as *transparency* [146].

Transparency allows a decision maker to make use of their expert knowledge, supplemented by the systems' ability to process large amounts of data. For a system to be interpretable, it should provide enough information for its decision process to be understood, without overloading a user. Therefore, a system should be designed to be transparent, without inducing high *cognitive load*. Unfortunately, any features added to a system are likely to incur additional cognitive load, therefore when building DSS it is important to scrutinise the cost-benefit of features. We have identified decision maker preferences [113] and explanation [73] as two system features that should help improve transparency in DSSs.

Trust is especially challenging when working with dynamic data; a decision maker does not have time to ascertain if a black box system has made a mistake, and therefore it is highly beneficial to provide provenance data to the decision maker, ensuring that the information motivating a recommendation is readily available. Data provenance provides a historical record of data and its origins, which allows the user to trace and assess data quality and suitability. In addition to the underlying evidence, it is also important that the user has some understanding of the space of possible solutions; as a result, some form of explanation mechanism is required that explains how a recommendation has been arrived at, and/or describes the relationship between alternative options. Therefore provenance is another feature of DSS that has potential to improve transparency and hence increase trust. A latent construct is an idea which cannot be observed or measured directly; trust and transparency are two examples of latent constructs. As a result of their latency,

a difficult challenge in the field is evaluating how trust and its antecedents are affected by these decision support features.

1.5 Approach

The approach taken in this work involved compiling a list of desiderata or desired features/characteristics for trustable dynamic decision support. To explore and analyse our desiderata, we utilise two dynamic MCDM case studies: train journey planning and harbour management. A system is developed for each, and the harbour management scenario is used as part of a evaluation of MCDM methods and a user study to assess trustability.

1.5.1 Desiderata

A framework for trusted dynamic decision support has been developed, comprising a bundle of desiderata for DSSs. Through analysis of the literature and the train journey planning case study, we began with a set of 5 desiderata. The differentiating characteristic between an algorithm that generates a set of pareto-optimal solutions, and a DSS, is the ability to recommend a specific solution to a user. The ability to dynamically revise this solution, makes a DSS dynamic. Therefore, declarative specification of preferences and dynamic revision of recommendations are deemed the cornerstones of dynamic decision support. Declarative specification means that the preferences are expressed without describing how the DSS will interpret them [83]. This allows the system to elicit preferences in an intuitive fashion.

In addition to the provision of preferences and dynamic updates, data provenance has been identified as a feature that provides trustability. To support the underlying evidence, it is also important that the user has some understanding of the space of possible solutions; as a result, some form of explanation mechanism is required that makes explicit how a recommendation has been arrived at, and/or describes the relationship between alternative options.

All this is required in a context where there may be genuine uncertainty relating to criteria that inform a recommendation. As such, it is important for maintaining trust to ensure that the uncertainty intrinsic in a recommendation is either presented to a user or able to be reflected within the decision-making process.

Drawing this together, we arrive at the following desiderata for trusted dynamic multi-criteria DSSs:

- declarative specification of preferences,
- dynamic revision of recommendations,
- provenance capturing the data underpinning decisions,
- explanation of outputs, and
- explicit support for uncertain data.

Different methods can produce different rankings when applied to an identical problem, even with identical user preferences. For an MCDM problem, it is often impossible to say which resultant ranking is optimal [152]. Therefore, selecting an appropriate MCDM method for a problem is difficult. A solution to this is to pick some alternate desired characteristics for a method [72], for example, consistent trade-offs or high stability of results. Utilising our second case-study as a test-bed environment we revised our set of desiderata to include these desired characteristics. The revised set of desiderata for the harbour management task adds three desired characteristics of MCDM methods for dynamic decision support:

- high stability of results,
- high diversity of options,
- consistent trade-offs between criteria.

A desirable characteristic for dynamic decision support problems is high stability of results. In this work we refer to the propensity to reorder under changes to criteria values as the *stability* of a ranking. For a ranking to be functional, the frequency of change must be less than the time it takes for a decision maker to act. For this to be fulfilled, the rankings must take into account changing criteria values, without reordering significantly when small changes are made. Therefore, it is desirable for the rankings to be stable under small changes to criteria values.

We also noted that, it is impossible to capture every nuance of a problem within a DSS. Consequently, it is important that the expert is presented with a diverse array of options, rather than multiple similar solutions which may fall prey to similar pitfalls that have been overlooked by the system. If our diverse set of results still doesn't present the user with a suitable solution, the decision maker can alter their preferences.

To make effective use of preferences, it would be useful for changes in criteria weights to have predictable effects. An aspect underpinning the predictability of changes in criteria weights is the *consistency* of trade-offs between criteria. It is expected that as the weighting for a criterion increases, the trade-offs become more favoured towards that criterion. However, this relationship is not always predictable, as small changes in criteria weightings can lead to large changes in how an algorithm values certain trade-offs. We view consistency in trade-offs as a desired feature, as it gives rise to predictable effects when decision makers adjust their preferences. After expanding our framework with these points in mind, we arrived at the following set of revised desiderata as shown in Figure 1.1.

1. declarative specification of preferences,
2. dynamic revision of recommendations,
3. high stability of results,
4. explicit support for uncertain data,
5. explanation of outputs,
6. provenance capturing the data underpinning decisions,
7. high diversity of options, and
8. consistent trade offs between criteria.

Figure 1.1: The eight proposed desiderata for trusted dynamic multi-criteria DSSs

1.5.2 Train Journey Planning Case Study

The first case study considers an application relating to train journey planning. We assume that a user can state where they need to go *from* and *to*, along with the proposed start time. We also assume that the most suitable journey time for a user may depend on different criteria, specifically the *arrival time* of the journey, the *price* of the journey, and the *number of changes*. One such criterion, arrival time, indicates the expected arrival time of a journey. This is subject to change, as trains may be delayed or lines closed. Ticket prices are also subject to change until the time of purchase.

To investigate how our desiderata could be supported using SPEs, a dynamic DSS for train journey planning has been developed. The approach demonstrates how user preferences *(Desiderata 1)* can be combined with a continuously running genetic algorithm to provide a dynamically revised ranking of recommendations *(Desiderata 2)*. The system shows how the uncertainty in-

herent in train journeys can be captured and quantified to help guide decision makers *(Desiderata 4)*. The system also gives an explanation of the ranking of routes *(Desiderata 5)* and captures the provenance data detailing changes to the train schedule *(Desiderata 6)*.

1.5.3 Harbour Management Case Study

The second case study is a harbour management task. In this task, a decision maker takes the role of a harbour master managing a harbour. The harbour master controls a single drone to identify ships close to the harbour zone. The job of the user is to select a route for the drone, identifying ships before they reach the harbour zone. The length of these routes is limited by the fuel of the drone. Once the drone has run out of fuel, it must return to the refuel point, located within the harbour zone. We assume that the most suitable route may depend on different criteria: *the time spent by unidentified ships in the harbour*, *the average time between identification of a ship and its arrival in harbour* and *the amount of fuel used to identify each ship*.

To demonstrate how these desiderata can be supported, we outline a dynamic DSS. This system applies an MCDM method as a fitness function within a continuously running genetic algorithm. The system takes into consideration user preferences *(Desiderata 1)*, to generate a continuously updated ranking *(Desiderata 2)*, with a mechanism to control the diversity of options *(Desiderata 7)*.

This DSS is then used as a test-bed environment. This environment can be controlled through a User Interface (UI), to assess UI features of DSSs, or run in headless mode, to assess characteristics of MCDM methods. Using this test-bed environment to determine an appropriate method for dynamic DSSs, we evaluate MCDM methods with respect to the stability of their rankings *(Desiderata 3)* and the consistency of trade-offs between criteria *(Desiderata 8)*. The methods evaluated are the WPM [20], the AHP [118], the TOPSIS [150] and the PROMETHEE [16].

1.5.4 Harbour Management Trustability Study

Trust is a principal influence in the interactions between a user and a DSS. As a result, dynamic DSS architects should try to understand what characteristics of a DSS govern trust. What are the principal drivers of trust? What DSS features affect these drivers? We have carried out a study that investigates this in the context of the harbour management scenario by developing a trust-based model for interactions with real-time DSSs. We have validated this model by applying the Partial Least Squares Structural Equation Modeling (PLS-SEM) technique [147], providing empirical evidence that transparency is a strong determinant of a decision maker's trust and satisfaction with a system. This study also assessed the effect of explanation, preferences and dynamic updates on our model. To collect data for validation, we used our harbour management test-bed environment. Users of the system were provided with a random selection of interface features enabled/disabled. The users then completed a series of tasks before filling out a questionnaire. The features and questionnaire answers for experiment users were then compiled for analyses.

1.6 Aims and Objectives

The overarching aim of the project is to investigate the suitability of decision support features and methodologies for trusted dynamic DSSs. To achieve this we accomplished the following research objectives:

1. To identify and demonstrate desiderata for dynamic DSSs, through the development and analysis of a train journey planning application.

2. To show how to support and evaluate desiderata for dynamic DSSs through a test-bed based on harbour management.

3. To evaluate the suitability of WPM, AHP, TOPSIS and PROMETHEE for dynamic DSSs through the assessment of the stability of results and consistency of trade-offs for each method.

4. To assess the effect of enabling/disabling explanation, preferences and dynamic updates, through the production of a trust-based model for interactions with real-time DSSs, which was then validated using PLS-SEM.

1.7 Contributions

This thesis provides contributions in the field of dynamic decision support and the role of trust in governing interactions between a user and a dynamic DSS. The contributions are as follows:

1. A set of desiderata for dynamic decision support, along with examples of how they can surface in specific applications. The set of desiderata includes eight desired features/characteristics that form the basis of trustable dynamic DSSs.

2. A dynamic genetic algorithm that can be used to incrementally refine recommendations, with a specific emphasis on the production of diverse recommendations. This algorithm applies the principles of DMOEAs, combined with MCDM methods as a fitness function, to support the cornerstones of dynamic decision support.

3. An evaluation of MCDM methods in terms of our desiderata for dynamic decision support. WPM, AHP, TOPSIS and PROMETHEE were assessed on their ability to provide high stability of results and consistent trade-offs between objectives.

4. A theoretical framework modelling trust and its antecedents in a real-time DSS. To create this, we have applied a methodology for the assessment of DSS and their features in the absence of clear success criteria.

5. An assessment of the effect of explanation, preferences and dynamic updates on our model is also included. This gives designers of DSS an indication of which user interface features are likely to assist decision making in a dynamic environment.

1.8 Thesis Outline

The structure of the thesis and flow through the chapters is shown in Figure 1.2.

This chapter has introduced the concept of trustable dynamic decision support, motivated the work and outlined the aim and contributions of the thesis.

In Chapter 2, multi-criteria decision making methods are described along with background on the streaming methodologies employed to process dynamic data.

In Chapter 3, details are given for the rail journey planning case study and the set of desiderata designed and implemented with this study in mind.

In Chapter 4, a description is given of the harbour management case study and the revised set of desiderata for dynamic decision support. This chapter features an approach to real-time decision support driven by a dynamic genetic algorithm, along with a discount function designed to encourage diversity.

In Chapter 5, various MCDM methods are evaluated in relation to the revised set of desiderata. The consistency of trade-offs and the stability of results for each algorithm are considered to determine an appropriate method for dynamic decision making.

In Chapter 6, a methodology is outlined for the assessment of decision support features without a clear metric for success. The methodology applies PLS-SEM to a user case study to assess the impact of dynamic updates, explanation and pairwise preferences on trust and its antecedents.

Finally, Chapter 7 presents the conclusions from the research and identifies areas of potential future investigation.

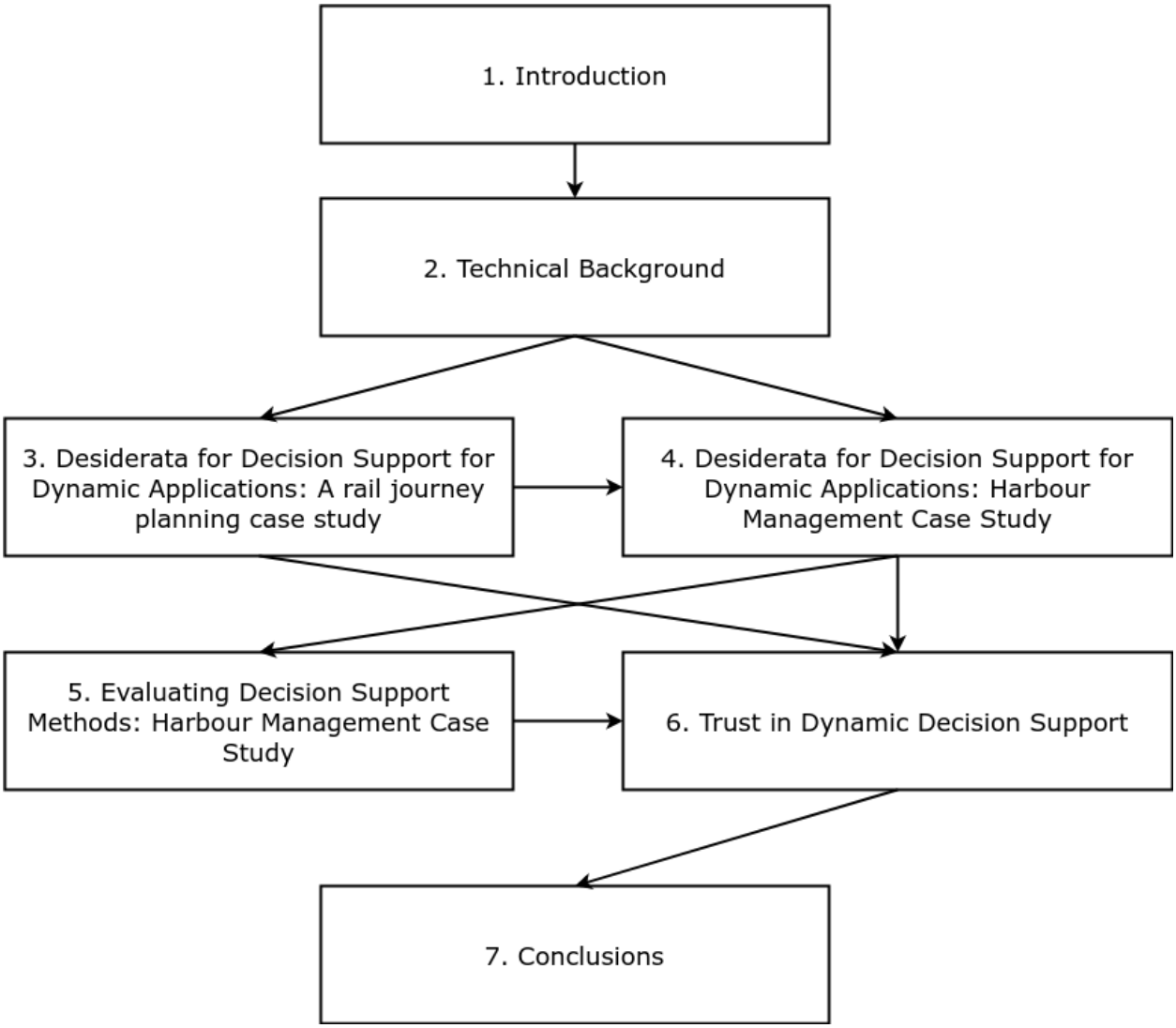


Figure 1.2: Overview of the thesis chapters.

2 Technical Background

In this chapter the concepts that are built upon and evaluated in our case studies are described. Firstly, the MCDM methods that are integrated into our dynamic DSSs are detailed. Secondly, the chapter introduces basic constructs of stream processing and describes how stream processing can be combined with MCDM methods to produce a dynamic DSS.

2.1 Multi-criteria Decision Making Methods

MCDM methods provide a methodology for synthesising a set of conflicting criteria relating to an overall goal, a set of alternatives which relate to each criterion, and an expression of a decision maker's preferences into a ranking. This ranking indicates how appropriate each alternative is for fulfilling the overall goal, ordered from most to least appropriate.

In this section we describe four of the predominant MCDM methods;

1. the Weighted Product Model (WPM);
2. the Analytic Hierarchy Process (AHP);
3. the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS);
4. the Preference Ranking Organization METHod for Enrichment Evaluation (PROMETHEE).

These four were chosen to represent the principal schools of MCDM methods; WPM typifies the early approaches to MCDM, AHP belongs to the category of value measurement models,

TOPSIS is a goal, aspiration and reference model and PROMETHEE exemplifies the French school of decision support. We explore the background and concepts behind each methodology, the advantages and disadvantages they give rise to, and present details of our implementation.

2.1.1 Weighted Product Model

2.1.1.1 Background

The WPM is a modification of the Weighted Sum Method (WSM) proposed by *P. Bridgman* in 1922. WSM is the earliest multi-dimensional decision making method [136]. WSM combines scores for criteria in a linear model. The criteria values are normalised, then multiplied by the weighting of the criterion and summed for each alternative. This sum represents the global score for the alternative. WPM overcomes some of the weaknesses of the WSM approach [20]. These weaknesses of WSM include rank reversal under different normalisation methods and rank reversal on removal of an alternative [132]. Such occurrences of rank reversal under WSM are caused by the interdependence between scores of alternatives incurred by normalisation. WPM raises weights as powers of the criteria value (positive powers for benefits and negative powers for costs), eliminating any units of measure.

The main benefit of the WPM approach is that the different units do not require normalisation [103]. As a result, WPM is often referred to as providing dimensionless analysis [135]. The lack of normalisation means that all scores for alternatives derived through WPM are independent and therefore the method suffers less from rank reversal. A drawback of the method is that it is required that the decision maker's preferences are encoded as a vector of weights, expressing the relative values of criteria. This process of expressing preference as a vector of weights is often non-intuitive to decision makers.

2.1.1.2 Implementation

WPM scores alternatives by a simple multiplicative model. The algorithm involves the following steps:

WPM Step 1. Weights are used to score each alternative using Equation 2.1. The criteria value of alternative a is represented as a_j for criterion j. The weighting for criterion j is given as w_j. For criteria we seek to minimise, we replace w_j with $-w_j$.

$$WPM(a) = \prod_{j=1}^{n} (a_j)^{w_j} \tag{2.1}$$

WPM Step 2. Rank alternatives according to the value of $WPM(a)$.

2.1.2 Analytic Hierarchy Process

2.1.2.1 Background

AHP was developed by *T. Saaty* in the 1970s as an alternative to these simplistic multi-dimensional models [118]. AHP is a structured technique for organising and analysing complex decisions. AHP consists of an overall goal, a group of options or alternatives for reaching the goal, and a group of factors or criteria that relate the alternatives to the goal. The decision maker's preference between two alternatives are quantified on a scale of 1 to 9, with 1 representing no preference of x over y through to 9 representing a strong preference of x over y. The same methodology can be used to compare criteria, resulting in a weighting vector expressing a decision maker's preferences. These judgements can then be used to synthesise an overall ranking of alternatives.

This approach proved popular; by 2008 there were more publications that reported application of AHP than any other MCDM method [142]. AHP fits into the category of value measurement

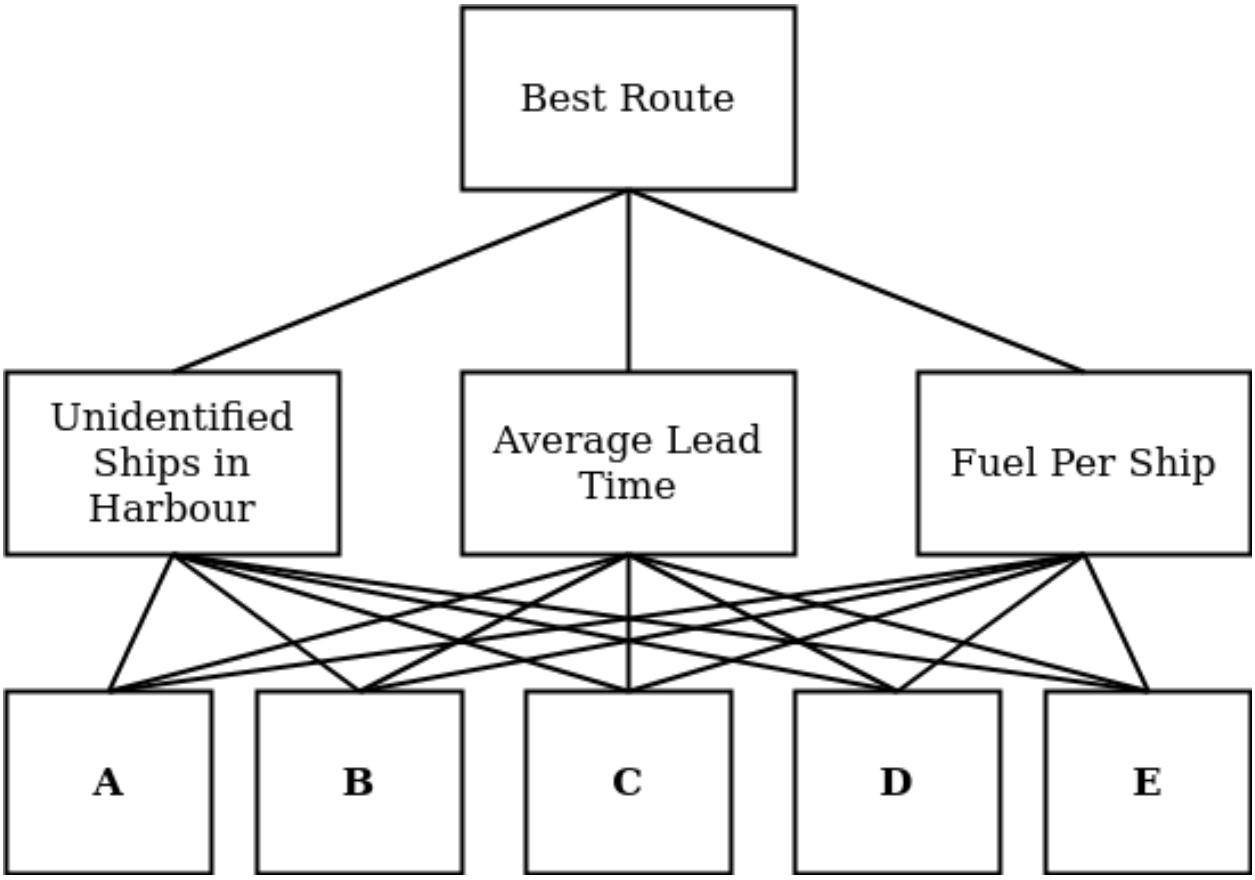


Figure 2.1: AHP hierarchy structure for the harbour management task; A-F represent the candidate routes

models, sometimes referred to as the American school of multi-criteria decision analysis [85].

2.1.2.2 Implementation

AHP is implemented in the following steps:

AHP Step 1. The problem is modelled as a hierarchy. The goal of the problem is at the highest level, with the criteria below it. These criteria can be divided further into sub-criteria, then at the lowest level we have the alternatives. Figure 2.1 shows the hierarchy for the harbour management task (discussed in detail in Chapter 5).

AHP Step 2. Criteria values are normalised according to the range of values across all alternatives using the formula given in Equation 2.2, where minX and maxX are the smallest and largest criteria values respectively.

$$Norm(x) = \frac{x - minX}{maxX - minX} \tag{2.2}$$

AHP Step 3. Priorities are established across the hierarchy by constructing a pairwise comparison matrix for each level. In our experiments, we evaluate a fixed set of weights and therefore it is only alternatives that require comparison. Each alternative has a value assigned according to each of the criteria, we refer to these as criteria values. To produce a ranking, criteria values must be scored. The normalised values are compared pairwise to generate a comparison matrix.

For three alternatives a_1, a_2 and a_3 and a criterion X with normalised criteria values x_1, x_2, x_3, we would generate a comparison matrix C.

$$\mathrm{C} = \begin{array}{c} \\ a_1 \\ a_2 \\ a_3 \end{array} \begin{array}{c} \begin{array}{ccc} a_1 & a_2 & a_3 \end{array} \\ \begin{bmatrix} 1 & f(x_1, x_2) & f(x_1, x_3) \\ f(x_2, x_1) & 1 & f(x_2, x_3) \\ f(x_3, x_1) & f(x_3, x_2) & 1 \end{bmatrix} \end{array}$$

Equation 2.3 is applied to compare criteria values. This formula maps two normalised values (x, y) to the fundamental scale proposed by Saaty [118].

$$f(x, y) = e^{x-y} \tag{2.3}$$

AHP Step 4. Comparison matrices generated through AHP have a concern with departure from consistency between judgements. When a matrix is inconsistent, the resultant vector of relative weights is viewed as untrustworthy. Therefore, the next step in AHP is to check the consistency of the matrix. In general practice, this is done by calculating the consistency ratio

(CR) and discarding matrices with a CR greater than 0.1 [127]. A matrix $C = (c_{ij})$ is termed *consistent* if:

$$c_{ij} \cdot c_{jk} = c_{ik} \; \forall \; i, j, k = 1, 2, ..., n \quad [10]$$

It is unnecessary to check consistency in our application of AHP as the comparison formula $f(x)$ produces comparison matrices that satisfy this equation for all values of i, j, k as shown below.

$$f(i,j) \cdot f(j,k) = f(i,k)$$

$$e^{\,i-j} \cdot e^{\,j-k} = e^{\,i-k}$$

$$e^{\,i-k} = e^{\,i-k}$$

AHP Step 5. The principal eigenvector P_j of the comparison matrix for each criterion j is calculated; this vector represents the priorities for each alternative. This priority vector P_j is then multiplied by the weighting w_j of each criterion j and summed to produce a global score vector G. For a problem with m alternatives and n criteria, the formula is given by Equation 2.4.

$$G = (g_i)_m = \sum_{j=1}^{n} P_j \cdot w_j \tag{2.4}$$

AHP Step 6. Rank alternatives according to their global score. For alternative i, the global score is entry g_i in G.

2.1.3 TOPSIS

2.1.3.1 Background

Another school of MCDM methods is comprised of goal, aspiration and reference models. These models formulate the problem as a comparison to a goal rather than assigning value directly from the criteria values.

The first such model, TOPSIS, was developed in the 1980s by *Hwang and Yoon* [150]. TOPSIS defines a positive ideal solution which represent the best alternative. This imaginary solution is created by collecting the best possible values across all criteria. The same is done with the worst criteria values to create a negative ideal solution. The best alternative is then determined by minimising the euclidean distance from the positive ideal solution and maximising the euclidean distance from the negative ideal solution. These distance metrics are used to score each alternative, to form an overall ranking of solutions. The distances along each dimension of the problem are scaled using a weighting vector. This weighting vector is an encoding of decision maker preferences. Unfortunately, TOPSIS provides no method to derive this vector from pairwise comparisons or any other natural expression of a decision maker's preferences. Consequently, this method is often combined with AHP, with AHP being utilised to create a weighting vector from a pairwise comparison of criteria.

2.1.3.2 Implementation

TOPSIS is described by the following steps:

TOPSIS Step 1. Create an evaluation matrix $(x_{ij})_{m\times n}$ consisting of m alternatives and n criteria, with the criteria values for each alternative i and criterion j given as x_{ij}.

TOPSIS Step 2. Calculate the normalised evaluation matrix $R = (r_{ij})_{m\times n}$ by applying the formula given in Equation 2.5.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{k=1}^{m} x_{kj}^2}}, \;\; i = 1, 2, \dots, m, \;\; j = 1, 2, \dots, n \tag{2.5}$$

TOPSIS Step 3. Calculate the weighted normalised decision matrix T = $(t_{ij})_{m\times n}$ by applying the formula from Equation 2.6.

$$t_{ij} = r_{ij} \cdot w_{ij}, \;\; i = 1, 2, \dots, m, \;\; j = 1, 2, \dots, n \tag{2.6}$$

TOPSIS Step 4. Compute the positive (A^+) and negative (A^-) ideal solutions. These serve as imaginary perfect and worst points in the solutions space, from which we can calculate the distance from real solutions as a form of evaluation.

$$A^+ = \{x_1^+, x_2^+, \dots, x_n^+\}$$

$$\text{where } x_j^+ = \{max(x_{ij}) \text{ if } j \in B; min(x_{ij}) \text{ if } j \in C)\}$$

$$A^- = \{x_1^-, x_2^-, \dots, x_n^-\}$$

$$\text{where } x_j^- = \{min(x_{ij}) \text{ if } j \in B; max(x_{ij}) \text{ if } j \in C)\}$$

where B is associated with benefit criteria (values we seek to maximise) and C with cost criteria (values we seek to minimise).

TOPSIS Step 5. Calculate the L^2-distance from positive ideal (d_i^+) and negative ideal (d_i^-)

solutions for each alternative.

$$d_i^+ = \sqrt{\sum_{j=1}^{n}(x_{ij} - x_j^+)^2}, \ \ i = 1, 2, \ldots, m$$

$$d_i^- = \sqrt{\sum_{j=1}^{n}(x_{ij} - x_j^-)^2}, \ \ i = 1, 2, \ldots, m$$

TOPSIS Step 6. Calculate the similarity to the worst condition for each alternative (s_i^-).

$$s_i^- = \frac{d_i^-}{d_i^- + d_i^+}, \ \ i = 1, 2, \ldots, m$$

TOPSIS Step 7. Rank the alternatives according to the similarity to the worst condition (s_i^-).

2.1.4 PROMETHEE

2.1.4.1 Background

The French school was founded by *B. Roy*, who produced the series of ELimination Et Choix Traduisant la REalité (ELECTRE) methods [116]. This served as inspiration for the family of outranking methods, characterised by the limited degree to which a disadvantage on one criterion may be compensated by advantages in another. PROMETHEE is an outranking method developed by *J.P Brans* [16]. PROMETHEE ranks a set of alternatives on the basis of several criteria by identifying pros and cons of the alternatives in a pairwise fashion. Criteria for MCDM problems can fall across a wide-range of scales, with different utility for similarly valued trade-offs. For example, when selecting a car, price may create an exponential range of values, whereas

horsepower falls across a linear range. This is caused by the net worth of individuals forming an exponential scale, whereas horsepower is limited by engineering. To capture these difference between criteria, PROMETHEE defines a preference function ($P(x)$) for each criterion.

The provision of $P(x)$ enables a more flexible approach to the comparison of criteria values. In a similar fashion to TOPSIS, PROMETHEE requires information on the relative importance of the criteria and is therefore often paired with AHP.

Multiple versions of PROMETHEE have been introduced [15, 17, 18]. PROMETHEE I produces a partial ranking, whereas PROMETHEE II computes a complete ranking of alternatives. In our case, a complete ranking of alternatives is required. Consequently, PROMETHEE II has been implemented as outlined below.

2.1.4.2 Implementation

In our work, PROMETHEE is defined by the following procedure:

PROMETHEE Step 1. Pairwise comparisons $d_j(x, y)$ are made between each criteria value x_{ij} for alternative i and criterion j using Equation 2.7.

$$d_j(x_{ij}, x_{kj}) = x_{ij} - x_{kj} \tag{2.7}$$

PROMETHEE Step 2. Unicriterion preference degree is calculated by applying a preference function $P(x)$ to the difference as shown in Equation 2.8.

$$\pi_k(x_{ij}, x_{kj}) = P[d_k(x_{ij}, x_{kj})] \tag{2.8}$$

This function can be different for each criterion. Six types of preference function are proposed; usual criterion, quasi criterion, criterion with linear preference, level criterion, V-shape with indifference criterion, and Gaussian criterion [15]. In this work, the criterion with linear

preference is applied, as shown in Equation 2.9.

$$P(x) = \begin{cases} 0 & x \leq 0 \\ x & 0 < d \leq 1 \\ 1 & d > 1 \end{cases} \tag{2.9}$$

PROMETHEE Step 3. A multi-criteria (global) preference degree $\pi(x, y)$ is computed to globally compare every pair of alternatives as shown in Equation 2.10.

$$\pi(x_{ij}, x_{kj}) = \sum_{k=1}^{q} \pi_k(x_{ij}, x_{kj}) \cdot w_k \tag{2.10}$$

PROMETHEE Step 4. Calculate the positive ($\phi^+(a)$) and negative ($\phi^-(a)$) preference flows for each alternative.

$$\phi^+(a) = \frac{1}{n-1} \sum_{x \in A} \pi(a, x)$$

$$\phi^-(a) = \frac{1}{n-1} \sum_{x \in A} \pi(x, a)$$

PROMETHEE Step 5. Calculate the net preference flow $\phi(a)$.

$$\phi(a) = \phi^+(a) - \phi^-(a)$$

PROMETHEE Step Six. Rank alternatives according to the net preference flow.

2.2 Stream Processing

A growing number of large-scale data processing use-cases involve data which is produced continuously over time. As a result, streaming analytics is a growing area of data science with

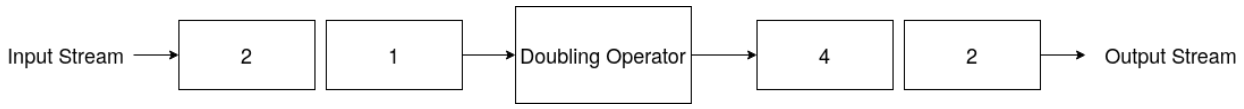


Figure 2.2: An example of an operator consuming a stream of random integers and doubling them to produce another stream.

considerable commercial interest. To handle these use-cases, a number of SPEs have been created, such as Apache Storm [23], Apache Spark Streaming [151] and Apache Flink [134]. These platforms offer an API to process data as it is produced, utilising the abstraction of data streams.

In this section we introduce the generic streaming concepts employed by SPEs. We also include background on the two SPEs used as components of our dynamic DSSs: Apache Storm and Apache Flink. Finally, we explain how we can build upon the extensibility points of SPEs to provide decision support over dynamic data.

2.2.1 Streaming Concepts

Data streams are an abstraction for modelling dynamic data. A data stream is an infinite sequence of elements. Elements are made up of a piece of data and a timestamp, indicating when an element was produced or when it was made available for processing. This could be the content of a tweet and the time it was sent, or an update to the price of a car and when it was updated.

Such data streams are manipulated through the application of operators. Operators are functions which consume zero or more streams to produce zero or more streams. A mapping is an operator which consumes a single stream to produce a single stream, with each element in the input stream corresponding to an element in the output stream. Figure 2.2 shows an example of a mapping consuming a stream of random integers and doubling them to produce another stream.

Windows are another common operator which are one of the core building blocks of streaming applications. Windows apply a function over a finite section of a data stream. For example, a window could be used to calculate the number of tweets made in the last 10 minutes. The most common forms of windows are time-based and count-based windows, bound by either time or

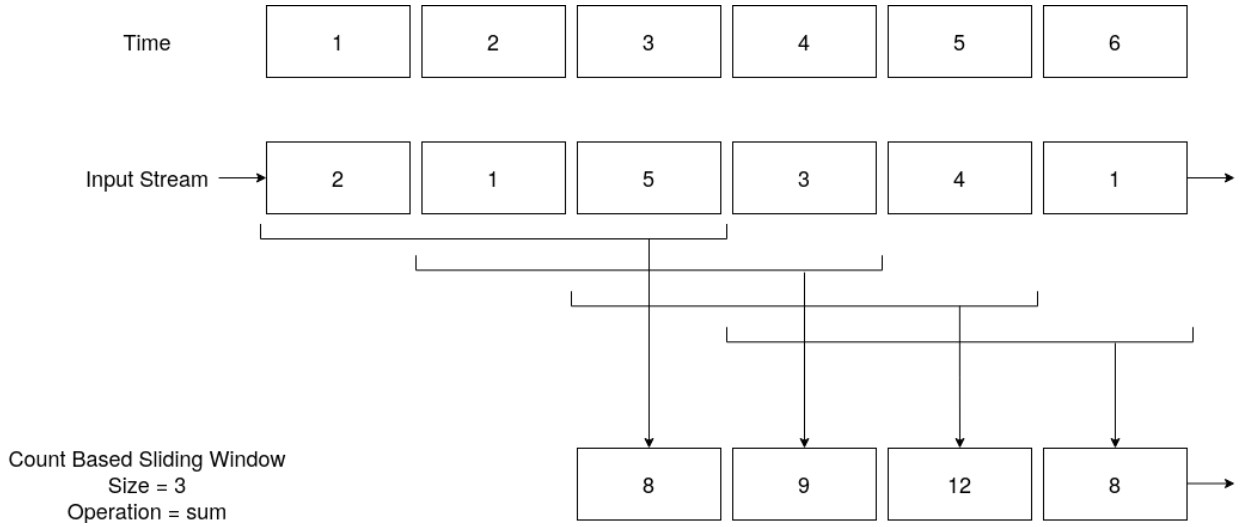


Figure 2.3: An example of a count-based sliding window (n=3) calculating the sum of a stream of random integers.

number of elements respectively.

