Many-Objective Genetic Type-2 Fuzzy Logic Based Workforce Optimisation Strategies for Large Scale Organisational Design

Andrew J. Starkey

A thesis submitted for the degree of Doctor of Philosophy in

Computer Science

School of Computer Science and Electronic Engineering

University of Essex

2018

Acknowledgements

I would like to thank the following people for their contributions to this PhD and my career, for which I will always be indebted:

Professor Hani Hagras, the academic supervisor, I would like to extend my sincere gratitude and thanks for his supervision, guidance and support. He has been an exceptional mentor and role model, which has led to the best possible start to my career.

Dr Gilbert Owusu, one of the industrial supervisors, I would like to thank for the opportunities given to me with respect to this PhD and the early development of my professional life. I would also like to thank him for the experience and guidance given for implementing commercial, impactful and professional research applications.

Dr Sid Shakya, one of the industrial supervisors, I would like to thank him for his exceptional guidance on evolutionary algorithms and operational research. I would also like to thank him for guiding me on best research practices and supporting me in my work.

Richard Chambers, the senior patch architect, I would like to thank him for his great contribution to this work. He has provided detailed context, data, insight and ideas around the development of iPatch. Without his drive to implement the results of this work, the real-world benefits and opportunities may never have been realised.

I would like to thank British Telecom for supporting this PhD.

Finally, I would like to thank my family and friends, for their love, support and guidance.

Publications and Awards Arising from this Work

Journal Papers

- A. Starkey, H Hagras, S Shakya, G Owusu, "iPatch: A Many-Objective Type-2 Fuzzy Logic System for Field Workforce Optimisation", IEEE Transactions on Fuzzy Systems (UNDER REVIEW)
- A. Starkey, H Hagras, S Shakya, G Owusu, "A Genetic Algorithm Based System for Simultaneous Optimisation of Workforce Skills & Teams", Künstliche Intelligenz pp.1-16, 2018.
- A Starkey, H Hagras, S Shakya, G Owusu, A Mohamed, D Alghazzawi, "A Cloud Computing Based Many Objective Type-2 Fuzzy Logic System for Mobile Field Workforce Area Optimization", Journal of Memetic Computing, vol. 8, no. 4, pp. 269-286, 2016.
- A. Starkey, H Hagras, S Shakya, G Owusu, "A multi-objective genetic type-2 fuzzy logic-based system for mobile field workforce area optimization", Information Sciences, vol. 321, no. 1, pp. 390-411, 2015.

Conference Papers

- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "Fuzzy Dominance Rules for Real-World Many Objective Optimization" , Proceeding of the 2017 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2017), Naples, Italy, July 2017

- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "A Genetic Algorithm Based Approach for the Simultaneous Optimisation of Workforce Skill Sets and Team Allocation", Proceedings of the 2016 International Conference of the BCS SGAI International Conference on Artificial Intelligence, Cambridge, Cambridge, UK, December 2016
- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "A Many-Objective Genetic Type-2 Fuzzy Logic System for the Optimal Allocation of Mobile Field Engineers", Proceedings of the 2016 World Congress on Computational Intelligence (WCCI 2016), Vancouver, Canada, July 2016
- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "A Comparison of Particle Swarm Optimization and Genetic Algorithms for a Multi-Objective Type-2 Fuzzy Logic Based System for the Optimal Allocation of Mobile Field Engineers", Proceedings of the 2016 World Congress on Computational Intelligence (WCCI 2016), Vancouver, Canada, July 2016
- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "A Genetic Algorithm Based Approach for the Optimisation of Workforce Skill Sets". Proceedings of the 2015 International Conference of the BCS SGAI International Conference on Artificial Intelligence, Cambridge, Cambridge, UK, December 2015
- A.Starkey, H.Hagras, S.Shakya, G.Owusu, "A Genetic Type-2 Fuzzy Logic Based Approach for the Optimal Allocation of Mobile Field Engineers to their Working Areas", Proceeding of the 2015 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE 2015), , Istanbul, Turkey, August 2015.

Awards & Nominations

- Awarded the Best Paper Award at the 36th International Conference of the BCS SGAI International Conference on Artificial Intelligence, Cambridge, December 2016 for the paper entitled "*A Genetic Algorithm Based Approach for the Simultaneous Optimisation of Workforce Skill Sets and Team Allocation*"
- Won with BT and University of Essex, The 2015 Global Telecoms Business Innovation Award for Business Innovation of the Year
- Nominated for Young IT Professional of the Year by the 2017 British Computer Society IT Awards.
- Highly Commended in Model-Based Engineering by the 2016 Institution of Engineering and Technology (IET) Innovation Awards,
- A 2015 IEEE Travel Grant for FUZZ-IEEE 2015, for the paper titled "*A Genetic Type-2 Fuzzy Logic Based Approach for the Optimal Allocation of Mobile Field Engineers to their Working Areas*"

Awards Contributions

- With BT and University of Essex won the 2017 Global Telecoms Business Innovation Award
- IEEE Outstanding Organisation Award for BT, presented at FUZZ-IEEE 2017 in Naples, Italy.

Abstract

Workforce optimisation aims to maximise the productivity of a workforce and is a crucial practice for large organisations. The more effective these workforce optimisation strategies are, the better placed the organisation is to meet their objectives. Usually, the focus of workforce optimisation is scheduling, routing and planning. These strategies are particularly relevant to organisations with large mobile workforces, such as utility companies. There has been much research focused on these areas. One aspect of workforce optimisation that gets overlooked is organisational design.

Organisational design aims to maximise the potential utilisation of all resources while minimising costs. If done correctly, other systems (scheduling, routing and planning) will be more effective.

This thesis looks at organisational design, from geographical structures and team structures to skilling and resource management. A many-objective optimisation system to tackle large-scale optimisation problems will be presented. The system will employ interval type-2 fuzzy logic to handle the uncertainties with the real-world data, such as travel times and task completion times.

The proposed system was developed with data from British Telecom (BT) and was deployed within the organisation. The techniques presented at the end of this thesis led to a very significant improvement over the standard NSGA-II algorithm by 31.07% with a P-Value of 1.86^{-10}.

The system has delivered an increase in productivity in BT of 0.5%, saving an estimated £1million a year, cut fuel consumption by 2.9%, resulting in an additional saving of over £200k a year. Due to less fuel consumption Carbon Dioxide (CO_2) emissions have been reduced by 2,500 metric tonnes.

Furthermore, a report by the United Kingdom's (UK's) Department of Transport found that for every billion vehicle miles travelled, there were 15,409 serious injuries or deaths. The system saved an estimated 7.7 million miles, equating to preventing more than 115 serious casualties and fatalities.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AD	Average Distance
AW	Amount of Work
BBComp	Black Box Optimisation Competition
BSC	British Computer Society
BT	British Telecommunications Plc
CO_2	Carbon Dioxide
CPU	Central Processing Unit
CTSP	Clustered Travelling Salesman Problem
DDoS	Distributed Denial of Service
FDR	Fuzzy Dominance Rules
FL	Fuzzy Logic
FLS	Fuzzy Logic System
FTE	Full Time Employment
GA	Genetic Algorithm
GB	Gigabyte

GTSP	Generalised Travelling Salesman Problem
ID	Identifier
IET	Institution of Engineering and Technology
km	Kilometre
MOGA	Multi Objective Genetic Algorithm
MOPSO	Multi Objective Particle Swarm Algorithm
NP	Nondeterministic Polynomial time
NSGA-II	Non-Dominated Sorting Genetic Algorithm II
NSGAIIFDR	Non-Dominated Sorting Genetic Algorithm II with Fuzzy Dominance Rules
PA	Patch Average
PCFLS	Patch Constructor Fuzzy Logic System
PSO	Particle Swarm Optimisation
RMS	Root Mean Square
RMSE	Root Mean Square Error
RVGA	Real-Valued Genetic Algorithm

SD	Standard Deviation
SDP	Service Delivery Point
SDPA	Service Delivery Point Average
SGAI	Specialist Group on Artificial Intelligence
SOGA	Single Objective Genetic Algorithm
TAFLS	Task Allocation Fuzzy Logic System
TSP	Travelling Salesman Problem
UK	United Kingdom
USA	United States of America
VRP	Vehicle Routing Problem
WA	Working Area
WFM	Workforce Management
WSRP	Workforce Scheduling and Routing Problem

Chapter 1. Introduction

1.1 An Introduction to Workforce Management

Workforce management (WFM) is broadly defined as a term that encompasses all the operational and decision activities needed to maintain a productive workforce, by providing an optimal plan, which will lead to significant cost reductions and performance improvements [1] [2].

For any organisation, having a well-managed workforce comes with a number of benefits, like reduced operating costs and increased capacity to handle incoming demand. The number of potential benefits increases when the workforce is mobile. Mobile workforces are far more difficult to manage as working locations continually change, so any task scheduling system has to factor in travelling times between locations. The larger the organisation, the more complex the management of the workforce becomes. Examples of organisations with both large and mobile workforces are delivery companies and companies that supply utility services, such as electricity, water, gas and telecoms. Demand for utility services are high, as they contribute to living standards and there is large amounts of infrastructure needed to supply utilities, which constantly need upgrading and maintaining. These factors mean there is high amounts of demand and pressure on workforces in these types of organisations.

WFM can be derived from a number of optimisation problems. Effective workforce management can be measured using metrics like productivity, efficiency and output. The objective in any profit maximising organisation would be to maximise these measures through advanced optimisation algorithms.

Optimisation of such complex real-world problems is often better suited to computer algorithms, rather than manual optimisation by a human. One notable reason is that there can

be, with only a few variables, many millions of possible solutions to the problem. For example in the Travelling Salesman Problem (which will be discussed at length in Chapter 2) a salesman has to travel to a number of cities, visiting each city only once and minimising the distance travelled. If there are just ten cities, there are over 181,000 possible routes the salesman can take, 16 cities would lead to more than 650 billion possible routes.

According to [3], such problems are looking for an object from a possibly countable infinite set, which are a class of problems called Combinatorial Optimisation (CO) problems.

Algorithms designed to tackle CO problems usually aim for a metaheuristic approach [4] because the optimisation must be completed within a reasonable amount of time. This is especially true for real-world optimisation problems as the environment changes on a frequent basis. A common approach to tackling these large scale complex optimisation problems are Evolutionary Algorithms (EAs) such as Genetic Algorithms (GAs) [5], [6], [7] and Particle Swarm Optimisation (PSO) [8], [9], [10].

The WFM challenges that will be discussed in this thesis are combinatorial optimisation problems, thus, metaheuristics will also be used as the proposed method to tackle these problems. The application of these optimisation methods to WFM is the practice of workforce optimisation.

1.2 Aims of the Thesis

This thesis aims to investigate and implement a system for optimising the organisational design of a large mobile workforce. The core aim is to develop fuzzy logic systems to handle the uncertainties present in real-world problems. Fuzzy logic is a well-known method for handling uncertainty. These uncertainties come from the input data, such as estimated travel times and task times and stem from unreliable collection methods, traffic & road conditions and unexpected issues on jobs or in the field. The uncertainties impact the quality of the solutions

and cause performance issues in optimisation techniques that rely on Pareto Dominance. For an in-depth look at the theory of Fuzzy Logic (both Type-1 and Type-2), please see Appendix A. The remaining aims of the thesis are as follows:

- To investigate the most suitable optimisation methods for organisational design
- To examine the potential benefits of implementing fuzzy logic to handle the uncertainties in the data.
- To develop a system which should produce near optimal geography and team designs, reducing the amount the mobile workforce travels and increase the number of tasks the workforce completes.
- To develop a system in which each proposed organisational design should consider the wide range of complex real-world constraints, to generate results that can easily be implemented into the real-world environment on which it is based.

As the proposed system is complex and spans a wide range of workforce management techniques, the proposed system features are listed as follows:

- A novel neighbourhood-based clustering algorithm for the design of the mobile workforce's geographical areas.
- A multi-objective approach to the geography optimisation problem
- A simultaneous optimisation-based approach to the skilling and team allocation problem
- A novel fuzzy logic-based workforce simulation
- A novel fuzzy logic-based approach to improving the clustering algorithm
- A novel fuzzy logic-based approach to improving multi-objective Pareto based algorithms in optimising many-objective problems

Over the first two years of deployment by BT, this application has increased productivity by 0.5% across the mobile workforce. The application has also helped reduce travel by approximately 7.7 million miles over the same period, which has reduced fuel consumption by 2.9%. These outcomes have led to a productivity benefit of £1million a year and an additional saving of over £200K a year in fuel costs.

There are additional secondary benefits of this system, which were not initially planned for. As the engineers are travelling less, this has saved an estimated 2,500 Metric Tons of CO_2 and potentially preventing the number of serious traffic casualties or fatalities, in the UK, by more than 115.

1.3 Thesis Layout

The thesis will be structured as follows; Chapter 2 will give an overview on workforce optimisation, covering its origins within the travelling salesman problem, how Working Areas (WAs), which are also known as patches, play their part in modern large-scale organisations and how this work expands beyond the traditional Workforce Scheduling and Routing Problem (WSRP).

Chapter 3 will give an overview of optimisation algorithms. This section describes how some of the most common algorithms in this domain work, this includes; GAs, PSO algorithms and Simulated Annealing. The topic will then expand to multi-objective optimisation for problems with more than one objective, and then a description of many-objective problems and how they are measured and evaluated through hypervolumes is given.

Chapter 4 will give an overview of the large-scale optimisation problems that this work will attempt to solve. These problems extend from geographical and structural optimisation problems to resource skilling and team setup optimisation problems.

Chapter 5 presents the type-2 fuzzy logic system for field workforce optimisation detailing how the overall system is set up, how fuzzy logic can be used to improve the optimisation algorithm and an initial evaluation of the first version of the optimisation tool created for the real-world implementation of this system.

Chapter 6 expands on the work presented in Chapter 5. The work here looks at the practical and theoretical benefits of deploying the real-world optimisation tool in a cloud environment. Chapter 6 also looks at addressing the weaknesses in multi-objective algorithms when attempting to solve many-objective problems. Thus, a simple distance metric is described and implemented overcome this challenge.

Chapter 7 describes the work completed on skill optimisation, team optimisation and the simultaneous approach for optimising both strategies together. This chapter aims at addressing the weaknesses in organisational design related to individuals and resource, as opposed to just the geographical or management hierarchy addressed in previous chapters.

Chapter 8 describes the fuzzy dominance rules for real-world many-objective optimisation. At the core of the work in the previous chapters is a many-objective optimisation problem, this chapter describes a method for improving the results generated for many-objective problems in a general context, outside of organisational design and workforce management, but still within the context of real-world problems.

Chapter 10 presents the conclusions of this thesis, the real-world impact and finally the potential future work is discussed.

Appendix A is an overview of fuzzy logic systems. Firstly, describing Fuzzy Logic in a general context then describing the basics of type-1 fuzzy systems and then type-2 fuzzy systems.

Chapter 2. An Overview on Workforce Optimisation

There are many workforce optimisation strategies for WFM, the selection of which is dependent on the workforce type, industry and time-period. Mobile field workforces are some of the most complex to manage. A mobile field workforce can be defined as a workforce that is required to travel between tasks and are usually expected to visit more than one work location on any given day. The exceptions to this are when the time to complete the task is greater than the time available to work on that day. This problem is comparable to the Travelling Salesman Problem (TSP).

2.1 The Travelling Salesman Problem

2.1.1 The Orginal Travelling Salesman Problem

The travelling salesman problem is a defined as follows in [11]: given a set of cities, along with the cost of travel between each of them, find the cheapest way of visiting all cities and return to the starting point. A single solution to this problem is known as a tour or circuit, and simply lists the cities in the order they should be visited by the salesperson. The ‘cost’ here is kept general as it could be any metric that is deemed suitable for evaluating a tour, for example, fuel expense, time, distance, a cost function that combines these metrics or any other that are deemed suitable. See Figure 2.1, for an example of a number of tours through the same set of cities.

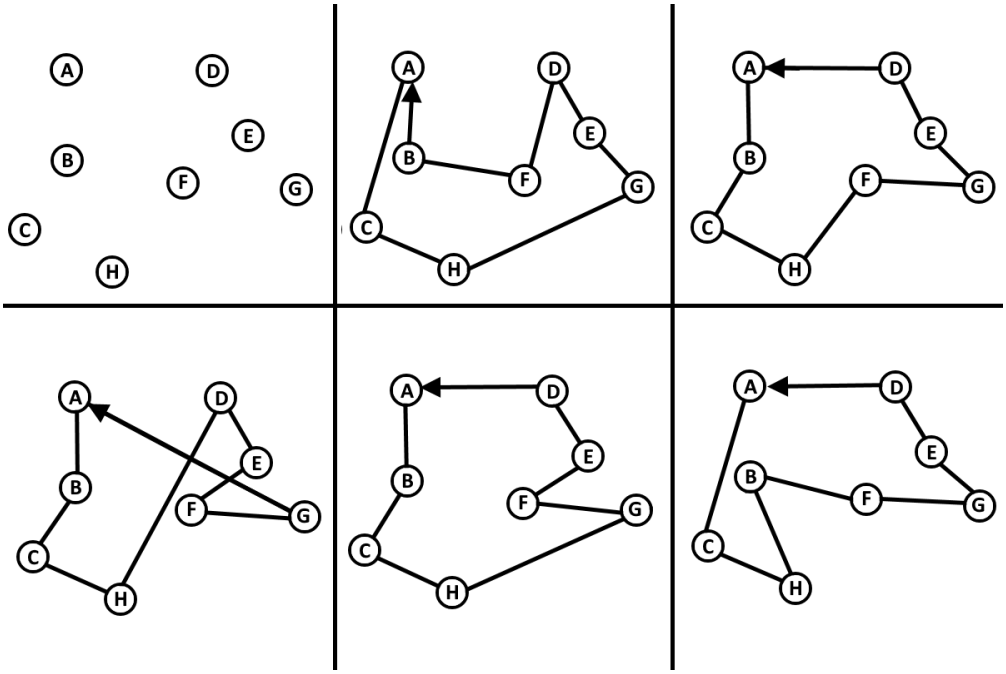

Figure 2.1: Tours in the Travelling Salesman Problem (TSP)

In [11] the TSP is traced back to a German handbook from 1832, where numerous salesmen were interested in planning the most economical routes to their customers. It is quite possible that the problem has its origins much further back than 1832, it would not be unreasonable to assume that salesmen on the historic 'Silk Road' trading route between Europe and China would have been concerned with minimising their travelling costs. Indeed, a legitimate objective of this time may have been to minimise deaths in the group for each trip.

Given that the problem has persisted throughout the centuries and has yet to be solved by a single algorithm, the complexity of the problem cannot be overstated. The TSP falls into a category of problems called Nondeterministic Polynomial Time-Hard or NP-Hard. Problems for which there is a good algorithm, where the problem can be solved in polynomial time, with a polynomial algorithm, are known as P class problems. If there is a possible solution to a problem, i.e. find the tour with the least cost, but the time taken to find the solution is unknown then the problem is classed as NP.

The concepts of computational complexity theory generated much research in the 1970s. An impressive survey is presented by Garey & Johnson [12]. Although the NP-Hardness of the TSP does not imply the exponential worst-case running times for its solutions are unavoidable, it does serve to reinforce one's belief that existence of a polynomial algorithm for the TSP is extremely unlikely [13].

The potential reward for coming up with the proof P = NP is so great that there is currently a $1,000,000 prize on offer by the Clay Mathematics Institute [11] [14].

2.1.2 The Generalised Travelling Salesman Problem

The Generalised Travelling Salesman Problem (GTSP) is a variation of the Traveling Salesman Problem in which not all nodes need to be visited by the tour [15]. The locations are clustered together, and only one location from each cluster needs to be visited by the salesman. For example, there are a number of cities across Europe, the cities are grouped by the country they are in, and a delivery driver has to visit one city from each country. This example would work in the same way in the United States of America (USA), where the driver had to visit one location within a selected number of states [16].

Figure 2.2 gives an example of the GTSP. The standard TSP is a type of GTSP, where each set of cities to visit is of size one [17].

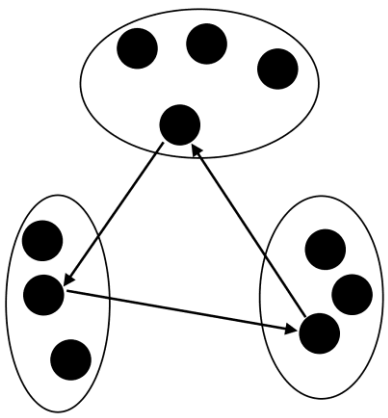

Figure 2.2: Tours in the Generalised Travelling Salesman Problem (GTSP) [17]

2.1.3 The Clustered Travelling Salesman Problem

The final type of TSP that will be discussed is the Clustered Travelling Salesman Problem (CTSP) [18]. The CTSP is a variant of the TSP where cities or locations are clustered together, the salesperson then travels to each cluster, but visits each location within each cluster before moving on to the next. The key difference between the GTSP and the CTSP is that the salesperson must visit all the cities in the cluster, instead of just one. There is an extra cost factor known as inter-cluster costs, the cost of travelling between clusters [18]. This can be seen in the real world for international travel. If the cities are clustered by country, there may be additional inter-cluster costs for travelling across borders, such as extra time due to border checks, visa costs, import duties etc. Figure 2.3 gives an example tour using the CTSP.