Windows can further be distinguished as *sliding* windows, calculating an output with the arrival of each new element, or *tumbling* windows, which can accumulate multiple elements before a new calculation is performed [14]. Figure 2.3 shows an example of a count-based sliding window, calculating the sum over a stream of random integers.

2.3 Multi-Criteria Dynamic Genetic Algorithm

A GA is a meta-heuristic search algorithm modelled on the process of natural selection. GAs rely on mutate, crossover and selection operators to define the search process. New solutions are derived from a previous population through a combination of mutations and crossovers, with the highest fitness solutions selected for the next population [97].

In this work, we apply SPEs to solving dynamic multi-criteria decision making problems. We do this by using an incremental genetic algorithm utilising MCDM methods as a fitness function. Incremental genetic algorithms are designed to handle problems which undergo frequent minor modifications [89].

This approach builds upon the GA methodology to maintain a solution set ranked according to preferences, as the situation and therefore solutions, evolve over time. To apply this architecture to a dynamic multi-criteria decision making problem it is must be formulated as a combination of solutions, criteria, and state.

2.3.1 Formulating the problem

A solution (x) is an encoding of an alternative, a criterion (C) is a property of an alternative that contributes towards the goal and the state (S_t) is a snapshot of the problem which allows the criteria values for a solution to be calculated at a given time. For each criterion it is required that we define a function as follows:

$$\text{Let } x \text{ be a solution,}$$

$$\text{Let } S_t \text{ be the state at time } t,$$

$$x_{ct} = C(x, S_t)$$

This function calculates the value of x_{ct}, which is the value of the criterion C at time t for solution x. Together these data structures form an interface with the algorithm. For example, if the problem is selecting an appropriate train journey to a destination, the solutions are the potential journeys and the state is the information regarding line closures or delays. A journey duration criterion function could then be applied to calculate a predicted duration for each journey.

To implement this architecture, we must also define two of the operators for the genetic algorithm: mutate(x) and crossover(x, y). The mutation operator exists to maintain the diversity of the population. This function takes a solution and produces a slightly modified but new solution at random. Whereas the crossover operator combines two solutions together to produce a new solution, inheriting characteristics from both parents. The forms for mutate and crossover

functions are shown below:

$$\text{Let } x, y, z \text{ be solutions of the problem,}$$

$$y = \text{mutate}(x),$$

$$z = \text{crossover}(x)(y)$$

2.3.2 Architecture

Figure 2.4 shows the flow of data through the algorithm. The *Solution Creator* takes the current state of the scenario and the previous ranking, and applies mutate(x) and crossover(x,y) operations to create new solutions. For the first generation it creates a random set of solutions. The *Criteria Calculator* then calculates the criteria values for each solution (x) using the current state of the scenario (S_t) by applying criterion functions. This produces criteria values $x_{c_n t}$ for each solution (x) and each criterion (C_n), valid at time t.

A time-based sliding window is taken for all solutions valid at time t, closing when a solution marked with time $t + 1$ arrives. This window comprises the current generation of solution. The *Solution Ranker* uses this window as context to score each of the solutions validated at time t. This is done by applying one of the MCDM methods described in Section 2.1. Initial calculations for each algorithm are applied within the solution ranker, by producing scores for each criterion.

These intermediary values and criteria weightings are then passed to the *Fitness Calculator*. Within the *Fitness Calculator*, solutions are assigned a fitness value calculated using the MCDM method. When selecting solutions for the next generation of the algorithm, we first choose the solution with the highest fitness. We then apply a fitness penalty to solutions that share characteristics with the set of solutions chosen for the next generation. This is intended to promote a diverse set of solutions for recommendation. This set of solutions forms a ranking, ordered by

the fitness assigned to each solution. The ranking can be used as an input to the *Solution Creator*, acting as a basis for the next generation.

2.4 Conclusions

The decision support literature outlines a wide variety of MCDM methods. These methods apply different approaches to the problem of synthesising a ranking from a list of potential solutions and decision maker preferences. Generally, these methods are applied in a static context, with a fixed set of solutions and preferences. If the set of solutions and preferences are subject to change, then the problem is dynamic [22].

SPEs provide a set of tools for tackling dynamic problems. They handle dynamic data by utilising the abstraction of data streams. Acting upon these streams, SPEs employ a variety of streaming concepts to enable users to create dynamic applications. We have outlined an approach to solving dynamic multi-criteria decision making problems using SPEs. This approach applies streaming abstractions to build a multi-criteria dynamic genetic algorithm that maintains a ranking of solutions as both solutions and preferences change over time.

In later chapters we discuss this approach in the context of both a train journey planning case study and a harbour defence case study; further we evaluate the suitability of the outlined MCDM methods as a fitness function.

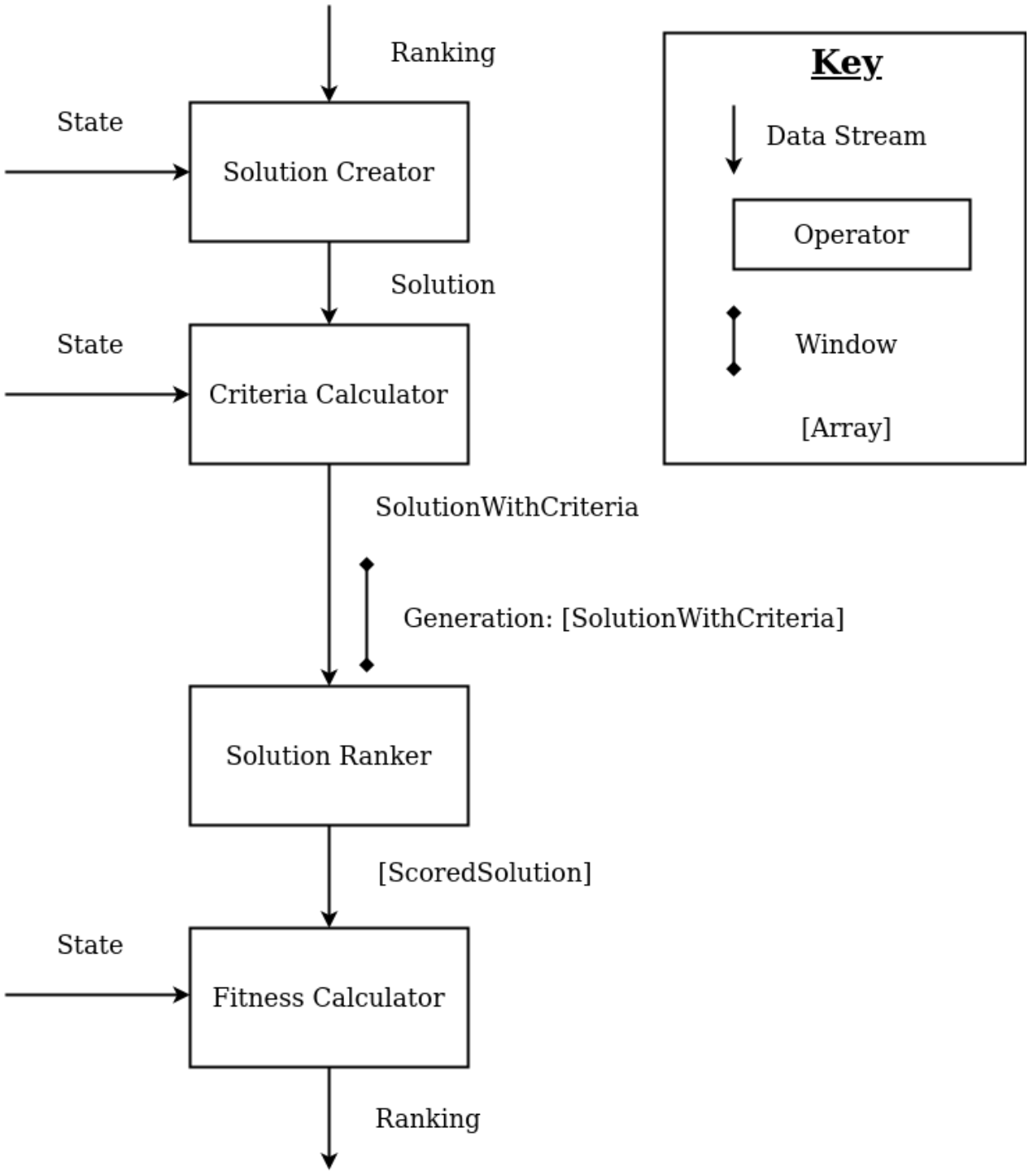


Figure 2.4: The architecture for the dynamic genetic algorithm

3 | Desiderata for Decision Support for Dynamic Applications: A rail journey planning case study

In this chapter, we identify some of the key issues and concepts in dynamic decision support. We explain how these give rise to our 8 desiderata for dynamic decision support. The literature review continues by giving examples of dynamic DSSs and outlining which of our desiderata are enabled. We then give a rail journey case study to illustrate how we can enable 5 of our desiderata, namely:

1. declarative specification of preferences,
2. dynamic revision of recommendations,
4. explicit support for uncertain data,
5. provenance capturing the data underpinning decisions, and
6. explanation of outputs.

The application helps a user to plan a journey between two stations according to the *arrival time*, *price* and the *number of changes*.

3.1 Related Work

In this section we introduce the concepts inspiring our list of desired features for dynamic decision support. We explain how these concepts inform our desiderata and we discuss methodologies that can enable the provision of our desiderata in dynamic DSSs. Finally, we give examples of dynamic DSSs and detail which desiderata are supported.

3.1.1 Concepts

In this subsection we describe dynamic decision making as a concept and the desiderata which fall out of the dynamic aspect of the problems. We then discuss the issues revolving around trust calibration and features which can assist in the process. Together, these sections give rise to our desiderata for dynamic decision support.

3.1.1.1 Dynamic Decision Making

As a result of the dimensionality of MCDM problems there is no single optimal solution, instead we define solutions as dominated or undominated. A solution is dominated by another solution if it performs worse under every criterion. A solution which is not dominated by any other is called pareto-optimal.

The set of pareto-optimal solutions is known as the Pareto-optimal front (POF). It is impossible to say which ranking of a POF is optimal [152]; the differentiating factor is the trade-offs between criteria, consequently to compare solutions within the POF we require information regarding the relative importance of criteria. These criteria can have different importance to different decision makers, therefore common decision analysis techniques provide methods to elicit user preferences. These preferences allow the decision maker to traverse the POF, guiding the process of selecting an appropriate solution.

For dynamic MCDM problems, it is not just the decision maker preferences that change, but

also the values of criteria and the set of solutions. As a result, the best solution is likely to change and it is important to dynamically revise the recommendations. Dynamic problems are subject to real-time constraints; a decision maker must select a solution as the problem continues to evolve and therefore, for a functional ranking, the frequency of change must be less than the time it takes to act. If a ranking of solutions completely reorders under small changes to the problem, it becomes difficult to make a decision in a timely manner. Campanella *et al.* [22] noted this problem and introduced a framework for improving stability in dynamic decision making. This approach provides stability by combining historical values for criteria through an aggregation function.

3.1.1.2 Calibrating Trust

Trustability is an important characteristic of the human-computer team comprising the DSS and the human decision maker. If a DSS is not trusted then the results can end up discarded and the team is only as capable as the human alone [9, 78]. Issues can also arise from a DSS being too trusted; common sense being overridden can lead to problems that would be avoided by a human decision maker.

Lack of trust can be especially detrimental in a dynamic environment, as the decision maker must act under real-time constraints. This means that more time deliberating over a decision may affect the outcome. Consequently, trust calibration is a vital aspect of any DSS.

An important aspect of trust calibration is communicating the level of uncertainty within a system [139]. Uncertainty is often divided into aleatoric and epistemic uncertainty; aleatoric uncertainty is representative of unknowns that differ each time we run an experiment (modelled probabilistically) whereas epistemic uncertainty results from a lack of knowledge [133, 145].

For DSSs, there is often both aleatoric and epistemic uncertainty relating to the criteria that inform a recommendation. As such, it is important to propagate both kinds of uncertainty to ensure that the uncertainty intrinsic in a recommendation is either presented to a user or able to

be reflected within the decision-making process.

Epistemic uncertainty is often difficult to quantify, and as a result several authors argue for the need to use verbal expressions (linguistic variables) for risk assessment [35, 64]. An example of this is using “trustworthy“ or “untrustworthy“ to represent the uncertainty underlying the reliability of a piece of information. Another method for dealing with epistemic uncertainty is understanding the source of information. Data provenance is a method for supporting this within information systems. Data provenance provides a historical record of data and its origins, which allows the user to assess data quality and suitability [21]. Understanding the source of information allows a decision maker to judge for themselves the limitations, and therefore epistemic uncertainty of the underlying data [69]. This allows a decision maker to judge the correct level of trust to give data within a system.

Tomsett *et al.* [133] proposed that for trust calibration, AI systems should communicate explanations for all outputs. For DSS, explanations complement the underlying evidence by allowing the user some understanding of the space of possible solutions; this means explaining how a recommendation has been arrived at, and/or describing the relationship between alternative options. Rudin *et al.* [117] noted that it is possible for machine learning training data to be flawed in unknown ways; this is also the case for the data underpinning DSSs. Provenance and explanation together can help to expose these flaws, allowing trust to be calibrated to an appropriate level.

When flaws are exposed in the recommended solution, it becomes the role of the decision maker to navigate the space of options to choose an appropriate alternative. It is impossible to avoid flaws in the underlying data and therefore it is imperative that DSSs provide capabilities to deal with them.

If all the solutions are very similar it is more likely that a single flaw could permeate the entire ranking. To solve this, Evans [43] argues that algorithms for supporting problem solving should generate a diverse set of alternatives. Following this logic, a diverse offering of solutions is a desired feature of DSSs [138]. Approaches to generating a diverse set of alternatives have

employed agent technology [137] and genetic algorithms [45].

If a diverse ranking is insufficient for selecting an appropriate solution the decision maker can navigate the space of options by altering their preferences. To make effective use of preferences, it is crucial for changes in criteria weights to have predictable effects. This helps to maintain consistency in the search, an important goal of any MCDM method [130]. Promoting predictability has been proposed as a foremost responsibility of leadership in project management [101]. For dynamic decision support, this becomes the responsibility of the system. An aspect underpinning the predictability of changes in criteria weights is the consistency of trade-offs between criteria. It is expected that as the weighting for a criterion increases, the trade-offs become more favoured towards that criterion. However, this relationship is not always predictable, as small changes in criteria weightings can lead to large changes in how an algorithm values certain trade-offs [25, 127]. We view consistency in trade-offs as a desired feature, as it gives rise to predictable effects when decision makers adjust their preferences.

3.1.2 Desiderata

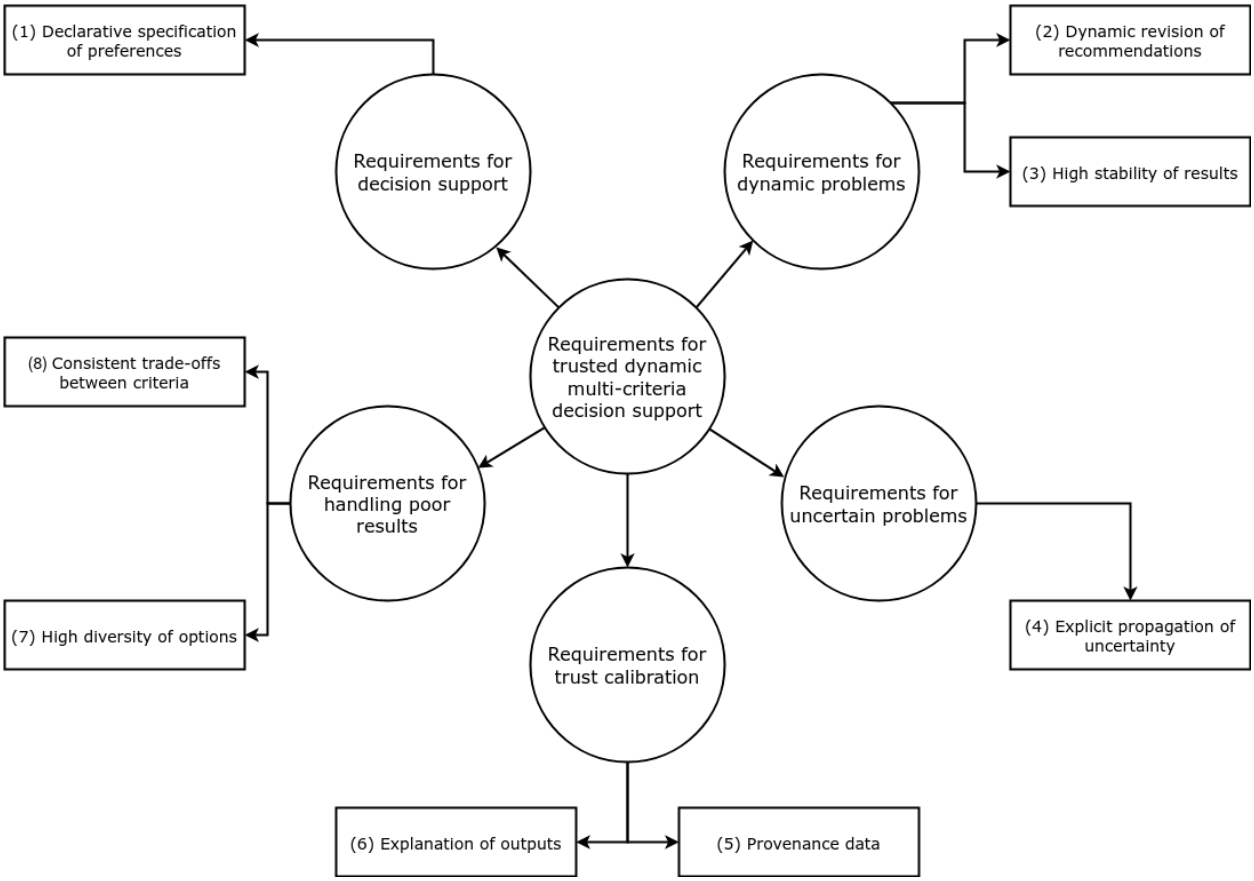


Figure 3.1: The eight proposed desiderata for trusted dynamic multi-criteria DSSs

In summary, to enable efficient decision making in a dynamic environment, in addition to the provision of preferences (*Desiderata 1)*, it is required for the system to dynamically revise recommendations (*Desiderata 2)* with a high stability of results (*Desiderata 3)*.

All this is required in a context where there may be genuine uncertainty relating to the criteria informing a recommendation. As such, it is important to ensure that the uncertainty intrinsic to a recommendation is either presented to a user or able to be reflected within the decision-making process (*Desiderata 4)*.

In an environment with dynamic uncertain data, calibrating trust is more important than ever.

Two features are put forward to enable the calibration of trust: data provenance (*Desiderata 5)* and explanation of outputs (*Desiderata 6).*

Once trust is properly calibrated, two desiderata are proposed to enable an efficient and intuitive search of the solution space. These are providing the users access to a high diversity of options (*Desiderata 7)* and the ability to change preferences, with consistent trade-offs between criteria (*Desiderata 8).* Drawing this together, we arrive at the 8 desiderata for trusted dynamic multi-criteria DSSs shown in Figure 3.1.

We can break our desiderata down into five categories of requirements: decision support, dynamic problems, uncertain problems, trust calibration and handling poor results. Declarative specification of preferences is a basic requirement for decision support, allowing a user to select the right solution from the POF. Dynamic revision of results and high stability of results allow this to be done in a dynamic environment.

For uncertain problems, it is important to propagate the uncertainty through to the decision maker, this is therefore a requirement for problems with uncertainty in the underlying data. An example of this would be choosing an appropriate train based on train times, it is important that the decision maker understands the uncertainty inherent to train arrivals. Provenance and explanation provide a means to calibrate trust. These desiderata allow the human decision maker to assess the evidence and reasoning behind a recommendation. The decision maker can then make an informed decision on whether to follow a recommendation. In the case that a recommendation is rejected, we are required to continue searching the solution space.

A high diversity of options and consistent trade-offs between criteria allow us to deal with poor results by finding alternatives efficiently.

These desiderata gives us a framework for comparing dynamic decision support systems but cannot definitively say that an individual system is good and trustable. Instead the framework gives us direction for improving dynamic decision support systems. Most systems will not fulfil all eight desiderata but it gives architects an idea of the effects of different features and char-

acteristics. For example, if a problem is uncertain we can improve the system by propagating uncertainty from the underlying data through the decision making logic. If the users are having difficulty calibrating trust, then we can add explanation or provenance data. If the users are struggling to handle poor results, then we can either improve the consistency of trade-offs or the diversity of the results. These examples show how such a framework is important for the production and improvement of dynamic decision support systems.

3.1.3 Methodologies

In this section we describe the literature surrounding the methodologies that allow us to enable our desiderata in dynamic DSSs. First we describe MOEAs, the methodology we employ to provide declarative specification of preferences with dynamic revision of recommendations. This methodology enables the dynamic revision of rankings for problems where the size of the solution space makes the reapplication of MCDM methods intractable.

3.1.3.1 Dynamic Multi-objective Evolutionary Algorithm

GAs have been widely applied to multi-objective optimisation problems, employing a heuristic search methodology to calculate the most appropriate solution from the solution space [53]. GAs applied to multi-criteria decision making are known as Multi-Objective Evolutionary Algorithms (MOEAs).

GAs have been highlighted as superior to conventional optimisation algorithms for multi-objective problems as a result of the following features [12, 13]:

1. GAs search with a population of candidate solutions rather than a single point. Thus, they are less likely to be trapped in a local optimum.

2. GAs use only the values of the payoff (objective function) information, and not the derivatives or other auxiliary knowledge.

3. GAs work with a coding (representation) of a parameter set not the parameters themselves. Thus the search method is naturally applicable for solving discrete and integer programming problems.

4. GAs use randomised parents selection and crossover from the old generation. Thus they efficiently explore the new combinations with the available knowledge to find a new generation with better fitness values.

Bingul *et al.* [12] identified these advantages of GAs and proposed a genetic algorithm for real-time multi-objective problems. The approach was applied to a war resource allocation problem, controlling the *blue* side allocations with four criteria to optimise: minimise the territory the *blue* side loses, minimise the *blue* side aircraft lost, maximise the number of *red* side strategic targets killed and maximise the number of *red* side armour killed. The GA had three possible fitness functions, shown in Equations 3.1, 3.2 and 3.3 respectively.

$$F_1 = f_1^4 + f_2^3 + f_3^2 + f_4 \tag{3.1}$$

Where F_1 is a fitness score, f_1 is the smallest criteria value, f_2 is the second smallest criteria value, f_3 is the third smallest criteria value, and f_4 is the largest criteria value.

$$F_2 = f_1^2 + f_2^2 + f_3^2 + f_4^2 \tag{3.2}$$

Where F_2 is a fitness score and f_1, f_2, f_3 and f_4 are the MOEAs criteria values.

$$F_3 = E_{max} - \sum_{i=1}^{4} (f_{max} - f_i)^2 \tag{3.3}$$

Where E_{max} (16 for this case) is the maximum value of total fitness score and f_{max} (2 for this case) is maximum value of each fitness score.

These fitness functions were applied in a GA with a fixed population size of 50, a crossover probability of 0.7 and a mutation probability of 0.02.

In a later paper [13], Bingul *et al.* introduced an adaptive system for setting the mutation and crossover rates. This change was designed to improve the convergence speed and stability of the results. To do this, the authors use the mean, variance and the best fitness value of each generation to set the mutation and crossover rates going forward. The results found that the adaptive GA had a higher rate of convergence and converged to a higher fitness value. Convergence rate is an important characteristic that underpins the stability of resultant rankings as a slow convergence can unsettle rankings as the scenario evolves over time.

Following similar reasoning, Deb *et al.* [31] introduced the Non-dominated Sorting Genetic Algorithm II (NSGA-II), an elitist genetic algorithm for multi-objective optimisation. This algorithm is often used as a baseline for comparing new MOEAs [51]. The algorithm employs a fast non-dominated sorting approach with $O(MN^2)$ computational complexity, where M is the number of objectives and N is the population size. The elitist selection operator creates a new generation by combining the parent and offspring populations and selecting the best N solutions. To do this, the solutions are first sorted into dominated and non-dominated (pareto-optimal) solutions. These non-dominated solutions form the first level non-dominated front. These solutions are then removed and the process is repeated to find the second level non-dominated front. For each solution, this hierarchy is used to calculate two entities: the domination count (the number of solutions dominating a solution) and the set of solutions dominated by a solution. The fronts are then integrated into the next generation, beginning with the first level non-dominated front. The final front is then sorted using the crowded-comparison operator, that preserves diversity by selecting solutions in a less crowded region of the solution space.

This approach approximates a pareto-optimal front efficiently, providing a means for dynamic revision, but integration with an MCDM method is required to solve the problem of recommending a single solution to the user.

Example	Preferences	Dynamic Revision	Stability	Uncertainty	Provenance	Explanation	Diversity	Trade-offs
1	✓	✓	×	✓	✓	✓	×	×
2	✓	✓	✓	×	×	✓	✓	✓
3	✓	✓	×	×	×	×	×	✓
4	✓	✓	×	×	×	×	×	✓
5	✓	✓	✓	×	×	×	×	×
6	✓	×	×	×	×	×	×	×

Table 3.1: The fulfilment of our desiderata in a bundle of dynamic decision support case studies; 1 - Train journey planning DSS, 2 - Harbour management DSS, 3 - A dynamic decision support system for evaluating peer-to-peer rental accommodations in the sharing economy [131], 4 - A dynamic decision support system for sustainable supplier selection in circular economy [6], 5 - A knowledge based system for supporting sustainable industrial management in a clothes manufacturing company based on a data fusion model [141], 6 - Urbanization suitability maps: a dynamic spatial decision support system for sustainable land use [24].

3.1.4 Examples

In this section we outline some recent dynamic decision support case studies from the literature and highlight how our desiderata surface within each system.

Tavana *et al.* [131] designed a dynamic DSS for evaluating peer-to-peer rental accommodations in the sharing economy. This system compares rental accommodation based upon 28 different criteria divided into 5 categories of evaluation, property, neighbourhood, economic and distance factors. The system applies TOPSIS over dynamic data to enable the provision of preferences in a dynamic environment. The system does not automatically revise results but allows for recalculation on demand. This system is applied in a context with no uncertain data and as the results are not revised repeatedly, stability of results is not an issue that has been addressed. The system also addresses a problem with a small number of solutions and therefore promoting a high diversity of options is not a priority.

Alavi *et al.* [6] presented a similar dynamic DSS for sustainable supplier selection in circular supply chains. This system also considered a large number of criteria and therefore chose to apply the Best Worst Method (BWM). This method is similar to AHP but focuses on reducing the number of pairwise comparisons needed by deducing the values of missing comparisons,

reducing the number of pairwise comparisons required from $\frac{n(n-1)}{2}$ in AHP to $2n-3$. The system enables dynamic revision through the re-application of BWM over dynamic data. The authors also investigate the consistency of the trade-offs by applying sensitivity analysis to assess the robustness of rankings under changes to the weightings of criteria, concluding that the rankings achieve a reasonable level of sensitivity. In our framework, we ignore the sensitivity of trade-offs in criteria in favour of a focus on consistency (*Desiderata 8).*

Vieira *et al.* [141] delineated a knowledge based system for supporting sustainable industrial management in a clothes manufacturing company based on a data fusion model. They apply a D-MCDM model proposed by Campanella *et al.* [22]. The model uses an aggregation function to synthesise historical and predicted future criteria values to create a ranking which aims to achieve a high stability of results. The model also allows for the specification of user preferences and is recalculated as the situation unfolds as a means for dynamic revision. Overall, in the literature most dynamic DSSs aim to fulfil a handful of our desiderata but fail to adequately address all our desired features and characteristics for dynamic decision support.

Cerreta *et al.* [24] outlined a dynamic spatial DSS for guiding and managing sustainable land use. The system applies AHP to select suitable land for consumption whilst minimising environmental impacts of spatial planning. Through the AHP, decision makers can express their preferences across a hierarchy of factors, such as geomorphology or the natural resources and ecological network. Their approach is not dynamic in an integrated temporal sense but instead models land consumption as a dynamic process, predicting the scenarios that might result from the implementation of city planning strategies. The possible outcome is predicted but the characteristics of the uncertainty are hidden and not propagated to the decision maker.

3.2 Rail Journey Case Study

This section introduces a rail journey case study, which is used to give illustrate how a DSS can be built which supports 5 of our desiderata, namely;

1. declarative specification of preferences,
2. dynamic revision of recommendations,
4. explicit support for uncertain data,
5. data provenance, and
6. explanation of outputs.

Further details of how we enable and evaluate (3) *high stability of results* , (7) *high diversity of options* and (8) *consistent trade-offs between criteria* are given in Chapter 5.

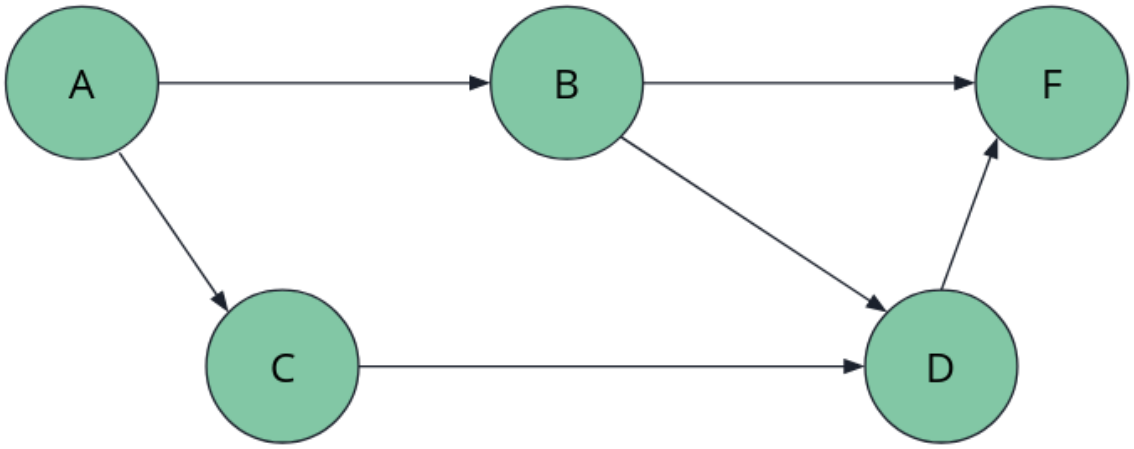
3.2.1 Motivating Example

To illustrate multi-criteria decision support over streams, we consider an application relating to train journey planning. We assume that a user can state where they need to go *from* and *to*, along with the proposed start time. We also assume that the most suitable journey time for a user may depend on different criteria, specifically the *arrival time* of the journey, the *price* of the journey, and the *number of changes*. For example, in Figure 3.2, a decision maker must choose a route from A to F in a way that takes into account price, arrival time and number of changes.

Solution	Price (£)	Changes	Arrival Time
ABF	15	1	14:00
ABDF	16	2	14:00
ACDF	9	2	14:40

Table 3.2: The solutions to figure 3.2.

Figure 3.2: Example Train Routing Scenario



Circles and arrows depict stations and trains respectively.

Table 3.2 shows the solutions to this example. We note that the solution *ABF* dominates *ABDF* as it is equal or better for all criteria values. This leaves us with two potential solutions; *ABF* and *ACDF*. A business person may prefer *ABF* because it is quicker, whereas a student may prefer to save money and take *ACDF*. There is no optimal solution for everyone and so we require user specification of criteria preferences *(Desiderata 1)*.

One such criterion, arrival time, indicates the expected arrival time of a journey. This is subject to change, as trains may be delayed or lines closed. Ticket prices are also subject to change up until the time of purchase. If a train is delayed or the price increases, the resulting solution may no longer be optimal, therefore dynamically revising recommendations *(Desiderata 2)* to reflect the most recent information is clearly beneficial. The user may also move between stations as a part of their interaction with the system, hence requiring an entirely new set of solutions. A decision maker may see these solutions and choose option *ACDF* because they believe it will only take 10 minutes. However, this route may be unreliable due to engineering works, so it may be important for the user to understand the source and derivation of criteria values *(Desiderata 3)*

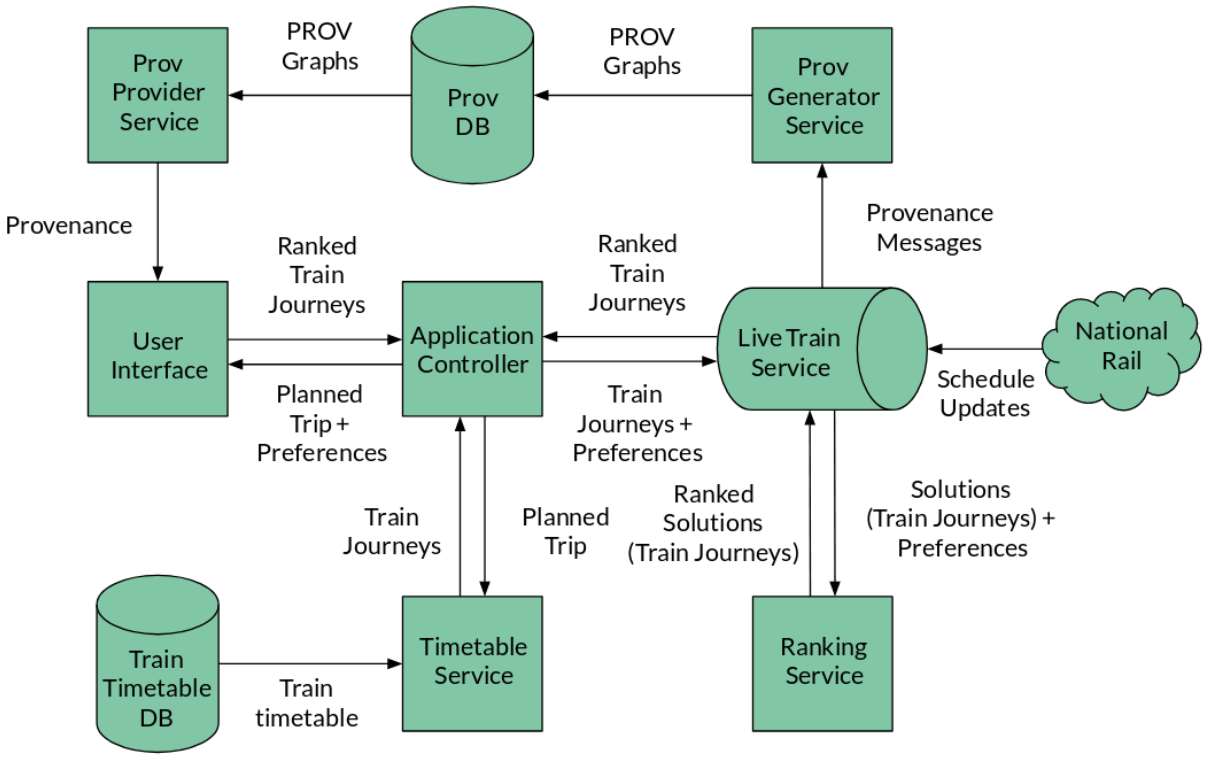


Figure 3.3: Prototype Architecture; PROV is defined in Subsection 3.2.3

to improve trustability, or to understand the uncertainty that is characteristic of this particular train service *(Desiderata 5)*.

Finally, after expressing their preferences, accepting criteria values and understanding uncertain aspects, a user is left with a recommended journey. It may be difficult to trust this recommendation without understanding why it was selected. Therefore we should provide the user with an explanation of where the recommendation falls in the solution space, so that they can understand the trade-offs being made, and how this ties into their preferences for criteria *(Desiderata 4)*.

3.2.2 Architecture

To evaluate our approach, a prototype platform has been developed. This platform implements our desiderata from Section 1.5.2, whilst providing decision support for train route plan-

ning. The system utilises a micro-services architecture shown in Figure 3.3.

The decision maker operates the DSS through the user interface. The user inputs details for a planned trip; an origin station, a destination station and a departure time. The user also must specify their preferences with regard to the criteria. This information is sent with a request to open a web-sockets connection to the *Application Controller*. The *Application Controller* holds the state of the train journeys (solutions) within the system. The controller uses the planned trip to build a http request to send to the *Timetable Service*.

Our architecture requires a solution service to generate the initial solution space. The *Timetable Service* is the implementation of the solution service for the train route planning scenario. The service generates a list of train journeys between the requested origin and destination stations at the specified departure time. Initial values are then calculated for all criteria. The *Timetable Service* returns an unranked list of train journeys which are passed from the *Application Controller* to the *Live Train Service*. A streaming component is also required to update the dynamic criteria and to produce a new ranking in real-time. The *Live Train Service* is an implementation of this component for the train scenario. In this case the live train service must update the expected train arrival time. The *Live Train Service* is initialised with a list of train journeys, which are ranked by the *Ranking Service*. A stream of UK-wide train updates from *National Rail* is filtered, and matching updates are used to update criteria values. The updated list of train journeys is then re-ranked by the *Ranking Service*. The output stream of ranked train journeys is communicated to the *User Interface* over web-sockets.

The *Ranking Service* accepts a specification of preferences and a list of solutions, to produce a ranking. This ranking is calculated through the application of the AHP.

The criteria and criteria behaviour are specified through the configuration. For example, we specify that price is a criterion and should be minimised. This allows the service to remain generic. The other generic component is the provenance sub-system. The provenance sub-system generates, stores and serves provenance data within the platform. This subsystem is made up of a

Operator	Input	Output
NationalRailSpout	N/A	<timestamp :: Timestamp, id :: trainID, destination :: String, newExpectedArrival :: Timestamp>
DelayBolt	NationalRailSpout	<timestamp :: Timestamp, journeys :: [Journey]>
RankingBolt	DelayBolt	<timestamp :: Timestamp, rankedJourneys :: [<score :: Double, journey :: Journey>] >

Table 3.3: Input and Output types for each operator

message queue, a database (*Prov DB*) and two services; one for generating provenance (*Prov Generator Service*) and one for serving it (*Prov Provider Service*). The sub-system receives messages from the streaming service which are processed to produce provenance graphs.

3.2.3 Architecture Components

In this subsection, we provide further details of the components in Figure 3.3.

Live Train Service

SPEs are programming frameworks designed to enable the intuitive manipulation of streaming data. The live train service makes use of Apache Storm to transform streams of live train updates. Apache Storm is an open source SPE that utilises three abstractions: spouts, bolts and topologies. Spouts produce streams; bolts consume any number of streams to produce new output streams; and a topology describes a network of spouts and bolts. Within our streaming component we instrument these operators to extract provenance data.

We extend the base classes for bolts and spouts to produce two new provenance aware classes: ProvenanceAwareBolt and ProvenanceAwareSpout. An example of a bolt extending this class is shown in Listing 3.1. *Execute* defines how a bolt processes each tuple and *declareOutputFields* declares the shape of tuples in the output stream. An operator inheriting from these classes will write provenance information concerning its inputs and outputs to the provenance sub-system.

For the train route scenario we have three operators: *NationalRailSpout*, *DelayBolt* and *RankingBolt* . The *NationalRailSpout* produces a stream of delays; the *DelayBolt* applies relevant delays to a list of journeys; and the *RankingBolt* interfaces with the *Ranking Service* to calculate a score

for each journey. Table 3.3 shows the input and output tuples for each operator. We instrument all the operators to supply us with provenance regarding the history of solutions, their criteria values and the resulting ranking.

```
public class ExampleBolt extends ProvenanceAwareBolt {
  public void execute(Tuple tuple) {}
  public void declareOutputFields(Declarer declarer) {}
}
```

Listing 3.1: Code for a provenance aware bolt

Ranking Service

To calculate a recommendation we apply the AHP [118], using pairwise comparisons between criteria to generate weightings. The details for this process are given in Subsection 2.1.2. We applied the methodology as outlined with two potential formulas for comparing criteria values under consideration, depending on whether the values fall along a linear scale (3.4) or an exponential scale (3.5). These formulas map two normalised values (x, y) to the fundamental scale proposed by Saaty [118]. For the train route planning scenario, we apply the first formula (3.4), because all criteria form a linear scale. For example, train prices might be £10, £15, £20 for three alternative routes and not £10, £100, £1000.

$$f(x, y) = |(x - y) \times 8| + 1 \qquad (3.4) \qquad\qquad f(x, y) = \frac{e^x}{e^y} \qquad (3.5)$$

The ranking service operates over web-sockets. The service requires a configuration file when a connection is opened, providing information about criteria. Critically, the configuration indicates the number of criteria and whether numerical criteria should be maximised or minimised.

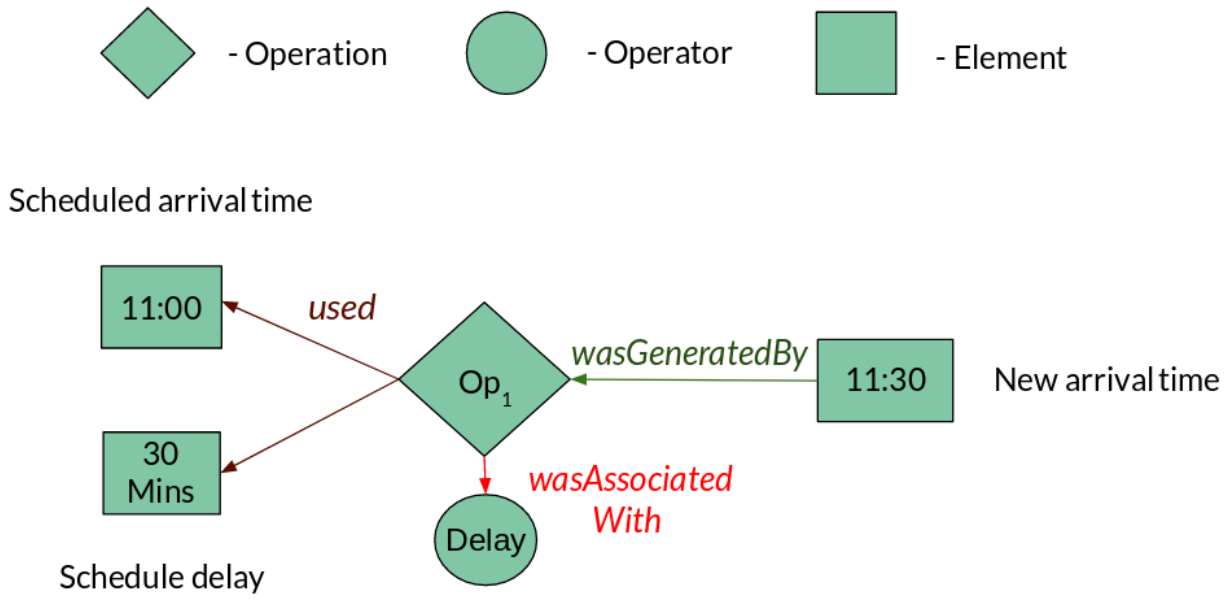


Figure 3.4: Provenance graph for a train schedule update

The configuration also allows us to indicate how we should compare non-numerical criteria. Once a connection is opened, AHP is applied to a stream of solutions, producing a stream of rankings.

Provenance Sub-system

The provenance sub-system processes messages from the streaming system and stores the output in a database for future querying. To store this data we choose to conform to the PROV standard [98]. PROV defines a data model consisting of a set of vertices and edges for modelling provenance as graphs. We adapt a subset of these to map to concepts from data stream analysis. For vertices we use entities, activities and agents. For edges we use *wasGeneratedBy*, *used* and *wasAssociatedWith*. The PROV data model describes entities as "an immutable piece of state", activities as "dynamic aspects of the world which produce entities" and agents as "parties which take a role in activities". We model stream elements as entities; stream operations as activities; and stream operators as agents. Note, we call a set of inputs and outputs a stream operation. The stream operator refers to the operator applied to these inputs to produce the outputs.

Edges describe the relationships between two entities. *wasGeneratedBy* links an entity to the activity which generated it. *used* links an activity to an entity it consumed. *wasAssociatedWith* links an activity to an agent associated with it. We say a stream element was generated by a stream operation. These operations *used* a stream element or window of elements. The operation also *wasAssociatedWith* the operator which was applied.

An example provenance graph is shown in Figure 3.4. This example shows the derivation for an expected train arrival time. The new *arrival time wasGeneratedBy* an operation which *used* the *scheduled arrival time* and the *schedule delay*. The operation *wasAssociatedWith* the delay operator (*DelayBolt*).

3.2.4 Framework Concepts

In the remainder of this section, we explain what we mean by *explanation* and *uncertainty* and how these concepts surface within our architecture.

Explanation

The AHP algorithm outputs a weight vector for criteria and a score for each solution. Whilst this is useful for constructing a ranking, these values are difficult for a human to interpret. Therefore we require some further explanation of how the system arrived at a recommendation. Fundamentally we describe explanation as a description of how a set of criteria preferences are used by AHP to select a solution from a solution space. Perhaps the most important aspect is an explanation of the trade-offs and benefits of a recommendation and how this ties into the specified user preferences. For instance, in the case of train route planning, a user could specify that price is critical to them. Assuming the system recommends *ABC*, the cheapest option, a simple explanation would be that *ABC* is the cheapest train and price is the most important criterion.

Our recommendations are dynamic and so it is important that an explanation can be processed by the user quickly. This lead us towards visual forms of explanation such as bar and spi-

der charts. Spider charts visualise multi-variate data as a shape constructed from three or more quantitative variables across axes stemming from the same point. Typically a chart with a larger area represents a better solution, but these charts can be misleading as the order of criteria can greatly affect the area. For this reason we chose instead to visualise the solution space through bar charts where the values for each criterion and solution are plotted side-by-side. Bar charts are one of the most simple forms of data visualisation, leaving less room for misinterpretation.

Uncertainty

Uncertainty is modelled using Cumulative Distribution Functions (CDFs) drawn from historical data. These functions capture information regarding the potential values of an uncertain criterion for a particular solution. Arrival time is an uncertain criterion for train route planning. We derive a CDF of arrival times for a journey from the historical performance of the trains travelling the same route. Such CDFs are a simple model, capturing the distribution of potential criteria values. Through this distribution we can view the probability of the potential risks (lateness) for a journey. CDFs serve as alternatives to criteria values for uncertain criteria but we require a method of comparing two CDFs. To do this we extract three key values from the distribution; optimistic, expected and pessimistic values. For a CDF f, we define optimistic, expected and pessimistic values as x such that $f(x) = 0.05$, $f(x) = 0.5$ and $f(x) = 0.95$ respectively. An example for train arrival times is shown in Figure 3.5. The user interface allows the decision maker to toggle which of these three values is fed into the ranking algorithm.

3.2.5 Motivating Example Application

In this section we explain how the user interacts with the system and how this interface supports the desiderata from Section 3.1.2. The user interface aims to target end-users, rather than decision scientists [128]. The user interface for the train route planner is shown in Figure 3.6.

For a decision maker planning a train journey, the first task is to specify the planned trip. The

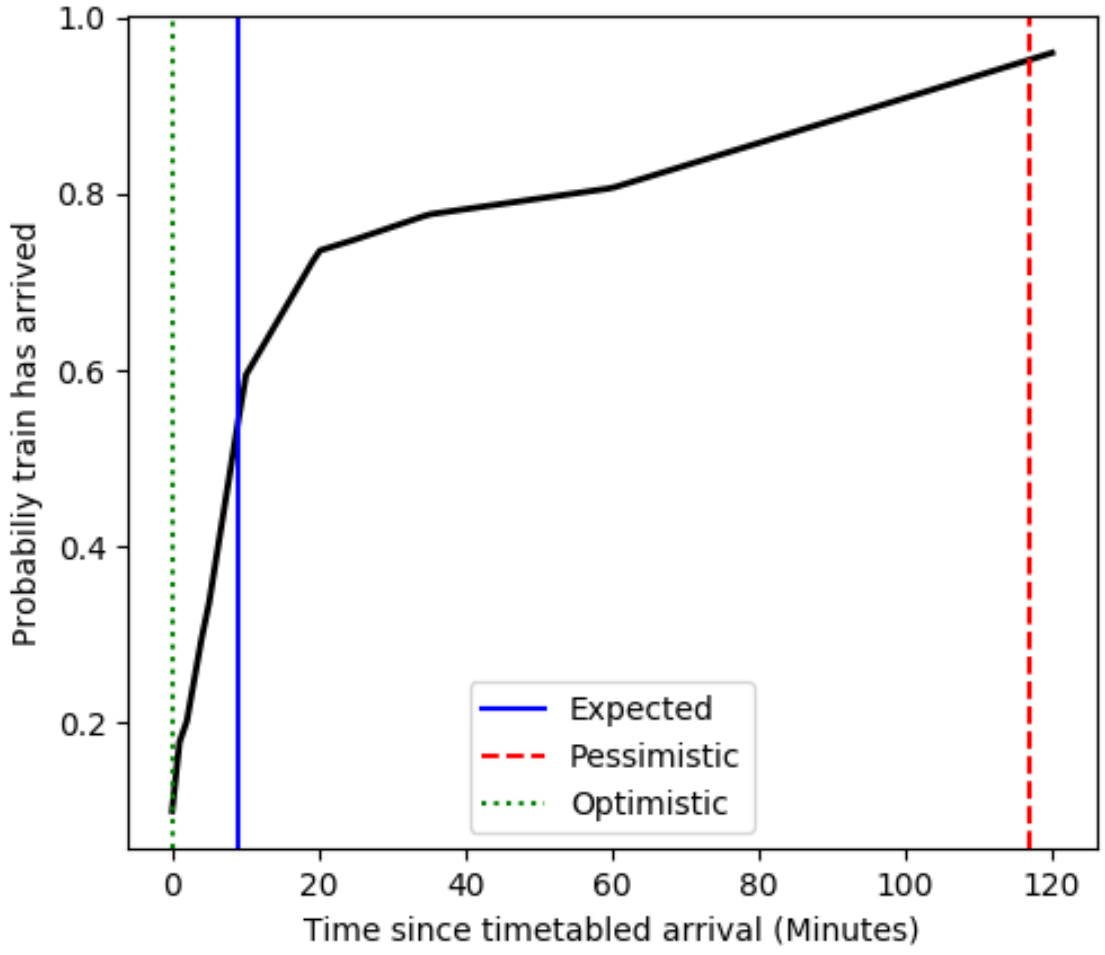
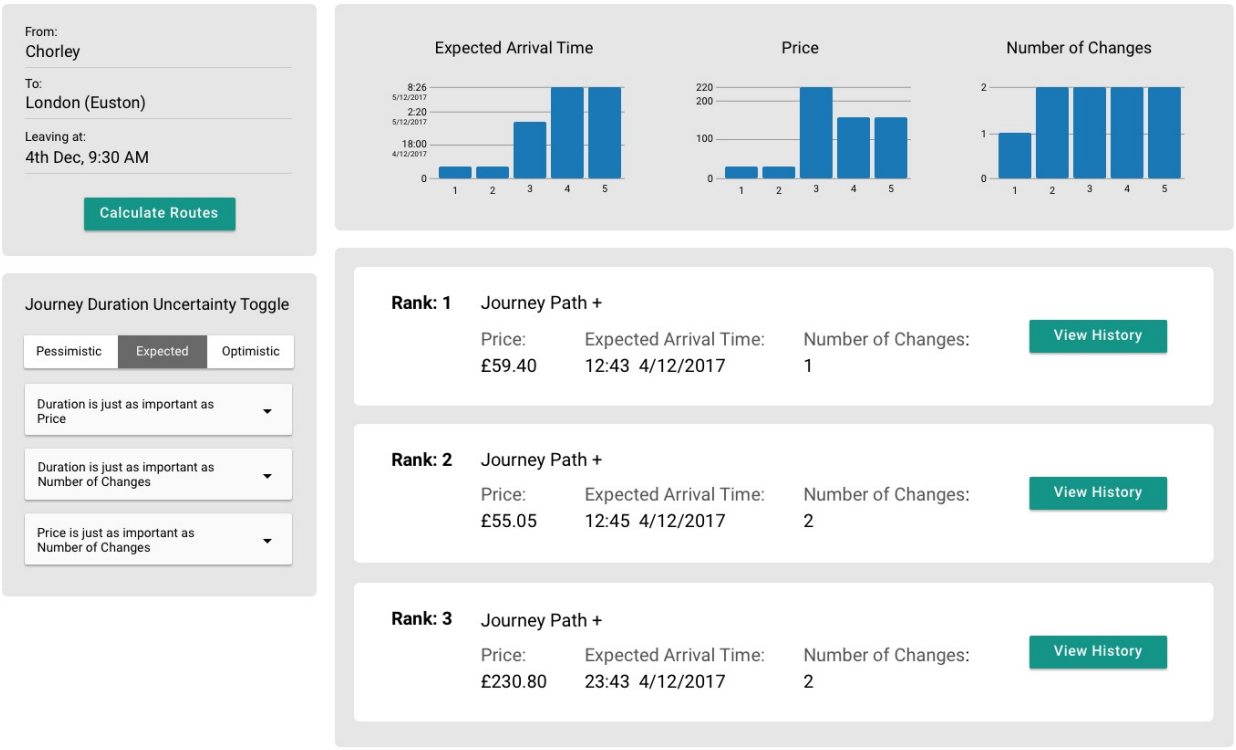


Figure 3.5: Cumulative Density Function for Arrival Time

top left corner shows the trip input form, where the user can input where they wish to travel *From* (Origin Station), *To* (Destination Station) and the time they are *Leaving At* (Departure Time). Once these values are set the user can click *Calculate Routes* to generate a set of possible journeys. The next task is for the user to specify their preferences *(Desiderata 1)*. In our user interface these pairwise user preferences are located in the bottom left. In Figure 3.6 the preferences are set to default, with all criteria equal. Each pair can be set through a drop-down menu one of five potential values:

Figure 3.6: Route Planning User Interface



1. *X is much more important than Y,*
2. *X is more important than Y,*
3. *X is just as important as Y,*
4. *X is less important than Y,*
5. *X is much less important than Y.*

These preferences can be changed at any point, triggering the system to re-rank the journeys. Once the planned trip and preferences have been detailed the user is presented with the top five ranked journeys (the fourth and fifth fall below the fold). Immediately the user can view criteria

values of each journey (*Price*, *Arrival Time* and *Transfers*). These values and the resultant ranking are updated continuously once routes have been calculated *(Desiderata 2)*.

To prevent information overload some extra details are hidden. Clicking the plus next to *Journey Path* displays the information needed to undertake a journey, including the journey path and the trains of which the journey is composed. Each journey also has a *View Detail* button, which allows the user to view provenance information in a pop-up window *(Desiderata 3)*. The design for this window is shown in Figure 3.7. Here the user can view the history of values for *Arrival Time* and the data sources.

The values for each of the criteria are shown in the bar charts at the top of Figure 3.6, with the x-axes ordered according to the ranking. These charts allow the user to visually compare a recommendation (the furthest left value) to the solution space (all other values). The charts are also ordered according to the weighting calculated through AHP, with the most important criteria appearing on the left. This means a user can both understand the trade-offs of a recommendation and how this ties into their specified preferences *(Desiderata 4)*. Finally, the user can toggle between *Pessimistic*, *Expected* and *Optimistic* modes for the predicted arrival time by clicking the corresponding button. These modes simply change the value extracted from the CDF, as described in Section 3.2.4 *(Desiderata 5)*. *Expected* values are more useful for users making a journey many times (such as commuters) whereas *pessimistic* values would be more important in a scenario where a user is travelling for something more time critical (such as a job interview).

3.3 Conclusions

In this chapter, we identified and demonstrated desiderata for dynamic DSSs, through the development of a train journey planning application. We presented a literature review revolving around the key issues and concepts in dynamic decision support. This discussion explains how these issues give rise to our eight desiderata. The chapter contributes these eight desiderata;

jointly making up a framework for trustable dynamic decision support.

Our train journey planning case study illustrated how our eight desiderata surface in a real application. The developed DSS helps a user to plan a journey between two stations according to the *arrival time*, *price* and the *number of changes*. This DSS support the five desiderata that can be viewed as desired features (rather than characteristics), namely:

1. declarative specification of preferences,
2. dynamic revision of recommendations,
4. explicit support for uncertain data,
5. data provenance, and
6. explanation of outputs.

For our train journey planning DSS, declarative specification of preferences and dynamic revision of recommendations are enabled by employing a dynamic GA with AHP applied as a fitness function. This is integrated with the uncertainty inherent to train arrival times, allowing the user to swap between expected, optimistic and pessimistic criteria values for uncertain criteria. Data provenance is provided, informing a decision maker on the source of information relating to train delays. Finally, the system provides a visual comparison of the alternatives as a means of explaining the rankings to the decision maker. This aims to provide an explanation of the outputs in a medium suitable for real-time decision making.

The final three desiderata are desired characteristics of dynamic DSSs, with the details of how we enable and evaluate these desiderata given in later chapters. In Chapter 4, we outline a methodology for enabling high diversity of options *(Desiderata 7)*, along with an evaluation. In Chapter 5, we evaluate four MCDM methods as fitness functions for our dynamic GA, assessing their stability of results *(Desiderata 3)* and the consistency of trade-offs between criteria *(Desiderata 8)*.

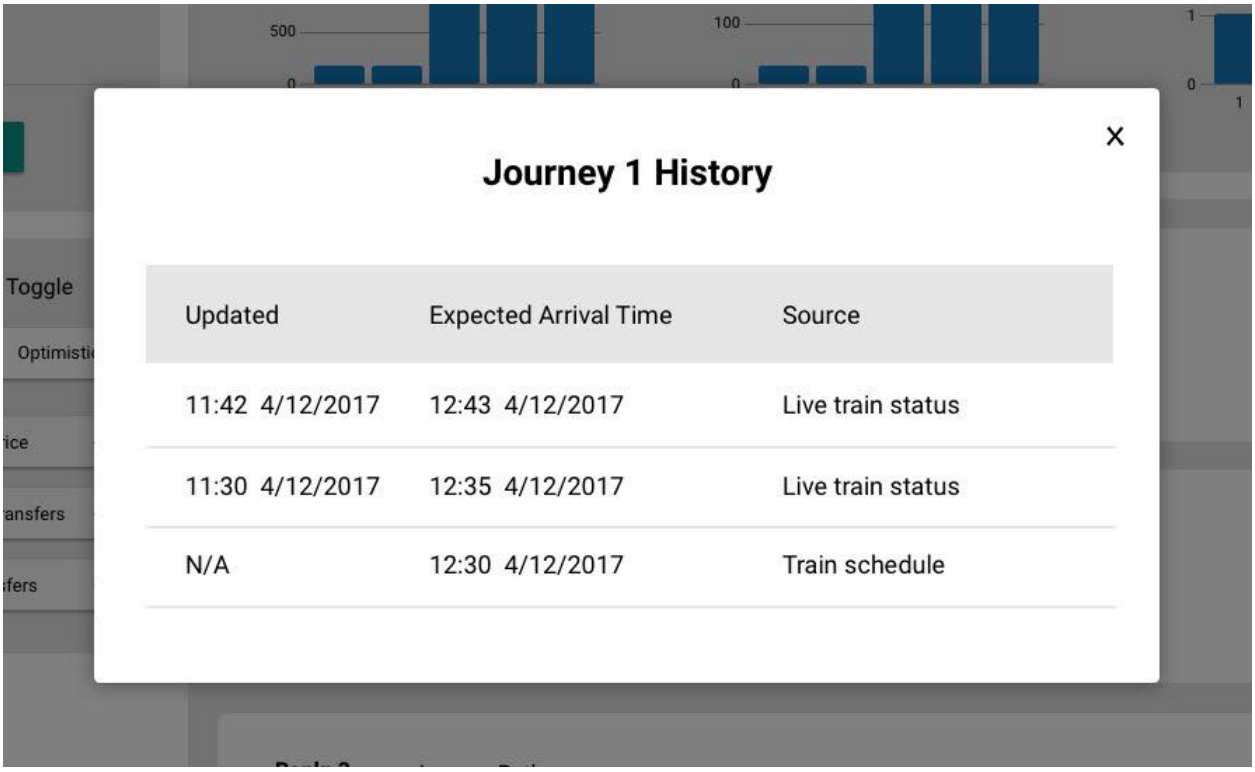


Figure 3.7: Provenance Data for an Arrival Time

4 | Harbour Management Case Study

Maritime facilities face challenging demands, including monitoring ocean traffic, port safety and emergency response. New technology is required to tackle these challenges, under the stresses of higher levels of traffic and an increased need for rigorous safety and prompt emergency response [104, 123]. Drones are one technology that has been identified for this role, and managing these drones for various tasks is one aspect of the expanding role of harbour management. Situational awareness has been highlighted as crucial in domains where the effects of ever-increasing technological and situational complexity on the human decision maker are a concern [95].

Drones provide a means for aerial situational awareness, within a harbour and beyond [48, 88]. One task, which has the potential to improve situational awareness, is automatic identification of ships approaching a harbour. Drones can be employed to take photos of ships, for the identification of traffic and potential threats. Selecting an appropriate route for the drone is a complex decision that can be informed through a DSS.

In this chapter we present our harbour management case study for situational awareness using drones. This case study is used as the basis for a test-bed environment that we used to assess appropriate MCDM methods for decision support in Chapter 5 and to analyse the effect of UI features on trust and its antecedents in Chapter 6.

An overview of the literature related to DSS applications to vehicle routing and drones is given. We then explain the motivation for using a DSS to manage a Unmanned Aerial Vehicle

(UAV) for situation awareness. Next, we explain how our desiderata relate to the task and give details of the criteria for this case study, when formulated as a MCDM problem. Finally, we present our DSS, showing the architecture and the user interface that a decision maker uses to interact with the system.

4.1 Related Work

In this section, an overview of the literature surrounding the problems solved and techniques used in the harbour management case study is presented. Firstly, an introduction to the Vehicle Routing Problem (VRP) and some studies producing VRP DSSs are described. We then introduce the UAV Task Assignment Problem (UAVTAP) and the application of MOEAs to solve them.

4.1.1 Vehicle Routing Problem

VRP is a well known problem in operational research and combinatorial optimisation. In VRP, routes must be assigned to a set of vehicles that must visit a set of customers such that the total cost of the operation is minimised. VRP has been tackled in a wide array of real-world systems.

Santos *et al.* [122] introduce a web-based spatial DSS for waste collections. The system provides static solutions with inputs in terms of constraints: shift time limit, vehicle types, capacities and an attribute to maximise/minimise. The system is therefore limited by its ability to optimise towards multiple objectives, requiring the user to specify a single attribute to optimise towards. Addressing this issue in the Safe Route Planner, a DSS designed to provide drivers with recommendations for the safest route between two locations, Sarraf *et al.* [124] integrate MCDM methods. The integration of user preferences allows the system to optimise towards multiple objectives at once and trade-off between the shortest, fastest and safest routes. They assess the AHP, Fuzzy AHP, TOPSIS, Fuzzy TOPSIS and PROMETHEE for suitability, concluding that AHP is most appropriate due to its simplicity and robustness.

Abbatecola *et al.* [1] introduce a decision support approach for postal delivery and waste collection. This system provides static routes for offline optimal planning of vehicle assignments. They apply a two-phase heuristic algorithm based on a clustering strategy and a fast insertion heuristic for solving a travelling salesman problem. This approach scales very well compared to a mixed integer linear programming approach, allowing results to be quickly recalculated. This improvement paved the way for future work, including plans to modify in real-time the planning of routes. Continuing the research, the approach was applied to a more complex problem, now including time and shift constraints [44]. Abbatecola *et al.* [3] provides further assessment under these conditions, concluding that the approach obtains a set of well balanced routes with respect to the vehicle travelling times and assigned loads. The latest paper [2], applied this algorithm to a dynamic VRP, encapsulating the routing problem of pick-up and delivery services considering both time windows and capacitated vehicles. The paper shows how the method can be applied by the vehicles for both planning the workday of the pick-up services, and adapting the routing plan to manage the ongoing requests.

These papers highlight user preferences as a desired feature for future VRP DSSs. This is because the suitability of a route in the real-world depends upon a wide array of criteria. To complement the addition of preferences, explanation is put forward as another useful feature for decision support. Explanation refers to the addition of capabilities allowing the user to distinguish the trade-offs made between solutions according to the user's preferences. We also identify dynamic updates as a desired feature of VRP DSSs.

4.1.2 Situational awareness with drones

The UAVTAP consists of finding an optimal assignment of UAVs to a set of tasks [28]. In this section we discuss previous research in the area and their approach to solving UAVTAP as an MCDM problem.

Ries and Ishizaka [114] present a multi-criteria support system for a dynamic UAVTAP. Their case study employs UAVs for surveillance to investigate ships in a maritime environment. The approach applies AHP to calculate weightings for PROMETHEE, ranking ships from least to most suspicious as a means of facilitating an efficient priority for surveillance. The system then assigns a UAV to investigate the most suspicious ship. This approach differs from our own as it calculates only the next ship to be visited, whereas in our own system we attempt to qualify an entire tour, limited by the fuel of the drone. This approach allows us to plan further ahead to find more fuel and time efficient routes. As a result of planning further ahead, our case study features an exponentially larger set of possible solutions. To explore this large set of possible solutions, we integrate a genetic algorithm as a meta-heuristic search function.

Ramerirez *et al.* first formulated UAVTAP as a constraint satisfaction problem, with the mission being modelled and solved using constraint satisfaction techniques [109]. In a later paper [111], UAVTAP is formulated as a multi-objective optimisation problem. Their objective consists of minimising the number of drones employed, the total flight time, the total fuel, the total distance, total cost and the time taken. The original approach was combined with a MOEA [110], this algorithm provides an estimate of the POF, i.e. the set of all non-dominated solutions of the problem. A solution s_1 is dominated by s_2 if s_1 is not better in any objective and s_2 is better in at least one. A solution is non-dominated if it is not dominated by any solution in the solution set. Their conclusion is that, as the complexity of the mission increases the number of solutions in the POF becomes huge, and therefore the time needed to calculate the complete POF becomes intractable.

To improve on this approach, Ramirez *et al.* [107] introduce a Knee-Point MOEA intending to reduce the POF to a set of the most likely best solutions. This reduces the size of the POF from hundreds to tens of solutions. This however, still leaves a difficult task for decision makers who must select the most appropriate mission plan. In a later paper, the authors rank the outputted POF using a selection of MCDM algorithms according to user preferences [108]. The

work assumes that the decision maker cannot provide *a priori* information regarding preferences on criteria. A limitation of this approach is that it can become costly in a dynamic setting, as the POF must be recalculated each time the mission scenario is updated.

Coelho *et al.* [27] also proposed a multi-objective UAVTAP. Taking inspiration from a multi-criteria view of real systems, the approach considers seven different objective functions which it seeks to minimise using a mixed-integer linear programming model solved by a matheuristic algorithm. This produces an estimate of the POF but the paper does not attempt to rank the non-dominated solutions.

In our work, we apply the MCDM methods as a fitness function of a genetic algorithm, rather than applying an MOEA. We take this approach as it is infeasible to calculate a POF in a dynamic setting. Our approach is enabled by the fact that the decision maker's preferences are available prior to route calculation. If they are not available, we assume the decision makers priorities are split evenly between objectives. Using a genetic algorithm with MCDM methods as a fitness function allows us to maintain a solution set ranked according to preferences, as the situation and therefore solutions, evolve over time.

4.2 Harbour Management Case study

In this section, we describe an UAVTAP for aerial situational awareness. For this task, a decision maker takes the role of a harbour master managing a harbour. The harbour master controls a single drone to identify ships close to the harbour zone. The job of the user is to select a route for the drone, with a view to identifying ships, before they reach the harbour zone. The length of these routes is limited by the fuel of the drone. Once the drone has run out of fuel, it must return to the refuel point, located within the harbour zone. We assume that the most suitable route may depend on different criteria: *Unidentified Ships in the Harbour*, *Average Lead Time* and *Fuel per Ship*.

- **Unidentified Ships in the Harbour** - The number of unidentified ships which will arrive in the harbour over the course of the route.
- **Average Lead Time** - The average amount of time between a ship being identified and arriving in the harbour.
- **Fuel Per Ship** - The amount of fuel used by the drone per ship identified.

4.2.1 Desiderata for Situational Awareness with Drones

As with all MCDM problems, these criteria have different values for different decision makers, so we require a specification of preferences (*Desiderata 1)*.

In our case, the criteria are predicted values, based upon the current trajectory of ships in the area surrounding the harbour. Ships approaching a harbour can quickly change direction, causing the criteria values of a route to change. This has an impact on the ranking of options, necessitating dynamic revision of recommendations (*Desiderata 2)*.

Such changes cause the criteria values for routes to change rapidly, resulting in difficultly when selecting a route under real-time constraints. This creates a requirement for a high stability of results (*Desiderata 3)*.

In this environment, there is no uncertainty captured in the underlying data. This means that for this case study, uncertainty cannot be propagated through the decision making process (*Desiderata 4)*.

There is also only a single source for the incoming data, so we ignore the desired provision of provenance (*Desiderata 5)*.

Without provenance data, we require another mechanism to enable to calibration of trust. In this case study, the harbour master has to select a route based on our three criteria. Understanding where each route falls within our solution set and the trade-offs being made is an important factor when creating understanding of the system. As a result, explanation is a desired feature

for calibrating trust (*Desiderata 6)*.

It is also worth noting that a decision maker may have knowledge outside the scope of the system. An example for harbour management could be multiple ships that do not require identification. If every route visits these ships then the decision maker is left to manually plot a route, rendering the system useless. This suggests that a diverse set of routes would be desired (*Desiderata 7)*.

When this diverse ranking is insufficient for selecting an appropriate route, the decision maker must generate a new ranking by altering their preferences. It is important that altering preferences has a predictable effect on the trade-offs between our three criteria: *Unidentified Ships in the Harbour*, *Average Lead Time*, *Fuel Per Ship* if the user is to make a timely decision. Hence, consistent trade-offs between criteria are desired (*Desiderata 8)*.

Bringing this together, we have the following list of desiderata for our harbour management case study:

1. declarative specification of preferences,
2. dynamic revision of recommendations,
3. high stability of results,
6. explanation of outputs,
7. high diversity of options, and
8. consistent trade-offs between criteria,

In this section we have detailed how we enable (1) *declarative specification of preferences*, (2) *dynamic revision of recommendations*, (6) *explanation of outputs* and (7) *high diversity of options*. Further details of how we enable and evaluate (3) *high stability of results* and (8) *consistent trade-offs between criteria* are given in Chapter 5.

4.2.2 Criteria

In this section we describe the motivation and calculations for each of the criteria used for route selection in our harbour management task.

4.2.2.1 Unidentified Ships in the Harbour

The primary goal of the situational awareness task is to identify ships before they reach the harbour. As a result, the first criterion is the number of unidentified ships which will arrive in the harbour over the course of the route. The system calculates the length of time needed to complete a route, then uses the current trajectory and speed of all boats to calculate which of the unidentified ships will reach the harbour before the drone returns. The amount of time each of these ships will spend in the harbour unidentified is then summed. The total time spent by *Unidentified Ships in the Harbour* should be minimised.

4.2.2.2 Average Lead Time

For situational awareness, information gained sooner is more valuable. If a ship is identified seconds before it passes the threshold into the harbour, there is no time to respond to the gathered information. As such, it is important to maximise the time between a ship's identification and its arrival in the harbour. We call this metric *lead time*. The system takes the speed and trajectory of each ship within a route to predict the lead time for each. The average of these values is calculated and used as the criterion *Average Lead Time*. The *Average Lead Time* should be maximised.