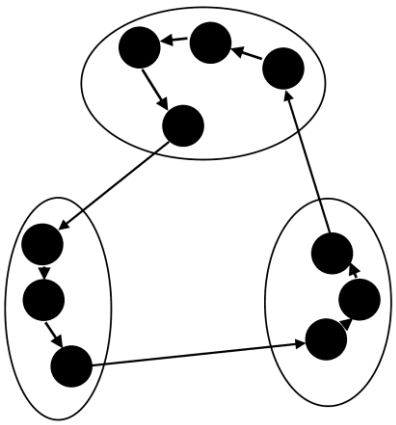

Figure 2.3: Tours in the Clustered Travelling Salesman Problem (CTSP) [19]

An additional dimension to the travelling salesman problem is the skills required to complete any given task at any given location. This means that the closest member of the workforce to a task may not necessarily have the required skill to complete the task. More will be discussed on multi-skilled workforces in section 2.4.

2.2 An Overview of Workforce Management Systems

For any organisation of meaningful size, managing the workforce effectively so that productivity is kept high is a complex problem. There are established methods to help tackle these workforce management problems.

One fundamental method for any organisation is scheduling. Scheduling can be defined as allocating tasks to resources. In many organisations, this can be real-time scheduling, which has the ability to adapt to ‘on the day’ changes and is the very-short-term. Any system in place that is tasked with scheduling will have to look at resource availability and skills. For mobile workforces, the complexity of this is increased by adding the resources current location and distance away from unscheduled tasks.

An additional layer of workforce management can occur before the scheduling phase. This phase has a number of names but is commonly referred to as ‘Planning’ or ‘Tactical Planning’ [20]. This phase is designed to support and inform sales teams and resource planners. The planning systems know which resources are available on what days, letting the sales team know if there is available resource capacity to deliver a service to a customer on any given day.

Described so far is a generalised view, so, for example, the planner will know there are 20 man-hours available next Wednesday, it takes two hours to travel to and deliver services to any customer, meaning there are ten slots available for customers on that day. If eight have been taken, it is still okay to book that customer in. If ten customers have been booked already, then another day will have to be chosen by the customer. Alternatively, the planners will know that two hours of overtime are needed and can plan that before impacting the customers first choice of date for service delivery. The planning phase is short to medium term from two days to two weeks ahead of scheduling.

The planning system cannot account for absences or emergency appointments on the day; this would be the job of the scheduler. However, the planner would know what the estimated ‘shrinkage’ would be and thus leave available time in each resource’s schedule to allow those resources to be allocated additional tasks on the day. Shrinkage here is the reduction in the expected supply due to factors that reduce overall productivity (i.e. illness, traffic and unexpected complications with a task). If too much time is left free the resources utilisation will become low, leave too little time and there is no capacity for error and appointments will be delayed, thus having an impact on customer service and causing a backlog to build up for the following day.

Scheduling and planning systems are vital for large organisations, and there are many publications and case studies that document the research and application of these systems [21] [22] [23].

There is one more area of workforce management, which is often overlooked, this is organisational design. Organisational design is particularly important for large organisations and especially large organisations that have a mobile workforce spread over a wide geographical area. Many large organisations have a hierarchical management structure and at the base of that structure are teams of resources, those teams are allocated responsibilities based on their skill sets. These teams can be anything from finance, human resources, call centre operators or sales. For mobile workforces, these teams could be a roaming sales force, utility engineers, home care workers or delivery personnel. Making sure each team has the right number of resources, in the right place, with the right skills is important to having an organisation set up correctly.

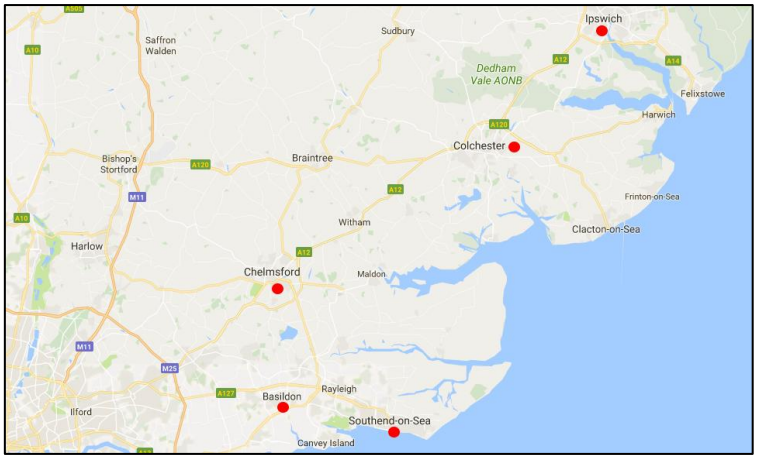

Figure 2.4 Local Area Map

For the case of a utility company, each engineering team at the base of the hierarchy is responsible for certain parts of the infrastructure. This is allocated on a geographical basis. For example, Southend-on-Sea and Basildon might be grouped as one area (see Figure 2.4 for reference), with the local engineers servicing the infrastructure in that area, Colchester and Ipswich might be another. Then, due to the hierarchical nature of the organisation, the Southend & Basildon area and the Colchester & Ipswich area would form part of the East Anglia Region. This structure means that an area manager can be overseen by a regional manager and there is a fixed reporting structure.

The challenge comes at the lower levels of this hierarchy. Is it the right decision to group Basildon and Southend in the first place, how broad should this area be? Who should work in these areas? If someone with the right skills lives in Chelmsford could they be assigned to Basildon & Southend, or maybe they are needed in the Colchester & Ipswich area. It's these sorts of questions that come under organisational design.

The benefits of getting these decisions correct can be significant. This means the planning systems will have better resourcing capabilities for the expected demands in each geographical area. It will also mean the organisation is more effectively set up to react to unexpected changes

in demand or resources. It won't necessarily negate all the effects of these changes, but it will mean the negative effect on the organisation and on customers will be reduced.

2.2.1 Working Areas (Patches)

One key aspect to organisational design for mobile workforces is the geography or territory the teams in the workforce are allocated. Each team is allocated a territory and the engineers in the teams will only receive tasks in that territory. These territories are known as Working Areas (WAs) or Patches. If an engineer is to do work in a neighbouring patch then this is usually at the request of a planner or patch manager who has noticed the neighbouring patch will be under resourced for a particular day, so the resource can be loaned across. However, the goal is to minimise this loaning as much as possible by configuring the patches and engineering teams correctly. Too much loaning can lead to decreased productivity, the resource being unfamiliar with the patch they have been allocated and instability for the engineer as they may not know where they will be working from one week to the next. The risk of this increases for an engineer the closer they are to patch borders.

The geographical patches are not fixed in their shape, they can be redesigned based on demand, this is a key aspect to organisational design for large scale workforces. The flexibility to change both the geographical shape of the patch and the team members that service that patch, give a greater range of flexibility to adapt to the ever-changing working environments. Figure 2.5 shows how the same infrastructure can be divided up into different patch designs.

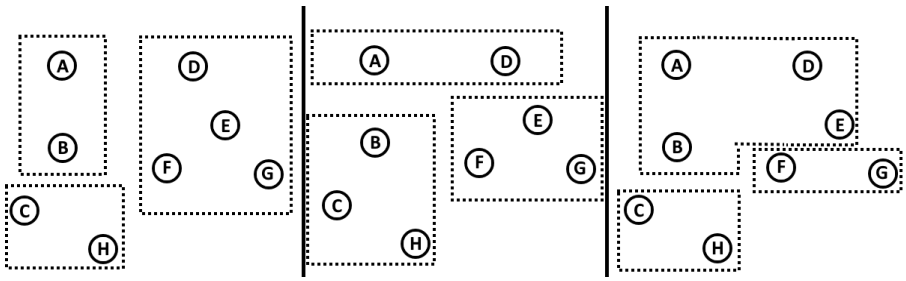

Figure 2.5 Possible Patch Designs

2.3 Large Scale Workforce Management Solutions

Due to the size of the workforce associated with utility companies, there have been several previous examples of successfully tackling parts of the workforce management problem. One such example is the scheduling system developed by British Telecom (BT) in the 1990s. The system was called Work Manager and was responsible for scheduling tasks to all of BT's 50,000 mobile engineers. In a paper summarising the success of this system, it states that the benefits of this solution were worth up to $250 million a year [24]

However, this summary of Work Manager also explained how each scheduling system had to work within the constraints of "175 separate groups that operate within non-overlapping geographical domains". This directly references the organisational design, where each of the 'domains' would be made up of a number of working areas as illustrated in Figure 2.6. The summary then goes on to say "(Geographical) domain-dependent requirements are rare. In fact, domains differ only on non-essential features, such as their road networks, their distribution of customers, the size and skills of their workforces, or their task-notification patterns".

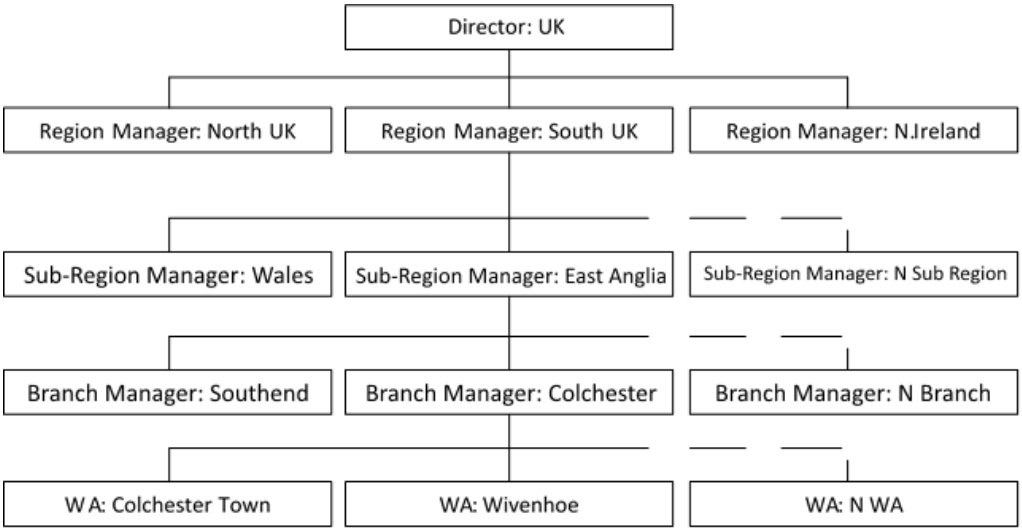

Figure 2.6 Large Organisational Structures

From this last statement, we can conclude that factors such as the distribution of demand, sizes of teams, skill profiles of teams and travel networks were not considered, as they are described

as ‘non-essential features’. The view that the organisational design was not important was a key factor in the lack of research and development in this area over the following decade. This meant the process of redesigning the organisational structure was a time-consuming manual process that required managers of domains and working areas to adjust teams and infrastructure responsibilities manually. In addition to this manual process being time consuming and unappealing, the results of the process were sub-optimal, due to the vast number of variables and constraints that would be needed to create new organisational designs effectively.

This example is specific to BT, however overlooking organisational design in favour of scheduling is not limited to infrastructure companies, [25] focuses on the scheduling and routing problem, very common for any mobile workforce, for healthcare workers that travel to patient homes. In this paper, they mention in relation to the feature of the problem, ‘the number of geographical areas (visit locations are grouped into areas)’. The features of each problem are then stated and ‘Number of Areas’ is one of just three major features. Lastly, when talking about preferences, the authors state ‘preferences include workers preferring to work in certain geographical areas, customers requiring workers with special skills’. The problem description is very similar to the one already stated by BT mobile engineering workforce. This highlights that there are many organisations that use a working area based organisational structure, yet the optimality of the organisation structure itself is overlooked in favour of scheduling.

In the journal article *Workforce scheduling and routing problems: literature survey and computational study* [26] there is mention of organisation design under the term ‘Clusterisation’. Where the reasons for clusterisation are stated as ‘employees may prefer not to travel more than a number of miles’, ‘companies assign employees to perform work only in certain geographical areas’ and ‘Clusters may also be created just to reduce the size of the problem by solving a number of clustered sub-problems’. This study also highlights the need for clusterisation in the home care service because ‘Clusterisation is based on municipalities’

borders to clearly define which authority (e.g. council, district, etc.) is responsible for each area'. For technicians, it states 'Companies with many branches across different regions use clusterisation to assign jobs to each branch when the scheduling is done centrally for all branches' for security personnel 'Customers are divided into regions (clusterisation) so that security guards living nearby are assigned to each region to reduce travelling time'. The conclusions of this survey say that for the workforce scheduling and routing problem clusterisation of locations is sometimes a characteristic that needs to be taken into account. However, the primary focus of the survey is to identify solutions to the broader problem of workforce scheduling and routing, of which it highlights a mixture of methods that include mixed integer linear programming (MIP), integer linear programming (IP) and a variety of meta-heuristics such as particle swarm optimisation (PSO) and simulated annealing (SA).

However, clusterisation is treated as a feature to the WSRP and not as a separate optimisation problem in itself. If sub-optimal clusters are used as a feature to the WSRP, then any optimisation algorithm attempting to achieve the best solutions for the WSRP will be restricted, making the optimisation much more difficult, or impossible, to achieve the desired results.

2.4 Workforce Optimisation Beyond the Workforce Scheduling and Routing Problem

2.4.1 The Vehicle Routing Problem (VRP)

There are a large number of real-world applications, both in North America and Europe that have widely shown that the use of computerised procedures for the distribution process planning produces substantial savings (generally from 5% to 20%) in the global transportation costs [27].

The success of the utilisation of Operations Research techniques is due to the development of computer systems, from both hardware and the software point of view and to the increasing integration of information systems into the productive and commercial processes [27]. Indeed, the growing prevalence of mobile applications in these processes is just one example of the mentioned increasing integration of information systems.

Problems concerning the distribution of goods between depots and final users are generally known as Vehicle Routing Problems (VRPs) [27]. VRPs can easily encompass the distribution of services too.

The VRP calls for the determination of the optimal set of routes to be performed by the fleet of vehicles to serve a given customer set, and it is one of the most important, and studied, combinatorial optimisation problems [27].

2.4.2 The Workforce Scheduling and Routing Problem (WSRP)

The Workforce Scheduling and Routing Problem (WSRP) is a natural extension to the VRP, and an in-depth survey has been conducted in [26]. It is widely used when employees have to travel between tasks, and the tasks have to be distributed among all employees so as to meet the organisation's objectives.

There are a number of objectives when it comes to the WSRP, some will be specific to the organisation others will be generic. They include:

- Maximising tasks completed,
- Minimising the total amount travelled,
- Minimising the task to travel ratio
- Minimising the number of vehicles

With a workforce of meaningful size (such as above 50) creating the most effective schedule is a challenging task without advanced software methods to assist in the decision-making.

The greater the workforce and the greater the number of tasks the workforce must complete the more scheduling and routing solutions become available. The possible solutions usually become so great that it falls into a combinatorial optimisation problem. Because the WSRP is a CO problem, metaheuristics are often used as a method of coming up with suitable schedules that meet the problems as best as possible.

To expand on the discussion of CO problems, CO is the field of discrete mathematics involving the resolution of the following problem.

Let X be a set of solutions and f a function that measures the value of each solution in X. The objective is to determine a solution s* ∈ X minimizing f, i.e. [28]:

$$f(s^*) = min_{s \in X} f(s) \tag{2-1}$$

Set X is presumed finite and is in general defined by a set of constraints. As an example, for a job scheduling problem on one machine, X can be made up of all job sequences satisfying precedence and priority constraints while *f* can correspond to the date at which the last job is finished (makespan) [28].

Depending on the industry, solving the WSRP in real-time adds to the complexity of the optimisation as real-time optimisation doesn't have the luxury of time as with offline optimisation task. However, if the real-time aspect is taken into account and the system produces results that are operationally good enough, where enough work is completed to meet task completion targets and reduce travel expenses, then this outcome is operationally acceptable. However, for larger, team-based multi-skilled workforces, this is only the first step to having a fully utilised workforce at the most efficient cost [28].

2.4.3 Multi-skilled Workforces

A multi-skilled workforce is one in which its employees can complete different types of tasks that require different skill sets, i.e. the workers possess a range of skills that allow them to participate in more than one work process [29]. For a simple example, a carpenter could be asked to do X and Y with no problem, but if asked to do Z would requires a different skill. Thus, would have to be trained in Z or reject the task.

For more complex industries such as utilities and construction, it is necessary to train employees on multiple skills to maximise each workers utilisation and to minimise workforce turnover (also known as 'churn'). Workforce turnover is defined as the percentage of new employees needed to replace the employees that have left the organisation. In large organisations a 5-10% churn is typical. However, if there is a much high rate of churn (over 20%), it could indicate a number of issues that should be immediately addressed. One issue could be that part of the workforce is only required part of the time and contractors are needed to make up this part of the workforce. This is because the organisation could not guarantee enough work for full-time employment.

Another issue is working conditions; workforce optimisation can help with working conditions to evenly balance workloads between employees and help tasks to be managed efficiently, rather than expected unreasonable amounts of work to be completed within a short deadline.

2.4.4 Team Allocation

When workforces are large enough to divide into teams the team structure, skills and responsibilities of the team are important factors to consider when designing optimal organisational structures. If the size of the team and the skills available do not match the demand profile of the tasks to be allocated to the team, this can have a noticeable impact on the utilisation of the team or can push up operating costs unnecessarily.

For mobile workforces, the task of optimal team allocation becomes more difficult. This is due to the location of the employees. A perfect candidate to meet the shortage in a team's skill set might be available, but if their starting location is too far away, they cannot be allocated to that team.

One aspect of team allocation that cannot necessarily be taken into account by any algorithm is the team dynamic. How employees work together, their own preferences, and with regards to mobile workforces and their local area knowledge. The area manager must consider these aspects. It is important to note as there may be hidden infeasible solutions to a problem when it comes to optimisation of humans.

2.5 Discussion

In this chapter, a description of the origins of workforce optimisation was given, explaining how this stems from the travelling salesman problem (TSP) and its variants. This chapter gave a detailed overview of three types of TSP, specifically highlighting that the clustered TSP is the most relevant to this work.

An overview of workforce management systems was described, explaining scheduling, planning and tactical planning. This chapter explained that organisational design is often overlooked in favour of these other solutions and hence why this problem is being tackled. It described how working areas (or patches) work in large organisations and how it is used for the division of labour and management responsibilities.

Other aspects of workforce optimisation that go beyond the traditional workforce scheduling and routing problem were described. The chapter also highlighted the aspects of multi-skilled workforce and team allocation as these directly link to some of the challenges of this thesis.

The next chapter will describe optimisation algorithms.

Chapter 3. An Overview of Selected Optimisation Algorithms

Optimisation is a procedure of finding and comparing feasible solutions until no better solutions can be found. Solutions are termed good or bad in terms of an objective which is often the cost of fabrication, amount of harmful gases, the efficiency of a process, product reliability or other factors [69].

When an optimisation problem modelling a physical system involves only one objective function, the task of finding optimal solutions is called *single-objective optimisation.* Currently, there exist single objective optimisation algorithms that work using gradient-based and heuristic-search techniques [69]. Heuristic search refers to the process of finding a solution to a problem that is 'good enough', because carrying out exhaustive search methods (looking at every single possible solution) is too costly, concerning time, computational power or monetary costs

Deterministic search principles (there is no randomness, and the same input to a system will always produce the same output) and stochastic search principles (where there is randomness, and the same input may produce a different output, depending on system and environmental variables) These allow optimisation algorithms to find globally optimal (the best possible) solutions more reliably [69].

In order to widen the applicability of an optimisation algorithm in various problem domains, natural and physical principles are mimicked to develop robust optimisation algorithms. Evolutionary algorithms and simulated annealing are two examples of such algorithms [69]

3.1 Genetic Algorithms

Concerning its internal functioning, a genetic algorithm is an iterative procedure which usually operates on a population of consistent size and is executed in the following way:

An initial population of individuals (also called “solutions”, “solution candidates” or “chromosomes”) is generated randomly or heuristically. During each iterative step (also called a “generation”) the individuals of the current population are evaluated and assigned a certain fitness value.

The fitness value is crucial to identifying strong individuals from weak individuals. There is usually a fitness function which takes the characteristics of the individual to compute the fitness value based on the environment and objectives of the optimisation.

In order to form a new population, individuals are first selected (usually with a probability proportional to their relative fitness value), and then produce offspring candidates, which in turn forms the next generation of parents. This ensures that the expected number of times an individual is chosen is approximately proportional to its relative performance in the population [70].

For producing new solution candidates, genetic algorithms use two operators, namely crossover and mutation:

- Crossover is the primary genetic operator. It takes two individuals, called parents, and produces one or two new individuals, called offspring, by combining parts of the parents’ characteristics (also known as “genes”). In its simplest form, the operator works by swapping (exchanging) substrings before and after a randomly selected crossover point. More on Crossover operators can be found in Section 3.1.2.2 [70]

- Mutation is the second genetic operator. It is essentially an arbitrary modification which helps to prevent premature convergence by randomly sampling new points in the search space. In the case of bit strings, mutation is applied by simply flipping bits randomly in a string, with a certain probability called mutation rate. See more on representation and mutation in sections 3.1.3 and 3.1.2.3 respectively [70]

Genetic Algorithms (GAs) are stochastic iterative algorithms, which cannot guarantee convergence (all individuals in the population are identical); termination is at this moment commonly triggered by reaching a maximum number of generations, by finding an acceptable solution or more sophisticated termination criteria, including permutation convergence.

3.1.1 Biological Terminology

The approximate way of solving optimisation problems by genetic algorithms holds a strong analogy to the basic principles of biological evolution. The fundamentals of natural evolution theory, as it is considered nowadays, refer to the theories of Charles Darwin, which were published in 1859 in his most well-known work "The Origin of Species: By Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life" [71]. In this work Darwin states the following five major ideas:

- Evolution, changes in lineages, occurs and occurred over time.
- All creatures have common descent.
- Natural selection determines changes in nature.
- Gradual change, i.e. nature changes somehow successively
- Speciation, i.e. Darwin claimed that the process of natural selection results in populations diverging enough to become separate species.