4.2.2.3 Fuel Per Ship

An important part of managing any operation is minimising cost. For this task, that means reducing the amount of drone fuel we expend. To maximise efficiency, another objective is therefore to minimise the fuel cost per ship visited. For our simulation, the drone burns a fixed amount

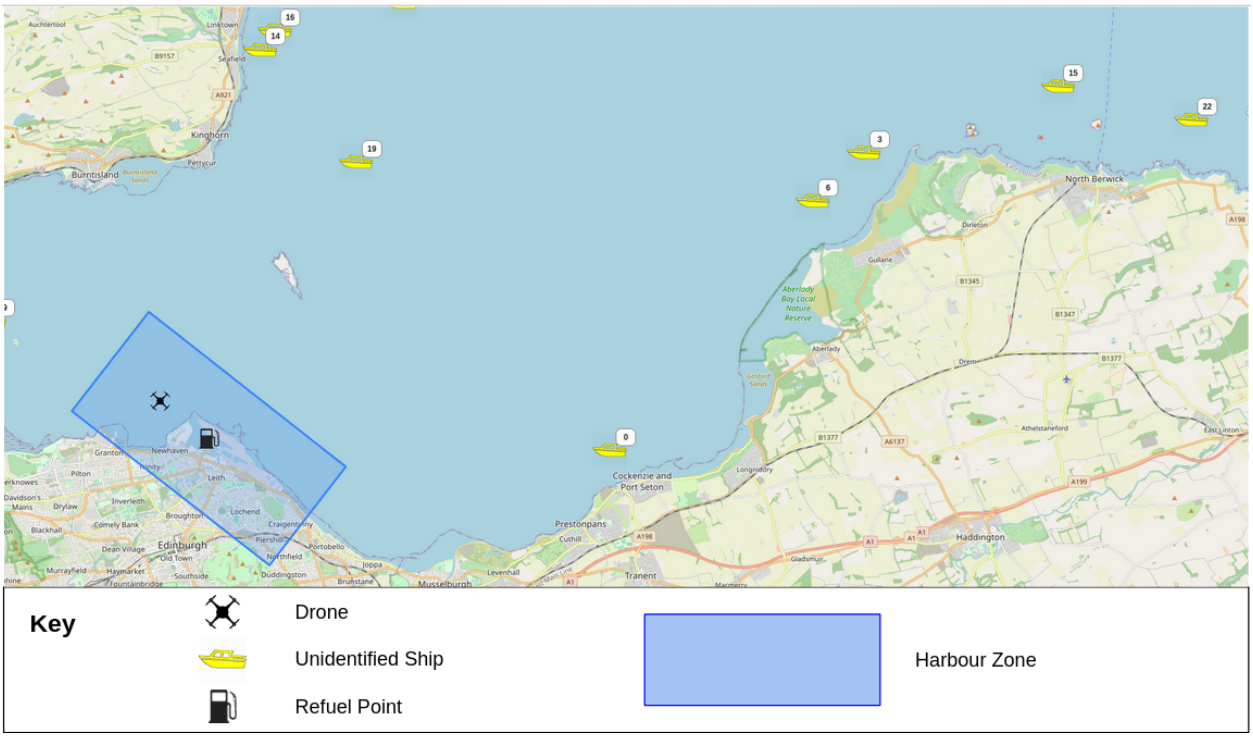


Figure 4.1: A visualisation of the Edinburgh scenario

of fuel per distance travelled. To compute a value for this criterion, we first determine the total fuel required for a route, by calculating the total distance of the route divided by the fuel efficiency of the drone. We calculate the distance of the route by predicting an intercept point for each ship then summing the distance between each intercept plus the distance to return to harbour. The total fuel is then divided by the number of ships identified, giving us *Fuel per Ship*. *Fuel per ship* should be minimised.

4.2.3 Scenarios

To analyse and evaluate our desiderata we have ten scenarios which simulate traffic in harbours around Great Britain and Ireland. The set of scenarios comprises simulations of the following locations:

1. Edinburgh

2. Liverpool
3. Dublin
4. Belfast
5. Portsmouth
6. Plymouth
7. Oban
8. Douglas
9. Dover
10. Hull

A scenario is defined by a set of 600 points in time and the position of all ships at each point. To maintain comparability between scenarios, each simulation includes 30 ships, of which 20 arrive at the harbour. The gap between each time step is equivalent to 30 seconds, therefore the entire simulation plots the routes of ships over a duration of five hours. We chose a five hour window of time for each scenario as this represents a sensible maximum flight time for a harbour management drone (based on domain expertise). This allows the drone time to identify roughly 20/30 of the ships. Therefore, if the correct route is picked, the drone should be able to achieve a perfect score. Unfortunately, due to the ships changing directions and speed, this route is impossible to predict every time, and often ships will arrive in the harbour before the drone reaches them or ignored ships will change path towards the harbour. Figure 4.1 shows a screenshot of the Edinburgh scenario visualised through a web interface.

Table 4.1: Criteria weightings - USH: Unidentified Ships in the Harbour; ALT: Average Lead Time; FPS: Fuel Per Ship

Weighting	USH	ALT	FPS
A	1	0	0
B	0	1	0
C	0	0	1
D	0.5	0.5	0
E	0.5	0	0.5
F	0	0.5	0.5
G	0.33	0.33	0.33

Table 4.2: The A-G represent the labels for the weightings.

4.2.4 Criteria Correlations

For a multi-criteria decision making problem to be interesting, it is important to have criteria that do not correlate strongly or have a negative correlation. For example, when purchasing a car, buyers may choose to minimise cost whilst maximising top speed. These criteria are correlated but conflicting, producing a need for trade-offs between objectives and therefore a need for MCDM methods. If the criteria are too strongly correlated then often all objectives can be maximised at the same time. As a consequence, strongly correlated criteria would be unfit for evaluating the capabilities of a MCDM system.

4.2.4.1 Experimental Set Up

To analyse the suitability of our criteria, we designed an experiment to calculate the Pearson Correlation between pairs of criteria. We generated 20 routes across each of the 10 scenarios outlined in Subsection 4.2.3 using the criteria weightings given in Table 4.1 for a total of 1400 generated routes. Each of these routes has a criterion value for *Unidentified Ships in the Harbour*, *Average Lead Time* and *Fuel Per Ship*. We then calculated the Pearson Correlation Coefficient between each pair of criteria values, with the results given in Table 4.3.

4.2.4.2 Results

Before analysing the results given in Table 4.3, it is worth noting that *Unidentified Ships in the Harbour* and *Fuel Per Ship* are both criteria we seek to minimise whilst maximising the *Average Lead Time* as stated in Subsection 4.2.2. With this in mind, the following can be observed:

- *Average Lead Time* and *Fuel Per Ship* have a high degree of correlation, but are conflicting.
- *Unidentified Ships in the Harbour* and *Fuel Per Ship* have a moderate degree of correlation.
- *Average Lead Time* and *Unidentified Ships in the Harbour* have a low degree of correlation.

The highest correlation occurs between *Average Lead Time* and *Fuel Per Ship*. These two criteria are highly correlated because *Fuel Per Ship* is low when a route identifies groups of ships which are close to the harbour zone and therefore fuel efficient. For these kinds of routes, the *Average Lead Time* is also low as only a short amount of time passes between the identification of ships and their arrival in the harbour. Fortunately, these criteria are conflicting as one is a cost (*Fuel Per Ship)* and the other is a benefit (*Average Lead Time)*, therefore the correlation is acceptable.

The second most correlated pair is *Unidentified Ships in the Harbour* and *Fuel Per Ship*. These two criteria are correlated as identifying many ships which are close to the harbour often includes ships which are most likely to pass into the harbour zone. The pair are not highly correlated though as the most fuel efficient route often includes ships which are moving away or are otherwise unlikely to enter the harbour. *Average Lead Time* and *Unidentified Ships in the Harbour* also have a weak correlation, explained as a transitive effect of the other correlations. Overall, the set of criteria are suitable as it is impossible to simultaneously minimise *Unidentified Ships in the Harbour* and *Fuel Per Ship* whilst maximising *Average Lead Time*.

Table 4.3: Criteria Pearson correlation coefficient - USH: Unidentified Ships in the Harbour; ALT: Average Lead Time; FPS: Fuel Per Ship

Criteria	USH	ALT	FPS
USH	1		
ALT	0.254679	1	
FPS	0.434399	0.644990	1

Criteria Objectives - Maximise: Unidentified Ships in the Harbour, Fuel Per Ship; Minimise: Average Lead Time

4.3 Genetic Algorithm for Route Selection

In our work, we apply MCDM methods as a fitness function, rather than applying an MOEA. This approach is enabled by the fact that the decision makers 'preferences are available prior to route calculation. The approach allows us to maintain a solution set ranked according to preferences *(Desiderata 1)*, via a continuously running evolutionary algorithm, as the situation and therefore solutions, evolve over time *(Desiderata 2)*. In this section we describe the genetic algorithm applied for route selection, how it fits into our streaming genetic algorithm architecture, and our approach to encourage diversification of results *(Desiderata 7)*.

4.3.1 Formulation of the problem

In this section we describe the genetic algorithm for the harbour management task. We include evaluation of the optimal number of generations to compute for future experiments.

4.3.1.1 Chromosome encoding of Routes

The routes for the drone in the harbour management task can be represented by Chromosomes that consist of genes. Each gene of the Chromosome represents a ship to be identified. The routes can therefore be modelled as follows:

Let S be the set of ships to be visited,

For $s_i \in S$,

$$r = (s_1, s_2, .. s_n)$$

For the harbour management task, this chromosome represents the route

$$s_1 \longrightarrow s_2 .. \longrightarrow s_n$$

To ensure the validity of the route the following conditions are needed:

1. Each ship may only be visited once.
2. The total distance of the route must be less than the range of the drone.
3. A route must visit at least one ship.

4.3.1.2 Fitness Function

The fitness function is used to evaluate the quality of the route. As this is an MCDM problem, a MCDM algorithm is used calculate the fitness. This MCDM method could be one of AHP, WPM, TOPSIS or PROMETHEE, outlined in Section 2.1. The method is given a set of criteria weights, which are used to calculate a score for each solution. A diversity discount is then applied to each solution (described in Section 4.3.3) and the scores are normalised with the following formula:

$$Norm(x) = \frac{x - minX}{maxX - minX}$$

4.3.1.3 Mutate Function

The mutation operator exists to maintain the diversity of the population. Generally, for travelling salesmen problems, mutate is defined as a swap operation, swapping the order of two cities in a route. This is because in the travelling salesmen problem, all cities must be visited for a route to be valid. In the harbour management task the drone can only visit a portion of the ships, limited by fuel. As a result we define mutate as a combination of two functions; *add a ship* (Equation 4.1) and *remove a ship* (Equation 4.2). When a route (r) is chosen to be mutated, one of the two functions is applied with equal chance of each. The two functions are defined below:

$$\text{Let S be the set of ships to be visited,}$$

$$\text{Let } r = (s_1, .. s_n),\ s_x \in S, s_x \notin r,$$

$$Add(r, s_x) = (s_1, .. s_x, .. s_n) \tag{4.1}$$

Such that the ship s_x is selected from S and its position in r is selected at random.

$$\text{Let S be the set of ships to be visited,}$$

$$\text{Let } r = (.. s_y, ..),\ s_y \in S,$$

$$Remove(r, s_y) = (.., ..) \tag{4.2}$$

Such that the ship s_y is selected at random from r.

4.3.1.4 Crossover Function

The crossover operator combines two selected routes together. For our harbour management task the crossover operator is a single-point crossover function $c(x, y)$ defined by the following steps.

1. Two parents are selected from the population as parents; denoted as x and y.

$$\text{For } s_i, t_i \in S,$$

$$x = (s_1, s_2, .. s_n), y = (t_1, t_2, .. t_m)$$

2. A sub-path z is selected from parent x at a random point j of length $k < n$.

$$z = (s_j, s_{j+1} .. s_{j+k})$$

3. All ships from sub-path z are removed from parent y except the first ship in the path. This produces an intermediate path y_1

$$\text{For } s_j = t_2, s_{j+1} = t_4, s_{j+k} = t_6,$$

$$y_1 = (t_1, t_3, t_5, t_6 .. t_m)$$

4a. The path is merged into parent y at the point of the first ship in the path.

$$\text{For } s_j = t_2,$$

$$c(x, y) = (t_1, s_j, s_{j+1}, .. s_{j+k}, .. t_m)$$

4b. If the first ship in the path doesn't exist in y then add the path to the end of y.

$$\text{For } s_j \notin y,$$

$$c(x, y) = (t_1, .. t_m, s_j, s_{j+1}, .. s_{j+k},)$$

4.3.1.5 Solution Selection

The selection operator is used to select individuals from the last generation. This is a significant operator for determining the balance between exploration and exploitation [40]. The choice of strategy for this operator is therefore important and previous studies have evaluated different strategies in the context of the travelling salesman problem [112]. One of the strategies that has been evaluated is elitism.

Elitism is the process of preserving the previous high fitness solutions from one generation onto the next [106]. This is typically accomplished by copying these elite solutions directly into the new generation. Various studies have implied that elitist strategies can considerably improve performance for MOEAs [31, 155]. Consequently, we have chosen to apply an elitist strategy for solution selection as outlined below.

To keep a high quality set of solutions, individuals are selected based on their fitness value. This fitness value is calculated by applying one of the MCDM methods described in Section 2.1. The route with the highest fitness is first selected for the next generation. To improve the diversity of the population, we then apply a fitness penalty to all other routes based upon their overlap

with the chosen route. Another route is then chosen and the process is repeated with all other routes being discounted according to the overlap with the set of selected routes. This process continues until enough routes have been selected for the population.

The discount function includes a diversity weighting W. This weighting can be set to control to what extent overlapping routes are discounted to promote diversity. A value of 0 for W means that no discount is applied. The discount function $d(x)$ is described in Equation 4.3.

Let $f(r)$ be the fitness of route r,

R refers to the set of selected routes,

$o(r, R)$ be the overlap between r and R,

$$d(r) = e^{W\, o(r,R)}\, f(r) \tag{4.3}$$

This discount formula calculates the overlap between a route r and the selected routes R using the formula described in Equation 4.4.

$|R|$ refers to the cardinality of R,

$|r|$ refers to the numbers of ships in r,

$N(R, s)$ is the number of routes in R visiting ship s,

$$o(r, R) = 1 - \frac{1}{|r|} \sum_{s}^{r} \frac{N(R, s)}{|R|} \tag{4.4}$$

The discount function $d(r)$ is introduced in an attempt to improve diversity (*Desiderata 3)*. We evaluate the effect changing the diversity weighting W has on diversity and the impact on quality of solutions in Subsection 4.3.3.

4.3.1.6 Solution Initialisation

The initialisation operator is used to create a set of individuals for the initial generation. To create a sensible set of solutions, a distance heuristic is applied. To begin, a single ship s_x is selected from S. The distance from this ship to all others is calculated and the reciprocal is taken, as shown in Equation 4.5.

Let S be the set of ships to be visited,

$$s_x, s_y \in S,$$

$d(s_x, s_y)$ is the Euclidean distance between s_x and s_y.

$$d^{-1}(s_x, s_y) = \frac{1}{d(s_x, s_y)} \tag{4.5}$$

The reciprocal of these distances are calculated and summed as shown in Equation 4.6.

$$sum(s_x) = \sum_{S}^{s_n} d^{-1}(s_x, s_n) \tag{4.6}$$

Finally, we select $s_i \in S$ from S with probability $prob(s_x, s_i)$, as shown in Equation 4.7.

$$prob(s_x, s_y) = \frac{d^{-1}(s_x, s_y)}{sum(s_x)} \tag{4.7}$$

We then repeat the process for s_i. This process repeats until the distance to each remaining ship is less than the fuel remaining for the drone. At this point the process for creating a single route is terminated. Routes are created by employing this procedure until the desired population size is reached.

4.3.2 Number of Generations Experiment

In this section we measure the convergence of the population in our streaming genetic algorithm (described in Section 2.3) to determine an appropriate number of generations for further experiments.

4.3.2.1 Evaluation Methodology

To determine the number of generations to run for future experiments, we measured the convergence of our population to a stable set of solutions. For this experiment we applied our genetic algorithm with WPM as the fitness function, to the first time-step of the 10 scenarios, with each of the 7 weightings outlined in Table 4.1. The population size for the genetic algorithm is chosen to be 30 solutions. First, we initialise the population using the procedure described in Subsection 4.3.1.6. We then iterate the algorithm given in Section 2.3 and at each generation we output the current ranking of solutions in the population.

To quantify the similarity between rankings of subsequent generations, we utilise the evaluation metric Average Overlap (AO) discussed by *Sarraf and McGuire* [124] and *Webber et al* [144]. AO is a similarity measurement algorithm that assigns more weights to the top of the list. It is based on simple set overlap, where the user compares the overlap of the two rankings at incrementally increasing depths. The formula to calculate AO is given in Equation 4.8. The AO

between generations is given with the results shown in Figure 4.2.

P_t is the population at generation t

P_{t+1} is the population at generation $t+1$

k is the evaluation depth,

d is the depth.

$$AO(P_t, P_{t+1}, k) = \frac{1}{k} \sum_{d=1}^{k} \frac{|P_t \cap P_{t+1}|}{d} \tag{4.8}$$

4.3.2.2 Results

The results show that different criteria weightings converge at different times. The difference between these weightings can be explained by observing that weightings C, E, F and G include a non-zero weighting for *Unidentified Ships in the Harbour*. This criterion favours longer routes, which naturally creates a larger search space of potentially optimal routes than *Average Lead Time* which is optimal for short routes. *Unidentified Ships in the Harbour* also takes longer to optimise than *Fuel per Ship*, because the heuristic used to initialise the population is most similar to *Fuel per Ship*.

As shown in Figure 4.2, weightings A, B and D converge the fastest, achieving an AO between populations of 1 after 103 generations. Weightings C, E, F and G converge more slowly, achieving an AO of 1 between populations after 789 generations. Therefore, 800 generations is chosen as the appropriate parameter for later experiments.

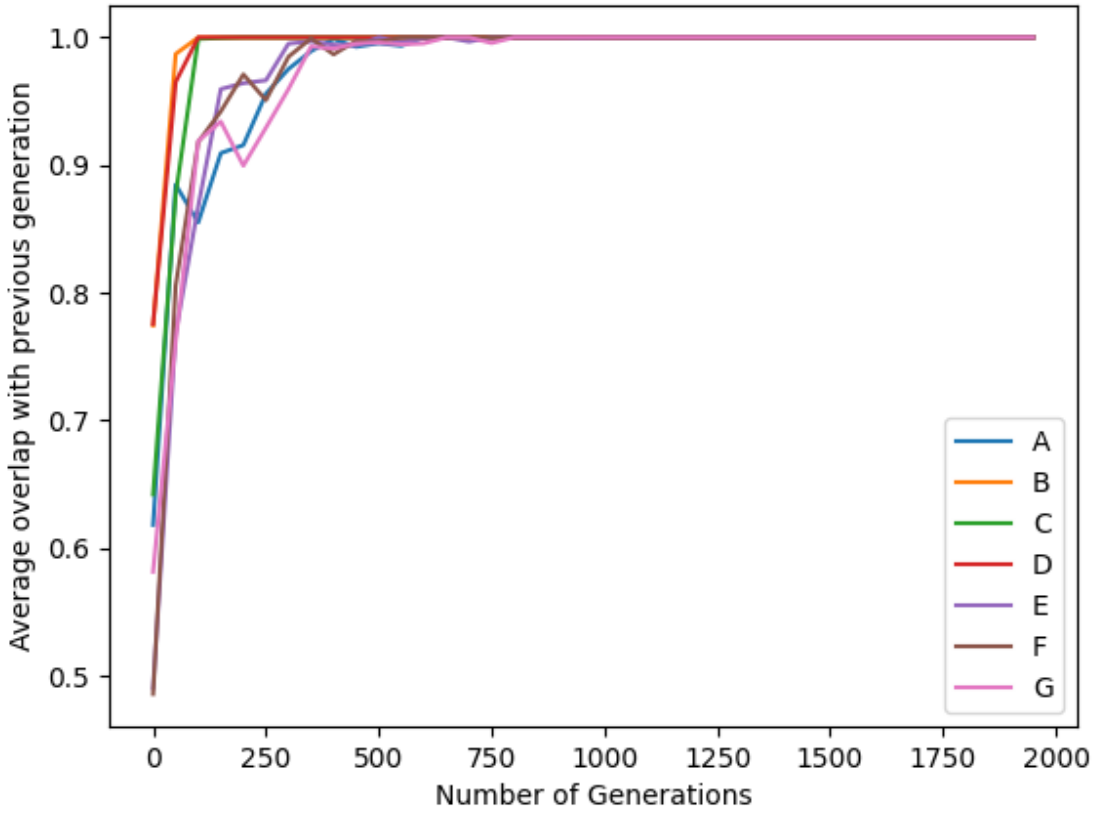


Figure 4.2: Average Overlap Between Generations

4.3.3 Diversification of Options

In this section, we analyse our approach to enabling a high diversity of options (*Desiderata 3*). We evaluate the effect of different diversity weightings on the discount function outlined in Subsection 4.3.1.5.

4.3.3.1 Evaluation Methodology

To evaluate the effect of the diversity discount function (shown in Equation 4.3), we generated rankings using different values for the diversity weighting W. A ranking was generated for each of the 10 scenarios, using each weighting from Table 4.1. The ranking was generated by running

the genetic algorithm for 800 generations, using WPM as the fitness function.

To calculate the effect of W, diversity was measured for each of the resultant rankings. To calculate diversity, we removed each route (one at a time) from the ranking then calculated the overlap (given in Equation 4.4) between the route and the routes remaining in the ranking. Diversity d was then calculated as the average of these overlap values as shown in Equation 4.9.

R is the ranking

$|R|$ is the number of routes in the ranking

r_i is a route in the ranking,

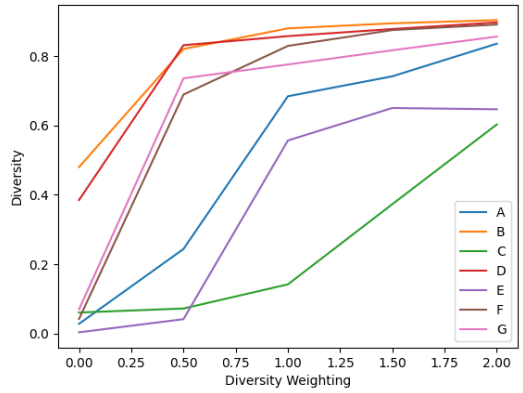
$R \setminus r_i$ is the ranking without r_i.

$$\mathrm{d}(R) = \frac{1}{|R|} \sum_{i=1}^{|R|} o(r_i, R \setminus r_i) \tag{4.9}$$

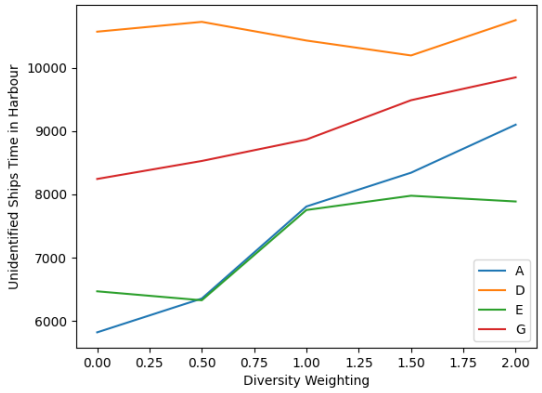
It is also important to note that changing the fitness function has an effect on the quality of the solutions in a ranking. To measure this effect, we recorded the criteria values for each criterion. The average of these criteria values across the rankings are compared as the diversity weighting W changes. When analysing criteria values, we plot only weightings that include a non-zero weight for the respective criterion. We analyse only these weightings because maximising a specific criterion is only an objective of the algorithm when the weighting is non-zero.

4.3.3.2 Results

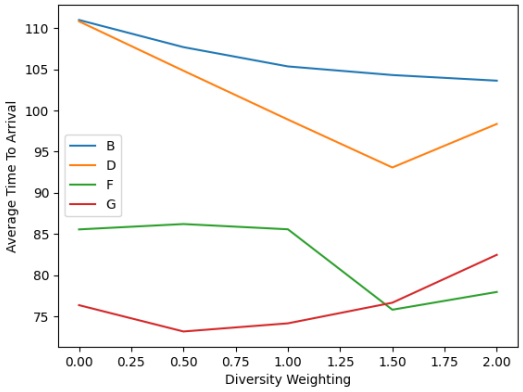
Figure 4.3(a) shows the effect of diversity weighting w on diversity. For each criteria weighting, increasing w increases the diversity of the rankings. It can also be observed that the criteria weighting has a significant effect on the diversity. Criteria weightings C and E both have lower



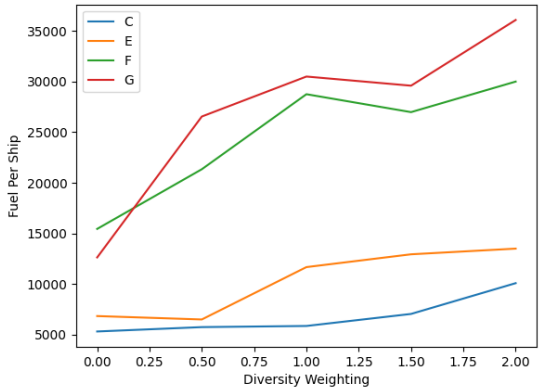
(a) Diversity of rankings.



(b) Unidentified Ships in the Harbour



(c) Average Time of Arrival.



(d) Fuel per Ship.

Figure 4.3: The effect of changing the diversity weighting on the diversity of rankings, Unidentified Ships in the Harbour, Average Time of Arrival and Fuel per Ship.

diversity, whereas weightings B and D both have higher diversity. This is because C and E are highly weighted towards *distance per ship*, and this criterion favours longer routes which therefore overlap more frequently, whereas B and D have a high weighting for *average lead time*, which favours shorter routes.

Figures 4.3(b), 4.3(c) and 4.3(d), show the effect of W on criteria values for *Unidentified Ships in the Harbour*, *Average Time to Arrival* and *Fuel per Ship*, respectively. *Unidentified Ships in the Harbour* is a (*cost*) criterion we seek to minimise, *Fuel per Ship* is another cost criterion. Whereas, we seek to maximise values for the *Average Time to Arrival* because it is a *benefit*. Weightings A, B and C are weighted towards only one criterion. Each of these weightings shows a decrease in quality as the diversity weighting increases.

The effect is more complex for weightings D, E, F and G as a result of the relationships between criteria. Figure 4.3(c) shows that weighting G features the highest *Average Lead Time* values at a diversity weighting of 2.0. This is caused by weighting G having equal weighting across criteria. *Average Lead Time* is correlated with the other criteria, and therefore more optimal values are attained as the solutions decline in overall quality.

4.3.3.3 Conclusion

Increasing diversity weighting W gives rise to the desired effect on diversity, at the cost of a decreased quality of solutions. This is because the optimal routes with 0 diversity weighting W often visit the same ships. It is worth noting that the recommended (first route) in each ranking is unchanged by altering W.

4.3.4 Architecture

The architecture for our test-bed environment is shown in Figure 4.4.

The decision maker operates the DSS through a web interface. To access this interface, a get request is made to the *Static File Server* (1) and the static files are returned. From here the user can

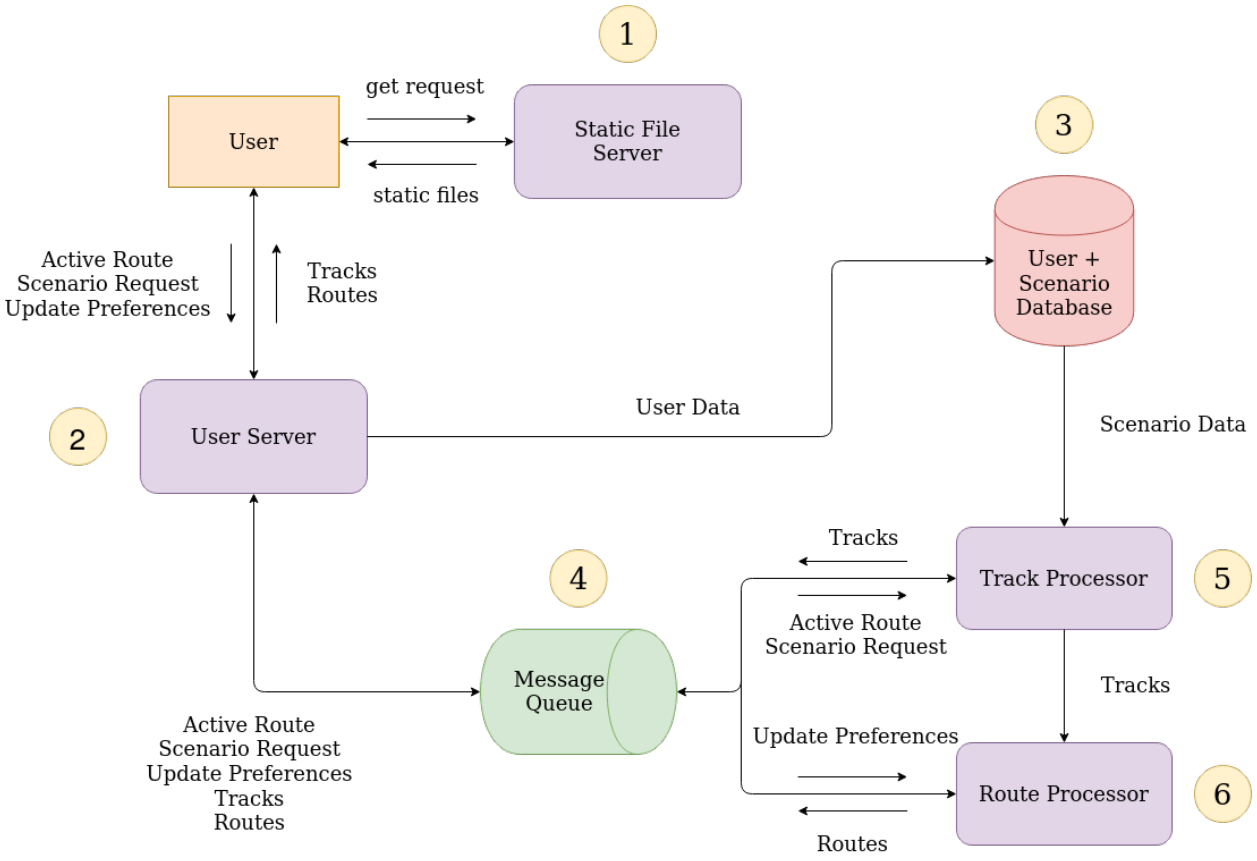


Figure 4.4: The system architecture

input their credentials and start a scenario. When a user starts a scenario, a scenario request is sent to the *User Server* (2) and the user is assigned a set of features. The *User Database* (3) contains the id for a user, their enabled/disabled features and their score for a session.

When a scenario request is authenticated it is placed onto the *Message Queue* (4). Once the *Track Processor* (5) receives a scenario request it retrieves scenario data from the *Scenario Database* (3). This database holds a set of scenarios defined by a set of ships and their position at each point in time called tracks. The *Track Processor* processes the stream of tracks, tracking the position of the drone and which tracks have been identified. The resultant stream of drone and ship tracks is passed to the front-end via the *Message Queue*. This stream is also processed further by the *Route Processor* (6). This component calculates a ranking of routes for each time-step and passes it to the front-end via the *Message Queue*.

Once a scenario has begun, a user can activate a route from the ranked recommendations or by drawing a path between tracks. The drone will then identify ships along the active route. If the recommendations are not appropriate, a user can input preferences to refine the selection of routes. This alters the weighting of criteria in the fitness function of the evolutionary algorithm.

4.3.5 Interface

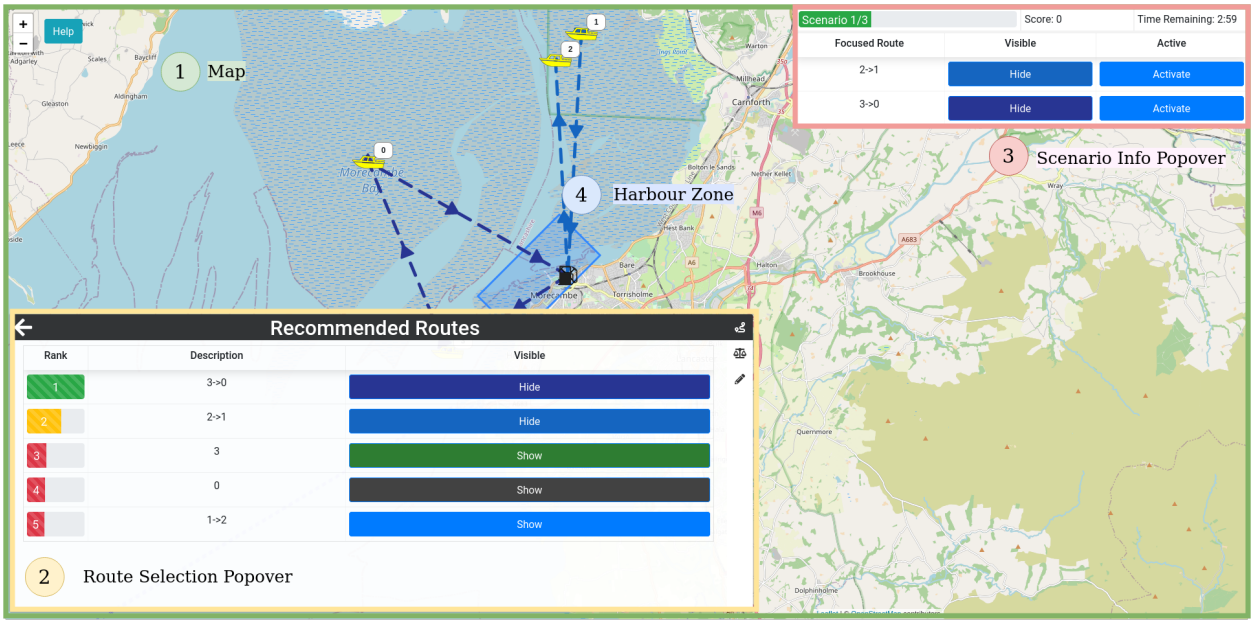


Figure 4.5: The user interface with no features enabled

The environment also provides a UI, which is used to evaluate the effectiveness of interface features in Chapter 6. In this section we explain this UI and how the user interacts with the system.

The UI for the harbour management task is shown in Figure 4.5. The window includes a *map* (1), the *route selection popover* (2) in the bottom left and a *scenario information popover* (3) in the top right. To render the map, Leaflet™ [76] was chosen as an open-source alternative to Google Maps™.

The *map* shows the current condition of the scenario, including: tracks, the drone, any visible

routes, the harbour zone and the refuel point. The tracks are indicated by a ship icon coloured yellow or green for unidentified or identified, respectively. The *harbour zone* (4) is displayed as a blue polygon and the position of the refuel point is shown as a fuel icon within.

The *route selection popover* shows the routes recommended by the system. These routes are ranked according to their fitness and similarity to the best possible route. The user can toggle each route as visible/hidden on the map. Any visible routes will be displayed within the *scenario information popover*. When a route is visible and has been deemed acceptable, the user can activate it by clicking *activate* in the *scenario information popover*. This popover also includes the current stage, the score for the current stage, and the time remaining. The recommended routes are dynamically updated until a route is activated, at which point the route selection popover is hidden.

If all routes recommended by the system are deemed unsuitable, the user can select the pencil icon to access the *draw a route* interface shown in Figure 4.6. Once this tab is open, a route can be drawn by clicking on ships in order. In this tab the user can view the criteria values of the drawn route, reset the drawn route, or activate the drawn route.

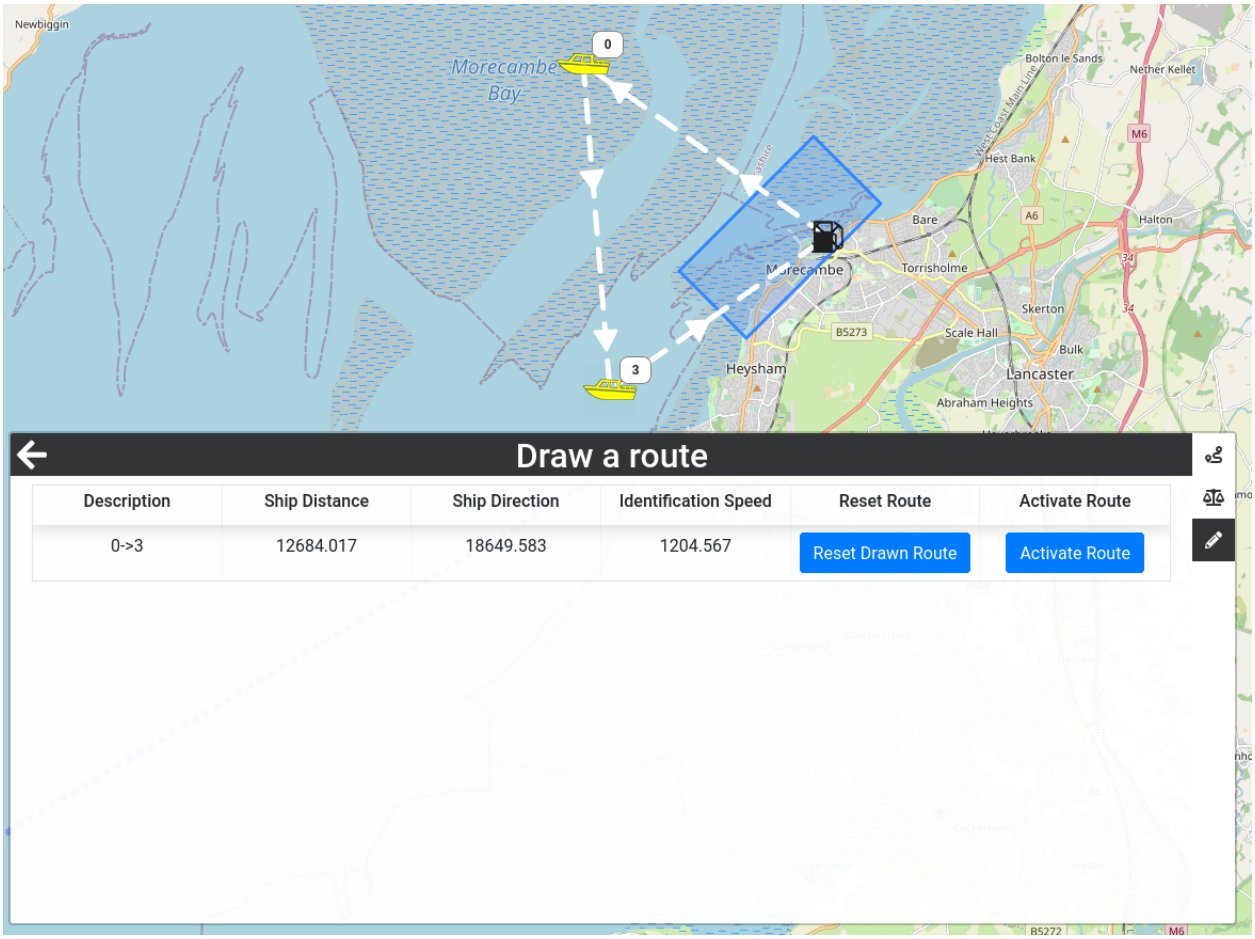


Figure 4.6: The interface for drawing a route

4.4 Conclusions

In this chapter, we aimed to show how we can support and evaluate desiderata for dynamic DSSs through a test-bed based on a harbour management case study.

The case-study employs a drone to provide situational awareness in a harbour area. The job of the user is to select a route for the drone, with a view to identifying ships, before they reach the harbour zone. The test-bed comprises a DSS for the case study and multiple scenarios over which we can evaluate the fulfilment of our desiderata. We apply the environment to carry out experiments on appropriate MCDM methods for decision support in Chapter 5, and to test the

effect of user interface features on trust and its antecedents in Chapter 6.

This case study has been formulated as a dynamic MCDM problem, including a validation of the suitability of the criteria. We assume that the most suitable route may depend on different criteria: *Unidentified Ships in the Harbour*, *Average Lead Time* and *fuel per ship*. The set of criteria were deemed suitable conflicting, as it is impossible to simultaneously minimise *Unidentified Ships in the Harbour* and *Fuel Per Ship* whilst maximising *Average Lead Time*. This is following the logic that if all objectives could be maximised simultaneously, the problem would be more suitably formulated as a single objective optimisation problem.

The test-bed DSS was built using a multi-criteria decision making genetic algorithm. This chapter contributes a dynamic genetic algorithm that can be used to incrementally refine recommendations, with a specific emphasis on the production of diverse recommendations. This algorithm applies the principles of DMOEAs, combined with MCDM methods as a fitness function, to support the cornerstones of dynamic decision support. The convergence of this algorithm has been analysed, indicating that all combinations of criteria converge after 789 generations. This has been used to support the choice of 800 generations as the appropriate parameter for assessing (3) *high stability of results*, (7) *high diversity of options*, and (8) *consistent trade-offs between criteria*.

We also have provided an outline and evaluation of our approach for enabling a *high diversity of options*. It was found that increasing diversity weighting W gives rise to the desired effect on diversity, at the cost of a decreased quality of solutions. This is caused by the optimal routes with 0 diversity weighting W often visiting the same ships. Our 7th desiderata is therefore provisioned within our DDS for situational awareness within a harbour area.

5 Evaluating Decision Support Methods for Harbour Management Case Study

In this chapter we present an evaluation of WPM, AHP, TOPSIS and PROMETHEE for suitability as components of a dynamic DSS using our situational awareness case study, as described in Chapter 4. Evaluation metrics are proposed for measuring the stability of results *(Desiderata 3)* and the consistency of trade-offs *(Desiderata 8)*. The methods are then compared according to these desired characteristics for dynamic MCDM problems.

5.1 Related Work

In this section we give examples of previous studies evaluating MCDM methods. These papers generally fall into two different categories: studies that outline issues or criteria and encapsulate them as evaluation metrics, and studies that compare rankings to a baseline method. Both kinds of studies are described below, including the methods that are evaluated and the metrics that are used.

5.1.1 Evaluating MCDM methods

Evaluating multi-criteria decision making methods is understood to be one of the most difficult problems in the field of decision analysis. *Zanakis et al.* [152] stated that it is impossible or difficult to answer questions such as: which method is more appropriate for what problem and what are the advantages or disadvantages of using one method over another.

In this section we discuss literature attempting to answer these questions. Specifically, we outline the application of concern, the methods compared, and their evaluation metrics.

Zanakis et al. [152] propose a simulation-based approach which compares the resultant ranking of an MCDM method to one generated through WSM, referred to in their paper [152] as the Simple Additive Weighting method. In the absence of any other objective standard, WSM is chosen as a result of its simplicity and the popularity of the method at the time of publication. The paper evaluates rankings produced by WPM (referred to in the paper as Multiplicative Exponent Weighting), AHP, ELECTRE and TOPSIS. The methods are evaluated through a bundle of metrics, notably mean squared error of weights, the mean squared error for ranks and Spearman's correlation for ranks, comparing both the weights assigned to alternatives and the resultant rankings. The paper concludes that AHP behaves most similarly to WSM, with ELECTRE being the least similar and the rest of the methods falling between the two.

The same study also evaluated the frequency of rank reversal. Rank reversal, first observed by *Belton and Gear* [11], refers to changes in the ranking of alternatives by addition or deletion of an alternative. In this paper, rank reversal was raised as an unintuitive characteristic of AHP. For their first example, the addition of alternative D to a ranking $B > A > C$ results in the ranking $A > B \sim D > C$. The ranking of A and B has reversed, despite no change to either alternative or the user preferences. This is an unintuitive and therefore undesirable characteristic of an MCDM method, as user understanding and faith in the results are imperative. Since then, rank reversal has also been noted to occur in other prominent MCDM methods, such as TOPSIS, ELECTRE and

PROMETHEE [50, 91, 143]. The study by *Zanakis et al.* introduces a new non-optimal alternative, then counts both how often the top ranked alternative remained the same, and the total number of ranks that are not altered as a percentage of the number of alternatives. This experiment found that TOPSIS suffered the least from rank reversal, followed by AHP and ELECTRE respectively.

Selmi, Kormi and Ali [126] compare the results of ELECTRE III, PROMETHEE I and II, TOPSIS, AHP and the Pareto-Edgeworth-Grierson method (PEG) in two case studies. The first study finds a similarity between PROMETHEE II and AHP, and observes the starkest difference between TOPSIS and ELECTREE III. The second study finds that ELECTREE III, PROMETHEE, and AHP agree on the first alternative being the best, while TOPSIS and PEG rank the third alternative as the highest. The study calculated the Gini Index to measure the rankings dispersion and uses the mean value of dispersion to perform comparisons between rankings. *Sarraf and McGuire* [124] highlight an issue with this approach as it does not consider that the top of the ranked list as being more important than the bottom. In this study, AHP-PROMETHEE II switches one pair at the bottom of the ranked list, whilst TOPSIS-PROMETHEE II switches a pair at the top of the list, yielding the same mean value of the Gini index. They propose that a change in the ranks at the top of the list must have a higher negative impact on the evaluation.

Sarraf and McGuire [124] compare the results of AHP, Fuzzy AHP, TOPSIS, Fuzzy TOPSIS and PROMETHEE through two real-world transport based case studies. They evaluate the similarity of each method compared to AHP. AHP is chosen as the baseline as it is the most widely used method in the literature. They attempt to rectify their issue with equal weighting being given to changes regardless of position in the *Selmi, Kormi and Ali* paper [126] by employing AO and Discounted Cumulative Gain (DCG) as evaluation metrics. AO is highlighted by *Webber, Moffat and Zobel* [144] as an approach to assign greater weighting to differences at the top of the list. It is based on simple set overlap where the overlap of the two rankings is compared at an incrementally increased depth. DCG was proposed by *Jarvelin and Kekalainen* [65] and is used in information retrieval to evaluate the relevance of results returned by search engines. This metric

is an alteration of cumulative gain that discounts scores of alternatives using a logarithmic discount function as their rank increases. The paper finds that the PROMETHEE ranking fits well with the AHP ranking and these two methods produce the best results. They found that fuzzy AHP produced acceptable results and that TOPSIS and fuzzy TOPSIS produced poor ranking results. The writers recommend AHP as a simple and robust method for the transportation field. The limitation of *Sarraf and McGuire's* [124] approach is that both DCG and AO only compare the rankings generated by each method to those generated through AHP.

For dynamic decision support, the alternatives are no longer static so the issues with rank reversal become more pronounced. As a result of rank reversal and other behaviour, applying MCDM methods designed for a static environment to dynamic problems can result in unstable rankings. Instability in rankings means that the best solution often changes rapidly, leading to distrust in recommendations which may only be valid for a short window. This effect spotlights the need for high stability of results.

In our research, we propose eight desiderata for dynamic decision support. Each desideratum is a desired feature or characteristic for dynamic decision support. Two of these desiderata, *(3) high stability of results and (8) consistent trade-offs between criteria*, have been captured as metrics to evaluate the suitability of MCDM methods for dynamic decision making problems. These two are chosen for evaluation as they comprise the desired characteristics in our framework. The other desiderata are features which are either present or not and therefore unsuited for evaluation.

5.2 Trade-off Evaluation

Our second desired feature is consistent trade-offs between criteria. Changing criteria weightings is an important part of a decision maker's process of exploring a solution space. For MCDM methods it is desirable for changing criteria weightings to create predictable effects on the outcomes of decisions.

In this section we generate rankings using each of WPM, AHP, TOPSIS and PROMETHEE, with various weightings for criteria. The top 3 routes for each ranking were then simulated to calculate the outcomes (in comparison to predicted values that were previously used as criteria). The smoothness of the change in criteria outcomes as criteria weightings increase is used to evaluate the consistency of trade offs between criteria *(Desiderata 8)*.

5.2.1 Motivation

In a dynamic setting, the outcomes of decisions are uncertain. For example, in our situational awareness case study, the decision maker selects a route based upon the *predicted outcomes*. These predicted values for the criteria, *Unidentified Ships in the Harbour*, *Average Lead Time* and *Fuel per Ship*, are not sure to be consistent with the actual values for the route. Therefore, in this experiment we simulate routes across the scenarios to calculate values for the outcome of a route. The outcome is measured through three criteria that map to a corresponding predicted criterion.

- **Score** - The number of ships that arrive at the harbour identified minus the number of ships that arrive unidentified.
- **Average Lead Time** - The actual average lead time for identified ships.
- **Fuel Per Ship** - The actual amount of fuel used per ship identified.

Score is a metric that captures the overall outcome of the scenario. The value for score is estimated through the predicted value for Unidentified Ships in the Harbour. The other metrics are simply the simulated outcome for the criteria values. Average lead time is the outcome that corresponds to the predicted average lead time and fuel per ship is the outcome for the predicted fuel per ship.

To measure the consistency of trade-offs in a dynamic setting, we compare the weighting for our criteria to the corresponding actual outcomes for the top 3 routes in a ranking. A linear relationship between the weighting and outcome is simple to understand and so leads to consistent

and predictable results. We therefore assess the consistency of trade-offs by the linearity of the relationship between these two variables. The linearity is measured through the Pearson correlation between the weighting and the outcome. A linear trade-off front is consistent because each change in weightings maps to a proportional change in the outcome.

5.2.2 Evaluation Methodology

To generate rankings, the GA was run for each of the 10 scenarios, for 800 generations using the weightings shown in Table 5.1. The weightings capture a smooth transition of the weight of C_1 between a weighting of 0 and 1. This process was repeated to generate 11 rankings for each criterion (C_1) and scenario pair. We then simulated the drone according to the first 3 routes, across all 600 time-steps of the corresponding scenario, allowing us to calculate each of the outcomes: *score*, *average lead time* and *fuel per ship*.

To combine these three outcomes into a single metric for each ranking, we used discounted cumulative gain (DCG) [65, 66]. We chose DCG because when providing recommendations, the further an option is down the ranking, the less likely it is to be picked. DCG quantifies this by giving a higher weighting to higher ranked solutions. The result is three evaluation metrics for a ranking R: the DCG of score (score-DCG(R)), the DCG of average lead time (ALT-DCG(R)) and the DCG of fuel per ship (FPS-DCG(R)). The formulas are given in Equations 5.1, 5.2 and 5.3 respectively.

Let R be a ranking

Let r_i be the ith route in a ranking

Let $\mathrm{sc}(r_i)$ be the simulated score of r_i

$$\text{score-DCG}(R) = \frac{\text{sc}(r_1)}{\log_3 2} + \frac{\text{sc}(r_2)}{\log_3 3} + \frac{\text{sc}(r_3)}{\log_3 4} \tag{5.1}$$

Let $\text{lt}(r_i)$ be the simulated average lead time of r_i

$$\text{ALT-DCG}(R) = \frac{\text{lt}(r_1)}{\log_3 2} + \frac{\text{lt}(r_2)}{\log_3 3} + \frac{\text{lt}(r_3)}{\log_3 4} \tag{5.2}$$

Let $\text{fps}(r_i)$ be the simulated fuel per ship of r_i

$$\text{FPS-DCG}(R) = \frac{\text{fps}(r_1)}{\log_3 2} + \frac{\text{fps}(r_2)}{\log_3 3} + \frac{\text{fps}(r_3)}{\log_3 4} \tag{5.3}$$

For each of the rankings generated through weightings 1-11 over a criterion C_1, the DCG of the corresponding outcome is calculated. The average DCGs across all scenarios of each algorithm for *Unidentified Ships in the Harbour*, *Average Lead Time* and *Fuel per Ship* are shown in Figures 5.1, 5.2 and 5.3 respectively.

To measure the smoothness of trade-offs, the Pearson correlation between the weight of a criterion and the corresponding outcome is calculated. This metric reflects the linear correlation of variables, and thus provides an indication of the consistency of trade-offs as the weighting changes. The correlation is calculated for each scenario, from which we calculate a mean and standard error of the mean. Tables 5.2, 5.3, 5.4 show the mean and standard error of the mean for

Weighting	C_1 Weight	C_2 Weight	C_3 Weight
1	0	0.5	0.5
2	0.1	0.45	0.45
3	0.2	0.40	0.40
4	0.3	0.35	0.35
5	0.4	0.30	0.30
6	0.5	0.25	0.25
7	0.6	0.20	0.20
8	0.7	0.15	0.15
9	0.8	0.10	0.10
10	0.9	0.05	0.05
11	1.0	0	0

Table 5.1: Weighting for criteria to evaluate changing C_1 weight

Unidentified Ships in the Harbour weight vs *score*, predicted *average lead time* weight vs *average lead time* and *fuel per ship* weight vs *fuel per ship*.

5.2.3 Results

Figure 5.1 shows the weighting for *Unidentified Ships in the Harbour* against the score DCG. It can be observed that as the weight for *Unidentified Ships in the Harbour* increases, the score DCG also increases. The exception to this can be seen at the highest weights where score DCG decreases. This is caused by the fact that the other criteria also have an effect on score. Increasing the weighting past 0.6 gives diminishing returns for the criterion at the cost to other criteria. *Unidentified Ships in the Harbour* is not a perfect predictor of score because ships can change direction unexpectedly, therefore having a small weighting towards *fuel per ship* improves performance.

Figure 5.2 shows the weighting for *average lead time* against the average lead time DCG. It can be observed that as the weighting for *average lead time* increases, the average lead time DCG also increases. Once the weight reaches 0.7 the average lead time DCG peaks with no further gains in outcome associated with a higher weighting. This is caused by the algorithm selecting

the routes with the highest average lead time once the weight is equal to or greater than 0.7.

Figure 5.3 shows the weighting for *fuel per ship* against the fuel per ship DCG. It can be observed that as the weight for *fuel per ship* increases, the average fuel per ship DCG decreases. This is because the *fuel per ship* is a cost criterion. The fuel per ship DCG decreases quickly, then reaches an optimum value at a weighting of 0.5 with no further gains in outcome associated with a higher weighting. The exception to this is with the PROMETHEE algorithm, which shows an increase in fuel per ship DCG as the criterion weight increases. The likely cause of this is the preference function being unsuitable for the range of values of *fuel per ship*. The criterion with linear preference function was applied to all criteria, as described in Subsection 2.1.4. The values for fuel per ship fall closer to an exponential scale, therefore the trade-offs made are inappropriate. This creates a linear correlation for the predicted values. It is worth noting that there is a difference between predicted values and the actual outcomes measured. This noise is particularly great for *fuel per ship* due to the wide range of possible values. As a result, the correlation is reversed.

Table 5.2 shows the mean Pearson correlation and standard error of the mean for *Unidentified Ships in the Harbour* weight vs *score*. TOPSIS has the highest correlation, followed by PROMETHEE, AHP and WPM respectively. TOPSIS has substantially more consistent trade-offs than the other algorithms, with a correlation of 0.833. Table 5.3 shows the mean Pearson correlation and standard error of the mean for *average lead time* weight vs *average lead time*. WPM has the highest correlation, followed by PROMETHEE then TOPSIS and AHP. All the correlations are strong, with little difference between the algorithms. Table 5.4 shows the mean Pearson correlation and standard error of the mean for *fuel per ship* weight vs *fuel per ship*. *Fuel per ship* is a cost criterion, therefore a high negative correlation corresponds to consistent trade-offs between criteria. The correlations are negative with the exception of PROMETHEE. Generally the correlation is weak, showing inconsistent behaviour of changing weightings. The most consistent trade-offs are made by the TOPSIS algorithm, which had a correlation of -0.64. TOPSIS was

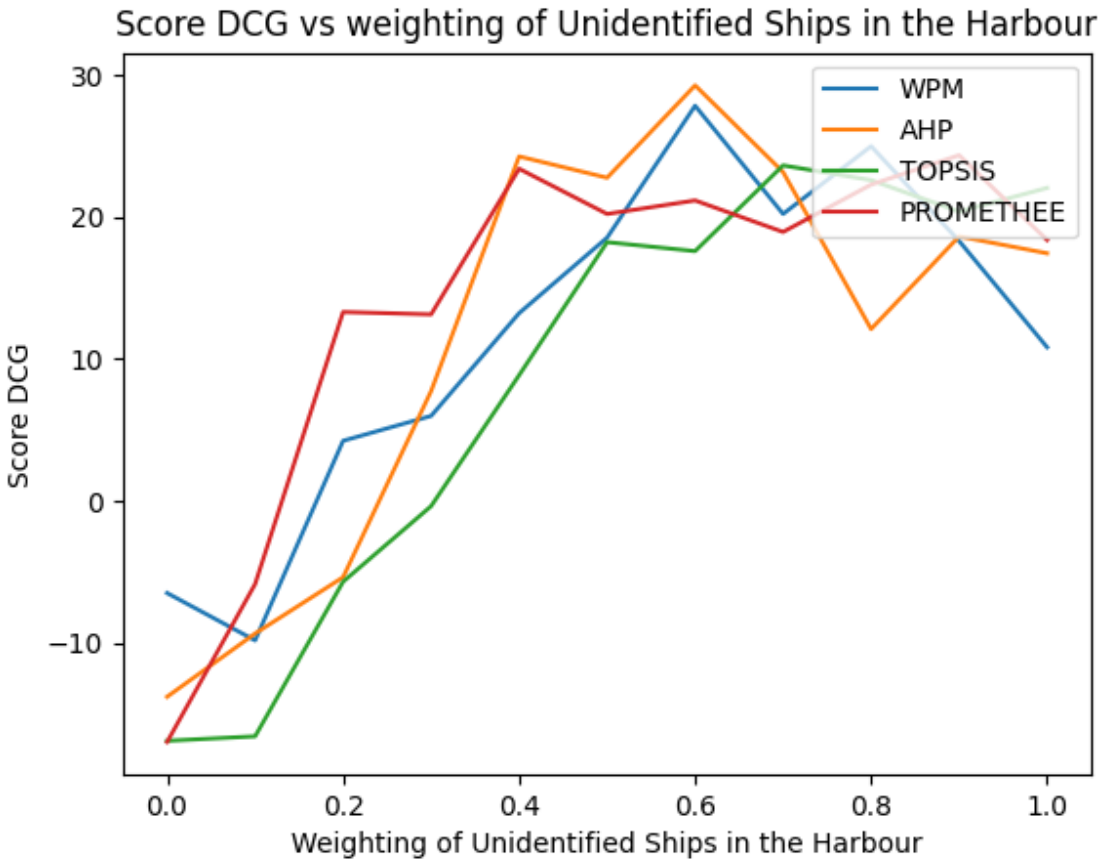


Figure 5.1: The change in score DCG as the weighting for Unidentified Ships in the Harbour increases

followed by WPM, AHP and PROMETHEE respectively.

5.2.4 Conclusions

Overall, TOPSIS was found to be the algorithm which made the most consistent trade-offs between criteria *(Desiderata 4)*, only under-performing another algorithm with respect to *average lead time*. AHP and WPM were the next most consistent algorithms with no significant difference in correlation between weights and outcome. Finally, PROMETHEE was the least consistent algorithm; this was caused by the implementation using a fixed preference function $P(x)$ for all criteria.

	USH weighting vs score DCG
WPM	0.541 +- 0.115
AHP	0.603 +- 0.049
TOPSIS	0.833 +- 0.0355
PROM	0.629 +- 0.0895

Table 5.2: Correlation between USH weighting and score DCG for each algorithm; USH refers to Unidentified Ships in the Harbour

	ALT weighting vs ALT DCG
WPM	0.942 +- 0.00541
AHP	0.885 +- 0.0415
TOPSIS	0.885 +- 0.0245
PROM	0.893 +- 0.0208

Table 5.3: Correlation between ALT weighting and ALT DCG for each algorithm; ALT refers to Average Lead Time

	FPS weighting vs FPS DCG
WPM	-0.268 +- 0.109
AHP	-0.166 +- 0.176
TOPSIS	-0.64 +- 0.0997
PROM	0.436 +- 0.159

Table 5.4: Correlation between FPS weighting and FPS DCG for each algorithm; FPS refers to Fuel Per Ship

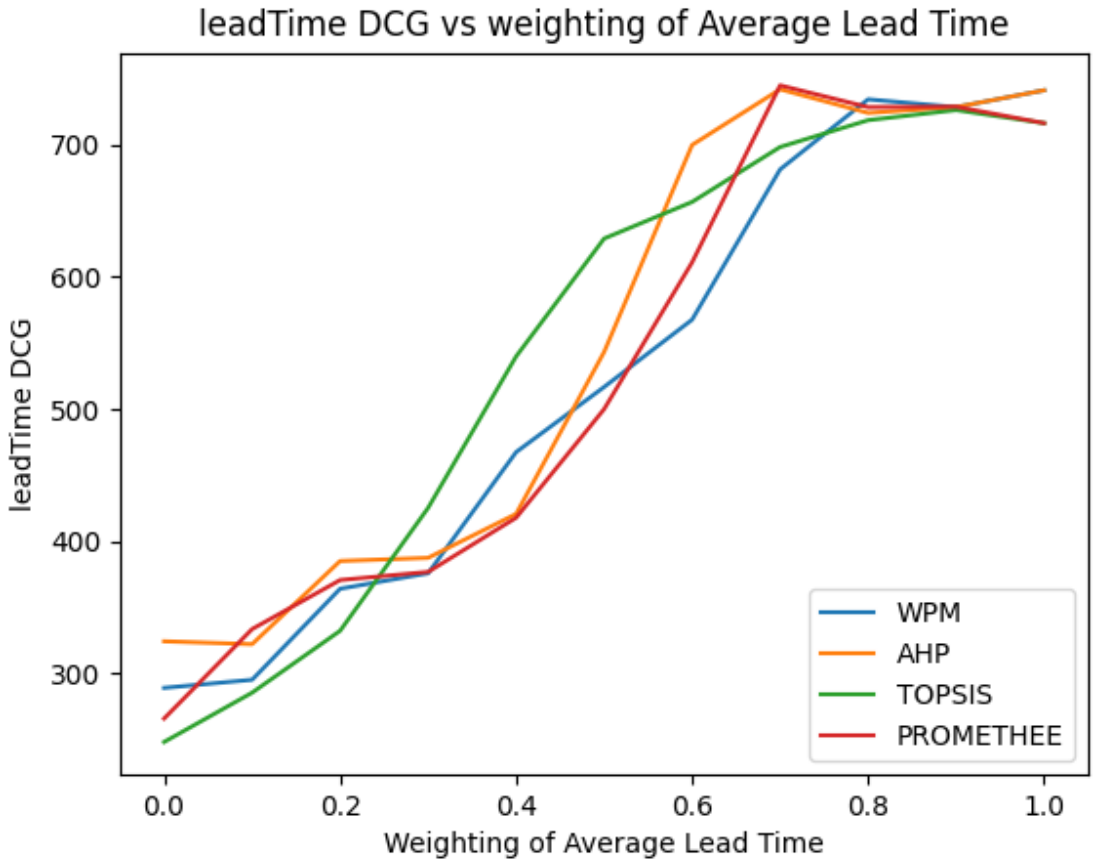


Figure 5.2: The change in Average Lead Time DCG as the weighting for Average Lead Time increases

5.3 Sensitivity Evaluation

In this section we generate rankings using each of WPM, AHP, TOPSIS and PROMETHEE as a fitness function for our dynamic GA. The stability of the resultant rankings is evaluated (*Desiderata 5)* under small changes to criteria values.

5.3.1 Evaluation Methodology

To evaluate the stability of rankings we used each MCDM method for 800 generations to generate a ranking. This process was repeated for each of the scenarios to generate a total of 10

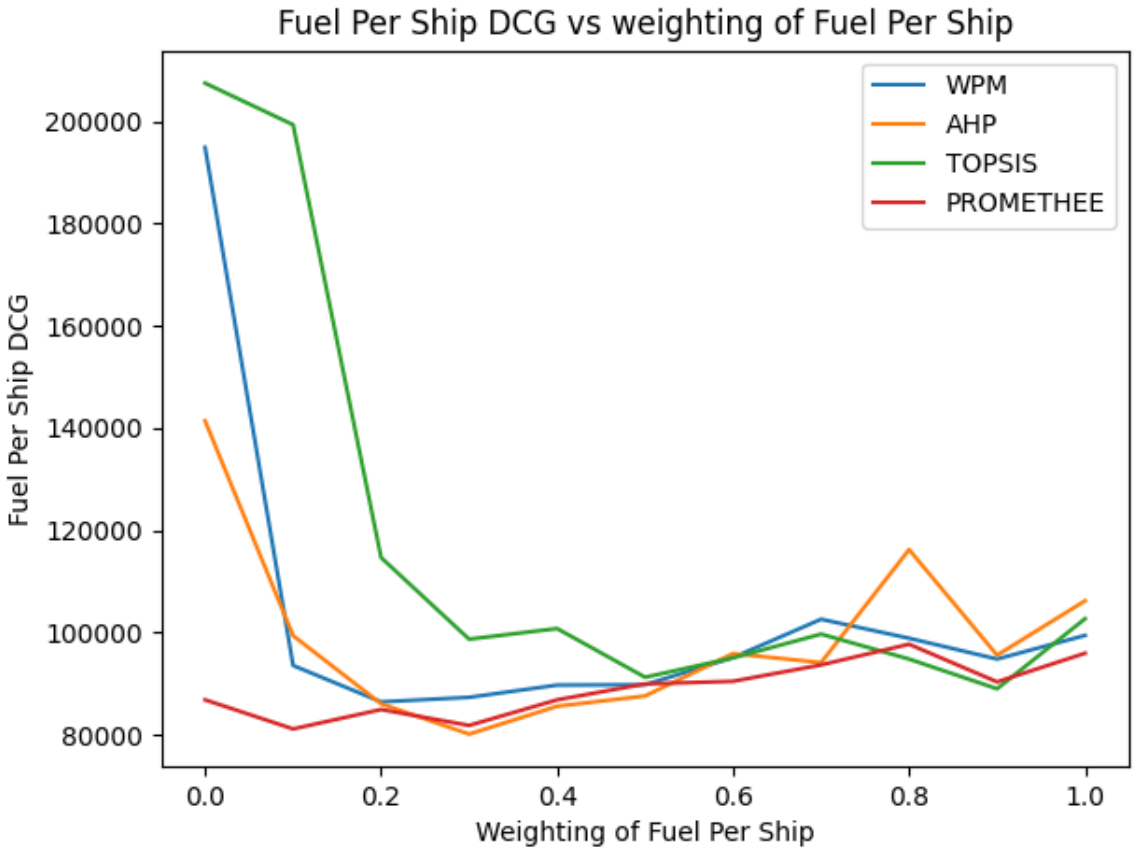


Figure 5.3: The change in Fuel per Ship DCG as the weighting for Fuel per Ship increases

rankings per method. Each algorithm used weighting G which assigns equal weighting to each criterion. For this experiment the diversity weight W was set to 0 to remove any effect of the diversity discount function on the outcome. Previous studies have analysed the sensitivity of MCDM algorithms over changes to criteria weightings [42, 126]. For a dynamic problem it is the values of criteria that are changing, therefore we assess the sensitivity over changes to criteria values.

Once we generated a ranking, we applied small changes to criteria values to model changes over time. The distribution is unknown so to do this we modelled a random variable (X) using a gamma distribution (Γ) for each criteria value of each route in the ranking [86]. The gamma

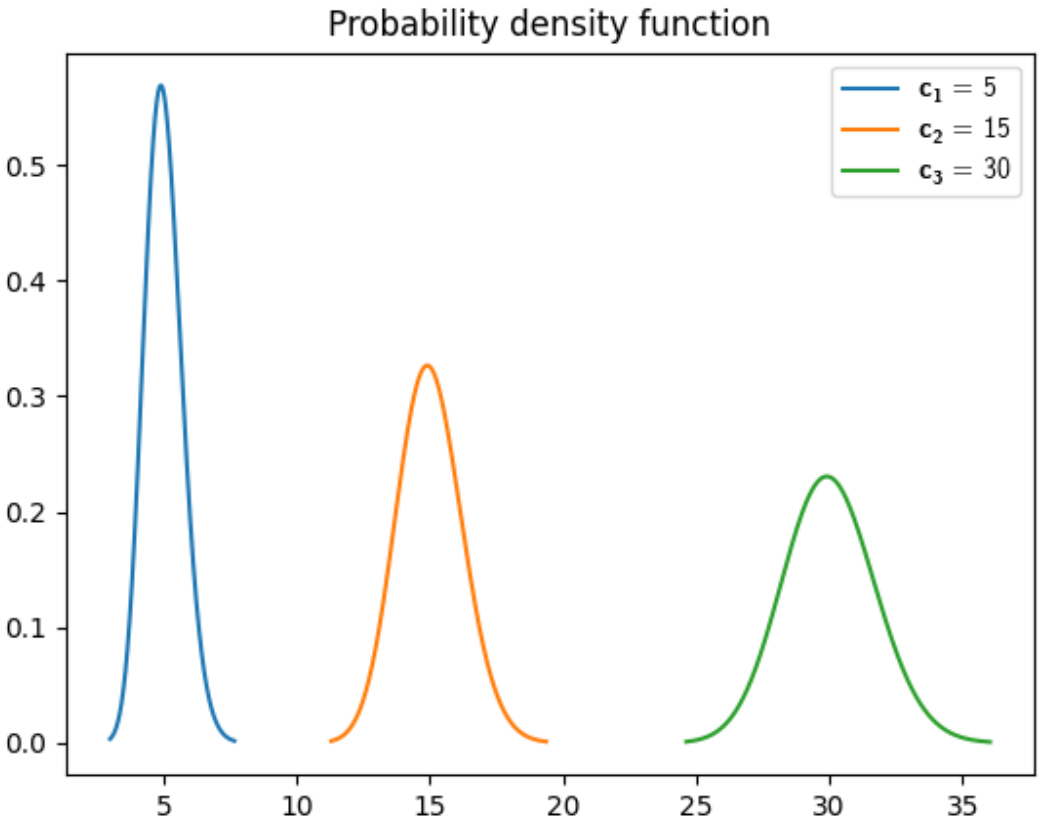


Figure 5.4: Gamma distribution probability density function with criteria values (c_x): $c_1 = 5$, $c_2 = 15$, and $c_3 = 30$.

distribution is a two-parameter family of continuous probability distributions. The gamma distribution is parameterised using shape k and scale θ as shown in Equation 5.4. For a gamma distribution with shape k and scale θ, the mean (μ) and variance (σ^2) are given in Equations 5.5 and 5.6 respectively.

$$X \sim \Gamma(k, \theta) = Gamma(k, \theta) \tag{5.4}$$

$$\mu = k\theta \tag{5.5}$$

$$\sigma^2 = k\theta^2 \tag{5.6}$$

The shape parameter effects the shape of the distribution rather than shifting it or stretching it. The scale parameter spreads the distribution across a larger range. We created a gamma distribution with the mean (μ) as the original value for the criteria (c_x) and the variance (σ^2) being 0.1% of the value for the criteria. To achieve this we parameterised the gamma distribution with θ and k as shown in Equations 5.7 and 5.8 respectively. Resultant gamma distributions are shown in Figure 5.4 for criteria values (c_x): $c_1 = 5$, $c_2 = 15$, and $c_3 = 30$.

$$\theta = \frac{1}{1000} \tag{5.7}$$

$$k = \frac{c_x}{\theta} \tag{5.8}$$

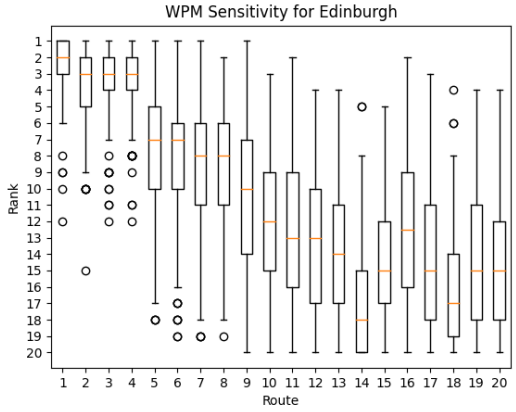
Each X was then sampled 200 times to generate 200 new sets of solutions with small changes to all criteria values. The sampled rankings were then re-ranked according to the MCDM method used to generate the original ranking. We then measured the average change in rank for each route.

The change in rank for each route for the Edinburgh scenario is shown in Figures 5.5(a), 5.5(b), 5.5(c) and 5.5(d) for WPM, AHP, TOPSIS and PROMETHEE respectively. The changes in rank were then averaged to calculate an average change across each ranking for each algorithm and scenario, as shown in Table 5.5. Finally, these scores were averaged across all scenarios to calculate the average change for each algorithm, given in Table 5.6.