This formed the solid foundations on which evolutionary biology has been based. From the field of evolutionary biology comes the terminology used in genetic algorithms [70].

- All living organisms consist of cells containing the same set of one or more chromosomes, i.e. strings of DNA. A gene can be understood as an "encoder" of a characteristic, such as eye colour. The different possibilities for a characteristic (i.e. brown, green, blue and grey) are called alleles. Each gene is located in a particular position (locus) on the chromosome [70].
- Most organisms have multiple chromosomes in each cell. The sum of all chromosomes, i.e. the complete collection of genetic material, is called the genome of the organism and the term genotype refers to the particular set of genes contained in a genome. Therefore, if two individuals have identical genomes, they are said to have the same genotype [70].
- Organisms whose chromosomes are arranged in pairs are called diploid, whereas organisms with unpaired chromosomes are called haploid. In nature, most sexually reproductive species are diploid. Humans, for instance, have 23 pairs of chromosomes in each somatic cell in their body. Recombination (crossover) occurs during sexual reproduction in the following way [70]:
- For producing a new child, the genes of the parents are combined to eventually perform a new diploid set of chromosomes. Offspring are subject to mutation where elementary parts of the DNA (nucleotides) are changed. The fitness of an organism (individual) is typically defined as its probability to reproduce, or as a function of the number of offspring the organism has produced [70].

For the sake of simplification, in genetic algorithms, the term chromosome refers to a solution candidate. The genes are either single bits or small blocks of neighbouring bits that encode as a particular element of the solution. Alleles are usually 0 or 1, however, for larger alphabets, more alleles are possible at each locus (i.e. Real-Value encoding) [70].

Despite human evolution being based on diploid representation, most applications of genetic algorithms are haploid representation. This is likely due to its simplicity in representation and implementation [70].

3.1.2 Genetic Operators

3.1.2.1 Selection

In genetic algorithms, once a fitness value has been assigned to each individual in a population, the set of solutions, that are to be "mated" in a given generation, is to be produced. In a standard genetic algorithm, the probability that a chromosome of the current population is selected for reproduction is proportional to its fitness. There are many methods available to accomplish this selection, Proportional Selection (also known as Roulette wheel Selection) and Tournament Selection are two of the most popular [72] [34] [35].

- Roulette Wheel Selection: In this method of selection, the expected number of descendants for an individual *i* is given as $p_i = \frac{f_i}{\bar{f}}$ with $f: S \rightarrow \mathbb{R}^+$ denoting the fitness function and $\bar{f}$ representing the average fitness for all individuals. Therefore, each individual of the population is represented by a space proportional to its fitness. By repeatedly spinning the wheel, individuals are chosen with random sampling with replacement [31] [33].
- Tournament Selection: There are a number of variations. However the most common is *k*-tournament selection where *k* individuals are selected from the population at random. Then the fittest individual of the *k* selected ones is considered for reproduction. In this variant selection pressure can be scaled quite easily by choosing an appropriate number for *k* [31] [33].

3.1.2.2 Crossover

In its easiest formulation, which is suggested in the canonical GA for binary encoding, crossover takes two individuals and cuts their chromosome strings at some chosen position. The produced substrings are then swapped to produce two new full-length chromosomes [31]. Conventional crossover techniques for binary representation include:

- Single Point Crossover

A single random cut is made, producing two head sections and two tail sections. The two tail sections are then swapped to create two new individuals (chromosomes). Figure 3.1 schematically sketches this crossover method which is also called one-point crossover [31].

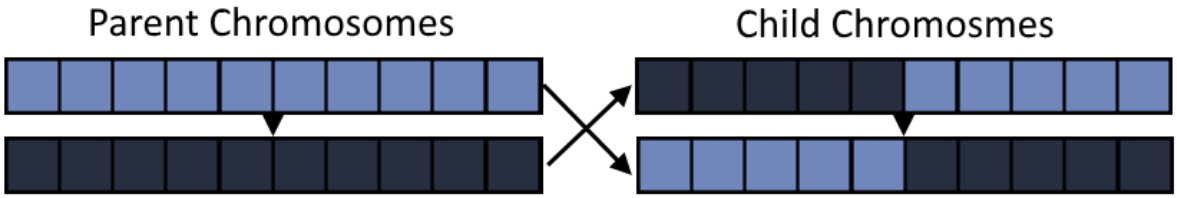

Figure 3.1 Single-Point Crossover [36]

- Multiple Point Crossover

One natural extension of the single point crossover is the multiple point crossover: In an n-point crossover there are *n* crossover points and substrings are swapped between the *n* points. According to some researchers, multiple point crossover is more suitable to combine good features present in strings, because it samples uniformly along the full length of a chromosome [37]. At the same time, multiple point crossover becomes increasingly disruptive with an increasing number of crossover points, i.e. the evolution of longer building blocks becomes more and more difficult. Decreasing the number of crossover points during the run of the GA may be a good compromise [31]. Multiple point crossover is illustrated in Figure 3.2.

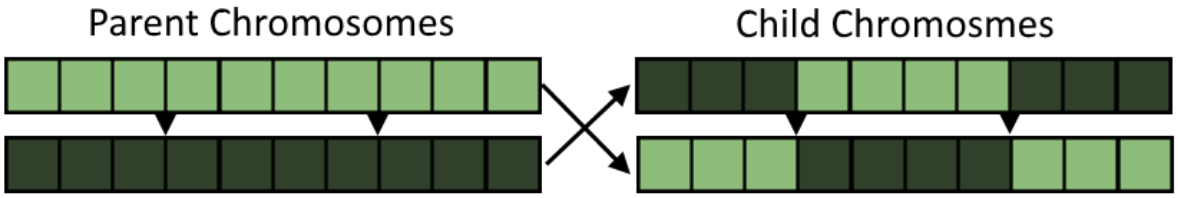

Figure 3.2 Multiple Point Crossover [36]

- Uniform Crossover

Given two parents, each gene in the offspring is created by copying the corresponding gene from one of the parents. The selection of the corresponding parent is undertaken by a randomly generated crossover mask: At each index, the offspring gene is taken from the first parent if there is a 1 in the mask at this index, and otherwise (if there is a 0 in the mask at this index) the gene is taken from the second parent. Due to this construction principle uniform crossover does not support the evolvement of higher order building blocks [31]. Uniform crossover is illustrated in Figure 3.3

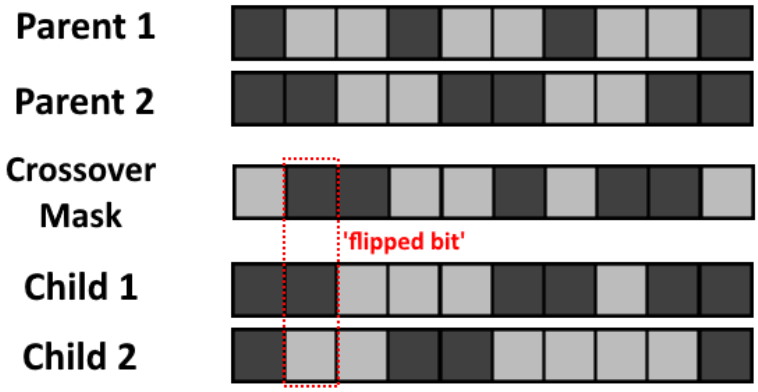

Figure 3.3 Uniform Crossover [38]

As the number of proposed problem-specific crossover techniques has been growing, a good discussion of crossover related issues can be found in [39] and [40].

3.1.2.3 Mutation

Mutation allows undirected jumps to slightly different areas of the search space. The basic mutation operator for binary coded problems is bitwise mutation. Mutation occurs randomly

and vary rarely with a probability p_m; typically, the mutation rate is less than ten percent. In some cases, mutation is interpreted as generating a new bit, and in others, it is interpreted as flipping the bit [31].

In higher order alphabets, such as integer numbering formulations, mutation takes the form of replacing as allele with a randomly chosen value in the appropriate range with probability p_m. However, for combinatorial optimisation problems, such mutation schemes can cause difficulties with chromosome legality; for example, multiple copies of a given value can occur which might be illegal for some problems (including routing). Alternatives suggested in the literature include pairwise swap and shift operations, as described in [41].

Also, adaptive mutation schemes similar to mutation in the context of evolutionary strategies are worth mentioning. Adaptive mutation schemes vary either the rate or the form of mutation, or both during a GA run. For instance, mutation is sometimes defined in such a way that the search space is explored uniformly at first and more locally towards the end, in order to do a kind of local improvement of candidate solutions [39].

3.1.3 Chromosome Representation

A key issue with most evolutionary algorithm techniques is the choice of a suitable encoding scheme, or how the solution to a problem will be represented through a chromosome. The choices are mainly binary, floating-point, or some grammar-based representation, see Figure 3.4. Holland [42] used the argument that a genome with a small number of alleles but long strings has a higher degree of parallelism than a numeric scheme with a larger number of alleles but short (floating point) strings. [43]

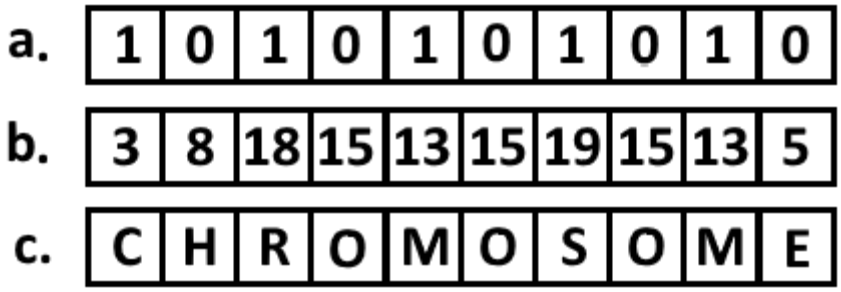

Figure 3.4 Chromosome Encoding a) Binary b) decimal c) Alphanumeric

However, as Mitchel [44] points out, for real-world applications it is frequently more natural to use a decimal or symbolic representation scheme, as this is an easier mapping to the actual representation of the problem space; for example, the weights in a neural network.

The text by Michalewicz [39] also offers a useful analysis of the relative merits of binary versus floating-point representations. The conclusion is that a floating-point scheme is faster, is more consistent between runs, and can provide a higher precision for large domain applications [43].

> *The binary alphabet offers the maximum number of schemata per bit of information of any coding and consequently the bit string representation dominated genetic algorithm research. This coding also facilitates theoretical analysis and allows elegant genetic operators. But the implicit parallelism does not depend on using bit strings and it may be worth-while to experiment with large alphabets. In particular for parameter optimisation problems with variables over continuous domains, we may experiment with real-coded genes together with special genetic operators developed for them* [39]

In relation to the workforce optimisation problems outline later in this thesis, floating point (or real-value) representation is the most suitable given the outlined advantages.

3.1.4 Implementing a Genetic Algorithm

The steps to genetic algorithms are shown in Figure 3.5 and the pseudocode for implementing a standard genetic algorithm is shown in Figure 3.6.

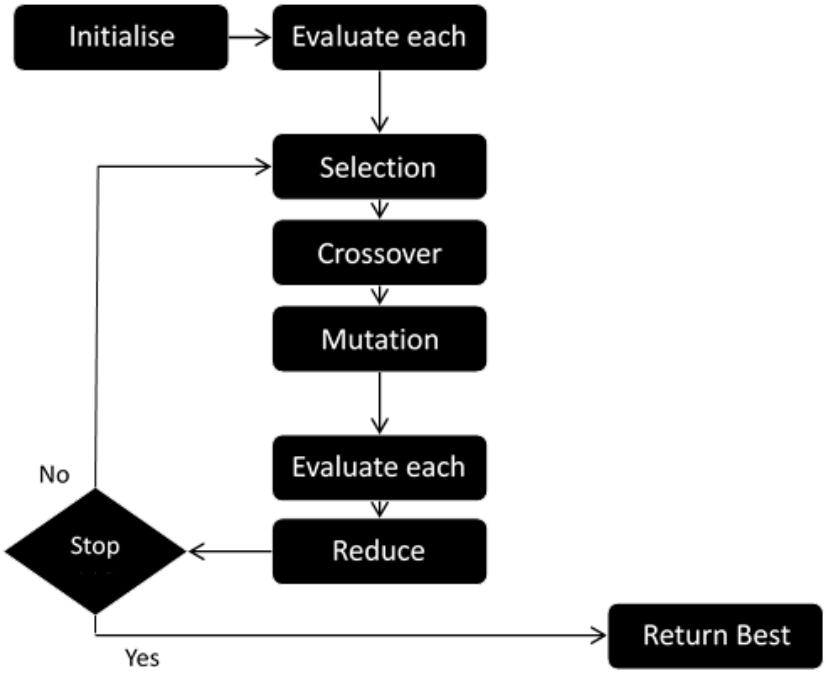

Figure 3.5: Flow of a Genetic Algorithm (GA)

```
t = 0;
initPopulation P(t);
evaluatePopulation(P);
Loop (until stopping criteria){
     For i = 1 to number of individuals
     P' = selectParents(P(t))
     P' = crossover(P')
     P' = mutate(P')
     evaluatePopulation(P')
     P = newPopulation(P')
     t++;
}
```

Figure 3.6: Pseudocode for a standard Genetic Algorithm [45]

3.2 Particle Swarm Optimisation

Particle Swarm Optimisation (PSO), which has its roots in artificial life and social psychology as well as engineering and computer science, differs from other evolutionary computation methods (such as the discussed GA in Section 3.1) in that the population members, called *particles*, are flown through the problem hyperspace. When the population is initialised, in addition to the variables being given random values, they are stochastically assigned velocities. Each iteration each particle's velocity stochastically accelerated towards its previous best position (where it had its highest fitness value) and towards a neighbourhood best position (the position of the highest fitness by any particle in the neighbourhood/population) [46]

The process of the optimisation is as follows [46]:

Each individual in the population, a particle, represents a potential solution to a problem. Each particle is treated as a point in a D-dimensional space. The *i*th particle is represented as $X_I = (x_{i1}, x_{i2}, \dots, x_{iD})$. The best previous position (the position giving the best fitness) of any particle is recorded and represented as $P_I = (p_{i1}, p_{i2}, \dots, p_{iD})$. The index of the best particle among all the particles in the population is represented by the symbol *g*. The rate of the position change (velocity) for particle i is represented as $V_I = (v_{i1}, v_{i2}, \dots, v_{iD})$. The particles are manipulated according to the following [46]:

$$v_{id} = v_{id} + c_1 \times R1 \times (p_{id} - x_{id}) + c_2 \times R2 \times (p_{gd} - x_{id}) \quad (3\text{-}1)$$

$$x_{id} = x_{id} + v_{id} \quad (3\text{-}2)$$

Where c_1 and c_2 are two positive constants, R1 and R2 are two random values in the range 0 to 1. The second part of equation (3-1) is the 'cognition' part. Which represents the private thinking of the particle itself. The third part of equation (3-1) is the 'social' part which represents the collaboration among the particles. Equation (3-1) is used to calculate the particles new velocity according to its previous velocity and the distance from its current

position from its own best experience (position) and the group's best experience. Then the particle flies towards a new position according to equation (4-2). The performance of each particle is measured by a predefined fitness function [47]. The fitness function here is like the fitness function of a GA. As mentioned in [48] a recommended choice for the constants c_1 and c_2 is 2.

3.2.1 Implementing a Particle Swarm Algorithm

The flow for a PSO is shown in Figure 3.7, and the Pseudocode for the PSO algorithm is shown in Figure 3.8 [46]:

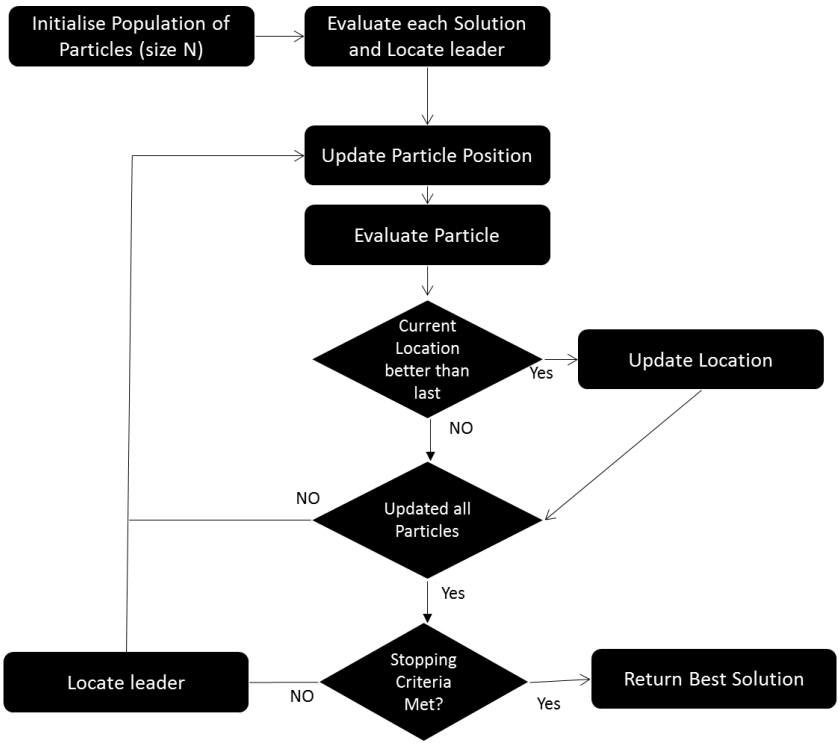

Figure 3.7: Flow of a Particle Swarm Algorithm

```
Loop (until stopping criteria){
     For i = 1 to number of individuals
          If(G(x⃗_i) > G(p⃗_i)){          //G()evaluates fitness
               For d = 1 to dimensions {
                    p_id = x_id          //p_id is best so far
               }
               Next d
          }
          g = i                         //arbitrary
          For j = indexes of neighbours
               If(G(p⃗_j) > G(p⃗_g))
                    g = j
          Next j
          For d = 1 to number of dimensions
               v_i(t) = v_i(t − 1) + φ_1(p_id − x_id(t − 1)) + φ_2(p_gd − x_id(t − 1))
               v_id ∈ (−V_max, +V_max)
               x_id(t) = x_id(t − 1) + v_id(t)
               If p_id < S(v_id(t)) then x_id(t) = 1; else x_id(t) = 0;
          Next d
     Next i
}
```

Figure 3.8: Pseudocode for the PSO algorithm [46]

3.3 Simulated Annealing

Simulated annealing is a technique for combinatorial optimisation problems, such as minimising functions of very many variables. Because many real-world design problems can be cast in the form of such optimisation problems, there is intense interest in general techniques for their solution. Simulated annealing is one such technique (it was introduced in 1983 by

Kirkpatrick et al. [49]) with an unusual pedigree: it is motivated by an analogy to the statistical mechanics of annealing in solids [50].

To understand why such a physics problem is of interest, consider how to coerce a solid into a low energy state. A low energy state usually means a highly ordered state, such as a crystal lattice; a relevant example here is the need to grow silicon in the form of highly ordered, defect-free crystals for use in semiconductor manufacturing. To accomplish this, the material is annealed: heated to a temperature that permits many atomic rearrangements, then cooled carefully and slowly, until the material freezes into a good crystal. Simulated annealing techniques use an analogous set of "controlled cooling" operations for non-physical optimisation problems, in effect transforming a poor, unordered solution into a highly optimised, desirable solution. Thus, simulated annealing offers an appealing physical analogy for the solution of optimisation problems, and more importantly, the potential to reshape mathematical insights from the domain of physics into insights for real optimisation problems [50].

For our purposes, a combinatorial optimisation problem is one in which we seek to find some configuration of parameters $\bar{X} = (X_1, X_2, \dots, X_N)$ that minimises some function $f(\bar{X})$. This function is usually referred to as the cost or objective function (like it is in GAs). Realistic design problems may require many parameters and a complex cost function. Consider, for example, deciding the placement of components on the surface of an integrated circuit in an optimal way. We may seek to maximise the ability to route wires to interconnect these components, minimise the overall chip area, maximise the manufacturing yield of the chip, minimise the deviation from specified timing constraints, and so forth. The cost function may be very sophisticated, and the number of parameters large: perhaps 10^3 to 10^5 variables to specify the positions for each component [50].

Heuristic strategies for solving such problems come in several styles. Sometimes constructive heuristics can be found, which build up a good answer directly, piece by piece. Of more interest are iterative improvement strategies, which attempt to perturb (alter or change) some existing, suboptimal solution in the direction of a better, lower-cost solution. The idea can be neatly illustrated with a "balls and hills" diagram, as shown in Figure 3.9. All the values of $f(\bar{X})$ define a cost surface. In Figure 3.9 it is shown schematically for N = 1, i.e. a single parameter, as a set of hills and valleys in the cost surface. The ball represents the current configuration we plan to perturb. In practice, iterative improvement algorithms often start with a random initial configuration or where possible, with a heuristically constructed initial configuration that is not as costly as a random solution [50].

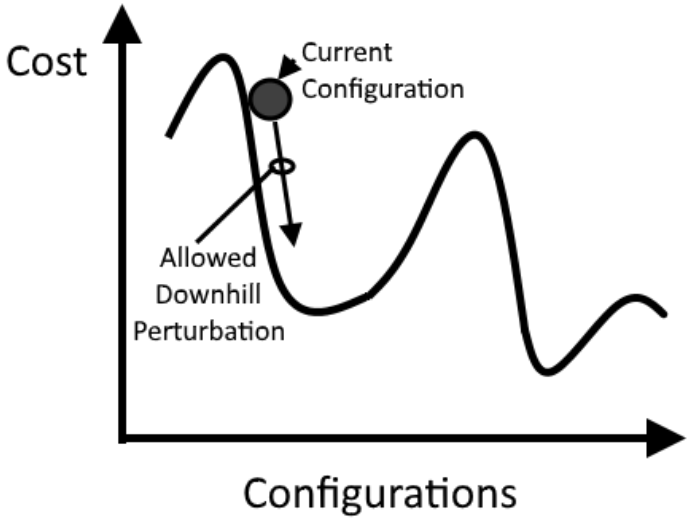

Figure 3.9: Ball and Hills Diagram [50]

From Figure 3.9, an obvious approach is to explore easily reached neighbouring configurations and to select the one with the least cost. In practice, some small random perturbation is attempted, to yield a nearby solution. This process can continue starting from the new configuration until no further improvements are obtained, at which point the process terminates. This strategy seems reasonable, but it has a serious problem, it is easily trapped in

local minima; solutions that look good in some small neighbourhood of the cost function but are not necessarily global optimal [50].

Standard iterative improvement is a *downhill-only* style. In Figure 3.9 each new perturbation moves to a configuration *downhill* from the previous one, thus becoming trapped in the local minima. In practice, one scheme to overcome this is simply to try many random initial configurations, improve each, and use the best answer found. However, for very large problems the computational expense is great. The number of random starts needed to sample the cost surface adequately is unreasonable, and we still have no guarantees of finding a good answer [50].

Simulated annealing offers a strategy very similar to iterative improvement, with one major difference: annealing allows perturbations to move uphill in a controlled fashion. Individual perturbations are now referred to as *moves*. As each move can now transform one configuration to a worse configuration, it is possible to jump out of local minima and potentially fall into a more promising downhill path. However, because the uphill moves are carefully controlled, we need not worry about getting close to a good solution, only to randomly jump uphill to some far worse one [50].

The relevant analogy here is physical annealing of a solid. To coerce some material into a low energy state, we heat it, then cool it very slowly, allowing it to come to thermal equilibrium (no heat flows between two systems when they are connected by a path permeable to heat) at each temperature. Simulating this process is very similar to a combinatorial optimisation task. For the physical system, the goal is to find some arrangement of atomic particles (a configuration) that minimise the energy (cost) of the system. The basic requirement for simulating this process it the ability to simulate how the system reaches thermodynamic equilibrium at each fixed temperature in the schedule of decreasing temperatures used to anneal

it. Toward this end, the Metropolis algorithm, developed in 1953 [51], can be employed. The algorithm is described in Section 3.3.1

3.3.1 Implementing a Simulated Annealing Algorithm

As mentioned in Section 3.3, the Metropolis algorithm, which is shown in Figure 3.10, can be used to simulate the annealing process, thus forming a simulated annealing algorithm for combinatorial optimisation.

```
M = number of moves to attempt
T = current temperature
For m = 1 to M{
     Generate a random move, e.g. move a particle;
     Evaluate the change in energy ΔE;
     If(ΔE < 0){
          //Downhill move, accept it.
          Accept this move and update the configuration;
     }else{
          //Uphill move, accept maybe.
          Accept with probability P = e^(-ΔE/T)
          Update configuration if accepted

     }
}
```

Figure 3.10: Pseudocode for Metropolis Algorithm [50]

The idea, as in iterative improvement, is to propose some random perturbation, such as moving a particle to a new location, then evaluating the resulting change in energy ΔE. If the energy is reduced, $\Delta E < 0$, the new configuration has lower energy and is accepted as the starting point for the next move. However, if the energy is increased, $\Delta E > 0$, the move *may* still happen: the

new, higher energy configuration may be accepted. In physical systems, jumps to higher energy actually do happen, but they are moderated by the current temperature T. [50].

At higher temperatures the probability of large uphill moves in energy is large; at low temperatures the probability is small. The Metropolis Algorithm models this with a Boltzmann distribution: the probability of an uphill move of size ΔE at tempreture T is $\Pr[accept] = e^{-\Delta E/T}$. In practice, this probabilistic acceptance is achieved by generating a uniform random number R in [0,1] and comparing it against the threshold $\Pr[accept]$. Only if $R < Pr\ [accept]$ s the move accepted. Thus, very probable moves can be rejected, and very improbable moves can be accepted, at least occasionally. By successively lowering the temperature and running the algorithm, we can simulate the material coming into equilibrium at each newly reduced temperature, and thus effectively simulate the physical annealing [50].

We can readily apply this simulated annealing procedure to arbitrary combinatorial optimisation problems concerning standard iterative improvement; the only addition is the notion of a temperature parameter. In physical systems, temperature has a physical meaning; in arbitrary nonphysical optimisation tasks, the temperature is simply a control mechanism. The idea is to employ a *cooling schedule*, a sequence of decreasing temperatures, to moderate the acceptance of uphill moves over the course of the solution [50].

Initially, the *effective* temperature parameter is high enough to permit an aggressive, essentially a random search of the configuration space. Most uphill moves are allowed: we tend to improve the value of the cost function here, but some local minima can also be avoided. At the coldest temperatures the solution is close to freezing into its final form, and very few disruptive uphill moves are permitted. In this temperature regime, annealing closely resembles standard downhill –only iterative improvement [50].

3.4 Multi-Objective Genetic Algorithms

A significant portion of research and application in the field of optimisation considers a single objective, although most real-world problems involve more than one objective. The presence of multiple conflicting objectives (such as simultaneously minimising the cost of fabrication and maximising product reliability) are natural in many problems and makes the optimisation problem interesting to solve [30].

Since no one solution can be termed as an optimal solution to multiple conflicting objectives, the resulting multi-objective optimisation problem resorts to a number of trade-off solutions. Classical optimisation methods can, at best, find one solution in one simulation run, thereby making those methods inconvenient to solve multi-objective optimisation problems [30].

Current evolutionary multi-objective optimisation applications can be roughly classified into three large groups: engineering, industrial and scientific [52].

Engineering applications are by far the most popular in the literature. Engineering disciplines normally have problems with better understood mathematical models which facilitate the use of evolutionary algorithms like genetic algorithms. Some examples include structural engineering [53] [54], robotics [55] [56] and telecommunications [57] [58] [52].

Industrial applications occupy the second place in popularity, with scheduling being the most popular sub-discipline [59] [60]. The industrial applications area is where the problems in this thesis sit. Particularly as some aspects are derived from scheduling problems. Other applications include design and manufacture [61] and management [62] [52].

Finally, there is a wide variety of scientific applications, with computer science being the most popular [63] [64]. Other applications include chemistry [65], physics [66] and medicine [67] [52].

3.4.1 Dominance

Most multi-objective optimisation algorithms use the concept of domination. Domination is described as the following:

We assume there are M objective functions. In order to cover both minimisation and maximisation of objective functions, we use the operator $\lhd$ between two solutions i and j as i$\lhd$j to denote that solution i is better than solution j on a particular objective. Similarly, i$\rhd$j for a particular objective implies that solution i is worse than solution j on this objective. For example, if an objective function is to be minimised, the operator $\lhd$ would mean the < operator, whereas if the objective function is to be maximised, the operator $\lhd$ would mean the > operator [30].

The following outlined conditions required for dominance covers both minimisation and maximisation objectives. A solution $x^{(1)}$ is said to dominate the other solution $x^{(2)}$, if both conditions 1 and 2 are true [30]:

1. The solution $x^{(1)}$ is no worse than $x^{(2)}$ in all objectives, or $f_j\left(x^{(1)}\right) \not\rhd f_j\left(x^{(2)}\right) for\ all\ j = 1, 2, \dots, M$ [30]
2. The solution $x^{(1)}$ is strictly better than $x^{(2)}$ in at least one objective, or $f_{\bar{j}}\left(x^{(1)}\right) \lhd f_{\bar{j}}\left(x^{(2)}\right)$ for at least one $\bar{j} \in \{1, 2, \dots, M\}$ [30]

If any of the conditions are violated, the solution $x^{(1)}$ does not dominate the solution $x^{(2)}$. If $x^{(1)}$ dominates the solution $x^{(2)}$ (or mathematically $x^{(1)} \preceq x^{(2)}$), it is also customary to write the following [30]:

- $x^{(2)}$ is dominated by $x^{(1)}$
- $x^{(1)}$ is non-dominated by $x^{(2)}$, or
- $x^{(1)}$ is non-inferior to $x^{(2)}$

Let us consider a two-objective optimisation problem, with five different solutions shown in the objective space as illustrated in Figure 3.11. Objective 1 needs to be maximised and Objective 2 needs to be minimised. Since both objectives are important to us it is difficult to determine which solution is best with respect to both objectives. We can use the dominance conditions to decide which solution is better among any two given solutions in terms of both objectives. For example, if solution 1 and solution 2 are to be compared, we observe that solution 1 is better in both objectives. Thus, both the dominance conditions are met in this case, so solution 1 is dominant, i.e. better [30].

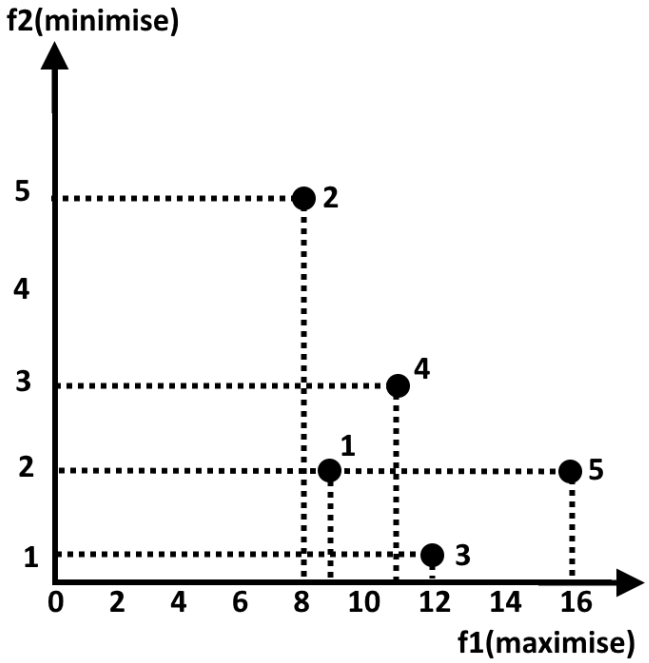

Figure 3.11: Five Solutions in a Two Objective Space [30]

3.4.2 Pareto Optimality

If we continue to analyse Figure 3.11, we can compare solutions 3 and 5. We observe that solution 5 is better than solution 3 in the first objective, while solution 5 is worse than solution 3 in the second objective. Thus, the first condition is not satisfied for both of these solutions. This simply suggests we cannot conclude that solution 5 dominates solution 3, nor can we say solution 3 dominates solution 5. When this happens, it is customary to say solutions 3 and 5

are non-dominated with respect to each other. When both objectives are important, it cannot be said which of the two solutions 3 and 5 is better [30].

For a given finite set of solutions, we can perform all possible pair-wise comparisons and find which solution dominates which and which solutions are non-dominated concerning each other. At the end, we expect to have a set of solutions, any two of which do not dominate each other [30].

This set also has another property. For any solution outside of this set, we can always find a solution in this set which will dominate the former. Thus, this particular set has the property of dominating all other solutions which do not belong to that set. This set is given special names; it is called the *non-dominated set* or the *Pareto-Optimal set.* Sets are also known as Fronts, and thus the term *Pareto Front* is commonly used to describe the non-dominated set. [30] Figure 3.12 marks the Pareto-optimal set with continuous curves for four different scenarios with two objectives.

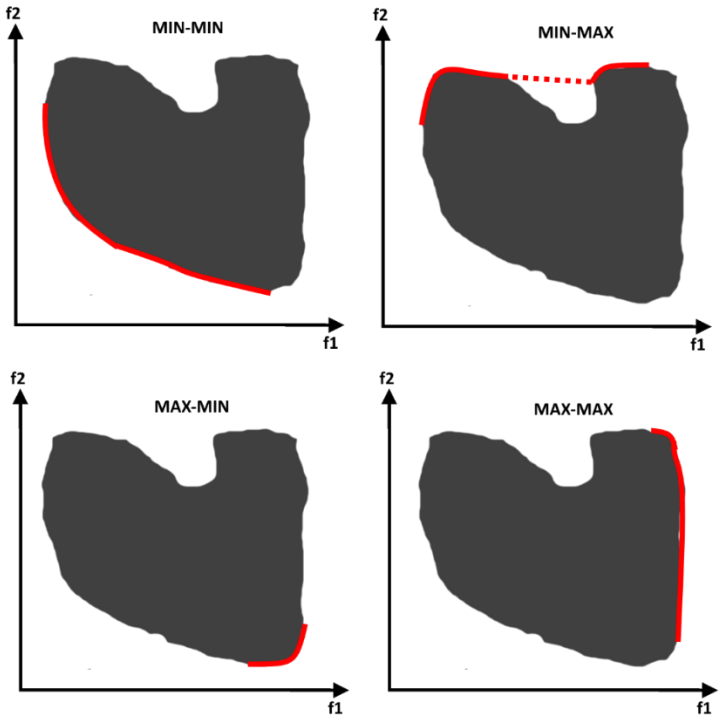

Figure 3.12: Pareto-Set 4 Different Scenarios [30]

One final example of Pareto optimality and how all solutions are grouped into sets is shown in Figure 3.13. This diagram shows solutions in a multi-objective problem with two minimisation objectives. All the solutions have been grouped into a total of four sets, or fronts, with the dominating set shown as the Pareto front. Figure 3.13 also shows an infeasible point, a point in the search space which is impossible to achieve given the optimisation and environmental constraints.

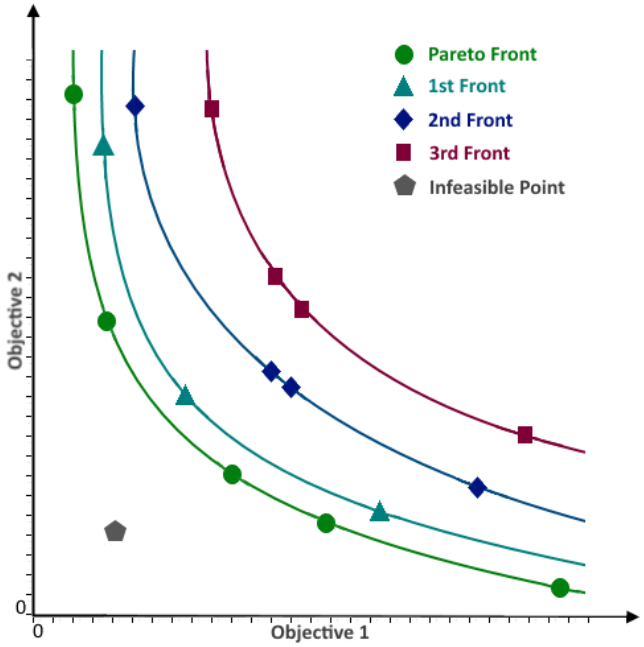

Figure 3.13 Fronts in a multi-objective optimisation problem

3.4.3 NSGA-II

The elitist Non-Dominated Sorting Genetic Algorithm II (NSGA-II) was proposed by Deb et al [68]. It was created to address a number of issues associated with multi-objective GAs that existed at the time. Issues such as:

- High Computational Complexity of sorting algorithm:
- Lack of elitism.
- The need to specify the sharing parameter.

Multi-objective GAs (MOGAs) differ from traditional (or single-objective) GAs in the fitness evaluation and comparison only. The remaining parts of the algorithm, such as Selection, Crossover, Mutation (as described in Section 4.1.2) are largely identical. NSGA-II concerns itself with the population of solutions.

In NSGA-II, when the offspring population Q_t is being generated from the parent population P_t the two populations are combined together to form R_t of size 2N (where N is the size of the initial population) [30].

Then a non-dominated sorting algorithm is used to classify the entire population R_t. Although this requires more effort compared to performing the non-dominated sort on Q_t alone, it allows a global non-domination check among the offspring and parent solutions. This global check is how elitism is handled in NSGA-II [30].

Once the non-dominated sorting is over, the new population is filled by solutions of different fronts, one at a time. The filling starts with the Pareto front and continues with solutions on the second front, then the third and so on. Since the overall population size of R_t is 2N, not all fronts may be accommodated in N slots. All fronts which could not be accommodated are simply deleted. Figure 3.14 illustrates this sorting procedure.

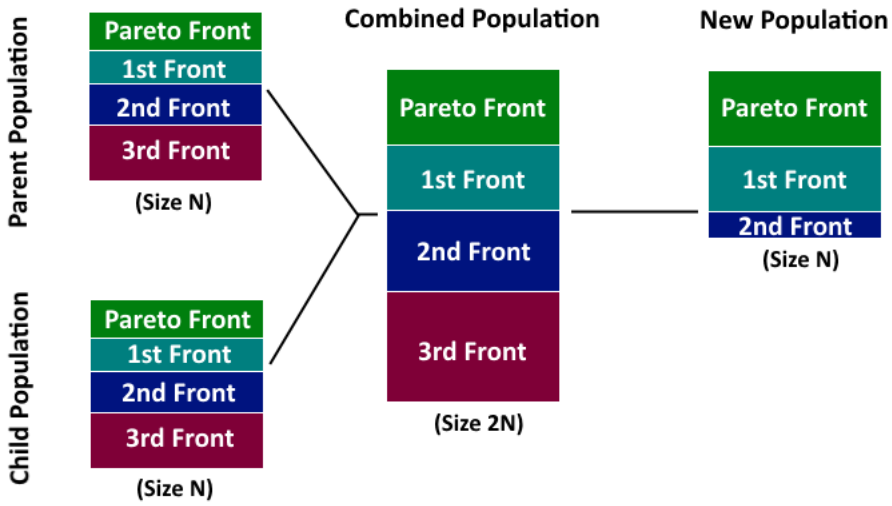

Figure 3.14: Illustration of the Sorting Procedure [30]

The following is an outline of NSGA-II steps, where initially a random population P_0 is created. Each member of the initial population is evaluated and ranked, then used to create an offspring population Q_0 of size N [30].

Step 1: Combine parent and offspring populations and create $R_t = P_t \cup Q_t$. Perform non-dominated sorting on R_t and identify the different fronts: $F_i, i = 1, 2, \dots, etc$. [30]

Step 2: Set new population $P_{t+1} = \emptyset$, set a counter i = 1. While $|R_{t+1}| + |F_i| < N$, perform $P_{t+1} = P_{t+1} \cup F_i$ and i = i +1 [30].

Step 3: Perform the crowding-sort $(F_i <_c)$ procedure and include the most widely spread $(N - |P_{t+1}|)$ solutions by using the crowding distance values in the sorted F_i to P_{t+1} [30].

Step 4: Create offspring population Q_{t+1} from P_{t+1} by using the crowding tournament selection, crossover and mutation operators [30].

3.4.3.1 Crowded Distance Tournament Selection

The crowded comparison operator $<_c$ compares two solutions and returns the winner of the tournament. It assumes that every solution i has two attributes [30]:

1. A non-domination rank r_i in the population
2. A local crowding distance (d_i) in the population. d_i of a solution i is a measure of the search space around i which is not occupied by any other solution in the population.

Based on these two attributes, we can define the crowded tournament selection operator as follows [30]:

1. If solution i has a better rank, that is $r_i < r_j$
2. If they have the same rank, but solution i has a better crowding distance than solution j, that is $r_i < r_j$ and $d_i > d_j$

This states that either the solution with the highest rank wins or if they have the same rank, the solution with the highest crowding distance wins [30].

3.4.3.2 Crowding Distance

The following is used to compute the crowding distance of each point in the set F [30].

Step 1: Call the number of solutions in F as $l = |F|$. For each i in the set, first assign $d_i = 0$ [30].

Step 2: For each objective function $m = 1, 2, \dots, M$, sort the set in worse order of f_m, or find the sorted indices vector: $I^m = sort\ (f_m, >)$ [30].

Step 3: For $m = 1, 2, \dots, M$ assign a large value to the boundary solutions, or $d_{i_1^m} = d_{i_l^m} = \infty$ and for all other solutions $j = 2\ to\ (l-1)$, assign [30]:

$$d_{I_j^m} = d_{I_j^m} + \frac{f_m^{(I_{j+1}^m)} - f_m^{(I_{j-1}^m)}}{f_m^{max} - f_m^{min}} \tag{3-3}$$

The index I_j denotes the solution index of the jth member in the sorted list. This metric denotes half the perimeter of the enclosing cuboid with the nearest neighbour solutions placed on the vertices of the cuboid, this is illustrated in Figure 3.15 [30].