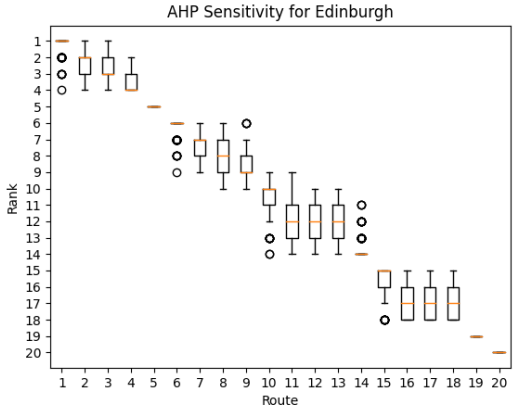
5.3.2 Results

Figure 5.5 show the ranks of routes across 200 samples for each algorithm over the Edinburgh scenario. The original rank of the route is shown across the bottom with the box plot showing the mean, upper quartile, lower quartile and range of ranks for each route, circles are drawn to represent outlying values.

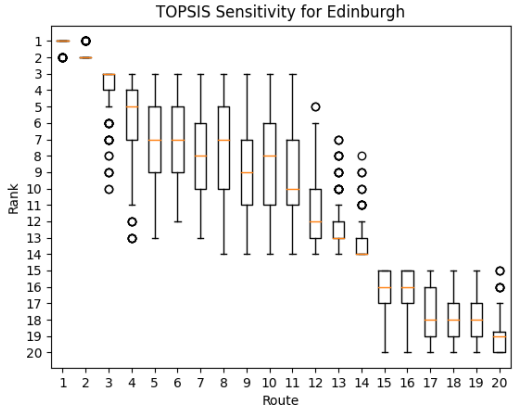
Figure 5.5(a) shows that the WPM ranking is very unstable, with an average rank change of



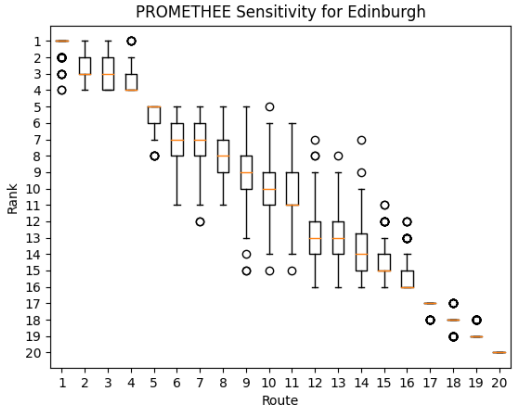
(a) WPM; showing an average rank change of 2.91 across 200 samples.



(b) AHP; showing an average rank change of 0.604 across 200 samples.



(c) TOPSIS; showing an average rank change of 1.43 across 200 samples.



(d) PROMETHEE; showing an average rank change of 0.853 across 200 samples.

Figure 5.5: Ranking sensitivity box plots; the bars show the range of rankings derived for each route after perturbations of criteria values.

Algorithm	WPM	AHP	TOPSIS	PROM	AVG
Edinburgh	2.91	0.604	1.43	0.853	1.45
Liverpool	0.294	0.185	0.222	0.0547	0.189
Belfast	1.57	1.3	1.11	1.79	1.44
Dublin	0.633	0.743	0.47	0.408	0.564
Portsmouth	4.90	1.51	1.35	3.27	2.76
Plymouth	2.32	0.41	0.486	0.661	0.969
Oban	4.98	2.17	1.00	4.82	3.24
Douglas	4.98	0.489	0.853	0.586	1.73
Dover	4.89	1.54	1.31	3.39	2.78
Hull	0.153	0.244	0.132	0.11	0.160

Table 5.5: Average change of rank across rankings for each algorithm and scenario; AVG refers to the average for all algorithms across each scenario; PROM refers to PROMETHEE.

2.91. It can be observed that the ranks of routes varies through almost the entire range of potential values, with the trend of increasing rank from left to right only visible through the mean and interquartile range. Figure 5.5(b) shows the most stable ranking, generated by AHP, resulting in an average rank change of 0.604. The top four routes are always the same, with positions varying more for lower ranked routes. Figure 5.5(c) shows a ranking generated by TOPSIS with an average rank change of 1.43. The first two routes are always the same, with positions varying over a large range for the rest of the ranking. Figure 5.5(d) shows a relatively stable ranking generated by PROMETHEE, with an average rank change of 0.853. For this ranking, the lowest ranked routes are the most stable, with the top results varying across a wider range of ranks.

Table 5.5 shows the average rank change for each algorithm across each scenario. It can be observed that the scenario has a significant effect on the stability of the rankings. For example, the Liverpool scenario on average generated much more stable rankings, resulting in an average rank change of 0.189. This was caused by the fact the scenario tended towards shorter routes which caused larger differences between criteria values in the resultant rankings. On the other hand, scenarios which tended towards longer routes, such as Oban which had an average rank change of 3.24, generated clusters of very similar routes. This created rankings with much smaller

Algorithm	Average Change in Rank
WPM	4.40 +- 0.0172
AHP	1.50 +- 0.0204
TOPSIS	0.931 +- 0.0107
PROM	2.88 +- 0.0403

Table 5.6: Average rank change and standard error for each algorithm across all scenarios; PROM refers to PROMETHEE.

differences between criteria values that were therefore much less stable under small changes.

Table 5.6 shows the average change for each algorithm across all scenarios. The standard error resulting from the randomness of sampling is also given. We found that applying TOPSIS resulted in the most stable rankings, followed by AHP, PROMETHEE and WPM respectively. The standard error for each algorithm was very small, meaning that all the differences were highly significant, under the epistemic uncertainty generated by random variable X from Equation 5.4. This was calculated via a t-test between all pairs of distributions (each generated by a different MCDM method).

5.3.3 Conclusions

TOPSIS was found to be the most stable method under small changes to criteria values *(Desiderata 5)*. AHP was the second most stable with a slightly higher change in rank than TOPSIS. AHP was followed by PROMETHEE and WPM respectively, which were found to be substantially less stable.

5.4 Discussion

In this chapter, we investigated the suitability of decision support methodologies for trusted dynamic DSSs. To do this, we evaluated the suitability of WPM, AHP, TOPSIS and PROMETHEE for dynamic DSSs in our test-bed environment according to their fulfilment of our desiderata.

There are few comparative studies addressing different MCDM methods. Such studies fall into two categories: those that compare the results of MCDM methods and those that compare specific attributes of the methods. The first category includes *Sarraf et al.* [124], a study that compares the results of different methods to a benchmark. Generally, these benchmarks are results generated using the most popular MCDM method at the time. The second category includes *Zanakis et al.* [152], a study that evaluates the frequency of rank reversal in MCDM methods. Rank reversal is an undesired characteristic for MCDM methods, therefore understanding its prevalence is invaluable when selecting an appropriate algorithm. Another paper, by *Triantaphyllou et al.* [136] presents a study based upon two criteria, comparing each method to a benchmark (category one) and determining the stability of a ranking when a non-optimal alternative was replaced with a worse alternative (category two). For example, given a ranking $A > B > C$, one of the non-optimal alternatives (B or C) would be replaced with a worse alternative (D). The ranking is then recalculated and the stability is analysed.

The second category compares methods according to the prevalence of unintuitive behaviour [152] and through the evaluation of desired criteria [136]. Both approaches can be seen as defining desired characteristics of a method, generally the absence of unintuitive behaviour, and creating an evaluation metric encapsulating this desideratum. Our study therefore falls into the second category, comparing desirable attributes of methods in the context of dynamic decision support.

This chapter builds on this framework through the assessment of two of our desiderata for dynamic decision support. These desiderata, namely high stability of results under small changes to criteria values *(Desiderata 3)* and the consistency of trade-offs *(Desiderata 8)*, have been put forward as important characteristics in the context of dynamic problems.

The effect of different strategies for the integration of criteria weightings is not well studied but is an important characteristic contributing to the suitability of an MCDM method. In our evaluation, TOPSIS was found to be the MCDM method that made the most consistent trade-offs between criteria. The trade-offs between criteria in TOPSIS are controlled by the scaling of

dimensions, each representing a criterion, in the solution space. This strategy has been shown to create consistently linear trade-offs, that can easily be understood by a decision maker.

TOPSIS was also found to be the most stable method under small changes to criteria values. This result is consistent with the study by *Zanakis et al.* [152], which found TOPSIS to be the most stable algorithm (suffering the least from rank reversals) when the members of a ranking are changed. Albeit using a different definition for stability, these results show the robustness of the goal, aspiration and references-based models. This is because this class of models relates the solutions more directly to an objective, rather than each other, leading to less sensitivity in the rankings when solutions are added, removed or criteria values are altered.

On the other hand, both AHP and PROMETHEE rely substantially on comparisons between all solutions and therefore have increased risk of rank reversal both under small changes to criteria values or when solutions are added or removed. *Zanakis et al.* showed that WPM is relatively stable under addition and removal of solutions, because there is no relative scoring of solutions beyond normalisation. Unfortunately, this normalisation causes a large amount of instability when criteria values change. When the largest criteria value changes, the multiplicative approach of WPM can cause a substantial change in the relative values of criteria. This leads to unstable rankings, under small changes to criteria values.

This chapter contributes two metrics, encapsulating stability of results under small changes to criteria values *(Desiderata 3)* and consistency of trade-offs *(Desiderata 8)*. These metrics are then used to carry out an evaluation of MCDM methods, capturing their suitability for application to dynamic decision support problems. WPM, AHP, TOPSIS and PROMETHEE were assessed, and from the results we have found that TOPSIS best fulfils both desiderata. Our final contribution is the suggestion of TOPSIS as an appropriate MCDM method for dynamic decision making.

6 | Trust in Dynamic Decision Support

In this chapter we aim to assess the effect of features of decision support on trust and its antecedents. We demonstrate a methodology applying PLS-SEM to assess these features in the absence of clear success criteria. A theoretical framework is applied to model the antecedents of trust in real-time decision support.

We conducted a user study with a drone-assisted maritime operations scenario to evaluate the effectiveness of declarative specification of preferences *(Desiderata 1)*, dynamic revision of recommendations *(Desiderata 2)* and explanation *(Desiderata 6)*.

6.1 Technical Background

Structural Equation Modeling (SEM) is a set of statistical methods used to create and validate models. These models are used to test complete theories and concepts in a wide variety of fields, such as marketing, business and decision support [74, 92]. This is done by creating constructs that represent latent concepts, such as a user's trust in a system, known as *latent variables*. Such latent variables are formed from indicator variables, which offer a measurable indication of the value for a latent variable, such as how often a user chooses to engage with a system.

The relationship between each latent variable construct and the associated indicator variables is called the *measurement model*. This relationship can either be reflective or formative. For a formative relationship, the indicators cause the latent variable, whereas in a reflective relationship

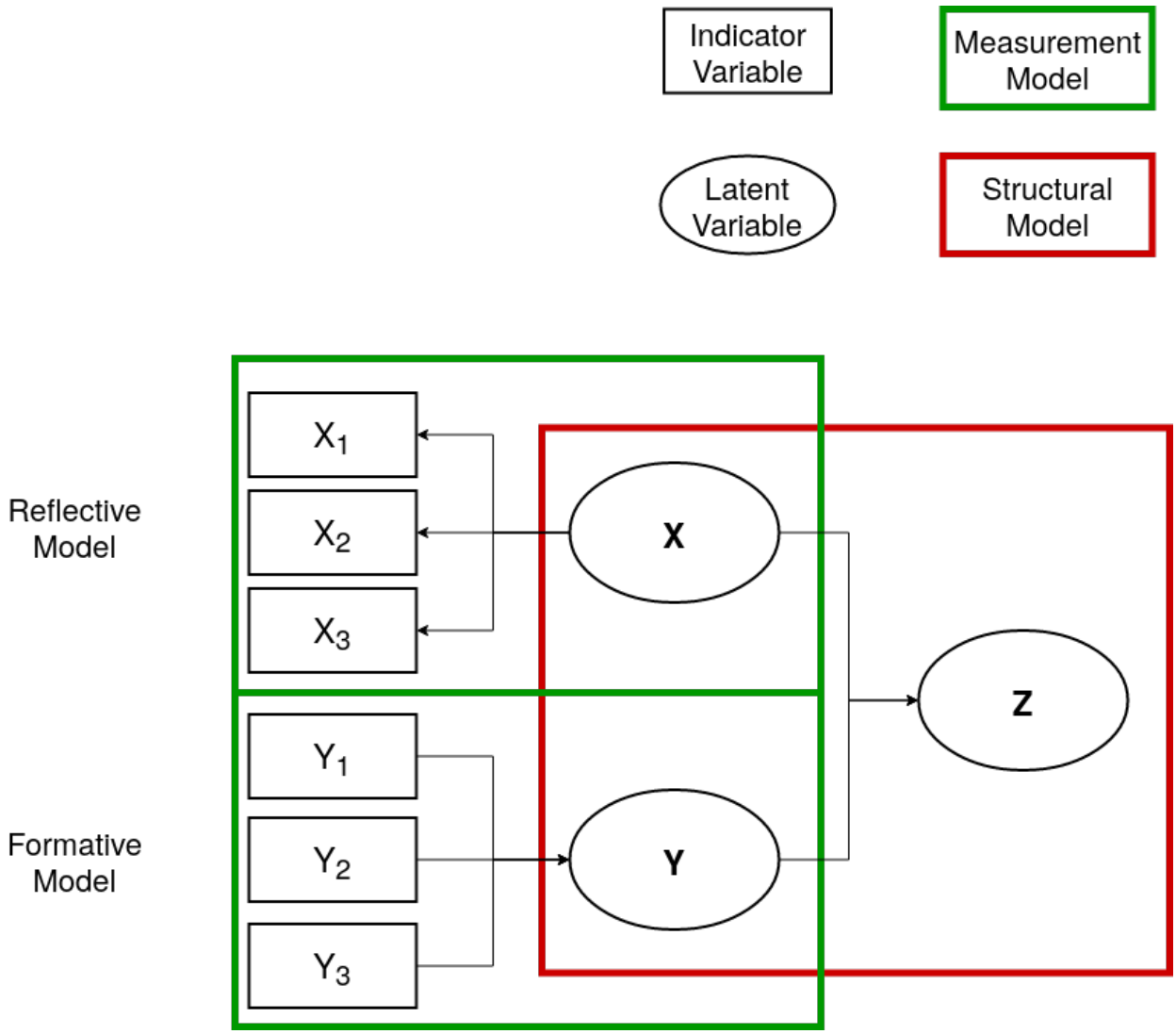


Figure 6.1: Theoretical SEM and constructs

the indicators are caused by the latent variable.

The second part of the model is called the *structural model*. The structural model represents the structural paths between the constructs. These paths capture the effect of one latent variable on another.

Figure 6.1 illustrates how these two models fit together. X, Y and Z represent the latent variables such as trust, transparency, cognitive load and satisfaction. X_1, X_2 and X_3 represent the indicator variables for X. In our model the indicator variables for these four latent variables are

calculated from the answers to our questionnaire, such as TRUST1, TRUST2, TRUST3, TRUST4 and TRUST5 for trust, as shown in Table A.1.

To assess these models, the causal relationship between variables is analysed using two powerful statistical approaches: exploratory factor analysis and structural path analysis. This enables the assessment of the measurement model and structural model simultaneously [79]. Two preeminent SEM methods are available to researchers: Covariance-based Structural Equation Modeling (CB-SEM) [68] and variance-based SEM (PLS-SEM) [67, 84]. The approach, strengths and weaknesses of each method are outlined below.

6.1.1 CB-SEM

CB-SEM is built on the common factor model [58]. This model approximates latent variables by common factors, as in common factor analysis [125]. The common factor model assumes the analysis should be based only on the common variance in the data. The specific variance and the error variance are removed before the model is examined. The method starts developing a solution by calculating the covariance between variables. This is done with the statistical objective of estimating the model parameters that minimise the differences between the observed sample covariance matrix and the covariance matrix estimated after the revised theoretical model is confirmed [55]. Once the optimal parameters are found the model can be used to confirm the existence of relationships between variables but not for predictions. As a result, CB-SEM is used principally for the confirmation of established theory.

6.1.2 PLS-SEM

Partial least squares is a prediction-oriented approach to SEM, primarily used for exploratory research, but also appropriate for confirmatory research. Maximising the variance explained in the dependent variables is the statistical objective of PLS-SEM [56]. This is done through an

iterative approach, that parameterises the latent variables by projecting the predicted variables and the observable variables to a new space. The algorithm then terminates after a predetermined number of iterations. or when an appropriate tolerance threshold is reached.

PLS-SEM is based on the composite model. This model approximates latent variables as weighted composites of observed variables, as in multivariate statistics such as canonical correlation analysis and principal component analysis [125]. The composite model includes common, specific, and error variance, and therefore uses all variance from the independent variables that can help to predict the variance in the dependent variables. This additional information allows the composite model approach to more effectively maximise the variance explained in the dependent variables. It is a result of this that when using PLS-SEM, a specific relationship is more likely to be statistically significant, given that it is present in a population. This is referred to as greater statistical power.

PLS-SEM achieves a greater statistical power than CB-SEM at all sample sizes, but particularly with a smaller sample size. Therefore, PLS-SEM is the recommended method for studies where $N < 100$ [59]. Consequently, PLS-SEM has been chosen as the methodology for analysing our theoretical framework of trust and its antecedents.

6.2 Related Work

In this section we outline some papers that apply PLS-SEM to measure trust and its antecedents. We then identify which factors are key drivers for success of DSSs.

6.2.1 PLS-SEM

The Swedish econometrician Herman Wold [147] developed the statistical methods underpinning PLS-SEM. PLS-SEM estimates partial model structures by combining principal component analysis with ordinary least squares regressions [57]. This statistical method allows us to analyse

complex interrelationships between observed and latent variables. A latent variable is a variable not observed, but inferred from the observed variables. Often, latent variables are aggregated observable variables, created to represent an underlying concept.

In the case of decision support, we can use latent variables to represent user trust and its antecedents in a system. Kim *et al.* [74] used PLS-SEM to assess the role of trust, perceived risk, and their antecedents in consumer decision making. The paper finds that user trust both directly and indirectly affects their intention to purchase online. PLS-SEM enable the authors to show that trust is a critical facet of the decision making process.

In another paper, Hsu *et al.* [63] use PLS-SEM to examine factors affecting intention to repurchase in online group-buying, including trust. The authors find that satisfaction with the website and satisfaction with sellers exert significant influences on user intention to repurchase. In addition, the results indicate that trust in a website is a strong predictor of satisfaction with a website, and trust in a seller is a strong predictor of satisfaction with the seller.

The literature indicates that both trust and satisfaction are key predictors for user intention to engage with a system or new technology. It is therefore important to gain an understanding of which features of DSSs influence trust and satisfaction. In our case, we observe the features enabled for each user: dynamic updates, user preferences and explanation, and assess the effect each feature has on our latent variables: transparency, cognitive load, trust and satisfaction.

6.2.2 Frameworks for trust and the acceptance of technology

The study of trust is difficult due to inconsistencies in the conceptualisation and measurement of trust in previous research [36]. One conceptualisation, proposed by Mayer et al. [93] is the Integrative Model of Organisational Trust, a theoretical framework examining trust in an organisational setting involving two individuals: a trustor (the individual trusting) and a trustee (the individual being trusted) [52]. In this model, trust is defined as "the willingness of a party to be vulnerable to the actions of another party based on the expectation that the other will perform

a particular action important to the trustor, irrespective of the ability to monitor or control that other party".

Trust of this kind is often broken down into three categories; affective, cognitive and overall trust [36]. Affective or emotional trust refers to the trust stemming from the emotional bond between the truster and the trustee. Affective trust implies a feeling of emotional security and belief that one's concern for another is reciprocated. On the other hand, cognitive trust is a willingness to be vulnerable to the trustee that is based on beliefs about the trustee's ability and integrity [93].

Mayer *et al.* also investigated the *competence* (knowledge, skills and competencies), *benevolence* (the extent to which a trustor believes that a trustee will act in the best interest of the trustor) and *integrity* (the extent to which the trustor perceives the trustee as acting in accord with a set of principles that the trustor finds acceptable). *Competence*, *benevolence* and *integrity* were postulated to be antecedents for the cognitive aspect of trust. Gill *et al.* [52] provided empirical evidence for the influence of these antecedents through a study that analysed participants propensity to trust a leader.

The cognitive aspect of trust is of particular interest when the trustee is not an individual but a technology or information system. We tend not to rely on emotional connections with information systems therefore in this domain the affective aspect of trust is less important. Mcknight *et al.* [94] have explored the idea of what makes people trust information systems, challenging the idea that "People trust people, not technology" [49]. The authors propose replacements for *competence*, *benevolence* and *integrity* when the trustee is technology. *Competence* is replaced with *functionality*, *benevolence* is replaced with *helpfulness*, and *integrity* is replaced with *reliability*. *Functionality* is the ability of a technology to deliver the capabilities promised, the competence of a person and the functionality of a technology are similar because they represent users' expectations about the trustee's capability. *Helpfulness* is the ability of the software help function to provide the advice necessary to complete a task, this replaces benevolence as technology has no

sense on moral agency. *Reliability* is the ability of technology to function with little or no down-time and to predictably respond to inputs, the reliability of technology is similar to the integrity of a person as they both define the ability to act in a predictable manner.

Another alternative model of trust in technology is the Technology Acceptance Model (TAM). TAM captures the intention of users when interacting with a computer system [29]. TAM is an adaption of the theory of reasoned action (TRA), an intention model that has been effective at predicting and explaining behaviour across many domains [8]. The goal of TAM is to provide the reasoning behind the acceptance of new technologies in a general way. TAM defines perceived usefulness and perceived ease of use as the two main drivers of a users attitude towards using a technology as shown in Figure 6.2.

For our framework, we accept the definition of trust as set out in the Integrative Model of Organisational Trust by Mayer [93], with the technological drivers of trust: *functionality*, *helpfulness* and *reliability* [94]. We also integrate into our model the concept of *satisfaction*. *Satisfaction* is how enjoyable a system is to use. We separate these two concepts as *trust* is needed for initial adoption whereas *satisfaction* is a key driver for continued use (citation needed).

Of the key drivers for *trust*, the decision support features we examine do not change the overall *functionality* or *reliability* but instead target *helpfulness*. This key driver of *trust* is closely related to *perceived ease of use* from TAM. For our framework we break the *perceived ease of use*/*helpfulness* into two adversarial factors: *transparency* and *cognitive load*. *Transparency* defines the ease of understanding the system and *cognitive load* is the amount of mental effort required to operate a system. This allows us to measure how these features contribute to *helpfulness* and transitively to *trust* by understanding whether they improve/reduce cognitive load or improve/reduce transparency. We also simultaneously assess the effect of these concepts on *satisfaction* to gain an understanding of how these features might effect continued use of dynamic decision support systems.

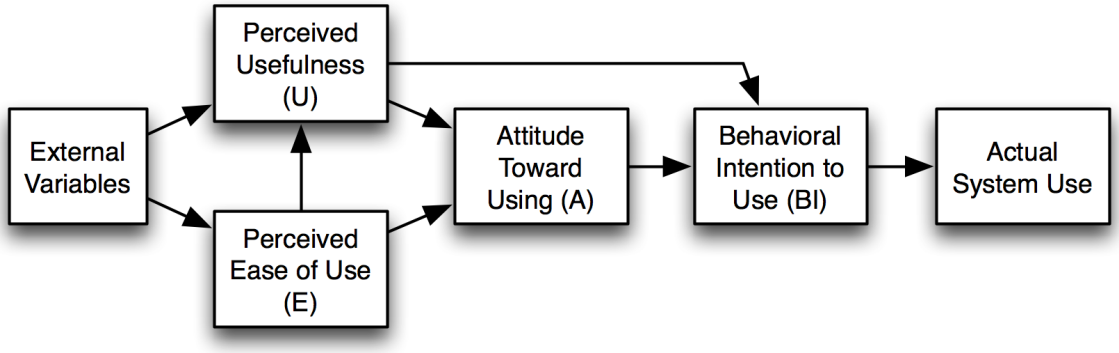


Figure 6.2: The technology acceptance model

6.2.3 Features

To investigate which features are useful for real-time decision support, we provided each experiment user with a set of features. In this section we will describe the features that can be enabled or disabled for different users.

6.2.3.1 Explanation

Explanation is the provision of reasoning describing how an output was reached. To facilitate explanation of the recommendations in the harbour management scenario, we added coloured bar charts. These charts display how a solution performed over a specific criterion, in the route selection popover, as shown in Figure 6.3.

A visual form of explanation was designed to quickly give decision makers an idea of how each criterion contributed towards the ranking of solutions. The size of the bar indicates how well a criterion value compares to the optimal value for the criterion. A bar is coloured green if it is greater than 75% of the optimal. A yellow bar for values between 50% and 75% and a red bar is used for values less than 50%. If the feature is disabled the user will see only the show/hide button, as shown in Figure 4.5.

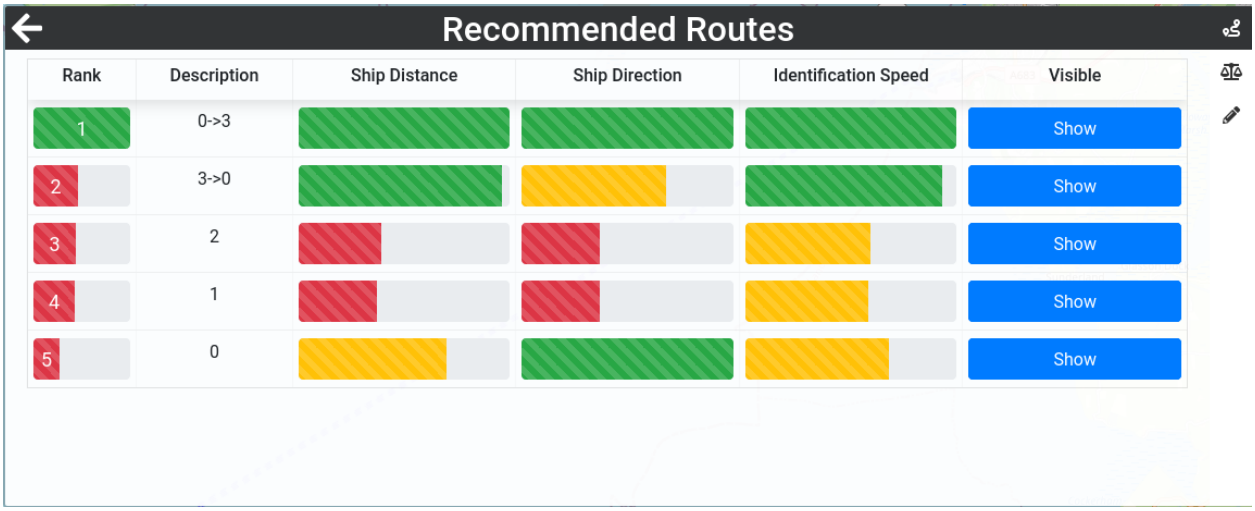


Figure 6.3: Routes with the explanation feature enabled

6.2.3.2 Preferences

The preferences feature allows decision makers to state their criteria preferences in the form of pairwise comparisons. For instance, a decision maker could express that "Ship Distance is more important than Ship Direction". The interface for setting preferences is shown in Figure 6.4. This menu is accessed by clicking on the scales icon on the route selection popover. Here the user can set their preferences through a drop down menu for each pair. AHP is applied to transform these pairwise comparisons into criteria weightings. The bars at the top of the interface provide visual feedback of the resultant weightings. If this feature is disabled the criteria tab is hidden and criteria weights are fixed evenly.

6.2.3.3 Dynamic Updates

The dynamic updates feature refers to the system's capacity to make new recommendations whilst the drone is in motion. If this feature is enabled, after a route is activated the route selection popover will not be hidden, and the system will continue to recommend updates to the plan. These updates could be small adjustments, or a complete change of plan, depending on how the scenario unfolds. If this feature is disabled when a route is active, the route selection popover is

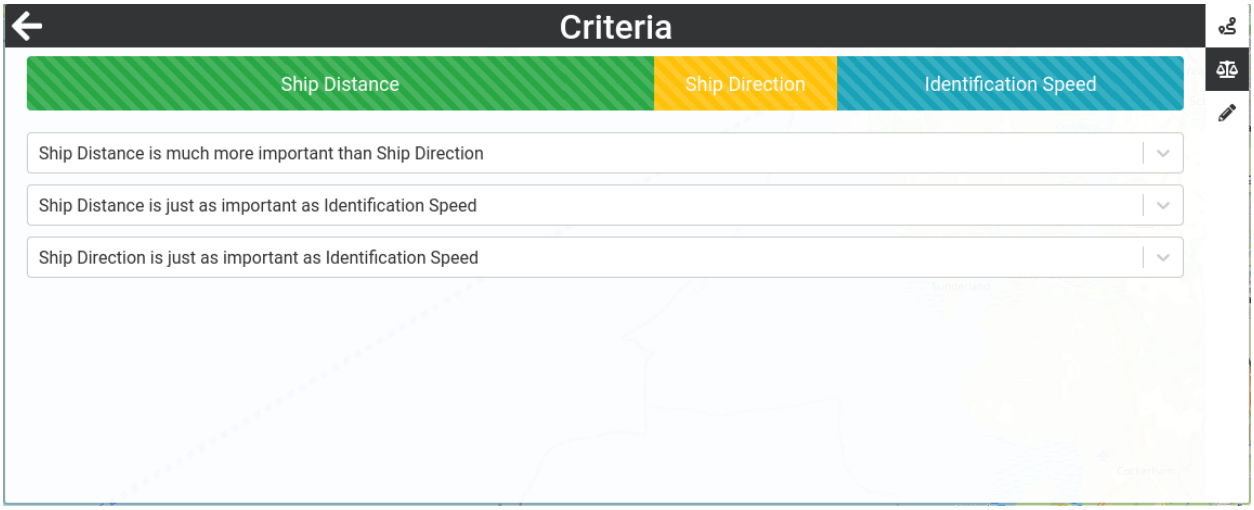


Figure 6.4: The interface for setting criteria preferences

hidden and no routes will be recommended until the drone returns to refuel.

6.3 Research model and hypotheses

6.3.1 Theoretical Model

Trust and satisfaction have been shown to be key measures of success for a DSS. The research model (Figure 6.5) shows how these measures are driven by two more practical targets, transparency and cognitive load. The model suggests that trust and satisfaction can both be improved by increasing the transparency of a system and lowering cognitive load.

6.3.1.1 Transparency

For our study, we define transparency as a measure of how well the user understands what actions are being performed. The understanding of how the recommendations are calculated, how an activated route will perform, and the current status of a task, are therefore all aspects of transparency. A system with greater transparency allows a decision maker to make more accurate judgements of the limits of a system. This means that transparency should improve a

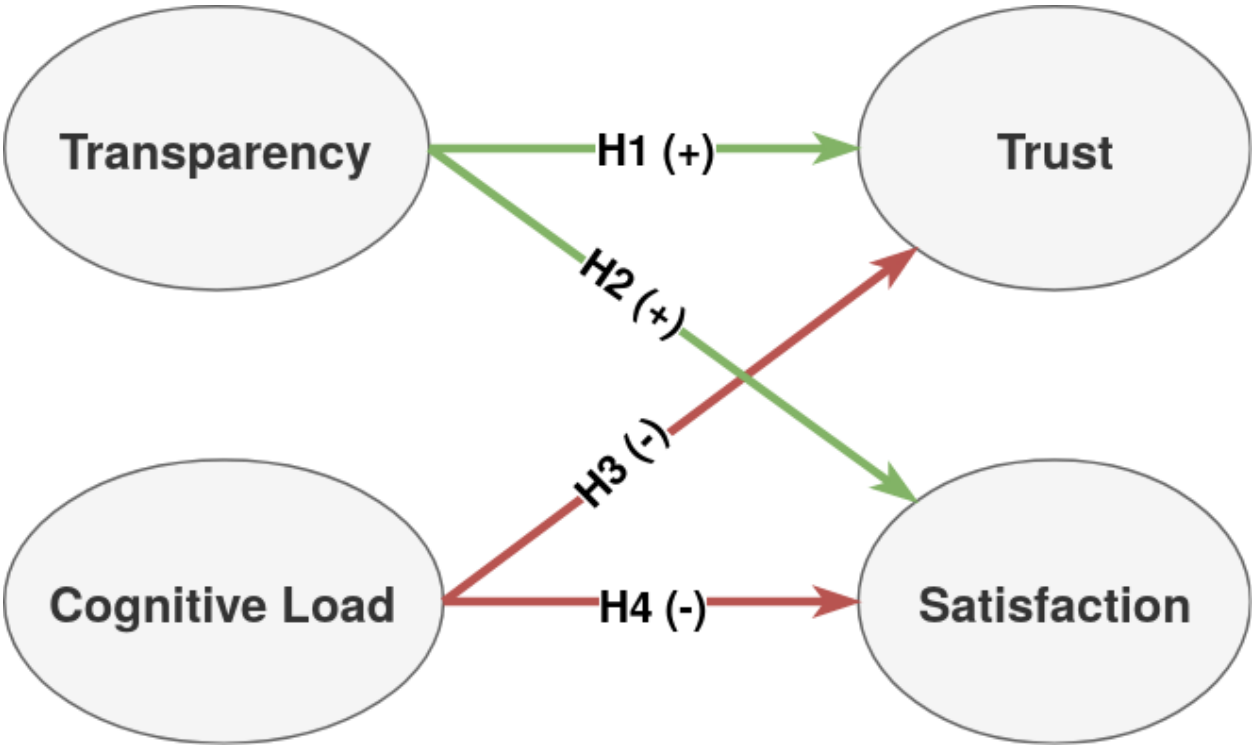


Figure 6.5: The basic theoretical framework

users' confidence in their decisions, and lead to greater trust in the system [146].

Understanding when to trust a recommendation creates a more competent human-computer team. This improvement in efficacy should create a higher user satisfaction with the system. Therefore we hypothesise:

Hypothesis 1. *Transparency increases trust.*

Hypothesis 2. *Transparency increases satisfaction.*

6.3.1.2 Cognitive Load

We define cognitive load as a measure of the amount of mental effort invested in operating a system. This mental effort detracts from the thought a user can put into selecting a solution. This gives the user less confidence in their decisions, undermining trust in the system. The added difficulty in correctly estimating the limits of the system may also cause lower user satisfaction.

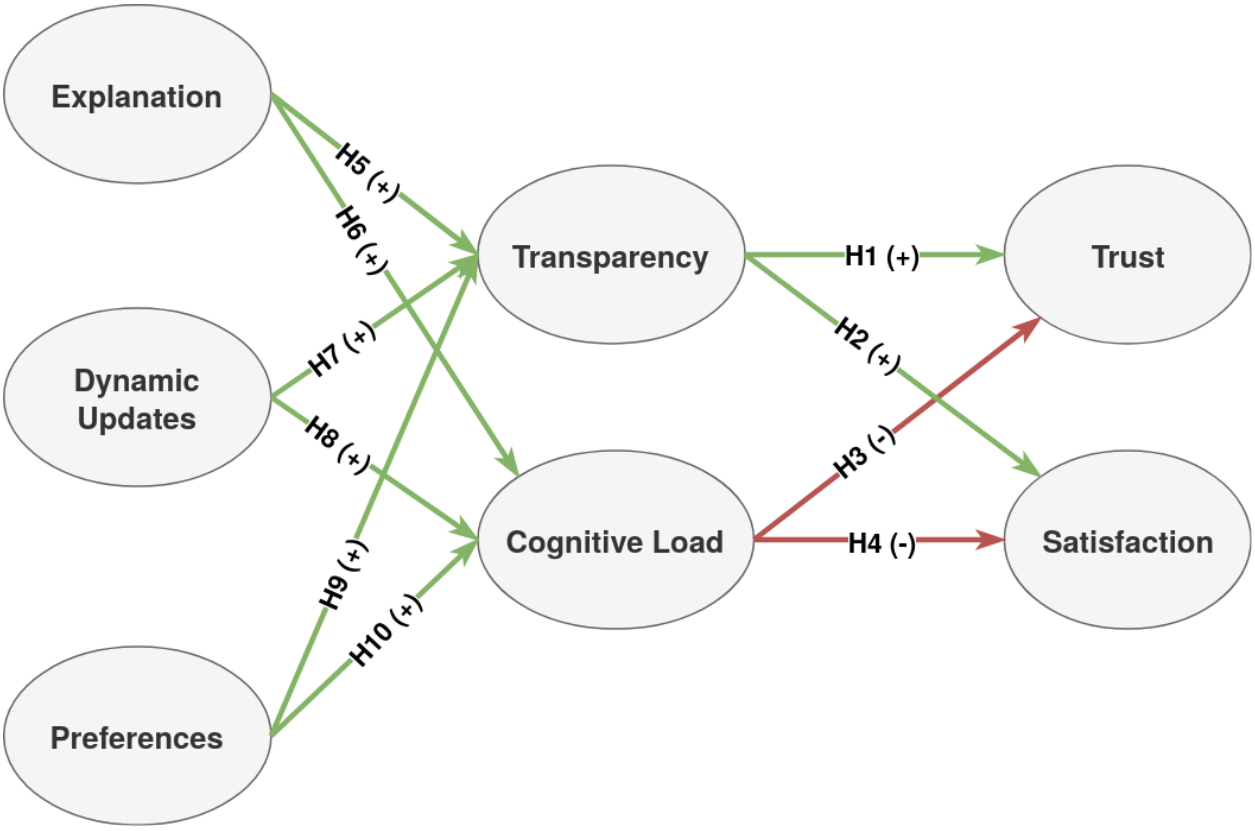


Figure 6.6: The research model

Therefore we hypothesise:

Hypothesis 3. *Cognitive load decreases trust.*

Hypothesis 4. *Cognitive load decreases satisfaction.*

6.3.2 Extended Research Model

Improving transparency can be specifically targeted by features of DSS. Architects of these systems should consider that all features are likely to increase cognitive load. Figure 6.6 shows how we hypothesise *explanation*, *dynamic updates* and *preferences* affect transparency and cognitive load.