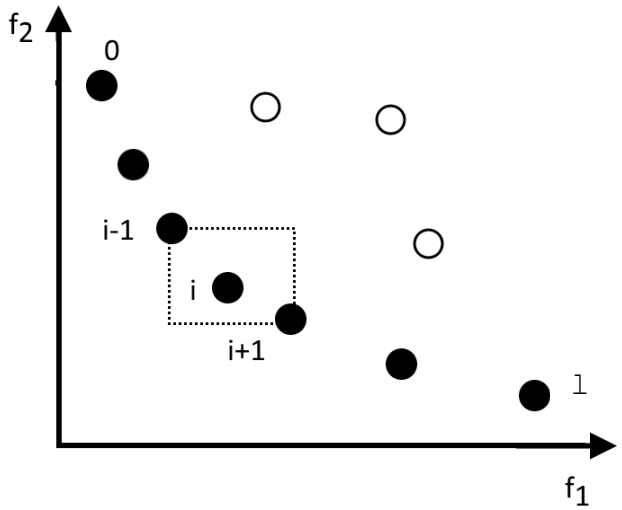

Figure 3.15 Crowding Distance - Enclosing Cuboid [30]

3.4.4 Implementing NSGA-II

The flow for NSGA-II is shown in Figure 3.16

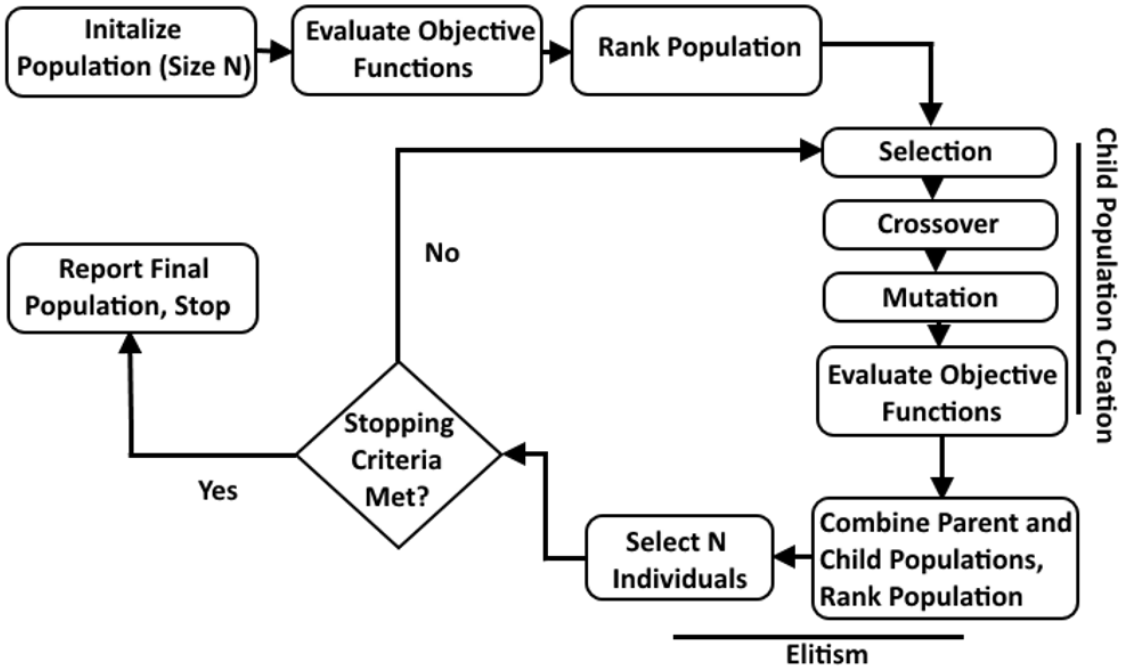

Figure 3.16 NSGA-II Flow Diagram

Pseudocode for implementing NSGA-II is as follows [68]:

```
For Each p ∈ P {
     Sp = ∅
     np = 0
     For each q ∈ P
          If (p ≺ q) then                    if p dominates q
               Sp = Sp ∪ {q}                 Add q to the set dominated by p
          Else if (q ≺ p) then
               np = np + 1                   Increment the domination count
     if np = 0 then                          p belongs to the first front
          prank = 1
          F1 = F1 ∪ {p}
i = 1                                  initialise the front counter
while Fi ≠ ∅
     Q = ∅                             used to store the members of the next front
     For each p ∈ Fi
          For each q ∈ Sp
               nq = nq − 1
          if nq = 0 then                       q belongs to the next front
               qrank = i + 1
               Q = Q ∪ {q}
i = i + 1
Fi = Q
```

Figure 3.17: Pseudocode for NSGA-II [68]

3.5 Many-Objective Problems

Real-world optimisation problems often contain several conflicting objectives that are simultaneously optimised. Problems with more than three objectives are defined as many-objective optimisation problems (MaOP) [69] [70]. The definition was first conceived by Farina et al. [71]

Pareto's definition captures the notion of "optimality" in a narrowly prescribed sense. In fact, the definition is relevant and useful for engineering and design problems, where typically the objective number is small, and the computational cost of each objective is high but is less suitable for many other kinds of problems (especially decision-making problems) where the number of objectives may be big (though computationally costless). Let us consider, for example, a minimisation problem with 50 objectives, $f_1, \ldots, f_{50}$ (a number which is unusual for engineering problems, but common for many real-world decision-making problems), and two points V_1 and V_2 such that in 49 objectives V_1 is better than V_2, and in just one objective j it holds $f_j(V_2) < f_j(V_1)$ (maybe for a small value ε) V_2 is better than V_1. It is obvious to any person would vote V_1 as a better solution than V_2. However, by Pareto definition, they are absolutely equivalent. [71]

3.6 Hypervolume

Once a MOGA produces a Pareto front we can measure the hypervolume of the shape [72], where the shape is produced by the Pareto solutions and reference points. The hypervolume for 2-dimensional problems would be more commonly known as area, and for 3-dimensional shapes, volume. A reference point (r in figure 3.18), represent the worst possible solution in the search space and acts as an anchor point to measure pareto fronts against. The reference point is the maximum value for all minimisation objectives and the minimum value for all maximisation objectives. Figure 3.18 illustrates the hypervolume for a 2-dimensional and 3-

dimensional problem. A hypervolume value gives us the quality of the Pareto front. The larger the hypervolume the better the quality of the Pareto front. This means for experiments, we can measure any improvements to a multi-objective algorithm by measuring the subsequent increase in the hypervolume after changes to the algorithm have been made.

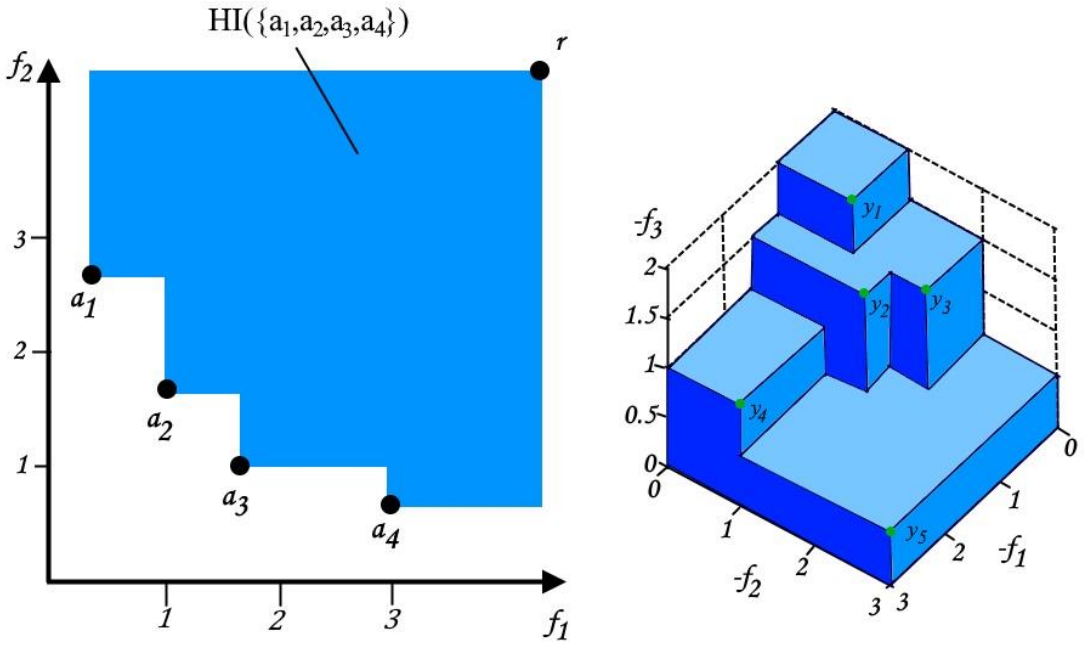

Figure 3.18: Hypervolume Indicator in two dimensions for a set A = {a1, ..., a4} ⊂ R 2 (left) and in three dimensions for a set Y = {y1, . . . , y5} ⊂ R 3 (right) [73]

3.7 Discussion

This chapter gave an overview of genetic algorithms (GAs), Particle Swarm Optimisation (PSO), simulated annealing and looked at multi-objective evolutionary algorithms.

The chapter when on to explain the issues with traditional multi-objective algorithms when attempting to solve many-objective problems (problems with four or more objectives). A brief overview on hypervolume was given, as a way to measure the quality of results from multi-objective algorithms.

The next chapter will give an overview of large-scale organisational design problems. These problems will be tackled by the systems in later chapters.

Chapter 4. An Overview of Large-Scale Organisational Design Problems

For large organisations, such as utility companies, an effective and responsive organisational design can limit the inefficiencies and reduce the impact unexpected events can have on the organisation. In Chapter 2 we disused how tactical planning and scheduling systems handle the allocation of tasks to resources on a weekly or daily basis. However, the effectiveness of these systems often relies on the organisation be set up as best as possible, else the organisation will hit a productivity ceiling, regardless of if there are enough resources to handle the overall demand. Organisational design is often overlooked when it comes to optimisation in large organisations; this is what we will be addressing in the next few chapters.

The problems being addressed are real-world problems that BT decided to address to improve their levels of productivity. Levels which could not be achieved by their current planning and scheduling tools. BT is responsible for much of the United Kingdom's communications network infrastructure and provides services such as telephone, television and internet services to households and businesses.

4.1 The Geographical Structure Optimisation Problem

As BT is primarily a utility company, their responsibilities include maintaining the communications infrastructure that extends across the UK; it also includes providing new communications infrastructure to new properties and upgrading the infrastructure as new technologies become available. To manage these complex responsibilities across such a large and diverse geography requires a management hierarchy based on geographical regions. This hierarchy is key to allow decisions made by executives at the top levels to flow down effectively to each part of the country. The deeper the level of the hierarchy the more increased level of specialist knowledge there is about people, geography and inventory. This type of

management hierarchy is common among many organisations, most famously the military [74]. See Figure 4.1 for how the UK might be divided up.

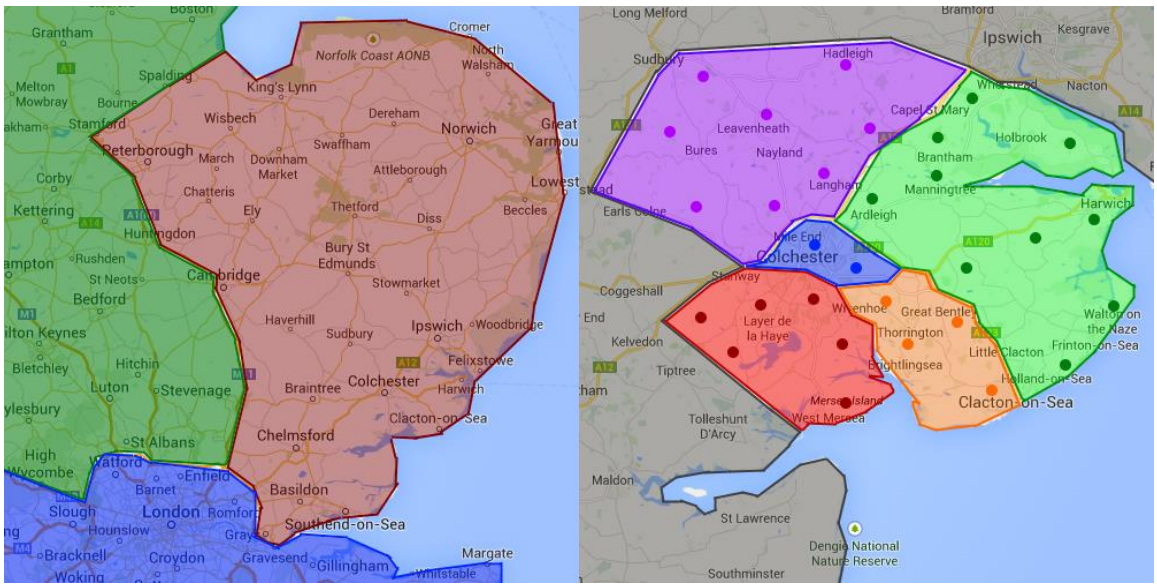

Figure 4.1: Possible Divisions of UK Geography

The organisational design problem that emerges from this are the decisions about what geographical regions should make up the lowest level of the hierarchy, which of these regions should be grouped to form the level above and so on. For BT we can group local geographical zones together based on how the infrastructure network is set up. Each property will be connected to a local Service Delivery Point (SDP), which is a building containing racks of network equipment to allow those properties to connect to the global network. Within BT there are over 5,500 SDPs across the country. The distribution trends towards population density, i.e. the more people there are, the more SDPs are needed to service them. This means more SDPs are required in urban areas than in rural areas.

Clustering together SDPs forms the patches a team of engineers will be responsible for. However, when it comes to real-world geography, there are some restrictions, requirements and special cases that need to be taken into account. For example, patches should not cross large rivers or other geographical obstacles.

Additionally, these constraints on the patches extend from the engineers in each of the teams. Examples of these constraints include; all the engineers will not all be working at all times (as some of them might fall sick, have holidays or day off), so there is a degree of workforce shrinkage that needs to be taken into account. Of the engineers that remain, they can only be assigned tasks that they are qualified to complete. Of these tasks, each engineer has preferred tasks that they work on. Taking this into account can help improve the average time taken to complete the tasks. Lastly, each engineer is limited by the amount of work they can do each day (travel time must be included in this), and each team has to be of equal size.

4.1.1 Objectives

For the particular geographical workforce optimisation strategy being tackled in this thesis, I have five potential objectives. If all of five objectives are used, it qualifies the optimisation problem as many-objective (discussed in Chapter 3.5). The objectives for the workforce optimisation process are as follows:

- *Maximise Coverage*: This is the basic measure of work that is expected to be completed by the engineers. This is measured as a percentage of total completed work. Equation (4-1) represents the sum of all engineers (n) expected completed work over the region's total work (R_{TW}) where the region contains all the patches being optimised. An individual engineer's coverage is represented by C_i while the total completed work is represented by C.

$$C = \frac{1}{R_{TW}} \sum_{i=1}^{n} C_i \tag{4-1}$$

- *Minimise Travel*: Minimising the amount an engineer travels increases the available productive time for each engineer. Reducing travel also reduces costs, due to less fuel consumption. Minimising travel conflicts with maximising coverage as an engineer will usually be required to travel to each task. As coverage increases, travel also

increases. In Equation (4-2), this is represented as the sum of all engineers' travel distance divided by the total number of engineers (n) representing travel as an average for the workforce. An individual engineer's travel is represented by T_i while the total travel is represented by T

$$T = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{4-2}$$

- *Maximise Utilisation*: Unutilised time is when the engineer is idle or travelling, and hence we want to maximise the utilisation of the workforce. Equation (4-3) shows the sum of each engineer completed work (C_i) divided by the engineer's available time (A_i), this sum is then divided by the total number of engineers (n).

$$U = \frac{1}{n}\sum_{i=1}^{n} \frac{C_i}{A_i} \tag{4-3}$$

- *Minimise Area Imbalance*: Patches should have an even distribution of demand. This will lead to smaller patches for urban areas, and larger patches for rural areas. This conflicts with minimising travel (and maximising utilisation) because the larger rural areas favoured by this objective will increase the distance a particular engineer has to travel in the rural area. Area balancing is the difference between the largest (P_{max}) and smallest (P_{min}) patches in hours of available work, shown in Equation (4-4).

$$AB = (P_{MAX} - P_{MIN}) \tag{4-4}$$

- *Minimise Team Imbalance*: Patches should have evenly balanced teams. This will conflict with the maximising the utilisation objective because having balanced teams a) doesn't mean the work is balanced in the patches and b) the number of engineers does not reflect the skill mix, work patterns and capabilities of the team. This objective is a management and human resource constraint. Having one team of 5 engineers and another of 35 engineers is not practical to manage, the manager of the larger team will be envious of the manager with the smaller team, and the smaller

team is less adaptable to spikes in demand. Team Balance is the difference between the largest (T_{max}) and smallest (T_{min}) teams, shown in Equation (4-5), and measured in the number of people.

$$TB = (T_{MAX} - T_{MIN}) \tag{4-5}$$

4.1.2 Complexity of the Problem

For any geographical area being optimised, the complexity of the optimisation can vary dramatically. This complexity is based on the number of SDPs in the area and the number of patches being optimised. The search space size can be calculated using the equation (4-6) where S is the number of SDPs in the area to be optimised, and P is the number of patches in the area.

$$S^P \tag{4-6}$$

Using 5-6, we can see for an average sized area of 109 SDPs with 150 engineers, 7 patches would be required. This results in a search space of $1.82e^{14}$ states. If it takes 0.1 seconds to run the simulation to evaluate each state, it would take almost 580,000 years to check every possible solution. The area with the most required patches, 13, has 106 SDPs. This would result in $2.13e^{26}$ states and would take 6.76 e^{17} years to search exhaustively.

4.2 The Resource Optimisation Problem

For any company with a large multi-skilled workforce, management of skills and teams poses many challenges. A multi-skilled workforce here is defined as one in which the members of the workforce are trained in multiple skills, allowing them to complete different types of tasks. The benefit is that a multi-skilled workforce is capable of completing a range of different tasks, with the aim of making the workforce more productive, more flexible to the changing demand and better at meeting customer needs [75]. This is part of the core principles of workforce optimisation and workforce management, which is about assigning the right employees with the right skills, to the right job, at the right time [1].

Additional arguments have been made for a multi-skilled workforce, such as employees with multiple skills are useful when demand is high, and the company wants to maintain a high level of customer satisfaction [76] [77]. Additionally, a multi-skilled workforce can help where the labour market is scarce of the types of people that are needed [78] [79]. Also, to get the most productivity out of a multi-skilled engineer, the skills they should be trained in should be correlated in some way [78].

The effect of the different mixture of skills in the workforce can have an impact on the utilisation of each member of the workforce and the overall performance of the company as a whole.

A study by the University of Texas in Austin looked at the effects of a multi-skilled workforce in the construction industry [79]. By conducting interviews with many large construction companies, they were able to evaluate the best practice for multi-skilling on large construction projects (where more than 200 workers are needed). They found that if all the workforce is multi-skilled, then there are no specialists, meaning more complex tasks take longer. If there are not enough multi-skilled engineers then there will be a significant increase in the hires and fires with the changing demand as the construction project develops, multi-skilling reduces this.

It is also mentioned that, as a result of a multi-skilled workforce, previous studies have shown a 5-20% reduction in labour costs and a 35% reduction in the required workforce. Similarly, we are investigating the most optimal configurations of skills to get the maximum benefit from the multi-skilled workforce, to further increase the reduction in operating costs.

Deciding which members of the workforce will produce the most benefit when they are trained with more skills can depend on various factors, such as the location of the engineer, the type

of tasks that are near to them and also the career pathway of the engineer to determine at which stage he is in terms of progression.

However, when engineers are trained with more skills, other engineers in the same area will have their utilisation impacted. This may be because an engineer has low-level skills that other engineers could train for and then pick up the work that engineer was doing. As a result, it may be more beneficial to move the low-skilled engineer to a neighbouring team, which is low on resources, and could benefit from the lower skilled engineer freeing up time from the higher skilled engineers in that team.

Due to these complex interactions, it may be more beneficial to evaluate the resultant effects of upskilling engineers at the time the selection of these engineers is evaluated. Upskilling is the process of training a resource and adding to their skillset. A multi-skilled workforce comes with the mentioned benefits, but there is little work in the optimisation of the workforce skill sets.

4.2.1 Multi-Skilled Engineers

Engineers could have varying numbers of skills based on the types of tasks they work on and how experienced they are. More experienced engineers are more likely to have more skills and more likely to have more advanced skills.

The skill sets of the engineers will differ between the different geographical areas that the groups of engineers (teams) are assigned to. So, a team with a given number of engineers and an optimised set of skills for each engineer will not necessarily be the best setup for another area.

In [80], it is noted that the skill optimisation problem is a combinatorial optimisation problem. The optimisation would need to be frequently run due to changes in customer demand and

churn of engineers. A common approach to tackling these large scale and complex optimisation problems is genetic algorithms, as discussed in Chapter 4, and examples are given in [5], [6] and [7].

4.2.2 Team Organisation Optimisation

Making sure engineers have the most optimal skill sets is just one part of the problem. This is because any change in the team's abilities can have sub-optimal implications. As such it would be necessary to reorganise the teams after the engineer skill sets have been changed.

An example of one of these implications would be that in an area of low utilisation a few engineers may be selected to train in a specialist skill, so they can pick up more work and hence be more utilised. However, for the engineer that was already a specialist, their work will be reduced. Possibly to a point where the engineer becomes grossly underutilised. As a result, it may be more beneficial to move that engineer into a neighbouring team. Especially if that area's team is near maximum utilisation but low completion of tasks (meaning there are more tasks than there is time available from the engineers). Another sub-optimal outcome form this scenario is that; if the engineers are trained in a specialist skill that is needed for the whole area, but the engineers live in a section of the geography that doesn't need those skills, the scheduling system may still allocate them the nearest work. As the scheduler is tasked with minimising travel. Thus, engineers may be trained in a new skill and rarely use it.

A potential solution here is to move someone in from a neighbouring area that is close to the specialist tasks, in this situation the engineer could already have the specialist skill, or they could then be trained in the specialist skill once they have moved team.

This additional layer of change adds more complexity to the problem, because if the re-organisation of teams happens after the skill optimisation has taken place, then the results will

be sub-optimal. The team re-organisation has to be done during the solution evaluation when engineers are being selected for training.

4.2.3 Objectives

From our list of primary objectives found in Section 4.1.1, the objectives applicable to this problem are the following:

- **Task Coverage (*C*)**: the percentage of the tasks estimated to be completed by the engineers at the end of the simulation. This is a maximisation objective.
- **Travel Distance (*T*)**: the distance in km an engineer, on average, has to travel in the simulated area. This is a minimisation objective.
- **Utilisation (*U*)**: the average utilisation of the engineers. This is a maximisation objective

If a single objective GA is used, the fitness function used can be given in Equations (4-7) and (5-8)

$$\boldsymbol{F} = \frac{(\frac{\boldsymbol{1}}{\boldsymbol{W}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\boldsymbol{C_i})(\frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\frac{\boldsymbol{C_i}}{\boldsymbol{A_i}})}{\frac{\boldsymbol{1}}{\boldsymbol{n}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{n}}\boldsymbol{T_i}} \tag{4-7}$$

$$\boldsymbol{F} = \frac{\boldsymbol{CU}}{\boldsymbol{T}} \tag{4-8}$$

We do not include any weighting factors in the fitness function. Business objectives change on a regular basis, so we will evaluate any solution to our problem with all objectives equal. This will help to determine what solutions to our problem are the best not only overall but in any particular objective.