6.3.2.1 Explanation

Explanation aims to provide reasoning describing how an output was reached. Our explanation feature provides this in the form of bar charts, shown in Figure 6.3, with detail given in Subsection 6.2.3.1. These charts show how the solutions perform over each criterion. This should give the user an understanding of how a ranking of solutions was reached, improving transparency [73]. This also gives the decision maker more information to consider; increasing cognitive load. We hypothesise:

Hypothesis 5. *Explanation increases transparency.*

Hypothesis 6. *Explanation increases cognitive load.*

6.3.2.2 Dynamic Updates

Dynamic updates is the ability to provide updated rankings of routes whilst the drone is in flight. Details for this feature are given in Subsection 6.2.3.3. This feature should increase transparency, as the user is provided with more information regarding the ranking of routes. This extra information comes at the cost of increased cognitive load whilst a route is active. Therefore we hypothesise:

Hypothesis 7. *Updates increase transparency.*

Hypothesis 8. *Updates increase cognitive load.*

6.3.2.3 User Preferences

The user preferences feature allows decision makers to state their criteria preferences in the form of pairwise comparisons. The details for our implementation of this feature are given in Subsection 6.2.3.2 and the interface is shown in Figure 6.4. This feature gives the user a lever to tailor recommendations. Another benefit is it gives the decision makers insight into the criteria

under consideration, and trade-offs made by the system, providing transparency [113]. At the same time, this creates an extra tab of input which must be read and understood, increasing cognitive load. We therefore hypothesise:

Hypothesis 9. *Preferences increase transparency.*

Hypothesis 10. *Preferences increase cognitive load.*

6.4 Approach

In this section we outline the method for collecting data for analysis by PLS-SEM using our harbour management DSS.

6.4.1 Data Collection

To test our theoretical framework, we surveyed users of our harbour management system. The users were Amazon Mechanical Turk (MTurk) [7] workers that attained our harbour management qualification in exchange for compensation at the rate of the UK minimum wage. MTurk is a crowdsourcing website for businesses (known as requesters) to hire remotely located "crowdworkers" to perform discrete on-demand tasks. One issue was that these workers are not specifically trained for harbour management. To address this issue we created a specific harbour management qualification, allowing us to test user efficacy and understanding of the harbour management task, before allowing them into the study.

As a further test for workers, the questionnaire also included two repeated questions. Workers giving different answers to the same question were marked as inappropriate. A total of 58 workers completed the task. After eliminating inappropriate responses and system errors, a total of 43 usable responses were included for construct validation and hypothesis testing. Of these 43 responses, 23 had explanation enabled, 24 had dynamic updates enabled and 24 had preferences enabled.

6.4.2 User Journey

The architecture of the system was previously given in Chapter 4, Figure 4.4. The interface was shown in Figure 4.5.

Users began the experiment by receiving a link to the *Static File Server* and credentials to authorise with the *User Server*. When authorised, the users were presented with the harbour management DSS. Each user was provided with a random selection of features enabled/disabled, recorded within the *User Database*. Users were required to complete a tutorial tailored to the features enabled within their interface. Once they had completed the tutorial, they were given three five-minute scenarios of increasing difficulty, presented as three stages to complete.

To incentivise and measure performance for this task, a score is recorded in the *User Database*. The user is awarded one point for each identified ship, and deducted one point for each unidentified ship, that enters the harbour zone. The score was recorded, and reset to zero between stages. The users were incentivised to maximise their score across all three stages through a bonus payment, paid to the best performing harbour manager.

After all three stages were complete, users were presented with the questionnaire, comprising questions relating to *trust*, *satisfaction*, *transparency*, and *cognitive load* (as shown in Table A.1). The questionnaire used 7-point Likert scales [70], with responses ranging from one (strongly disagree) to seven (strongly agree). The features and questionnaire answers for experiment users were then compiled for analysis. This analysis was completed by applying the answers to these questions as indicator variables that form a reflective or formative relationship with the relevant latent construct.

6.5 Results

To test our proposed research model we performed data analysis using PLS-SEM. Figure 6.7 shows the results calculated using the python package, PLSPM version 0.5.5 [105].

6.5.1 Reliability

To calculate internal consistency we used both Cronbach's Alpha (Alpha) and Composite Reliability (CR). Table 6.2 shows that the CR values are above 0.7 [47] and the Alpha values are all above 0.65 [77], satisfying the standard requirements for internal consistency. We also show that all Average Variance Extracted (AVE) values were higher than 0.50, the suggested minimum. An AVE greater than 0.5 indicates that more than 50% of the variance of the measurement items can be accounted for by the constructs.

6.5.2 Construct Validity

Construct validity was assessed via convergent validity and discriminant validity. Convergent validity is shown to be acceptable in Table A.1, with all item loadings greater than 0.50, and all items for each construct loading onto only one factor with an eigenvalue greater than 1.0.

To evaluate discriminant validity we applied the Heterotrait-Monotrait ratio of correlation (HTMT). Henseler *et al.* [60] proposed HTMT, providing evidence for its superior performance by means of a Monte Carlo-based simulation study, that showed that HTMT is able to achieve higher specificity and sensitivity rates (97% - 99%) compared with the Fornell-Lacker (20.82%). HTMT values close to 1 imply a lack of discriminant validity. Table 6.1 shows that all values are below the accepted threshold value of 0.9. This indicates discriminant validity among variables.

Table 6.1: HTMT ratios of correlation for constructs

Construct	TRAN	CL	TRUST
CL	0.892		
TRUST	0.872	0.619	
SAT	0.880	0.734	0.828

6.5.3 Structural model assessment

To assess the structural model we assessed both path coefficients and R^2. Both R^2 and path coefficients give us an indication of the model fit. Figure 6.7 shows the results. Transparency (TRAN) had a strong positive effect on trust (TRUST) and satisfaction (SAT). The path coefficients TRAN $\rightarrow$ TRUST and TRAN $\rightarrow$ SAT were both significant at the 0.01 level. Therefore Hypotheses 1 and 2 were supported.

Cognitive Load (CL) was shown to have no effect on trust, with a small negative effect on satisfaction. The CL $\rightarrow$ SAT path coefficient fell short of the 0.05 level, therefore we failed to find evidence to support Hypotheses 3 and 4.

For the paths from features, contrary to our hypothesis, explanation showed a small negative effect on transparency. This path and the hypothesised path from explanation to transparency were not significant, therefore not supporting Hypotheses 5 and 6.

The dynamic updates feature had a similar outcome, with neither Dynamic Updates $\rightarrow$ TRAN or Dynamic Updates $\rightarrow$ CL showing significance at $p < 0.05$. So hypotheses 7 and 8 were not supported.

The hypothesised paths from preferences to transparency and preferences to cognitive load showed weak positive and negative effects respectively, significant at $p < 0.05$. This validates Hypotheses 9 and 10.

The R^2 for trust and satisfaction were both 0.79, showing that transparency and Cognitive Load provide a strong explanation for trust and satisfaction in a system. On the other hand, transparency and cognitive load had an R^2 value of 0.27 and 0.21 respectively, implying that

Table 6.2: Descriptive statistics and reliability indices for constructs

Construct	Item	Model	Loading	AVE	CR	Alpha
TRAN	TRAN1	REFL	0.879	0.573	0.815	0.693
	TRAN2	REFL	0.774			
	TRAN3	REFL	0.442			
	TRAN4	REFL	0.851			
CL	CL1	FORM	0.866	0.758	0.926	0.879
	CL2	FORM	0.923			
	CL3	FORM	0.819			
TRUST	TRUST1	FORM	0.826	0.600	0.885	0.836
	TRUST2	FORM	0.831			
	TRUST3	FORM	0.795			
	TRUST4	FORM	0.772			
	TRUST5	FORM	0.631			
SAT	SAT1	FORM	0.832	0.676	0.873	0.802
	SAT2	FORM	0.882			
	SAT3	FORM	0.763			
	SAT4	FORM	0.808			

The questions corresponding to each item are given in Table A.1; REFL indicates a reflective relationship; FORM indicates a formative relationship.

these variables were largely driven by factors outside the scope of the study.

6.6 Conclusion and Discussion

In this chapter, we assessed the effect of enabling/disabling explanation, preferences and dynamic updates. To do this we produced a trust-based model for interactions with real-time DSSs, which was then validated using PLS-SEM.

PLS-SEM has been used extensively to investigate the antecedents of trust in electronic commerce [26, 74, 75] and the adoption of emerging technologies [80, 87, 90]. In these papers, the authors survey participants that have used pre-existing systems: online retailers, in the case of electronic commerce; or mobile banking services, as an example of an emerging technology. These systems are assessed for features and the effect of these features is measured on the re-

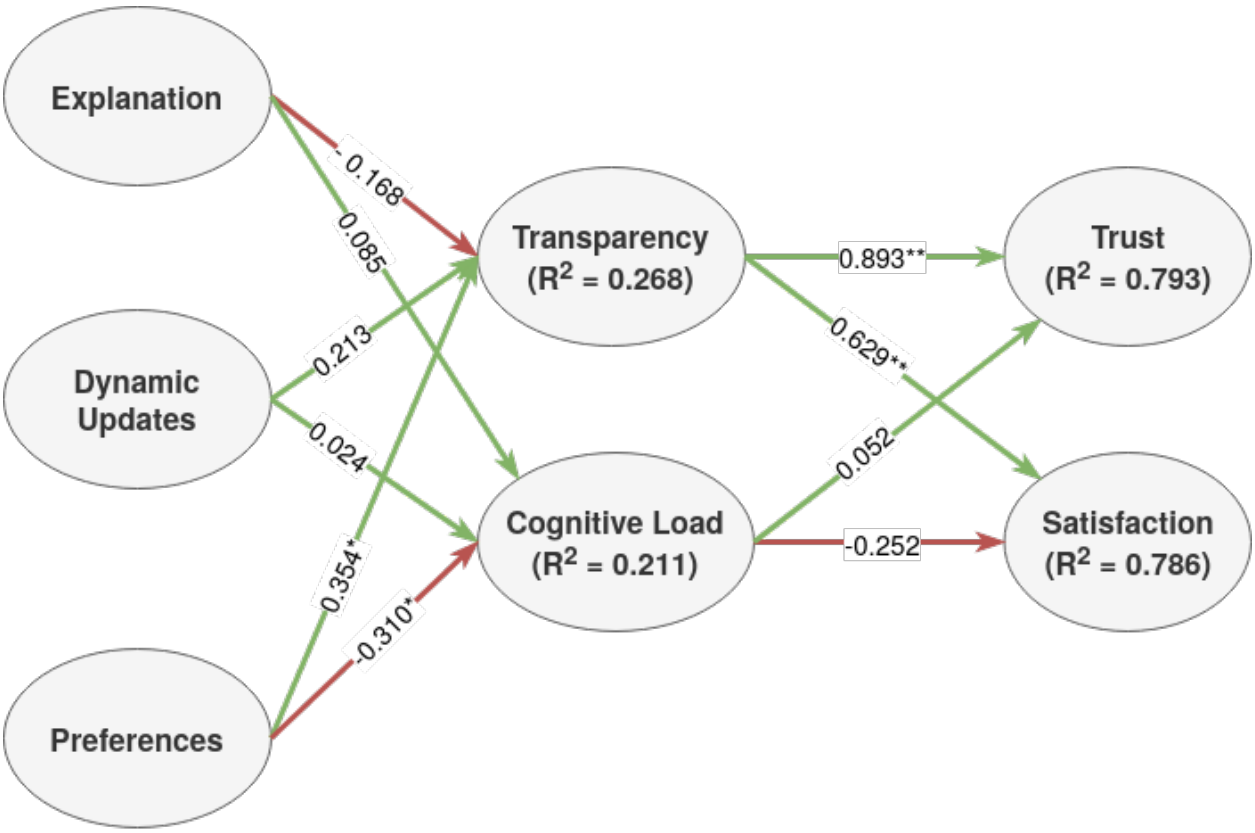


Figure 6.7: The results of PLS analysis

search model. Our study differs from this standard PLS-SEM approach as we assess participants engaging with a purpose-built system, controlling which features are enabled.

We have applied a methodology for the assessment of DSS and their features in the absence of clear success criteria. Through this method, we contribute a theoretical framework for the antecedents of trust in a real-time DSS. This model implies that transparency is a strong predictor of trust and satisfaction in a system.

An earlier study by Kawamoto *et al.* in clinical decision support has investigated validity of decision support features by assessing the improvement in clinical practice. This assessment was enabled through the measurement of patient outcomes or process measures [71]. The study

summarised 82 relevant comparisons of which 71 compared a clinical DSS with a control group (control-system comparisons) and 11 directly compared a system with the same system plus extra features (system-system comparisons). Another study conducted a similar assessment with 162 randomised control trials [115].

The above studies collate data from a multitude of papers, assessing features for purpose-built clinical DSS. This type of study is enabled in clinical decision support by the presence of clear success criteria (patient outcomes), to compare against the presence of decision support features. In lieu of this, our study instead assesses the effect of the presence of features on the constructs *trust*, *satisfaction*, *transparency* and *cognitive load* through the application of PLS-SEM.

The study by Kawamoto *et al.* investigated 15 decision support features. These were mostly specific to the clinical setting, but there was one feature that had significant overlap with our study: justification of decision support via provision of reasoning. We view this as a clinical domain-specific implementation of explanation. The study found that systems with this feature had a 12% uplift in success rate; this corresponds with Hypothesis 5. Unfortunately we did not find evidence to support this hypothesis. Instead, our implementation of explanation caused reduced transparency and increased cognitive load, therefore reducing trust and satisfaction. This may have been caused by our specific implementation of explanation for route selection.

The empirical results suggest that the transparency of a system positively impacted the trust and satisfaction that a user associated with a DSS. This is consistent with previous research, finding that transparency of a leader significantly affected trust [102]. While different, human teams share many similarities with human-computer teams. In both contexts, the leader or DSS prescribes a plan of action, with trust and transparency being significant factors shaping the outcomes of the interaction. Furthermore, a large portion of the variance within trust and satisfaction can be explained through the constructs of transparency and cognitive load. This validates the argument that architects should target transparency as an imperative aspect of real-time decision support.

The study also provides evidence that the ability to set user preferences improve transparency, indirectly improving trust and satisfaction. Our implementation of preferences was also found to reduce the cognitive load induced by a system. This feature provided necessary insight into the criteria considered for recommending routes. We hypothesise that the presence of explanation without preferences induced additional cognitive load, as users did not have adequate information to understand the system reasoning. This suggests that the ability to set preferences provides the user with a better understanding of the system and recommendations, enabling them to follow the reasoning provided by an explanation. Therefore reducing cognitive load.

Future work is needed to assess the generalisability of this model to other VRP and UAVTAP applications. We find that preferences improve transparency, but it is worth noting that our results are specific to the implementations within the harbour management system. Another limitation is that a low R^2 for transparency and cognitive load imply they were largely driven by factors outside the scope of the study. Therefore a future study taking into account more factors could provide an explanation of the drivers of these constructs. One such driver, outside the scope of this study is a users explicit propensity to trust, also known as dispostional trust [96]. Dispositional trust represents an individuals overall tendency to trust automation, independent of context or a specific system [61]. Previous studies have revealed that this type of trust is largely driven by biological and environmental factors such as: culture, age, gender and personality traits. The effect of dispositional trust can outweigh the effects of features of DSSs on trust. It is also evident from the low R^2 values that transparency (or understanding of the system from the user perspective) and cognitive load are largely driven by factors relating to the individual. As a result, we failed to achieve statistical significance regarding the effect of explanation and dynamic updates on cognitive load and transparency. This is likely caused by the relatively small impact of these two features. It would therefore be beneficial for a study to be conducted at a larger scale, to provide conclusive results of the effect of these features.

From a practical standpoint, our research highlights several features that an architect should

consider when building a real-time DSS. The findings imply that the implementation of these features should be carefully considered. For instance, we found that our implementation of explanation had an effect contrary to our hypotheses. This suggests that architects should be mindful that reasoning provided is concise and clear, because it is possible for an explanation to reduce transparency.

7 Conclusion

In this chapter, we summarise the contributions of the thesis then present some possible avenues for future research.

7.1 Reflections

This research investigated which decision support features and methodologies would be suitable to employ as part of trustable decision support for dynamic applications. Providing a trustable DSS for dynamic problems requires a broad range of problems to be tackled. To help deal with these problems, a framework has been proposed. This framework consists of eight desiderata for DSSs, together offering a basis for trustable dynamic DSSs. We illustrated these desiderata by showing how they surface in our two case studies, train journey planning and harbour management, and have given examples of how they can surface in similar applications. These desiderata help to inform architects of DSSs on how to equip their systems with the capabilities needed to handle dynamic problems.

The major issue facing an architect of a dynamic DSS is how to recommend a continuously revised recommendation, whilst incorporating decision makers' preferences. To solve this, we have outlined a dynamic GA that can be used to incrementally refine recommendations, with a specific emphasis on the production of diverse recommendations. This algorithm applies the principles of DMOEAs, combined with MCDM methods as a fitness function, to support the

cornerstones of dynamic decision support. The approach applies preferences in an a priori fashion to maintain a small set of candidate solutions, compared to previous algorithms that maintain a large POF. The POF then requiring navigation through a separate application of an MCDM method.

The research also includes an evaluation of MCDM methods in terms of our desiderata. WPM, AHP, TOPSIS and PROMETHEE were assessed on their ability to provide high stability of results and consistent trade-offs between objectives. The research concluded that TOPSIS is an appropriate method for trustable dynamic decision support. Applying TOPSIS as a fitness function for our dynamic GA therefore provides an approach to decision support that enables declarative specification of preferences *(Desiderata 1)*, dynamic revision of recommendations *(Desiderata 2)*, high stability of results *(Desiderata 3)*, high diversity of results *(Desiderata 7)*, and consistent trade-offs between criteria *(Desiderata 8)*.

To understand the factors contributing to trustability in dynamic decision support, we have put forward a theoretical framework modelling trust and its antecedents in a real-time DSS. We applied PLS-SEM to evaluate this model through our harbour management case study. This model highlights how transparency and cognitive load drive trust and satisfaction, giving architects of DSSs more tangible concepts to target when they design a UI. An assessment of the effect of explanation, preferences and dynamic updates on our model has also been included. This gives architects an indication of how UI features affect transparency and cognitive load and which features are likely to assist decision making in a dynamic environment.

7.2 Future Research

In this section we propose some directions for future research in the field of dynamic decision support.

7.2.1 Multi-objective ant colony optimisation for decision support

We outlined a dynamic GA applying MCDM methods as a fitness function as a means to provide dynamic decision support. GAs are a family of meta-heuristic search algorithms, inspired by the process of natural selection. An alternative family of meta-heuristic search algorithms is Ant Colony Optimisation algorithms (ACOs) [38]. ACOs are inspired by the methodology employed by ants to forage for food. These ants lay and follow pheromone trails, using pheromones as a means of coordinating their search [37]. ACOs copy this methodology, generating probabilistic paths through the parameter space. These paths are referred to as pheromone trails. A solution is generated by applying an "ant" to follow the pheromone trail. The algorithm begins with the most basic pheromone trail, which is equally likely to generate every solution (this trail generates a random solution from the space of options). Multiple ants are employed each generation to derive a set of solutions, a fitness value is calculated for each solution, and these values are used to update the pheromone trail. Through updating the pheromone trail, we amend the distribution of generated solutions to increase the chance of generating "good" solutions.

In nature, both search algorithms operate under dynamic environments. They are both therefore prime candidates as a starting point for dynamic decision making problems [37, 51]. The main differences between the two algorithms come from the abstraction that encodes the progress of the algorithm. For a GA, the progress of an ongoing search is represented by a collection of intermediate solutions. An intermediate solution of a dynamic GA can be generated by taking the solution from the population with the highest fitness. ACOs encode the progress of a search as probabilistic paths through graphs, where these graphs inform the process of building a solution. For a dynamic ACO, an intermediate solution can be generated by applying N ants to this graph to generate N solutions and taking the solution with the highest fitness.

For example, in our harbour management scenario, a solution may be $A \rightarrow B \rightarrow C$. The progress of an ongoing GA search would be represented by a population of these solutions, e.g.

$\{A \rightarrow B \rightarrow C, A \rightarrow C \rightarrow B\}$. The progress of an ongoing ACO search would be represented by a probabilistic graph that encodes the process of building a route, as shown in Figure 7.1. The route is then built by applying "ants" to follow the pheromone trail, each ant will derive a route.

ACOs have been found to lend themselves well to changing environments [32–34, 41, 153]. One reason for this is the nature of the ACO encoding of progression. The algorithm can adjust branches and their weights to efficiently adapt to dynamic problems. For example, if ship A is no longer available to visit, the GA is required to remove A from each solution in the population. The ACO is only required to remove the node from the pheromone trail. If this proves to be faster to adapt than the GA approach, it could cut down recalculation times, giving decision makers more time to react to rapidly evolving situations.

Further work could produce a dynamic ACO, applying MCDM methods as a fitness function, and compare the method with our GA-based methodology. A possible approach to this would be to analyse the convergence of each algorithm. Comparing the initial convergence time, and the convergence time as the situation evolves, could provide insight into potential benefits of ACOs over GAs for dynamic decision support.

7.2.2 Integrating frameworks for dynamic decision support

Our work identified stability of results as a desired characteristic of trusted dynamic decision support. Aggregation-based dynamic decision making frameworks have been highlighted in the literature as an approach to improving stability in dynamic DSSs [121].

Aggregation-based frameworks provide a methodology for taking a stream of criteria values and collating them into a single value for a time period through the application of aggregation functions. Further investigations could evaluate the effect of these frameworks on the stability of rankings. The framework put forward by Campanella *et al.* [22] suggested a variety of aggregation functions, including: average aggregation functions, conjunctive aggregation functions, disjunctive aggregation functions and mixed aggregation functions. Further work could evaluate

how these functions effect the stability and quality of results.

The crux of this approach is to include temporal factors as a method of selecting which solution will be best in the future (or over the period of time required to execute a recommendation). Therefore providing a generalisable approach to the assessment of dynamic criteria values. Our approach uses domain knowledge to assess what will happen when a route is carried out. Generally, with better domain knowledge comes better results; knowing a ship is a ship makes it much easier to predict its position in the future. Nevertheless, there is scope to combine both methods. As new information becomes available, the criteria values still evolve over time e.g. when a ship changes direction. It would therefore be of interest to evaluate how these frameworks compare to and complement the domain-based prediction approach.

7.2.3 Trust calibration versus improving trust

Trust is an important factor underpinning the effectiveness of the human-computer team formed when a decision maker utilises a DSS. The more a decision maker trusts a DSS, the more likely they are to follow a recommendation. If a DSS is trusted completely, assuming that everyone recommendation is taken, the human-computer team functions as well as the computer component. Whereas, if the DSS is completely untrustworthy, assuming that the human overrides every recommendation, the human-computer team will function only as well as the human component [9, 78]. Therefore, for the human-computer team to operate at a higher capacity than both of the components, it is required for trust to be calibrated effectively.

Trust calibration is a measure of the decision makers' ability to know when to follow a recommendation and when to override it. Calibrating trust can be posed as a problem of communication: the system must quickly communicate its abilities and limitations to a user [133]. The level of trust calibration can therefore be viewed as one facet of how well the decision maker understands the system (referred to as *transparency* in this research).

This research investigates the antecedents of trust (including *transparency*), by building a

model of trust and its antecedents then measuring the effect of various DSS features on the model. This yields understanding of trust and its role in real-time decision support, providing insights into how to build trust between the decision maker and the DSS. Whilst useful, this approach focuses on improving trust, rather than addressing the problem of trust calibration. Reducing the visibility of errors is a potential avenue to improve a decision makers trust in a DSS. It is therefore possible that ignoring the subtleties of trust calibration in favour of improving trust could undermine the raison-d'etre of DSSs, effectively removing the human from the loop.

Features that help the decision maker to understand when to override the computer are of particular relevance to trust calibration. In terms of our desiderata for dynamic decision support, these features are data provenance *(Desiderata 5)* and explanation of outputs *(Desiderata 6)*. Data provenance enables a user to check the integrity of the data underlying the recommendations and explanation provides insight into the reasoning. Both of these features should aim to bring into focus the limits of a DSS, surfacing errors in the data or reasoning respectively. It is difficult to know whether improving the visibility of errors in a system would undermine trust or build it. It is also important to note that either could be beneficial, depending on the circumstance. Therefore, for these two desiderata, and other features of DSSs that target trust calibration, it would be ideal if we could evaluate the effect on trust calibration.

An approach to this would be to evaluate the performance of the decision maker, the computer component and the human-computer team separately. By comparing these three metrics it would be possible to yield a latent variable representing how well trust is calibrated between the human-computer team. Through the application of PLS-SEM, researchers could measure the effect of DSS features on trust calibration, rather than on trust. It would also be beneficial to create a model of trust calibration (including trust and its antecedents) in the context of dynamic decision support, to provide a better understanding of the concept.

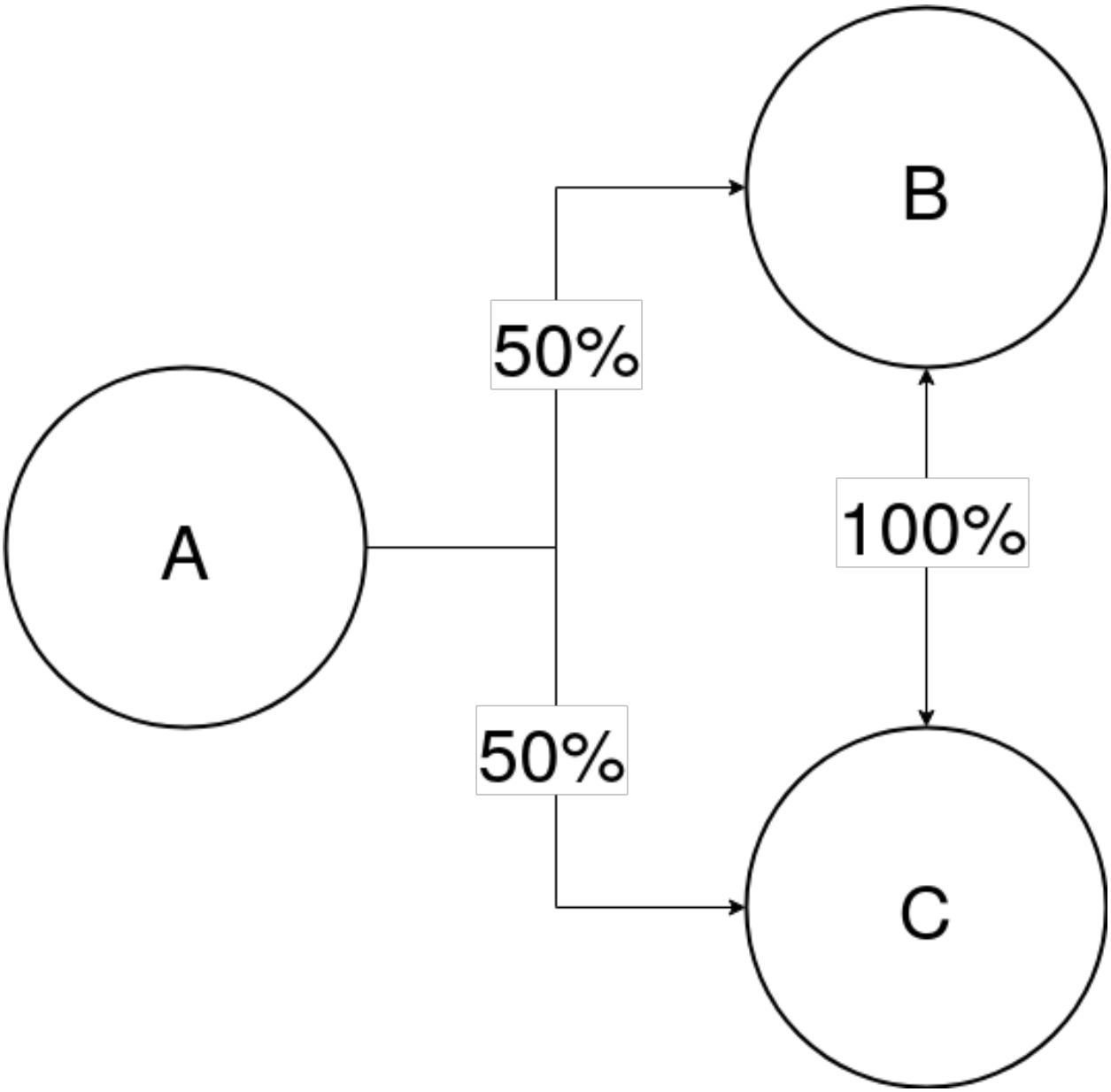


Figure 7.1: A pheromone trail for building a route visiting ships A, B and C; each route starts at A, selects B or C with even probability, then the route is finished with the unselected ship; this pheromone trail is equivalent to the population $\{A \to B \to C, A \to C \to B\}$ in a GA.

A Appendix

A.1 Harbour Management Questionnaire

Table A.1: Proposed Measurement Items for Constructs

Construct	Question	Loading
TRAN1	I understood why the recommended routes were provided over alternatives.	0.879
TRAN2	The recommended routes were clearly justified.	0.774
TRAN3	I was aware of trade-offs when I was choosing routes.	0.442
TRAN4	I felt I was kept up to date with events in the scenario.	0.851
		Eigenvalue : 2.18
CL1	The interface of the exercise was complex.	0.866
CL2	The way the interface presented information was distracting.	0.923
CL3	It was difficult to find relevant information for selecting routes.	0.819
		Eigenvalue : 2.42
TRUST1	I felt that I could trust the application.	0.826
TRUST2	The recommended routes were good.	0.831
TRUST3	The recommended routes helped me to make decisions.	0.795
TRUST4	The recommended routes often turned out as expected.	0.772
TRUST5	I believe the application has been designed to enhance my decision making.	0.631
		Eigenvalue : 3.05
SAT1	I feel satisfied with the overall experience of using the application.	0.832
SAT2	I enjoyed using the application.	0.882
SAT3	I think route recommendations are a good idea.	0.763
SAT4	I feel good about the decisions I made with the support of the application.	0.808
		Eigenvalue : 2.55

A.2 Harbour Management Qualification

The harbour management exam assessed users for the trustability study. The questions were designed to assess a users' comprehension of the briefing and to remove users that were completing tasks hastily.

Question	Answers	Correct
What colour is attributed to unidentified ships?	Red.	
	Blue.	
	Yellow.	✓
	Green.	
What are the three criteria for route recommendations?	Ship direction, ship distance, identification speed.	✓
	Ship direction, ship speed, identification speed.	
	Ship speed, ship direction, identification time.	
	Ship distance, ship direction, identification time.	
How many scenarios will you have to complete?	1.	
	2.	
	3.	✓
	4.	
If an unidentified ship reaches the harbour you will...	Gain 1 point.	
	Gain 2 points.	
	Lose 1 point.	✓
	Lose 2 points.	
How many routes can be active at once?	1.	✓
	2.	
	3.	
	4.	
Ship distance is...	A measure of the distance of a ship from the harbour.	
	A measure of the distance between ships on a route.	
	A measure of the distance of a ship from the drone.	
	A measure of the distance of ships on a route from the harbour.	✓
Ship direction is...	A measure of how directly ships along a route are travelling away from the harbour.	
	A measure of how directly ships along a route are travelling towards the harbour.	✓
	A measure of how directly a ship is heading towards the drone.	
	A measure of how directly a ship is heading away from the drone.	
Identification speed is...	A measure of the time taken for a route to be completed.	
	A measure of the time taken for a ship to arrive at a harbour.	
	A measure of the time taken for a drone to identify a ship.	
	A measure of the time taken to identify each ship over a route.	✓
The length of a route is limited by the...	Number of ships.	
	The fuel of the drone.	✓
	The speed of the drone.	
	The distance from the harbour.	
If none of the recommended routes are satisfactory...	There is no other option.	
	A route can be drawn by dragging a line between ships.	
	A route can be drawn by clicking ships in order, on the draw a route tab.	✓

Table A.2: Questions and answers for the harbour management qualification.

A.3 Acronyms

ACO Ant Colony Optimisation algorithm. 149

AHP Analytic Hierarchy Process. 12, 27, 60, 76, 105

Alpha Cronbach's Alpha. 140

AO Average Overlap. 94, 107

AVE Average Variance Extracted. 140

BWM Best Worst Method. 59

CB-SEM Covariance-based Structural Equation Modeling. 127

CDF Cumulative Distribution Function. 69

CPSS Cyber-Physical-Social System. 18

CR Composite Reliability. 140

DCG Discounted Cumulative Gain. 107

DMOEA Dynamic Multi-Objective Evolutionary Algorithm. 20, 147

DSS Decision Support System. 12, 16, 18, 32, 49, 75, 105, 128, 147

ELECTRE ELimination Et Choix Traduisant la REalité. 40, 106

GA Genetic Algorithm. 12, 44, 56, 110, 147

HTMT Heterotrait-Monotrait ratio of correlation. 140

MCDM Multi-Criteria Decision Making. 18, 32, 50, 75, 76, 105

MOEA Multi-Objective Evolutionary Algorithm. 19, 56, 76

MTurk Amazon Mechanical Turk. 138

PC Pairwise Comparison. 18

PEG Pareto-Edgeworth-Grierson method. 107

PLS-SEM Partial Least Squares Structural Equation Modeling. 28, 125, 148

POF Pareto-optimal front. 50, 78, 148

PROMETHEE Preference Ranking Organization METHod for Enrichment Evaluation. 12, 27, 105

SEM Structural Equation Modeling. 125

SPE Stream Processing Engine. 19, 43, 65

TAM Technology Acceptance Model. 131

TOPSIS Technique for Order of Preference by Similarity to Ideal Solution. 12, 27, 59, 76, 105

UAV Unmanned Aerial Vehicle. 75

UAVTAP UAV Task Assignment Problem. 76

UI User Interface. 27, 75, 101, 148

VRP Vehicle Routing Problem. 76

WPM Weighted Product Model. 12, 27, 105

WSM Weighted Sum Method. 33, 106

Bibliography

[1] Lorenzo Abbatecola et al. "A Decision Support Approach for Postal Delivery and Waste Collection Services". In: *IEEE Transactions on Automation Science and Engineering* 13.4 (Oct. 2016), pp. 1458–1470. ISSN: 1545-5955. DOI: 10.1109/TASE.2016.2570121.

[2] Lorenzo Abbatecola et al. "A Distributed Cluster-Based Approach for Pick-Up Services". In: *IEEE Transactions on Automation Science and Engineering* 16.2 (2019), pp. 960–971. DOI: 10.1109/TASE.2018.2879875.