4.2.4 Complexity of the Problem

The complexity of the resource optimisation problem stems from the uncertainty of the operational impact of any decision. When moving engineers between teams there is a utilisation

trade-off between the two teams that may not be equal. An engineer's skill and start location will be critical factors in how much they contribute to their new team. If the engineer has very little skills required by the new team, their impact will be limited.

Upskilling is no less trivial, as mentioned the algorithm will trend towards upskilling all engineers, but this is obvious. The reason upskilling is non-trivial is because of the limiting factor. The limiting factor is the maximum number of engineers that should be upskilled. The knock-on effects of upskilling one engineer may be difficult to determine until the simulation has taken place. Thus, it is difficult to know whom to choose for the second upskill and so on. Upskilling two engineers in the same location with the same skill may be very beneficial depending on the type of work, or it may be a waste of a training slot. Typically, any sub-region will have between 100-150 engineers; these subregions will contain 5-7 teams. The total possible operational choices for moving engineers is given in Equation (4-9), where *n* is the number of engineers, and *t* is the number of teams in the sub-region.

$$(n^2)^t \tag{4-9}$$

Meaning for any subregion there could be up to $2.92e^{30}$ possible move choices per region, in one example of this in Chapter 8 there are eight sub-regions. Thus, there can be $2.33e^{31}$ possible move choices for a regional manager. The size of this search space is too large for heuristic search to traverse in a reasonable time. If the simulation returns results within 0.1 seconds (a measure which is entirely dependent on the hardware the algorithm is run on), it will take approximately $7.40e^{22}$ years to arrive at a definitive answer. Hence, we use meta-heuristics, specifically GAs, which are explained in Chapter 3. This search space size does not account for the upskilling options, which add another complex dimension. Every engineer has two states within this optimisation, either their current skill set, or their upskilled skill set. The limiting factor states the maximum number of engineers that should be upskilled in the set of

total engineers. Given that the engineers have a binary state, we can calculate the total number of upskilling options (U) for a set of engineers (n) and a limiting factor (x) using (4-10)

$$U = \frac{n!}{x!((n-x)!)} \tag{4-10}$$

However, this will calculate the number of possible upskill options for the maximum number. Thus, we need to sum up all options from 0 to the limiting factor to obtain the true total of possible options. So, we can derive (4-11):

$$U = \sum_{m=0}^{x} \frac{n!}{m!((n-m)!)} \tag{4-11}$$

Hence, if we have 150 engineers and we set the limiting factor to 10 (i.e. we have a maximum budget for 10 training slots), we would have 1.26e15 upskilling combination options. If we apply the same computation time of 0.1s per simulation, this will result in a total computation time of $3.99e^{6}$ years to arrive at a definitive answer. Thus, if we try to compute the most optimal move and upskilling combination simultaneously it would take $2.89e^{92}$ years ($7.40e22^{3.99e6}$) or $2.09e^{81}$ lifetimes of the universe.

4.3 The Suitability of Simulated Annealing

As discussed in section 4.1.1 and 4.2.4, the search spaces for problems associated with large-scale organisational design are potentially infinitely vast. To add to the complexity of solving these problems, there are a number of real-world constraints that irregularly warp the search space.

In a short survey of GAs versus simulated annealing, there seems to be evidence that there are more benefits to using a GA for large optimisation problems.

[81] concludes that simulated annealing is a popular contemporary placement method; however, the results of this study indicate that genetic algorithms may lead to better results.

[82] states that the results show better convergence of shortest length chromosome using GA than simulated annealing.

[83] concludes that their outcomes showed that both of the algorithms are able to tackle the problem. However, the GA could return better results in a shorter computation time.

Finally, [84] concludes, simulated annealing needed longer computation times compared to the genetic algorithm.

As a result of this survey and the size and complexity of the problems, simulated annealing solutions will not be developed to reduce the scope of this work.

4.4 Discussion

This chapter gave an introduction to the specific large-scale organisational design problems that will be tackled. It then explained the two distinct domains, geographical structure optimisation problem and the resource optimisation problem. The objectives for these problems were detailed, which are the following

- Coverage: How much work is completed
- Travel: How much the engineers travel
- Utilisation: How utilised is the workforce (are they idol or travelling a lot?)
- Area balancing: also known as patch balancing, the measure of how balanced each of the areas is.
- Team balancing: a measure of how equally balanced the teams are regarding Full-Time Employment (FTE)

The first three objectives are applicable to both problem areas; the last two objectives are only applicable to the geographical optimisation problem.

The computational complexities of the optimisation problems were detailed, as a result it is reasonable to try and solve these problems with meta-heuristics, such as those described in Chapter 3.

The next chapter will give an overview of the type-2 fuzzy logic system for field workforce optimisation.

Chapter 5. The Genetic Type-2 Fuzzy Logic System for Field Workforce Optimisation

Figure 5.1 provides an overview of the framework of the multi-objective genetic type-2 fuzzy logic-based system for mobile field workforce area optimisation.

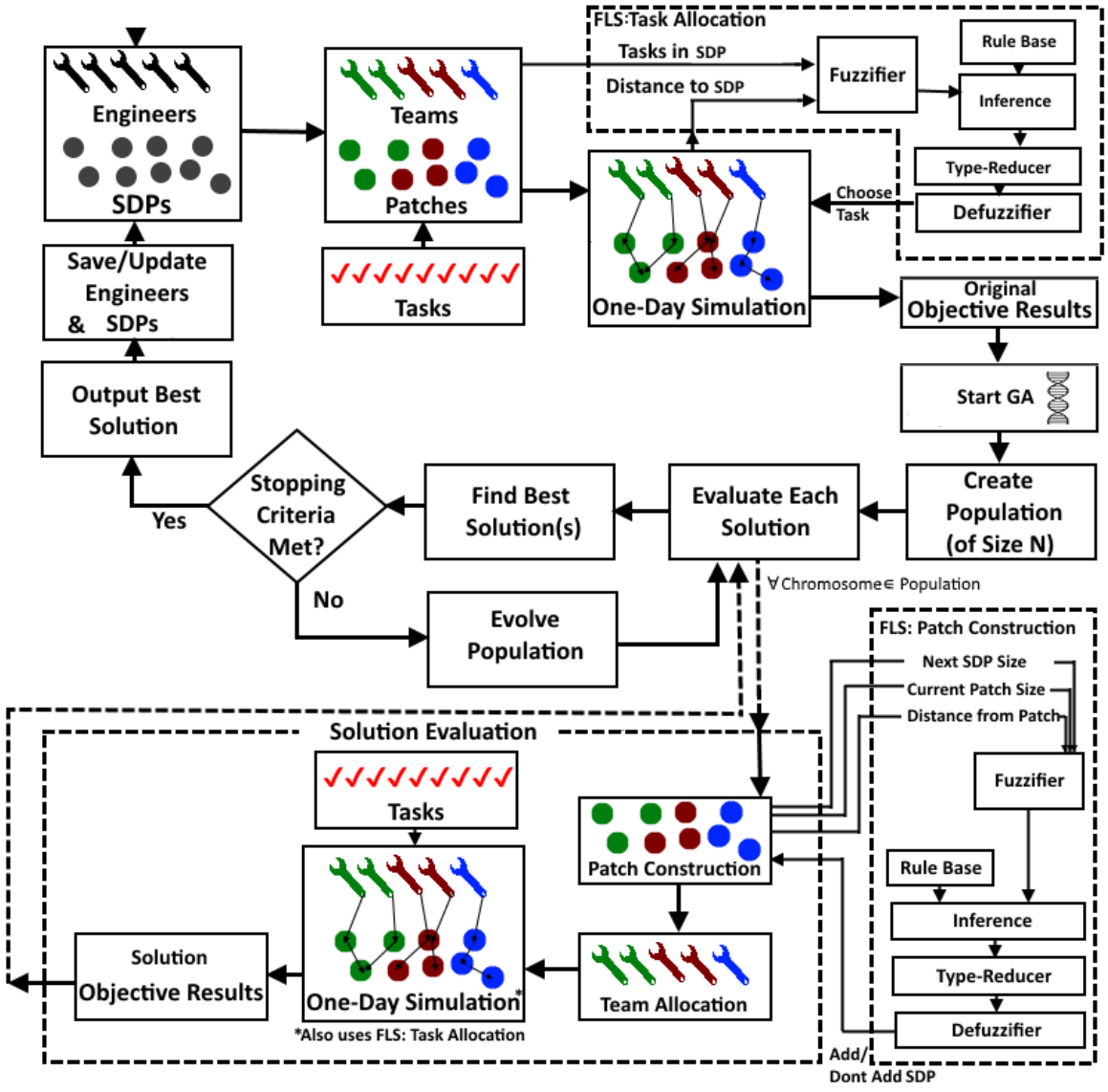

Figure 5.1: The multi-objective Genetic Type-2 Fuzzy Logic Based System for Mobile Field Workforce Area Optimisation

The first step in this system is to collect the list of engineers and the list of SDPs to optimise. The engineers and SDPs will already be grouped together into teams and patches from their

current set-up in the real-world environment, so the system organises the entities into the groupings from the data presented.

The system now has the current setup of patches with their respective teams. This configuration is then put through the one-day simulation to assess how the current setup is performing. The one-day simulation cycles through each engineer and assigns them tasks, based on their skills and the patch they are in. The simulation will attempt to assign the closest relevant tasks to the engineer but also considers task density, as it would be less efficient to send an engineer to a location far away if there is only an hour's worth of work there. Once a task has been assigned, it will be removed from the available task list.

Each engineer will be assigned tasks until their time has been filled or there are no more tasks available for that engineer. Each engineer is allocated 7 hours, and each task has an estimated completion time attached to it. When an engineer is assigned to a task the time will be added to their utilised time, while the time it takes to travel to the task will be added to the engineer's travel time (part of the engineers unutilised time). The distance travelled per task is also stored for each engineer. The simulation will stop assigning tasks once the utilised time combined with the travel time is over 7 hours. Any remaining time an engineer has will be idle time, which is part of the engineer's unutilised time.

The one-day simulation step is where the Task Allocation Fuzzy Logic System (TAFLS) can be applied. When choosing which task to assign to an engineer the distance and time to the task is fuzzified. The number of tasks at the SDP is calculated and fuzzified, due to the uncertainty around the completion time and the number of tasks on any given day. This helps the simulation take into account the uncertainty of the travel time and to direct the engineer to SDPs that will better reflect real-world conditions. More on this can be found in Section 5.1.

Once each engineer has been cycled through, the system will calculate the objective results. The objectives calculated within the simulation are coverage, travel and utilisation. The area balancing objective is a trivial calculation, but an important one. For this stage of the research, the Team Balancing objective was not used. The details of these objectives were discussed in Section 5.1.1.

Given that the current live organisational structure is available and can be evaluated using the proposed simulation, the values generated from the current design can be used as a simple benchmark for the optimisation process to improve upon. The system gives the user the option to adjust any of the GA's parameters (crossover rate, mutation rate, population size, number of generations and elite solutions) before the optimisation process is started.

When the GA begins, it will create a new population of solutions. Each member of the population has P genes, where P is the number of patches to optimise for. Each gene is the centre location of a patch, and the rest of the patch will be constructed from these points.

As each solution needs to be evaluated, the first step to this is building the patch setup from the centre points. Certain restrictions apply to the patch construction. SDPs in the same patch cannot be separated by rivers or by other patches, as described in Section 5.1. A high-level description of how the patch construction works is given in the following: Each centre point (extracted from the genes in the solution's chromosome) works out who its neighbouring SDPs are. Then out of these neighbours, works out which is the closest. If no other patch has deemed that SDP to be the closest it will be added to the patch. The next patch will do the same. Each time an SDP is removed from the list and added to a patch, each patch has to recalculate who its available neighbours are.

The patch construction is where the Patch Construction Fuzzy Logic System (PCFLS) can be applied. This will allow the system to account for the uncertainty in travel times across the

patches. When it is being decided if an SDP should be added to a patch, the list of all neighbouring SDPs will be passed through the FLS whose inputs are the size of the SDP (in hours), the size of the patch (in hours) and the distance to the SDP from the centre point. More on this can be found in section 5.2.

Once the patches have been constructed from the centre points, the teams for each patch need to be assigned. This first step in this process is to assign each engineer to the patch they live in (or are closest to, if they do not live in any patch). This will usually mean the teams are extremely unbalanced as city/town patches will have overpopulated teams and rural patches will have underpopulated teams.

So, the next step is to balance out the teams. This is done by a bidding process. The system will cycle through each overpopulated patch and 'sell off' its engineers to the highest bidders. Each underpopulated patch will cycle through the current overpopulated patch's engineers and give each a bid value. If there are no other bids for this engineer, they will move over to the underpopulated patch and if there are other bids the highest bid wins. The bid value is made up of the distance the engineer is from the underpopulated patch, how much their skills are needed and the level of under-population the patch is at. Once the bidding process is complete, the engineers should be spread as best as possible between the patches.

The newly constructed patches and teams will then go through the same one-day simulation process just as the original setup did (also using the TAFLS if specified) if the generated solution is valid. There are certain criteria that if not met the solution will be rejected or altered before the one-day simulation is run on it. This includes the number of patches constructed. As the user specifies the number of patches and each gene represents a patch centre, any solution cannot have two genes that represent the same centre point. Also, all SDPs must be added to the patch design, so the list of unassigned SDPs has to be empty before the simulation can be

run. If there are any SDPs on the list, they will be assigned the same patch as their closest neighbour.

Once the solution has passed the checks and is deemed valid, the objective values for this solution will be calculated. The GA will carry out the 'Solution Evaluation' for every solution it generates. More about how the single objective and multi-objective algorithms affect the optimisation can be found in section 5.3.

With each solution in the population evaluated, regular GA processes are resumed. The stopping criterion that is currently being used in the system is the maximum number of generations reached. Once the GA has stopped the results are reported, and output files can be generated. The output files list each engineer and their newly assigned patch and the structure of these new patches.

5.1 Task Allocation Fuzzy Logic System

The closer the one-simulation is to replicating real life, the better the result of the optimisation will transfer into the real world. One key part of the simulation is how an engineer is allocated tasks. The simplest form of this is to allocate the nearest available task the engineer is capable of doing. The more complex and detailed solution to this is to implement a version of the organisation's full scheduling system.

The simplest form is not representative of real decision-making. Indeed, choosing the tasks with the smallest travel distance may actually increase overall travel. For example, the closest tasks to an engineer may only be small tasks that take a total of one hour at that location. The engineer will have to travel to the next location only after this short time. If the next closest task at this point is also only another hour in duration the engineer will spend a lot longer travelling, and less time completing tasks. This is compared to the engineer choosing a location with four or five hours' worth of work but is a further away.

The complex form of allocating tasks to engineers is far too computationally expensive for a population-based evolutionary algorithm. Implementing such a complex scheduling system to the solution evaluation stage would increase the run time of the algorithm to a point where it is not practical to use on a daily basis.

The middle ground of these two scenarios is the proposed Task Allocation Fuzzy Logic System (TAFLS). This is because the fuzzy logic can handle the uncertainty about the quantity and complexity of tasks, in a genralised way, whilst not having to know the exact tasks that might appear on any given day over the next few months. By using fuzzy decision-making system designed by an expert, the computational time of these decisions is relatively cheap and lead to more realistic task allocation decisions that a crisp system.

The TAFLS is compatible with both Type-1 and Type-2 FLS, and the below describes the more complex Type-2 version. The type-1 TAFLS can be inferred from the membership functions with 0% uncertainty in these sets (thus there is no footprint of uncertainty and $\underline{f}^M = \overline{f}^M$ or $Y_{TR} = y_l = y_r$).

Figure 5.2, Figure 5.3 and Figure 5.4 show the interval type-2 fuzzy sets used to decide which tasks to allocate to the engineer. The average distance to a task (AD in Figure 5.2) is calculated for the area being optimised and is done before the initial one-day simulation when the teams and SDPs are first loaded into the system. The average amount of work in an SDP for the area (AW in Figure 5.3) is also calculated at this point. Figure 5.4 shows the output of the interval type-2 FLS which represent the probability of picking a task. This interval type-2 FLS uses the Centre of Sets type-reduction as it has reasonable computational complexity.

The footprints of uncertainty, shown in Figure 5.2, Figure 5.3 and Figure 5.4 as the grey areas, is variable. The uncertainty value is given to the system as an input, and the footprint extends each side of the base point by the required percentage. The base points of the membership

functions were tuned by running experiments to find the most suitable setup, with a human expert.

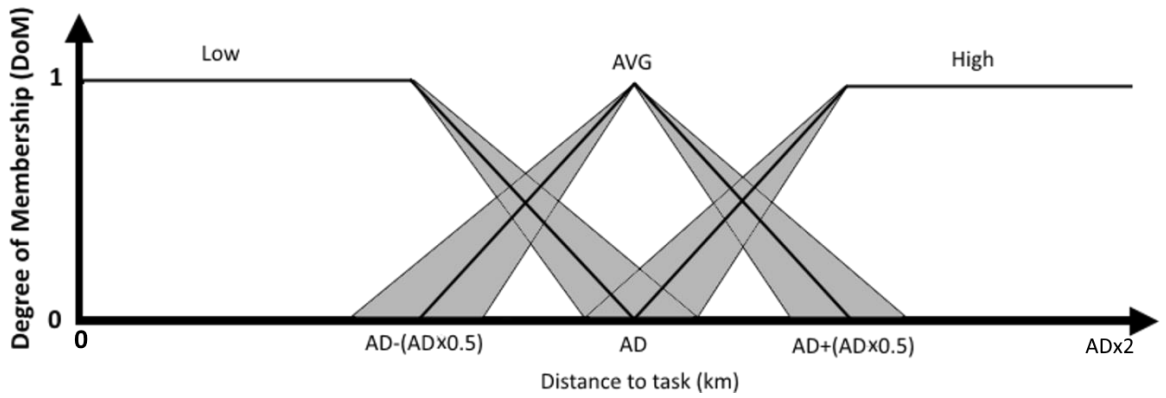

Figure 5.2: 'Distance to Task' Type-2 Fuzzy Sets

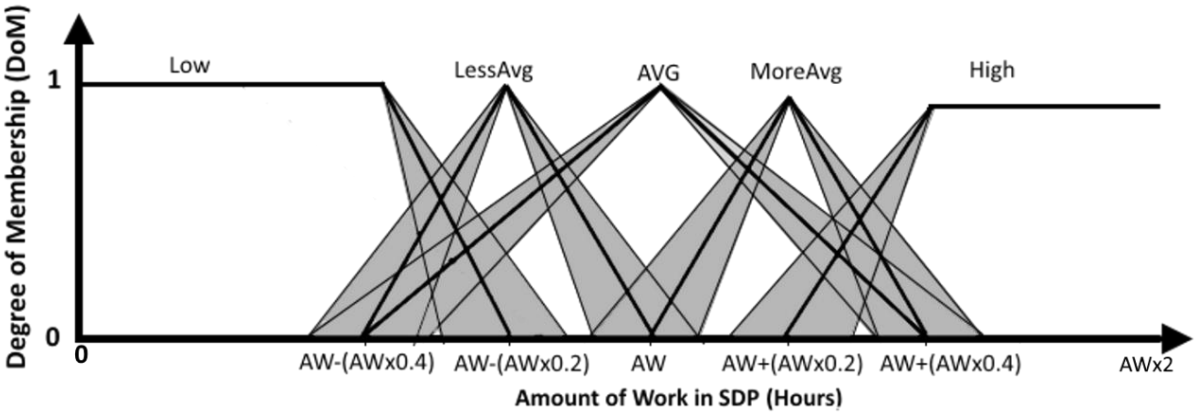

Figure 5.3: Jobs in SDP Type-2 Fuzzy Sets

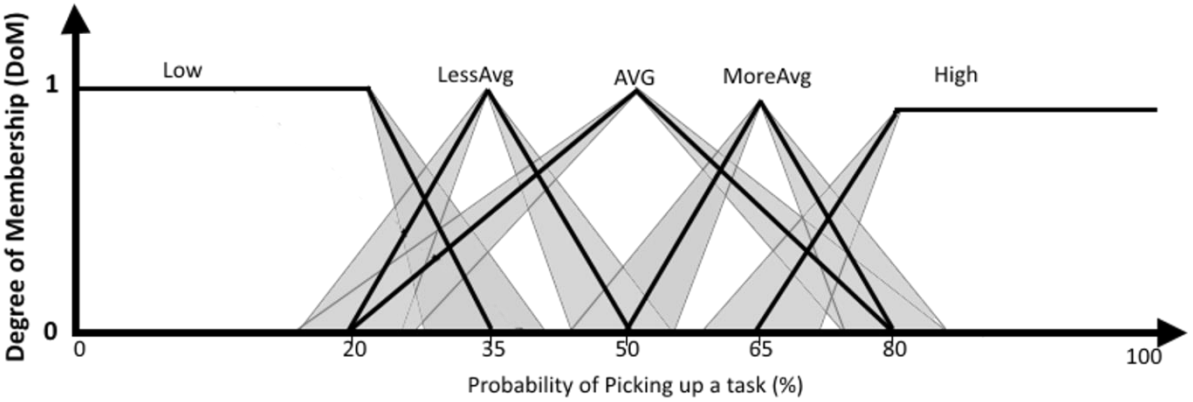

Figure 5.4: Probability of Picking Task Type-2 Fuzzy Sets

The values for the average distance (AD) and average work (AW) had to be calculated so that their values were relative to the area that was being optimised. For example, an average

distance per job in London might be 100 meters, but in the Scottish Highlands, this value might be 5km or more. Having the base points relative to the area is important, else input values will be wrongly categorised relative to the local area. The reason for the triangular and trapezoid membership functions is that they easy to explain to the non-technical experts. In addition, due to the need to generate the membership functions dynamically, it is faster to use the triangular and trapezoid membership functions generated from calculated base points and scale them accordingly. Table 5-1 shows the list of rules used in this FLS.