[3] Lorenzo Abbatecola et al. "A New Cluster-Based Approach for the Vehicle Routing Problem with Time Windows". In: *2018 IEEE 14th International Conference on Automation Science and Engineering (CASE)*. 2018, pp. 744–749. DOI: 10.1109/COASE.2018.8560419.

[4] Edward Abel, Ludmil Mikhailov, and John Keane. "Group aggregation of pairwise comparisons using multi-objective optimization". In: *Information Sciences* 322 (Nov. 2015), pp. 257–275. ISSN: 00200255. DOI: 10.1016/j.ins.2015.05.027.

[5] Edward Abel et al. "Pairwise comparisons or constrained optimization? A usability evaluation of techniques for eliciting decision priorities". In: *International Transactions in Operational Research* (Nov. 2020), itor.12907. ISSN: 0969-6016. DOI: 10.1111/itor.12907.

[6] Behrouz Alavi, Madjid Tavana, and Hassan Mina. "A Dynamic Decision Support System for Sustainable Supplier Selection in Circular Economy". In: *Sustainable Production and*

Consumption 27 (July 2021), pp. 905–920. ISSN: 23525509. DOI: 10.1016/j.spc.2021.02.015.

[7] *Amazon Mechanical Turk*. https://www.mturk.com/.

[8] Icek Azjen. "Understanding attitudes and predicting social behavior". In: *Englewood Cliffs* (1980).

[9] Gagan Bansal et al. "Beyond Accuracy: The Role of Mental Models in Human-AI Team Performance". In: *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing* 7.1 (2019), p. 19.

[10] L. Basile and Livia D'Apuzzo. "Transitive matrices, strict preference order and ordinal evaluation operators". In: *Soft Computing* 10.10 (2006), pp. 933–940. ISSN: 14327643. DOI: 10.1007/s00500-005-0020-z.

[11] Valerie Belton and Tony Gear. "On a short-coming of Saaty's method of analytic hierarchies". In: *Omega* 11.3 (Jan. 1983), pp. 228–230. ISSN: 03050483. DOI: 10.1016/0305-0483(83)90047-6.

[12] Z. Bingul, A. Sekmen, and S. Zein-Sabatto. "Evolutionary approach to multi-objective problems using adaptive genetic algorithms". In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics* 3 (2000), pp. 1923–1927. ISSN: 08843627. DOI: 10.1109/ICSMC.2000.886394.

[13] Z. Bingül et al. "Genetic algorithms applied to real time multiobjective optimization problems". In: *Conference Proceedings - IEEE SOUTHEASTCON* (2000), pp. 95–103. ISSN: 07347502. DOI: 10.1109/SECON.2000.845432.

[14] Irina Botan et al. "SECRET: a model for analysis of the execution semantics of stream processing systems". In: *Proceedings of the VLDB Endowment* 3.1-2 (2010), pp. 232–243.

[15] J. P. Brans, Ph Vincke, and B. Mareschal. “How to select and how to rank projects: The Promethee method”. In: *European Journal of Operational Research* 24.2 (Feb. 1986), pp. 228–238. ISSN: 03772217. DOI: 10.1016/0377-2217(86)90044-5.

[16] J. P. Brans and Ph. Vincke. “Note—A Preference Ranking Organisation Method”. In: *Management Science* 31.6 (June 1985), pp. 647–656. ISSN: 0025-1909. DOI: 10.1287/mnsc.31.6.647.

[17] J. Pierre Brans and Bertrand Mareschal. “Promethee V: Mcdm Problems With Segmentation Constraints”. In: *INFOR: Information Systems and Operational Research* 30.2 (May 1992), pp. 85–96. ISSN: 0315-5986. DOI: 10.1080/03155986.1992.11732186.

[18] Jean Pierre Brans and Bertrand Mareschal. “The promethee VI PROCEDURE: How to differentiate hard from soft multicriteria problems”. In: *Journal of Decision Systems* 4.3 (Jan. 1995), pp. 213–223. ISSN: 21167052. DOI: 10.1080/12460125.1995.10511652.

[19] Berndt Brehmer and Peter Thunholm. “C2 after Contact with the Adversary: Execution of Military Operations as Dynamic Decision Making”. In: (June 2011), p. 38.

[20] Percy Williams Bridgman. *Dimensional analysis*. Yale university press, 1922.

[21] Peter Buneman, Sanjeev Khanna, and Tan Wang-Chiew. “Why and where: A characterization of data provenance”. In: *International conference on database theory*. Springer. 2001, pp. 316–330.

[22] Gianluca Campanella and Rita A. Ribeiro. “A framework for dynamic multiple-criteria decision making”. In: *Decision Support Systems* 52.1 (Dec. 2011), pp. 52–60. ISSN: 01679236. DOI: 10.1016/j.dss.2011.05.003.

[23] Paris Carbone et al. “Apache flink: Stream and batch processing in a single engine”. In: *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering* 36.4 (2015).

[24] M. Cerreta and P. De Toro. "Urbanization suitability maps: A dynamic spatial decision support system for sustainable land use". In: *Earth System Dynamics* 3.2 (2012), pp. 157–171. ISSN: 21904979. DOI: 10.5194/esd-3-157-2012.

[25] Hongyi Chen and Dundar F Kocaoglu. "A sensitivity analysis algorithm for hierarchical decision models". In: *European Journal of Operational Research* 185.1 (2008), pp. 266–288.

[26] Chao Min Chiu et al. "Re-examining the influence of trust on online repeat purchase intention: The moderating role of habit and its antecedents". In: *Decision Support Systems*. Vol. 53. 4. North-Holland, Nov. 2012, pp. 835–845. DOI: 10.1016/j.dss.2012.05.021.

[27] Bruno N. Coelho et al. "A multi-objective green UAV routing problem". In: *Computers and Operations Research* 88 (Dec. 2017), pp. 306–315. ISSN: 03050548. DOI: 10.1016/j.cor.2017.04.011.

[28] Walton Pereira Coutinho, Maria Battarra, and Jörg Fliege. "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review". In: *Computers and Industrial Engineering* 120.June 2017 (2018), pp. 116–128. ISSN: 03608352. DOI: 10.1016/j.cie.2018.04.037.

[29] F.D. Davis, R.P. Bagozzi, and P.R. Warshaw. "User Acceptance of Information Technology : a Comparison of Two Theoretical Models *". In: *Management Science* 35.8 (1989), pp. 982–1002.

[30] Kalyanmoy Deb. "Multi-objective optimisation using evolutionary algorithms: an introduction". In: *Multi-objective evolutionary optimisation for product design and manufacturing*. Springer, 2011, pp. 3–34.

[31] Kalyanmoy Deb et al. "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II". In: *International conference on parallel problem solving from nature*. Springer. 2000, pp. 849–858.

[32] Gianni Di Caro and Marco Dorigo. “Ant colonies for adaptive routing in packet-switched communications networks”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1998, pp. 673–682.

[33] Gianni Di Caro and Marco Dorigo. *AntNet: A mobile agents approach to adaptive routing*. 1997.

[34] Gianni Di Caro and Marco Dorigo. “Mobile agents for adaptive routing”. In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. Vol. 7. IEEE. 1998, pp. 74–83.

[35] Alina Diaz-Curbelo, Rafael Alejandro Espin Andrade, and Ángel Manuel Gento Municio. “The Role of Fuzzy Logic to Dealing with Epistemic Uncertainty in Supply Chain Risk Assessment: Review Standpoints”. In: *International Journal of Fuzzy Systems* (2020), pp. 1–23.

[36] Kurt T Dirks and Donald L Ferrin. “Trust in leadership: meta-analytic findings and implications for research and practice.” In: *Journal of applied psychology* 87.4 (2002), p. 611.

[37] Marco Dorigo and Thomas Stützle. “Ant colony optimization: overview and recent advances”. In: *Handbook of metaheuristics* (2019), pp. 311–351.

[38] Marco Dorigo and Thomas Stützle. “The ant colony optimization metaheuristic: Algorithms, applications, and advances”. In: *Handbook of metaheuristics*. Springer, 2003, pp. 250–285.

[39] Finale Doshi-Velez and Been Kim. “Towards A Rigorous Science of Interpretable Machine Learning”. In: (Feb. 2017).

[40] Haiming Du et al. “Elitism and distance strategy for selection of evolutionary algorithms”. In: *IEEE Access* 6 (Aug. 2018), pp. 44531–44541. ISSN: 21693536. DOI: 10.1109/ACCESS.2018.2861760.

[41] Frederick Ducatelle, Gianni A Di Caro, and Luca M Gambardella. “Principles and applications of swarm intelligence for adaptive routing in telecommunications networks”. In: *Swarm Intelligence* 4.3 (2010), pp. 173–198.

[42] Bapi Dutta et al. “Post factum analysis in TOPSIS based decision making method”. In: *Expert Systems with Applications* 138 (2019), p. 112806. ISSN: 09574174. DOI: 10.1016/j.eswa.2019.07.023.

[43] James R Evans and James Robert Evans. *Creative thinking in the decision and management sciences*. South-Western Pub, 1991.

[44] Maria Pia Fanti et al. “Decision support for a waste collection service with time and shift constraints”. In: *2016 American Control Conference (ACC)*. 2016, pp. 2599–2604. DOI: 10.1109/ACC.2016.7525308.

[45] B. Fazlollahi and R. Vahidov. “A method for generation of alternatives by decision support systems”. In: *Journal of Management Information Systems* 18.2 (2001), pp. 229–250. ISSN: 07421222. DOI: 10.1080/07421222.2001.11045683.

[46] Carlos M Fonseca, Peter J Fleming, et al. “Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization.” In: *Icga*. Vol. 93. July. Citeseer. 1993, pp. 416–423.

[47] Claes Fornell and David F. Larcker. “Evaluating Structural Equation Models with Unobservable Variables and Measurement Error”. In: *Journal of Marketing Research* 18.1 (Feb. 1981), p. 39. ISSN: 00222437. DOI: 10.2307/3151312.

[48] Marianne Harbo Frederiksen and Mette Præst Knudsen. “Drones for offshore and maritime missions: Opportunities and barriers”. In: *SDU Centre for Integrative Innovation Management* April (2018).

[49] Batya Friedman, Peter H Khan Jr, and Daniel C Howe. “Trust online”. In: *Communications of the ACM* 43.12 (2000), pp. 34–40.

[50] M. Socorro García-Cascales and M. Teresa Lamata. "On rank reversal and TOPSIS method". In: *Mathematical and Computer Modelling* 56.5-6 (Sept. 2012), pp. 123–132. ISSN: 08957177. DOI: 10.1016/j.mcm.2011.12.022.

[51] Sen Bong Gee, Kay Chen Tan, and Hussein A. Abbass. "A Benchmark Test Suite for Dynamic Evolutionary Multiobjective Optimization". In: *IEEE Transactions on Cybernetics* 47.2 (Feb. 2017), pp. 461–472. ISSN: 21682267. DOI: 10.1109/TCYB.2016.2519450.

[52] Harjinder Gill et al. "Antecedents of trust: Establishing a boundary condition for the relation between propensity to trust and intention to trust". In: *Journal of Business and Psychology* 19.3 (2005), pp. 287–302. ISSN: 08893268. DOI: 10.1007/s10869-004-2229-8.

[53] David E Goldberg. "Genetic algorithms in search". In: *Optimization, and MachineLearning* (1989).

[54] Salvatore Greco, Matthias Ehrgott, and José Rui Figueira. *Multiple Criteria Decision Analysis*. Springer, New York, NY: Springer, 2016. ISBN: 978-1-4939-3094-4.

[55] Joe F Hair et al. "An assessment of the use of partial least squares structural equation modeling in marketing research". In: *Journal of the academy of marketing science* 40.3 (2012), pp. 414–433.

[56] Joseph F Hair et al. "The use of partial least squares structural equation modeling in strategic management research: a review of past practices and recommendations for future applications". In: *Long range planning* 45.5-6 (2012), pp. 320–340.

[57] Joseph F. Hair et al. "When to use and how to report the results of PLS-SEM". In: *European Business Review* 31.1 (2019), pp. 2–24. ISSN: 0955534X. DOI: 10.1108/EBR-11-2018-0203.

[58] Joseph F Hair Jr et al. *A primer on partial least squares structural equation modeling (PLS-SEM)*. Sage publications, 2021.

[59] Joe F. Hair Jr. et al. “PLS-SEM or CB-SEM: updated guidelines on which method to use”. In: *International Journal of Multivariate Data Analysis* 1.2 (2017), p. 107. ISSN: 2396-8303. DOI: 10.1504/ijmda.2017.10008574.

[60] Jörg Henseler, Christian M Ringle, and Marko Sarstedt. “A new criterion for assessing discriminant validity in variance-based structural equation modeling”. In: *Journal of the Academy of Marketing Science* 43.1 (2014), pp. 115–135. ISSN: 00920703. DOI: 10.1007/s11747-014-0403-8.

[61] Kevin Anthony Hoff and Masooda Bashir. “Trust in automation: Integrating empirical evidence on factors that influence trust”. In: *Human Factors* 57.3 (Sept. 2015), pp. 407–434. ISSN: 15478181. DOI: 10.1177/0018720814547570.

[62] Jeffrey Horn, Nicholas Nafpliotis, and David E Goldberg. “A niched Pareto genetic algorithm for multiobjective optimization”. In: *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. Ieee. 1994, pp. 82–87.

[63] Meng Hsiang Hsu et al. “Determinants of repurchase intention in online group-buying: The perspectives of DeLone and McLean is success model and trust”. In: *Computers in Human Behavior* 36 (2014), pp. 234–245. ISSN: 07475632. DOI: 10.1016/j.chb.2014.03.065.

[64] Muhammad Saiful Islam and Madhav Nepal. “A fuzzy-Bayesian model for risk assessment in power plant projects”. In: *Procedia Computer Science* 100 (2016), pp. 963–970.

[65] Kalervo Jarvelin and Jaana Kekalainen. “IR evaluation methods for retrieving highly relevant documents”. In: *SIGIR Forum (ACM Special Interest Group on Information Retrieval)* (2000), pp. 41–48. ISSN: 01635840. DOI: 10.1145/3130348.3130374.

[66] Kalervo Järvelin and Jaana Kekäläinen. “Cumulated gain-based evaluation of IR techniques”. In: *ACM Transactions on Information Systems (TOIS)* 20.4 (2002), pp. 422–446.

[67] Karl G Joreskog. “The ML and PLS techniques for modeling with latent variables: Historical and comparative aspects”. In: *Systems under indirect observation, part I* (1982), pp. 263–270.

[68] Karl Jöreskog. “Structural Analysis of Covariance and Correlation Matrices”. In: *Psychometrika* 43 (Feb. 1978), pp. 443–477. DOI: 10.1007/BF02293808.

[69] Audun Josang, Ross Hayward, and Simon Pope. “Trust network analysis with subjective logic”. In: *Conference Proceedings of the Twenty-Ninth Australasian Computer Science Conference (ACSW 2006)*. Australian Computer Society. 2006, pp. 85–94.

[70] Ankur Joshi et al. “Likert Scale: Explored and Explained”. In: *British Journal of Applied Science and Technology* 7.4 (Jan. 2015), pp. 396–403. DOI: 10.9734/bjast/2015/14975.

[71] Kensaku Kawamoto et al. “Improving clinical practice using clinical decision support systems: A systematic review of trials to identify features critical to success”. In: *British Medical Journal* 330.7494 (2005), pp. 765–768. ISSN: 09598146. DOI: 10.1136/bmj.38398.500764.8f.

[72] Mehdi Keshavarz-Ghorabaee et al. “A comparative analysis of the rank reversal phenomenon in the EDAS and TOPSIS methods”. In: *Economic Computation and Economic Cybernetics Studies and Research* 52.3 (2018), pp. 121–134. ISSN: 18423264. DOI: 10.24818/18423264/52.3.18.08.

[73] Buomsoo Kim, Jinsoo Park, and Jihae Suh. “Transparency and accountability in AI decision support: Explaining and visualizing convolutional neural networks for text information”. In: *Decision Support Systems* 134 (July 2020), p. 113302. ISSN: 01679236. DOI: 10.1016/j.dss.2020.113302.

[74] Dan J. Kim, Donald L. Ferrin, and H. Raghav Rao. “A trust-based consumer decision-making model in electronic commerce: The role of trust, perceived risk, and their an-

tecedents". In: *Decision Support Systems* 44.2 (Jan. 2008), pp. 544–564. ISSN: 0167-9236. DOI: 10.1016/J.DSS.2007.07.001.

[75] Sanghyun Kim and Hyunsun Park. "Effects of various characteristics of social commerce (s-commerce) on consumers' trust and trust performance". In: *International Journal of Information Management* 33.2 (2013), pp. 318–332. ISSN: 02684012. DOI: 10.1016/j.ijinfomgt.2012.11.006.

[76] *Leaflet*. https://leafletjs.com/.

[77] Jae Nam Lee and Young Gul Kim. "Effect of partnership quality on IS outsourcing success: Conceptual framework and empirical validation". In: *Journal of Management Information Systems* 15.4 (1998), pp. 29–61. ISSN: 07421222. DOI: 10.1080/07421222.1999.11518221.

[78] John D. Lee and Katrina A. See. *Trust in automation: Designing for appropriate reliance*. Jan. 2004. DOI: 10.1518/hfes.46.1.50_30392.

[79] Lorraine Lee et al. "On the use of partial least squares path modeling in accounting research". In: *International Journal of Accounting Information Systems* 12.4 (Dec. 2011), pp. 305–328. ISSN: 14670895. DOI: 10.1016/j.accinf.2011.05.002.

[80] Francisco Liébana-Cabanillas, Juan Sánchez-Fernández, and Francisco Muñoz-Leiva. "Antecedents of the adoption of the new mobile payment systems: The moderating effect of age". In: *Computers in Human Behavior* 35 (2014), pp. 464–478. ISSN: 07475632. DOI: 10.1016/j.chb.2014.03.022.

[81] Sifeng Liu, Yingjie Yang, and Jeffrey Forrest. "Grey Numbers and Their Operations". In: *Grey Data Analysis : Methods, Models and Applications*. Singapore: Springer Singapore, 2017, pp. 29–43. ISBN: 978-981-10-1841-1. DOI: 10.1007/978-981-10-1841-1_3.

[82] Zhong Liu et al. "Cyber-physical-social systems for command and control". In: *IEEE Intelligent Systems* 26.4 (2011), pp. 92–96. ISSN: 15411672. DOI: 10.1109/MIS.2011.69.

[83] John W Lloyd. “Practical Advtanages of Declarative Programming.” In: *GULP-PRODE (1)*. 1994, pp. 18–30.

[84] Jan-Bernd Lohmöller. *Latent variable path modeling with partial least squares*. Springer Science & Business Media, 2013.

[85] F A Lootsma. *RAIRO. RECHERCHE OPÉRATIONNELLE*. Tech. rep. 3. 1990, pp. 263–285.

[86] Eugene Lukacs. *A Characterization of the Gamma Distribution*. Tech. rep. 2. 1955, pp. 319–324.

[87] Xin Luo et al. “Examining multi-dimensional trust and multi-faceted risk in initial acceptance of emerging technologies: An empirical study of mobile banking services”. In: *Decision Support Systems* 49.2 (2010), pp. 222–234. ISSN: 01679236. DOI: 10.1016/j.dss.2010.02.008.

[88] Giusy Macrina et al. “Drone-aided routing: A literature review”. In: *Transportation Research Part C: Emerging Technologies* 120 (Nov. 2020), p. 102762. ISSN: 0968090X. DOI: 10.1016/j.trc.2020.102762.

[89] Nashat Mansour, Mohamad Awad, and Khaled El-Fakih. “Incremental genetic algorithm”. In: (2006).

[90] Azizbek Marakhimov and Jaehun Joo. “Consumer adaptation and infusion of wearable devices for healthcare”. In: *Computers in Human Behavior* 76 (2017), pp. 135–148. ISSN: 07475632. DOI: 10.1016/j.chb.2017.07.016.

[91] B. Mareschal, Y. De Smet, and P. Nemery. “Rank reversal in the PROMETHEE II method: Some new results”. In: *2008 IEEE International Conference on Industrial Engineering and Engineering Management, IEEM 2008*. 2008, pp. 959–963. ISBN: 9781424426300. DOI: 10.1109/IEEM.2008.4738012.

[92] Francisco J. Martínez-López, Juan C. Gázquez-Abad, and Carlos M.P. Sousa. "Structural equation modelling in marketing and business research: Critical issues and practical recommendations". In: *European Journal of Marketing* 47.1 (Feb. 2013), pp. 115–152. ISSN: 03090566. DOI: 10.1108/03090561311285484.

[93] Roger C Mayer, James H Davis, and F David Schoorman. "An integrative model of organizational trust". In: *Academy of management review* 20.3 (1995), pp. 709–734.

[94] D. Harrison Mcknight et al. "Trust in a specific technology: An investigation of its components and measures". In: *ACM Transactions on Management Information Systems* 2.2 (2011). ISSN: 2158656X. DOI: 10.1145/1985347.1985353.

[95] "Measuring and Predicting Shared Situation Awareness in Teams". In: *Journal of Cognitive Engineering and Decision Making* 3.3 (Sept. 2009), pp. 280–308. ISSN: 15553434. DOI: 10.1518/155534309X474497.

[96] Stephanie M. Merritt et al. "I trust it, but i don't know why: Effects of implicit attitudes toward automation on trust in an automated system". In: *Human Factors* 55.3 (Nov. 2013), pp. 520–534. ISSN: 00187208. DOI: 10.1177/0018720812465081.

[97] Melanie Mitchell. *An introduction to genetic algorithms*. MIT press, 1998.

[98] Luc Moreau et al. *PROV-N: The Provenance Notation*. English. W3C Recommendation. United States: World Wide Web Consortium, Apr. 2013.

[99] Arrchana Muruganantham, Kay Chen Tan, and Prahlad Vadakkepat. "Evolutionary Dynamic Multiobjective Optimization Via Kalman Filter Prediction". In: *IEEE Transactions on Cybernetics* 46.12 (Dec. 2016), pp. 2862–2873. ISSN: 21682267. DOI: 10.1109/TCYB.2015.2490738.

[100] S. Muthukrishnan. *Data Streams: Algorithms and Applications*. 2600 AD Delft, The Netherlands: now, 2005. ISBN: 9781933019147.

[101] Paul Nightingale and Tim Brady. "Projects, paradigms and predictability". In: *Project-based organizing and strategic management*. Emerald Group Publishing Limited, 2011.

[102] Steven M Norman, Bruce J. Avolio, and Fred Luthans. "The impact of positivity and transparency on trust in leaders and their perceived effectiveness". In: *Leadership Quarterly* 21.3 (2010), pp. 350–364. ISSN: 10489843. DOI: 10.1016/j.leaqua.2010.03.002.

[103] G O Odu and O E Charles-Owaba. *Review of Multi-criteria Optimization Methods-Theory and Applications*. Tech. rep. 10. 2013, pp. 2–3.

[104] Vanessa Pirotta et al. *Consequences of global shipping traffic for marine giants*. Feb. 2019. DOI: 10.1002/fee.1987.

[105] *PLSPM*. https://github.com/GoogleCloudPlatform/plspm-python.

[106] Robin C Purshouse and Peter J Fleming. "Why use Elitism and Sharing in a MultiObjective Genetic Algorithm?" In: *Proceedings of the Genetic and Evolutionary Computation Conference* (2002), pp. 520–527.

[107] Cristian Ramirez-Atencia, Sanaz Mostaghim, and David Camacho. "A knee point based evolutionary multi-objective optimization for mission planning problems". In: *GECCO 2017 - Proceedings of the 2017 Genetic and Evolutionary Computation Conference* (2017), pp. 1216–1223. DOI: 10.1145/3071178.3071319.

[108] Cristian Ramirez-Atencia, Victor Rodriguez-Fernandez, and David Camacho. "A revision on multi-criteria decision making methods for multi-UAV mission planning support". In: *Expert Systems with Applications* 160 (2020), p. 113708. ISSN: 09574174. DOI: 10.1016/j.eswa.2020.113708.

[109] Cristian Ramirez-Atencia et al. "A simple CSP-based model for Unmanned Air Vehicle Mission Planning". In: *INISTA 2014 - IEEE International Symposium on Innovations in Intelligent Systems and Applications, Proceedings* (2014), pp. 146–153. DOI: 10.1109/INISTA.2014.6873611.

[110] Cristian Ramirez-Atencia et al. “Solving complex multi-UAV mission planning problems using multi-objective genetic algorithms”. In: *Soft Computing* 21.17 (2017), pp. 4883–4900. ISSN: 14337479. DOI: 10.1007/s00500-016-2376-7.

[111] Cristian Ramirez-Atencia et al. “Solving UAV mission planning based on temporal constaint satisfaction problem using genetic algorithms”. In: *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming*.

[112] N. Razali and J. Geraghty. “Genetic Algorithm Performance with Different Selection Strategies in Solving TSP”. In: 2011.

[113] P. Reichert et al. “Concepts of decision support for river rehabilitation”. In: *Environmental Modelling and Software* 22.2 (Feb. 2007), pp. 188–201. ISSN: 13648152. DOI: 10.1016/j.envsoft.2005.07.017.

[114] Jana Ries and Alessio Ishizaka. “A multi-criteria support system for dynamic aerial vehicle routing problems”. In: *2nd International Conference on Communications Computing and Control Applications, CCCA 2012* 241598 (2012), pp. 1–4. DOI: 10.1109/CCCA.2012.6417853.

[115] Pavel S. Roshanov et al. “Features of effective computerised clinical decision support systems: Meta-regression of 162 randomised trials”. In: *BMJ (Online)* 346.7899 (2013), pp. 1–12. ISSN: 17561833. DOI: 10.1136/bmj.f657.

[116] Bernard Roy. “The Outranking Approach and the Foundations of Electre Methods”. In: *Readings in Multiple Criteria Decision Aid*. Springer Berlin Heidelberg, 1990, pp. 155–183. DOI: 10.1007/978-3-642-75935-2_8.

[117] Cynthia Rudin and David Carlson. “The secrets of machine learning: Ten things you wish you had known earlier to be more effective at data analysis”. In: *Operations Research & Management Science in the Age of Analytics*. Informs, 2019, pp. 44–72.

[118] R. W. Saaty. "The analytic hierarchy process-what it is and how it is used". In: *Mathematical Modelling* 9.3-5 (1987), pp. 161–176. ISSN: 02700255. DOI: 10.1016/0270-0255(87)90473-8.

[119] Thomas L Saaty, Luis G Vargas, et al. *Decision making with the analytic network process*. Vol. 282. Springer, 2006.

[120] Thomas L Saaty and Luis G Vargas. "The analytic network process". In: *Decision making with the analytic network process*. Springer, 2013, pp. 1–40.

[121] Thomas L. Saaty. "Time dependent decision-making; dynamic priorities in the AHP/ANP: Generalizing from points to functions and from real to complex variables". In: *Mathematical and Computer Modelling* 46.7 (2007). Decision Making with the Analytic Hierarchy Process and the Analytic Network Process, pp. 860–891. ISSN: 0895-7177. DOI: https://doi.org/10.1016/j.mcm.2007.03.028.

[122] Luís Santos, João Coutinho-Rodrigues, and Carlos Henggeler Antunes. "A web spatial decision support system for vehicle routing using Google Maps". In: *Decision Support Systems* 51.1 (2011), pp. 1–9. ISSN: 01679236. DOI: 10.1016/j.dss.2010.11.008.

[123] Anthony Sardain, Erik Sardain, and Brian Leung. "Global forecasts of shipping traffic and biological invasions to 2050". In: *Nature Sustainability* 2.4 (Apr. 2019), pp. 274–282. ISSN: 23989629. DOI: 10.1038/s41893-019-0245-y.

[124] Reza Sarraf and Michael P. McGuire. "Integration and Comparison of Multi-Criteria Decision Making Methods in Safe Route Planner". In: *Expert Systems with Applications* 154 (2020), p. 113399. ISSN: 09574174. DOI: 10.1016/j.eswa.2020.113399.

[125] Marko Sarstedt and Heungsun Hwang. *Advances in composite-based structural equation modeling*. Feb. 2020. DOI: 10.1007/s41237-020-00105-9.

[126] Mbarka Selmi, Tarek Kormi, and Nizar Bel Hadj Ali. "Comparison of multi-criteria decision methods through a ranking stability index". In: *International Journal of Operational Research* 27.1/2 (2016), p. 165. ISSN: 1745-7645. DOI: 10.1504/ijor.2016.10000064.

[127] Sajid Siraj, Ludmil Mikhailov, and John A. Keane. "Contribution of individual judgments toward inconsistency in pairwise comparisons". In: *European Journal of Operational Research* 242.2 (Apr. 2015), pp. 557–567. ISSN: 03772217. DOI: 10.1016/j.ejor.2014.10.024.

[128] Sajid Siraj, Ludmil Mikhailov, and John A. Keane. "PriEsT: an interactive decision support tool to estimate priorities from pairwise comparison judgments". In: *ITOR* 22.2 (2015), pp. 217–235. DOI: 10.1111/itor.12054.

[129] David F. Steiner et al. "Impact of Deep Learning Assistance on the Histopathologic Review of Lymph Nodes for Metastatic Breast Cancer". In: *American Journal of Surgical Pathology* 42.12 (Dec. 2018), pp. 1636–1646. ISSN: 15320979. DOI: 10.1097/PAS.0000000000001151.

[130] TJ Stewart. "A critical survey on the status of multiple criteria decision making theory and practice". In: *Omega* 20.5-6 (1992), pp. 569–586. ISSN: 03050483. DOI: 10.1016/0305-0483(92)90003-P.

[131] Madjid Tavana et al. "A dynamic decision support system for evaluating peer-to-peer rental accommodations in the sharing economy". In: *International Journal of Hospitality Management* 91 (Oct. 2020), p. 102653. ISSN: 02784319. DOI: 10.1016/j.ijhm.2020.102653.

[132] Chris Tofallis. "Add or Multiply? A Tutorial on Ranking and Choosing with Multiple Criteria". In: *INFORMS Transactions on Education* 14.3 (2014), pp. 109–119. ISSN: 1532-0545. DOI: 10.1287/ited.2013.0124.

[133] Richard Tomsett et al. "Rapid Trust Calibration through Interpretable". In: *Patterns* 1.4 (2020), p. 100049. ISSN: 2666-3899. DOI: 10.1016/j.patter.2020.100049.

[134] Ankit Toshniwal et al. “Storm @Twitter”. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data*. 2014, pp. 147–156. ISBN: 9781450323765. DOI: 10.1145/2588555.2595641.

[135] Evangelos Triantaphyllou. “Multi-Criteria Decision Making Methods”. In: 2000, pp. 5–21. DOI: 10.1007/978-1-4757-3157-6_2.

[136] Evangelos Triantaphyllou and Stuart H. Mann. “An examination of the effectiveness of multi-dimensional decision-making methods: A decision-making paradox”. In: *Decision Support Systems* 5.3 (Sept. 1989), pp. 303–312. ISSN: 01679236. DOI: 10.1016/0167-9236(89)90037-7.

[137] Rustam Vahidov. “Intermediating user-DSS interaction with autonomous agents”. In: *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans* 35.6 (Nov. 2005), pp. 964–970. ISSN: 10834427. DOI: 10.1109/TSMCA.2005.851292.

[138] Rustam Vahidov and Fei Ji. “A diversity-based method for infrequent purchase decision support in e-commerce”. In: *Electronic Commerce Research and Applications* 4.2 (June 2005), pp. 143–158. ISSN: 15674223. DOI: 10.1016/j.elerap.2004.09.001.

[139] Anne Marthe Van Der Bles et al. “Communicating uncertainty about facts, numbers and science”. In: *Royal Society Open Science* 6.5 (May 2019). ISSN: 20545703. DOI: 10.1098/rsos.181870.

[140] Marius S Vassiliou, David S Alberts, and Jonathan Russell Agre. *C2 re-envisioned: the future of the enterprise*. CRC Press, 2014. ISBN: 9781466595804.

[141] Gasper G. Vieira, Leonilde R. Varela, and Rita A. Ribeiro. “A knowledge based system for supporting sustainable industrial management in a clothes manufacturing company based on a data fusion model”. In: *Lecture Notes in Business Information Processing*. Vol. 250. Springer Verlag, 2016, pp. 113–126. ISBN: 9783319328768. DOI: 10.1007/978-3-319-32877-5_9.

[142] Jyrki Wallenius et al. “Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead”. In: *Management Science* 54.7 (2008), pp. 1336–1349. ISSN: 00251909. DOI: 10.1287/mnsc.1070.0838.

[143] Xiaoting Wang and Evangelos Triantaphyllou. “Ranking irregularities when evaluating alternatives by using some ELECTRE methods”. In: *Omega* 36.1 (Feb. 2008), pp. 45–63. ISSN: 03050483. DOI: 10.1016/j.omega.2005.12.003.

[144] William Webber, Alistair Moffat, and Justin Zobel. “A similarity measure for indefinite rankings”. In: *ACM Transactions on Information Systems* 28.4 (Nov. 2010), pp. 1–38. ISSN: 10468188. DOI: 10.1145/1852102.1852106.

[145] Herbert I Weisberg. *Willful ignorance*. Wiley Online Library, 2014.

[146] Carl Westin, Clark Borst, and Brian Hilburn. “Automation Transparency and Personalized Decision Support: Air Traffic Controller Interaction with a Resolution Advisory System”. In: *IFAC-PapersOnLine* 49.19 (Jan. 2016), pp. 201–206. ISSN: 24058963. DOI: 10.1016/j.ifacol.2016.10.520.

[147] Herman Wold. “Model Construction and Evaluation When Theoretical Knowledge Is Scarce”. In: *Evaluation of Econometric Models*. Elsevier, Jan. 1980, pp. 47–74. DOI: 10.1016/b978-0-12-416550-2.50007-8.

[148] K Yamasaki. “Dynamic Pareto Optimum GA against the changing environments”. In: *Evolutionary Algorithms for Dynamic Optimization Problems* (2001), pp. 47–50.

[149] Shuli Yan et al. “Dynamic grey target decision making method with grey numbers based on existing state and future development trend of alternatives”. In: *Journal of Intelligent and Fuzzy Systems* 28.5 (June 2015), pp. 2159–2168. ISSN: 18758967. DOI: 10.3233/IFS-141497.

[150] Kwangsun Yoon and Ching-Lai Hwang. *Multiple Attribute Decision Making*. Vol. 186. Lecture Notes in Economics and Mathematical Systems. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011. ISBN: 978-3-540-10558-9. DOI: 10.4135/9781412985161.

[151] Matei Zaharia et al. "Discretized streams". In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP '13* 1 (2013), pp. 423–438. ISSN: 21508097. DOI: 10.1145/2517349.2522737.

[152] Stelios H. Zanakis et al. "Multi-attribute decision making: A simulation comparison of select methods". In: *European Journal of Operational Research* 107.3 (June 1998), pp. 507–529. ISSN: 03772217. DOI: 10.1016/S0377-2217(97)00147-1.

[153] Ying Zhang, Lukas D Kuhn, and Markus PJ Fromherz. "Improvements on ant routing for sensor networks". In: *International Workshop on Ant Colony Optimization and Swarm Intelligence*. Springer. 2004, pp. 154–165.

[154] Aimin Zhou, Yaochu Jin, and Qingfu Zhang. "A population prediction strategy for evolutionary dynamic multiobjective optimization". In: *IEEE transactions on cybernetics* 44.1 (2013), pp. 40–53.

[155] Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. "Comparison of multiobjective evolutionary algorithms: Empirical results". In: *Evolutionary computation* 8.2 (2000), pp. 173–195.

[156] Yeleny Zulueta et al. "A discrete time variable index for supporting dynamic multi-criteria decision making". In: *International Journal of Uncertainty, Fuzziness and Knowlege-Based Systems* 22.1 (Feb. 2014), pp. 1–22. ISSN: 02184885. DOI: 10.1142/S0218488514500019.

[157] Yeleny Zulueta et al. "A discriminative dynamic index based on bipolar aggregation operators for supporting dynamic multi-criteria decision making". In: *Advances in Soft Computing* 228 (Nov. 2013), pp. 237–248. ISSN: 18600794. DOI: 10.1007/978-3-642-39165-1_25.