Distance to Task	Tasks at SDP	Probability of Choosing SDP
Low	Low	Average
Low	LessAvg	Average
Low	Average	MoreAvg
Low	MoreAvg	High
Low	High	High
Average	Low	LessAvg
Average	LessAvg	LessAvg
Average	Average	Average
Average	MoreAvg	MoreAvg
Average	High	MoreAvg
High	Low	Low
High	LessAvg	Low
High	Average	Low
High	MoreAvg	LesAvg
High	High	Average

Table 5-1 Task Allocation Rule Base

The following is an example of how this fuzzy system might work:

The system wants to find the next best SDP to send an engineer to, so the system finds out that the average amount of work in all SDPs in the patch is five hours. The average distance to a task is calculated to be two kilometres. The current engineer has three SDPs to choose to go to next. The first is three kilometres away with five hours' worth of work. The second is one kilometre away with six hours' worth of work, and the third is two kilometres away with eight hours' worth of work.

Given these options, the fuzzy system would classify the first option as *High* distance and *Average* amount of work giving a *Low* probability of choosing that SDP. The second option would be classified as *Low* distance and *More than Average* amount of work giving a *High* probability of choosing the SDP. The third option would be classified as *Average Distance* and *High* amount of work giving a *More than Average* probability of being chosen. With these three results, their output defuzzified values are compared, which would give option two the highest value and this SDP would then be assigned to the current engineer.

5.2 Patch Construction Fuzzy Logic System

A key aspect to generating a solution from the chromosomes in the GA is the geographical design of the patches. This is a critical part of the evaluation process, as without the geographical structure no engineers can be allocated, and no work can be completed.

The way patches are constructed from the chromosomes is to map each gene in a solution to an SDP based on the genes value. Each member of the population has P genes, where P is the number of patches to optimise for. If we represent the total number of SDPs as ST, $P <= ST$. From a practical perspective it is always the case that $P < ST$, this means there are SDPs that are not assigned to any patch. The process of assigning the remaining SDPs to the patches is via a neighbourhood based clustering technique.

5.2.1 Neighbourhood Clustering For Patch Construction

Given that there are strict requirements for how patches for can be designed the way the SDPs are clustered together has to be intelligent.

Each gene in a solution represents an SDP to act as a centre point to each cluster. The clustering process is illustrated in Figure 5.5. Figure 5.5a shows three SDPs selected as the centre points. Figure 5.5b shows the immediate neighbours being added; Figure 5.5c shows the next few layers of SDPs being added. Finally, Figure 5.5d shows the final design created from the three SDPs selected by the GA in Figure 5.5a. If an SDP neighbours more than one cluster, we use a decision system to decide which cluster that SDP should be added to. In its most basic form, this decision is based on the closest distance.

This form of neighbourhood-based clustering ensures that all SDPs will be added if the geographical region is connected from one side to the other. This method also ensures patches are not split into more than one continuously connected grouping.

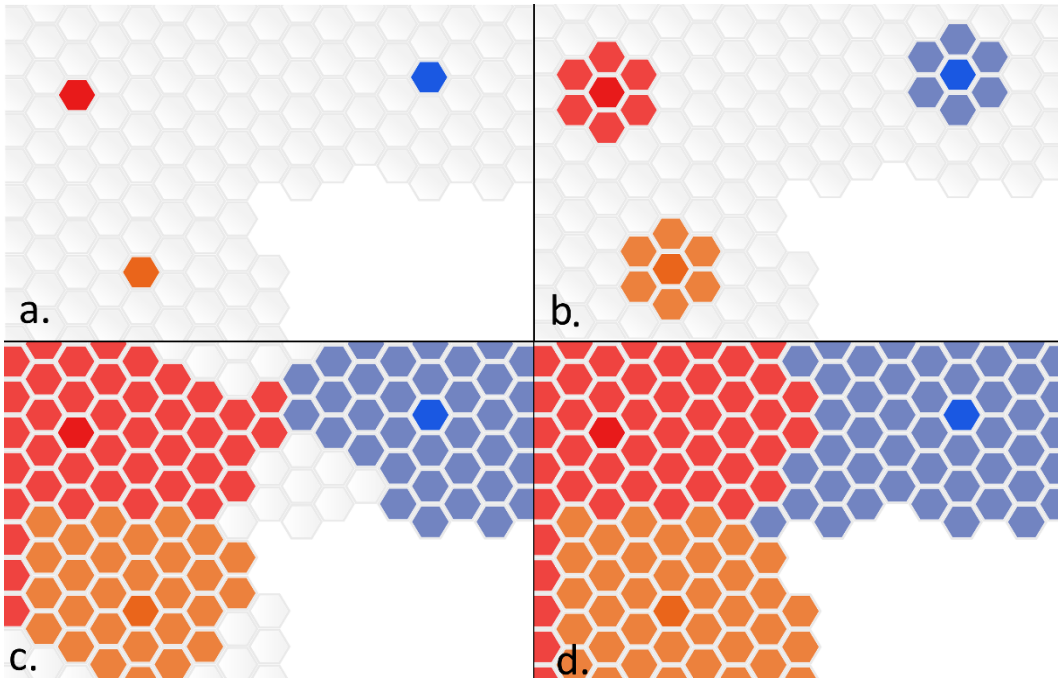

Figure 5.5: An example of the SDPs being clustered by their neighbours.

5.2.2 Fuzzy Neighbourhood Clustering For Patch Construction

When an SDP can be added to more than one patch, the decision-making process of just choosing the nearest patch by travel distance or travel time is weak. Adding any SDP has a significant impact on the whole design. This is particularly true in the early stages; a few key decisions can dramatically change the outcome of the design from the same centre points.

To make the decision-making process more intelligent a fuzzy logic system, known as the Patch Construction Fuzzy Logic System (PCFLS), has been developed. The aim of the PCFLS is to take into account the relative amounts of work within each SDP and within each of the constructed patches. It also aims to fuzzify the travel element so that an SDP does not simply get added to the patch that is 0.1km closer than another (or 2 minutes closer if time is used). This means the design understands there is much uncertainty around travel estimations.

The PCFLS is compatible with both Type-1 and Type-2 FLS, and the below describes the more complex Type-2 version. The type-1 PCFLS can be inferred from the membership functions with 0% uncertainty in these sets (thus there is no footprint of uncertainty and $\underline{f}^M = \overline{f}^M$ or $Y_{TR} = y_l = y_r$), just like the TAFLS.

Figure 5.6, Figure 5.7 and Figure 5.8 show the type-2 fuzzy sets that are used in the PCFLS. When the area to be optimised is initially loaded up, the average patch size in hours of work, Patch Average (PA), is calculated along with the average SDP size (SDPA). This is because these values can vary a lot between urban and rural areas. Hence, for London, the average SDP may carry 500 hours' worth of work, but in the Scottish Highlands, there may only be an average of 20 hours' worth of work, or even less.

The base points of the membership functions were tested to see if reasonable categorisation of SDPs and patch sizes were given. This interval type-2 FLS also uses the Centre of Sets type-reduction, again because it has a reasonable computational complexity.

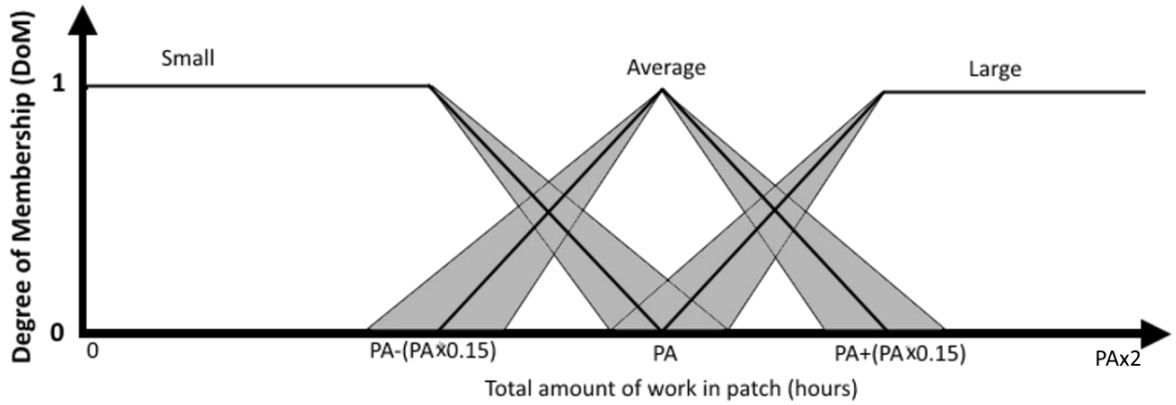

Figure 5.6: Patch Size Average Type-2 Fuzzy Set

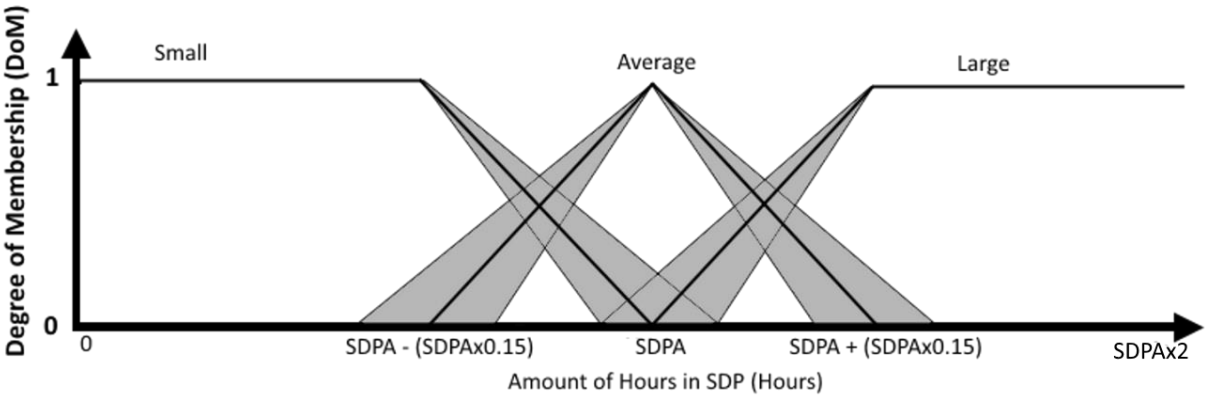

Figure 5.7: SDP Size Average Type-2 Fuzzy Set

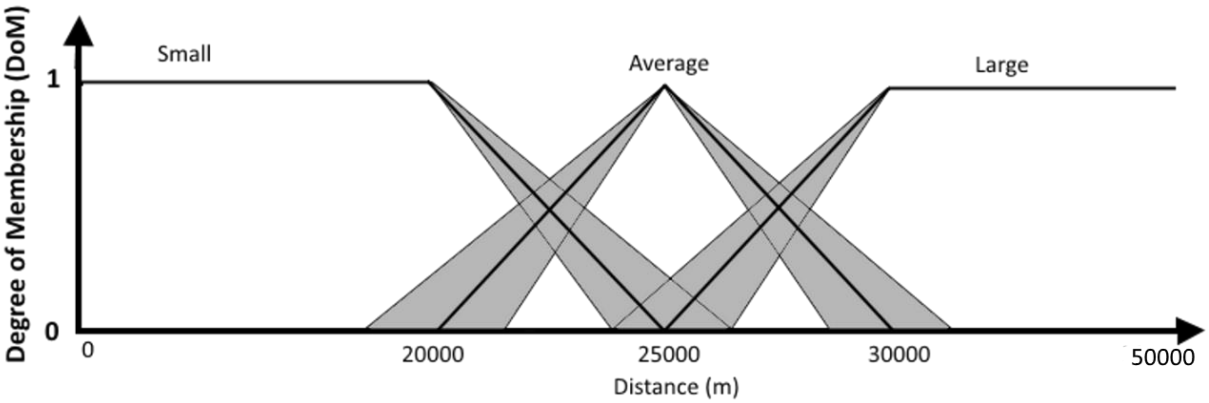

Figure 5.8: Average Distance Type-2 Fuzzy Set

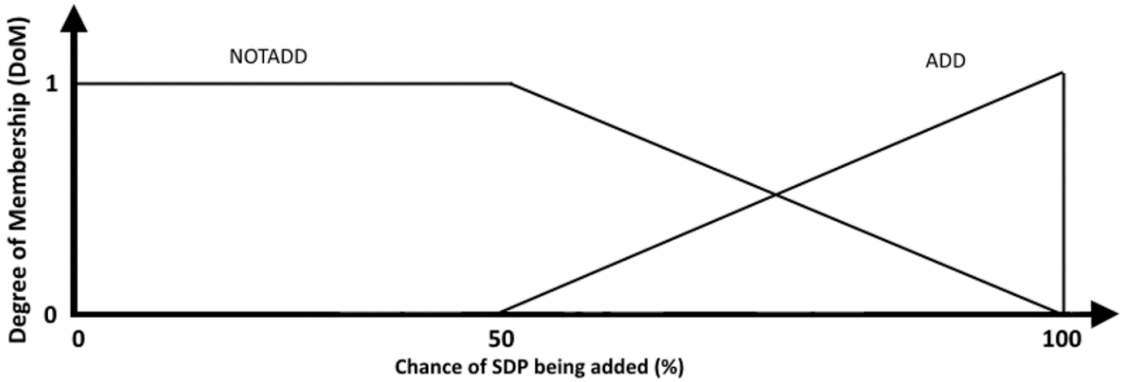

Figure 5.9: Add/Not Add Fuzzy Set.

WA Size	Distance to SDP	SDP Size	Consequence
Small	Small	Small	Add
Small	Small	Average	Add
Small	Small	Large	Add
Small	Average	Small	Add
Small	Average	Average	Add
Small	Average	Large	Add
Small	Large	Small	Add
Small	Large	Average	Add
Small	Large	Large	NotAdd
Average	Small	Small	Add
Average	Small	Average	Add
Average	Small	Large	NotAdd
Average	Average	Small	Add
Average	Average	Average	Add
Average	Average	Large	NotAdd
Average	Large	Small	Add
Average	Large	Average	NotAdd
Average	Large	Large	NotAdd
Large	Small	Small	Add
Large	Small	Average	NotAdd
Large	Small	Large	NotAdd
Large	Average	Small	NotAdd
Large	Average	Average	NotAdd
Large	Average	Large	NotAdd
Large	Large	Small	NotAdd
Large	Large	Average	NotAdd
Large	Large	Large	NotAdd

Table 5-2 Patch Construction Rule Base

The centre points of the patches are provided to the fuzzy system. The size of the patch is re-calculated each time an SDP is added to it. Figure 5.9 shows the type-1 fuzzy sets representing the output of the type-1 FLS which is the chance of an SDP being added.

The Add/NotAdd membership functions were designed in such a way that a rule with a Not Add consequence would have more of an impact on the outcome than an Add consequence. The output values are compared between the patches, with the SDP being added to the patch with the highest output value. Table 5-2 shows the list of rules used in the PCFLS.

The PCFLS will cycle through each patch. Initially, each patch only contains one SDP, but will still have at least one neighbour. There is no guarantee this one neighbour will not be the centre point of another patch; this would result in one of the patches only containing one SDP, which could be a valid solution if that SDP contains a significant amount of work (i.e. a city centre).

When the PCFLS is looking at a patch, it will look at all the neighbouring SDPs to choose the best one to add. The PCFLS finds out that the average amount of work in all SDPs in the area to be designed and computes a score for each SDP. The best one will be added, the available neighbours of this patch will be updated (by removing the SDP that was just added and adding its neighbours that aren't already assigned to another patch). The PCFLS will then cycle to the next patch, only adding one SDP to each patch at a time.

The decision-making process becomes a little more complex when an available SDP borders more than one patch. In this situation, the SDP will only be added to the current patch if it has the highest score from all the neighbouring patches. This means if it does not have the highest score, it is likely that SDP will be added to the better-suited patch when the PCFLS cycles around to that patch.

A short example of the PCFLS might work as follows:

The average amount of work in an SDP for an area is 5 hours. The current patch is deemed to be an Average sized patch based on its current total amount of work. The current patch has three SDPs to choose from to add to itself. The first is 3.0 kilometres away with 5 hours' worth of work. The second is 2.0 kilometres away with 6 hours' worth of work, and the third is 2.5 kilometres away with 2 hours' worth of work.

Given these options, the PCFLS would classify the first option as *Large distance* and *Average amount of work* giving a consequence of suggesting *Not to Add* this SDP to the current patch. The second option would be classified as *Low distance* and *Large amount of work* giving a consequence of suggesting *Not to Add* this SDP to the current patch. The third option would be classified as *Average Distance* and *Small amount of work* giving a consequence of suggesting too *Add* this SDP to the current patch. With these three results, their output defuzzified values are compared which would give option 3 the highest value and this SDP would then be added to the current patch. The PCFLS will then move to the next patch.

It is worth noting that it does not matter how low the score is from the PCFLS, the highest value always wins. This is to ensure that all exchanges are added to a WA, even if that means adding a *Large* SDP to a *Large* patch. Ultimately this will just mean this solution will perform badly in the patch balancing objective, yet it would still be a valid solution as all SDPs would have been added to the design.

5.3 Use of Genetic Algorithms

Both single objective and multi-objective genetic algorithms can be used with the proposed system and the different results that are given by each can be found in section 5.4. If a single objective GA is being used, then the following fitness function (equation 5-1) will be used to assess the solutions.

$$Fitness = \frac{(Coverage \times W_1) \times (Utilisation \times W_2)}{(Travel \times W_3) \times (Balancing \times W_4)} \quad (5\text{-}1)$$

W is the weighting of each objective, w_1 is the weighting of the coverage objective, w_2 is the weighting of the utilisation objective, w_3 is the weighting of the travel objective, and w_4 is the weighting of the balancing objective. Changing these values pushes the optimisation to find solutions that satisfy the objectives with the higher weightings. Any weighting could be set to 0 to remove that objective from the fitness function. If this is done, the objective value combined with the weighting defaults to a value of 1.

If a multi-objective GA is being used, there will be no single fitness value, only each individual objective value. The output will also be a set of solutions (provided there is more than one solution on the Pareto front). This allows managers to pick a setup that is best-suited based on local knowledge that could not be taken into account by the proposed system. This adds an extra layer of validation before any new organisational designs get rolled out to a live environment.

5.4 Initial System Experiments & Results

To test the proposed type-2 fuzzy logic system for field workforce optimisation, the techniques and methods need to be integrated into a user-friendly interactive tool. This allows non-technical users to run the algorithms and get a visual output of the results. An early version of the real-world tool created for this process is shown in Figure 5.10. This version of the tool allows the visualisation of SDPs (the dots) and the patches (coloured areas).

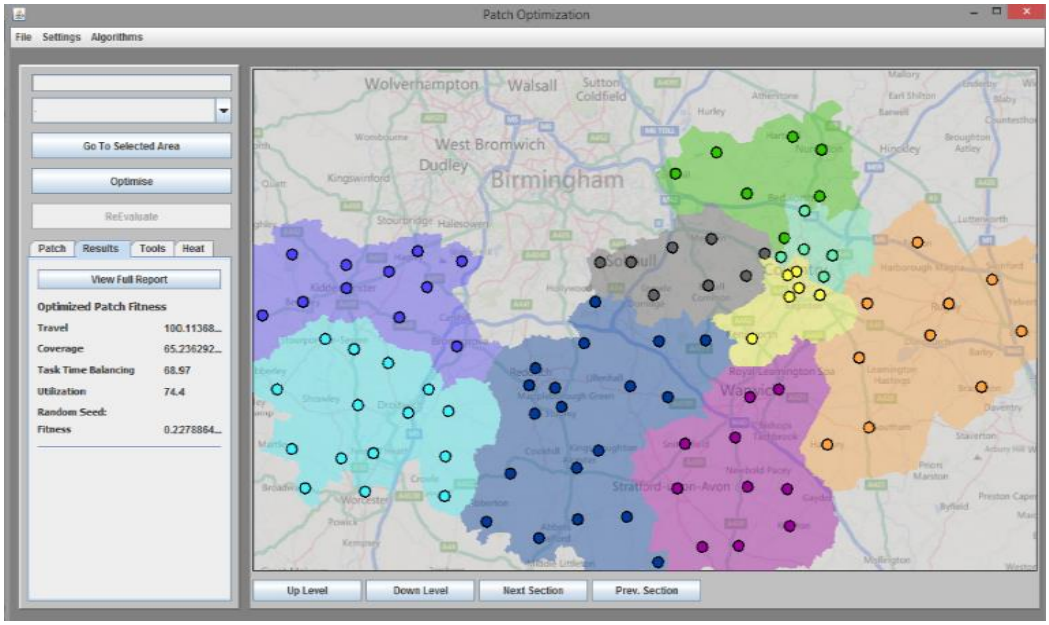

Figure 5.10: Version 1.0 of the Mobile Field Workforce Area Optimisation Tool

Once the initial system had been developed, the underlying algorithms and optimisation methods needed to be tested. This forms the first set of experiments.

These experiments aim to take an existing patch structure in a telecommunications domain, then run it through the optimisation process to see the levels of improvement that can be obtained. These experiments are then repeated with enhancements, to examine the impact these potential changes will make. The experiments involved altering the optimisation process by gradually increasing the use of more advanced optimisation methods.

The process started by comparing the use of single and multi-objective GAs and then progressed to evaluate the effect of employing type-1 and type-2 FLSs.

For all the experiments, both single objective and multi-objective genetic algorithms (GA and MOGA) were set to carry out 20 generations and have a population size of 40. Due to the complexity in generating designs of patches and simulating one-day, the time it took to complete one generation within this version of the tool, was significant enough, at this stage of development, to prevent more generations from being carried out. In addition, because the tool

is under the constraint of being used by non-technical users on a daily basis, so the user experience has to be factored into the time constraint of the optimisation process.

Both the GA and the MOGA ran with a crossover rate of 0.4 and a mutation rate of 0.05. These settings were already in place from the daily use of the single objective GA, which was already in use in the tool. These settings were kept for the following experiments for a fair comparison on how implementing fuzzy logic and a MOGA would affect those daily results.

5.4.1 Single Vs Multi-Objective GAs

The goal of this first experiment was to see if our chosen MOGA, NSGA-II, optimised more objectives than the standard Single Objective GA (SOGA). Where Travel is measured in kilometres (km) and balancing and coverage are measured in hours (hrs.).

Table 5-3 shows a sample of the results for three different areas when comparing single and multi-objective GAs to the current live design. Where a result is in bold, it indicates it has performed better than the current design.

Current Live Score			Single Objective			Multi-Objective		
Travel (km)	**Balancing (hrs.)**	**Coverage (hrs.)**	**Travel (km)**	**Balancing (hrs.)**	**Coverage (hrs.)**	**Travel (km)**	**Balancing (hrs.)**	**Coverage (hrs.)**
80.00	17.00	455.00	**73.86**	22.28	453.25	**73.20**	44.70	**460.59**
99.00	68.00	476.00	102.17	**38.21**	**485.64**	**97.39**	**64.38**	**492.11**
50.00	102.00	212.00	52.55	**12.13**	**214.73**	**45.86**	**48.40**	**214.74**

Table 5-3 Original Vs Single Vs Multi-Objective GA

The first row of results from Table 5-3 shows that the SOGA optimised in travel only, whereas the MOGA optimised in both travel and coverage. Although the SOGA did a better job of optimising in the balancing objective than the MOGA, neither beat the current system at balancing in this case.

In the second and third rows of results, the SOGA optimises in balancing and coverage but not travel. However, the MOGA optimises in all objectives when compared to the current patch set up. In the SOGA results, the balancing objective is better than the MOGA result. However, this is due to the fact that the SOGA has sacrificed the travel objective to reach this level of balancing. The goal is to optimise in all objectives; the SOGA fails to do this because a good result in one of the objectives out weights the poor result in the fitness function.

In the results presented in Table 5-3, the MOGA optimises in more objectives than the SOGA when compared to the current patch set up. This supports the concept that MOGAs are better at dealing with problems with multiple conflicting objectives.

5.4.2 Single Vs Multi-Objective GAs with Type-1 Fuzzy Logic

The next set of experiments aim to assess the impact of the inclusion of type-1 fuzzy logic in the patch construction and one-day simulation processes, i.e. adding in the type-1 PCFLS and TAFLS respectively. In the results shown in Table 5-4, there are two different areas (A1 and A2) that are optimised. Rows 1 to 3 show that in area 1 (A1) when a SOGA is used and the FLSs are used, the coverage is increased by 24.72%, reduce the imbalance between the patches by 46.10% and increase the utilisation by 24.72%. Coverage and utilisation are linked very closely together, so the rate of change of these values is almost the same, this pattern of improvement continues through all the results in Table 5-4. However, as a result of these significant improvements, there is an increase in the level of travel by 8.76%.

In rows 4 to 6, we see the results of the MOGA on area A1 with and without the FLSs. In this instance, we get a 64.53% reduction in travel, with a slight increase in coverage and utilisation when the FLSs are used. This very small increase may be due to the coverage being topped out by the MOGA (very little work left in the SDPs).

Area & System Type	Travel (km)	Coverage (hours)	Balancing (hours)	Utilisation (%)
A1 SOGA without Fuzzy	122.63	763.74	369.16	57.03
A1 SOGA with Type-1 FLS	133.37	952.53	170.18	71.13
A1 SOGA Effect with Type-1 FLS	**8.76%**	**24.72%**	**-46.10%**	**24.72%**
A1 MOGA without Fuzzy	135.70	1014.15	70.02	75.72
A1 MOGA with Type-1 FLS	48.14	1021.36	82.19	76.27
A1 MOGA Effect with Type-1 FLS	**-64.53%**	**0.71%**	**17.38%**	**0.73%**
A2 SOGA without Fuzzy	123.48	624.38	310.20	61.50
A2 SOGA with Type-1 FLS	145.87	739.75	174.01	72.97
A2 SOGA Effect with Type-1 FLS	**18.13%**	**18.47%**	**-43.90%**	**18.65%**
A2 MOGA without Fuzzy	165.44	799.16	74.80	78.72
A2 MOGA with Type-1 FLS	44.90	779.90	16.19	76.82
A2 MOGA Effect with Type-1 FLS	**-72.86%**	**-2.41%**	**-71.06%**	**-2.41%**

Table 5-4 Addition of Type-1 FLS to Patch construction and Job Allocation

As the MOGA improves over the SOGA results, the coverage value may have already hit the upper limits, so the potential improvements that could be made by the FLSs on coverage are very small. Hence the much-improved travel objective, as the FLSs cannot improve on coverage, they can improve on the rate of travel per task. In this example, it is the balancing objective that has suffered to the largest degree. However, when comparing this value to the SOGA with FLSs value we still get a 51.71% reduction in the imbalance of the patches, showing that MOGA is still outperforming the SOGA.

When the same experiments were run on area A2, we get similar results for the SOGA. Rows 7 to 9 show that when the FLSs are used we achieved an 18.47% increase in coverage, a 18.65% increase in utilisation and a 43.90% reduction in the imbalance of the patches.

When we look at the MOGA results for area A2, rows 10 to 12, we see that with the FLSs in use we get a 72.86% reduction in travel and a 71.06% reduction in the imbalance of the patches. As a result of these very large improvements, we suffer a small decrease in coverage and utilisation at a rate of 2.41% each. It would then be up to the user to decide if these significant improvements out-weighed the minor reductions.

If we take area A2 as an example and compare the SOGA without the FLSs and the MOGA with the FLSs, we see 63.64% reduction in travel, a 24.91% increase in coverage and utilisation and a 94.78% reduction in the imbalance of the patches, which is regarded as significant improvements in all areas and most notably in travel and patch balancing, which are the two primary areas where the FLSs are applied.

The results shown in Table 5-4 suggest that including the FLSs in the task allocation and patch construction procedures have the capability of a significant improvement on the results generated by the proposed system.

5.4.3 Type-1 FLSs Vs Type-2 Fuzzy FLSs

The third experiment aims to test the impact type-2 FLSs have on the results. The following results include the type-1 FLS results and the type-2 FLS results with different uncertainty values. If the uncertainty value is 1%, this means that the footprint of uncertainty extends 1% (of the average value) either side of the base point. For example, if the average SDP hours is 50, then the FOU will extend 0.5 hours either side of the base points.

For this experiment seeding was used in the GA to allow a more accurate comparison of the results. It is more accurate because the GA, for each run, is given the same starting population and conditions, giving a more accurate reflection of how the final outcome is affected by the different types of FLS and uncertainty values. A single objective GA was used in this

experiment so that the fitness values can be directly compared between results and there is no ambiguity as to which result is better.

Table 5-5 gives a sample of the results collected for the comparison of the type-1(T1) and type-2 (T2) FLSs. Any uncertainty (U) associated with the type-2 FLSs is noted in brackets.

Type (U)	Travel (km)	Coverage (hours)	Balancing (hours)	Utilisation (%)	Fitness
T1	180.30	819.33	133.28	62.93	**1.83**
T2 (1%)	165.22	833.72	111.47	64.03	**4.60**
T2 (3%)	157.25	794.94	161.30	61.06	**2.82**
T2 (5%)	180.30	819.33	133.28	62.93	**1.83**

Table 5-5 Type 1 FLS vs Type-2 FLS in Work Area Optimisation System

In Table 5-5, the type-1 FLSs gave an overall fitness value of 1.83. This is now compared with the results from the type-2 FLSs where three uncertainty values were tested. A 5% uncertainty gave the same result as the type-1 FLSs; this is possible because of the seeding and the same optimisation conditions. A 3% uncertainty value significantly improved on the fitness by 54%. Finally, an uncertainty value of 1% gave a fitness value of 4.60 a 151% increase over the type-1 FLSs.

The results shown in Table 5-5 suggest that upgrading from a type-1 FLS to a type-2 FLS can have significant improvements to the final results. However, the uncertainty values must be tuned correctly for these results to be realised.

5.4.4 Progressive Results

One final set of results aims to test the suggestions given by the previous experiments in one sequential real-time test. These results are not an average, not seeded, they use the same patch,

and run as if they would be in the real world. Coverage here is expressed as a percentage of the total amount of work available.

Method	Travel (km)	Coverage (%)	Balancing (hours)	Utilisation (%)
Current	**172.00**	**71.34%**	**68.96**	**63.88%**
SOGA	**187.16**	**68.86%**	**110.16**	**61.67%**
MOGA	**173.26**	**68.46%**	**54.21**	**61.30%**
MOGA-Fuzzy T1	**67.01**	**69.68%**	**62.09**	**62.40%**
MOGA-Fuzzy T2 (Tuned)	**68.15**	**71.25%**	**30.08**	**63.81%**

Table 5-6 Progressive Real-World Run Results

Table 5-6 shows the results of the progressive tests. The current patch's values are given in row one. The first step is to optimise this patch with the SOGA. Row two shows us that on this occasion the SOGA failed to optimise in any objective. This means that the optimisation would have to be run again and the GA setting would need to be tuned for this specific area to get a better result. This would cause frustration to the user and cost time.

Row three shows us the most suitable solution from the MOGA. On this occasion the MOGA has optimised in balancing, travel is less than 1% worse, so the difference here is negligible. However, the MOGA has failed to optimise in coverage and utilisation. If the user was looking to only improve on balancing and was happy to suffer the reduction in the other two objectives then this may be acceptable, else the optimisation would need to be run again.

Row four shows the most suitable solution from the MOGA using type-1 FLSs in the optimisation. Here we can see that the MOGA has now optimised in two objectives, with travel being significantly improved, now being only 38.96% of the original travel value. Coverage and utilisation still suffer. However, these objectives suffer less than if the MOGA did not use

the type-1 FLSs. There is a 1.80% increase in both coverage and utilisation over the MOGA that does not use any FLSs.

Finally, row five shows the most suitable result from the MOGA system with type-2 FLSs (that has been tuned to 1% uncertainty). On this occasion, two objectives have been optimised, and the remaining two do not suffer noticeably. The effects are less than 0.13% for coverage and less than 0.11% for utilisation. This gives the user a solid result and can confidently say that these new patches are better than the current patches. This is on one run of the optimisation and with no specific tuning of the GA required, which is great from a user's point of view.

Consequently, it can be said that these results support the use of a multi-objective genetic type-2 fuzzy logic-based system for mobile field workforce area optimisation.

5.4.5 Subjective Evaluations

To demonstrate how the visual representation of the results from the system may be interpreted there are images captured from the optimisation process of each tested method.

Figure 5.11, Figure 5.12, Figure 5.13 and Figure 5.14 show the visualisation of how the results change with each incremental improvement of the proposed system. Figure 5.11 shows that the SOGA divided the selected area into nine patches. The selected area includes both rural and urban areas, including the densely populated city area and surrounding suburbs. The single objective optimisation has split the city area (circled in Figure 5.11) up into three patches; this is not good as engineers will have to keep travelling in and out of the city. The other patches are either too large or too small.

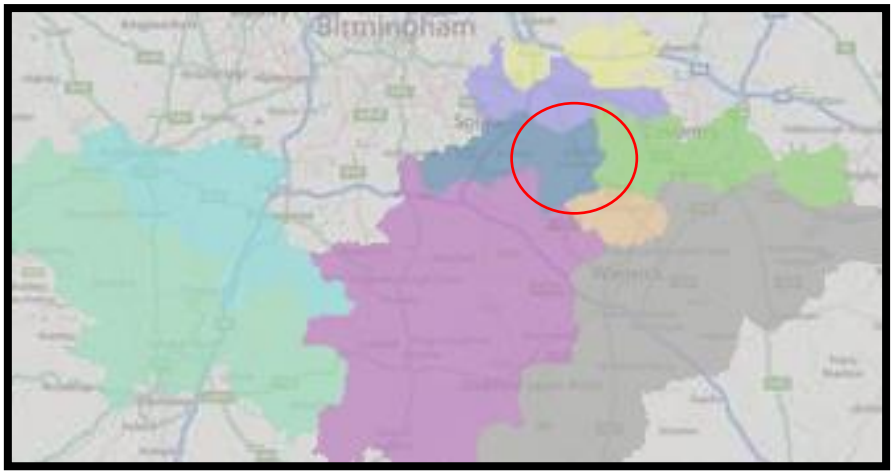

Figure 5.11: SOGA Optimisation Design (main city area is circled)

Figure 5.12 shows one of the solutions on the Pareto front from the MOGA with no FLS in use. This solution is slightly better as it has sectioned off the centre of the city. However, this city patch is now too small as the outside of the city forms part of another patch to the north. This has left one suburb in a very oversized patch (purple) and another in a small patch (light blue). The remaining patches are of reasonable size.

Figure 5.12: Multi-Objective Optimisation

Figure 5.13 shows a solution that used the MOGA with type-1 FLSs in the optimisation process. This has done a slightly better job of sectioning off the city, but there are a few SDPs that were not included in the city patch that should have been. There is also a patch in the west that is too small, and there is a suburb still in an oversized patch.

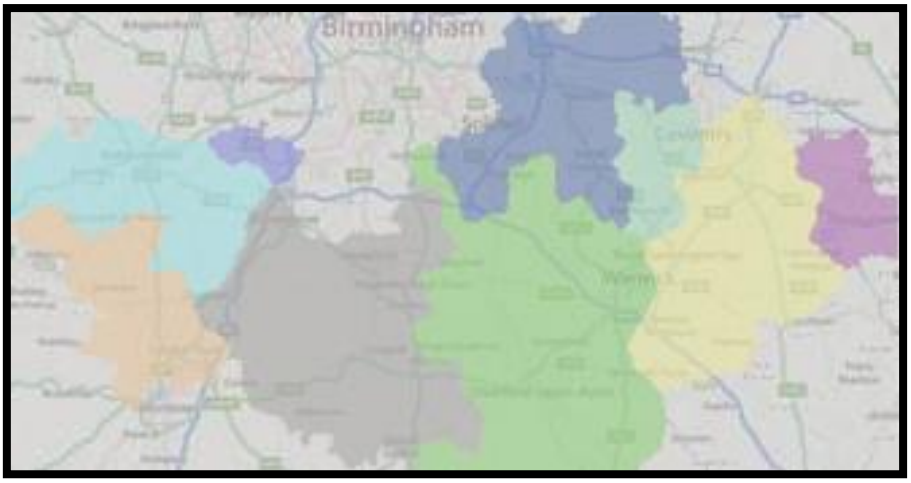

Figure 5.13: MOGA with Type-1 Fuzzy.

Figure 5.14: MOGA- with Type-2 Fuzzy

Figure 5.14 shows a solution that has replaced the type-1 fuzzy with type-2 fuzzy logic in the MOGA. This solution has done a good job of sectioning off the city. Each patch is more balanced in size and even the town to the west is its own patch. There also seems to be reasonable utilisation of the road networks in the area. The MOGA with type-2 FLSs has produced the most sensible patch designs from a visual perspective; this is important to the engineers and managers who have to accept these designs. Not only are the designs good, it has the best results from the simulation to back it up.

5.5 A Comparison of Particle Swarm Optimisation and Genetic Algorithms

The main aim of these experiments is to evaluate the differences when using a GA compared to a using a Particle Swarm Optimisation (PSO) algorithm for our system for mobile field workforce area optimisation. The best algorithm can then be used in the tool shown previously in Figure 5.10.

To avoid the issues associated with many-objective problems (as described in Section 4.5) only two objectives will be optimised. The first objective is utilisation of the engineers, the more an engineer is working, the more they are utilised. The second objective is patch balancing. Meaning that more balanced patches are better (patch balance is defined by the difference in demand, in hours, between biggest and smallest patches). These two objectives will help us to evaluate the performance of the engineers in the designed patches and how manageable the patches will be.

Our first set of experiments within this section is to evaluate the performance of the GA and PSO that use the fitness function to accommodate multiple objectives. Because this fitness function method may not be the most optimal the multi-objective algorithms, NSGA-II and Multi-Objective PSO (MOPSO), will be evaluated too. Finally, comments will be made on the difference between the best fitness function-based algorithm and the best multi-objective algorithm.

First, one of the 60 areas that need optimising was selected. This area contained 252 resources and 776 jobs (totalling 1265 hours of work) in 140 SDPs. When evaluated by the existing system, the area gave us the following results.

Objective	Value
Utilisation	78.72%
Balancing	428.74 Hours

Table 5-7 Initial Benchmark Values to Optimise

The following experiments were run on a machine with a CPU clock speed of 2 × 1.9GHz, 4GB of RAM and the capability of 4 CPU threads. The GA was run with a crossover probability of 0.4 and a mutation rate of 0.1. The first run of experiments uses the fitness function outlined in equation (6-2). This equation is a derivative of the full fitness function given in equation (6-1). For these experiments the weightings of each objective, given by W_1 and W_2, are equal. The GA was run on this area five times to get an average. These results are given in Table 5-8.

$$Fitness = \frac{(Utilisation \times W_1)}{(Balancing \times W_2)} \tag{5-2}$$

	Utilisation (%)	Balance (Hours)	Time (seconds)
	95.17	39.32	132
	95.06	45.55	139
	94.43	37.7	143
	97.06	68.47	144
	95.67	95.87	141
Average	**95.48**	**57.38**	**139.80**

Table 5-8 Genetic Algorithm Optimisation Results

Using this genetic algorithm resulted in an average increase of the utilisation by 16.76% when compared to the current design for this area. There was also a significant improvement to the balance of the patches. With the difference between the biggest and smallest patch being reduced on average to 57.38 hours from 428.74 hours, a reduction of 86.62%.

The second run of experiments was to run the PSO with the fitness function on this area. This produced the results in Table 5-9.

	Utilisation (%)	Balance (Hours)	Time (seconds)
	92.30	113.59	113.00
	93.67	94.04	140.00
	93.08	137.70	139.00
	92.28	142.28	118.00
	95.97	110.52	141.00
Average	**93.46**	**119.63**	**130.20**

Table 5-9 Particle Swarm Optimisation Results

Using the PSO also resulted in improvement in both objectives. There was, on average, an increase of 14.74% in utilisation. The patch balancing also improved with the difference between the smallest and largest patch reducing, on average, to 119.63 hours, a 72.10% reduction.

If each algorithm was compared separately, each algorithm could be seen as performing well. However, if a comparison is made between the GA results in Table 5-8 and the PSO results in Table 5-9, it can be seen that the GA has produced better results, on average, in both objectives. The GA increased utilisation by 2.12% more when compared to PSO. The GA has also improved in the balancing objective, reducing the difference by an extra 62.24 hours, or 14.52%

Table 5-8 and Table 5-9 contain a column of time. The amount of time in seconds to complete all 100 generations of the optimisation process with a population size of 20.

This is important as if the algorithm is used on areas significantly larger than the area for these experiments; the time may exponentially increase. Time is a factor of how practical the algorithm is to use in a live application. On average PSO is 9.6 seconds quicker than the GA, which could increase the more SDPs and patches there are. For these experiments, almost 10 seconds does not have a significant impact on the algorithms practicality.

The difference between the GA and PSO results led us to analyse what could cause this difference. As PSO moves around the search space using the numerical representation of the areas, it was conceived that changing the way the PSO sees the search space could have an impact. As SDPs are geographical locations, they were ordered them from nearest to furthest from an origin point. Whereas before they were ordered alphabetically by their name, which may not necessarily allow PSO to move to the next nearest neighbour if the next neighbour is represented by an ID value, which is further away from the current geographical position.

	Utilisation (%)	**Balance (Hours)**	**Fitness**
	91.62	150.62	0.30
	93.64	102.83	0.46
	93.32	121.68	0.38
	96.18	61.89	0.78
	93.81	96.54	0.49
Average	**93.71**	**106.71**	**0.48**
Standard Dev.	**1.46**	**29.26**	**0.16**

Table 5-10 PSO Geographic Organisation Results

Changing this value to a distance would allow the PSO to move to the nearest neighbour more easily. The results for this modification can be found in Table 5-10. There is no difference in run-time between the geographically ordered and alphabetically ordered runs of the PSO.

The geographically ordered PSO increases the utilisation by 0.27% and the patch balancing by 10.80%. The utilisation increase is not significant enough to say this option is better, but the patch balancing is. This could be because the patch balancing objective relies on the optimal selection of SDPs, where SDPs are geographical objects. Having the PSO search this geography seems to improve the construction of the patches.

The second set of experiments for this section is evaluating the multi-objective variations of these algorithms to see how they compared. The results for the multi-objective GA (NSGA-II) can be found in Table 5-11, and the results for the multi-objective PSO (MOPSO) can be found in Table 5-12.

NSGA-II performs significantly better than the MOPSO. NSGA-II is 5.48% better in utilisation and 59.40% better in balancing. NSGA-II also has a better standard deviation, with an 82.80% improvement in utilisation and 74.69% improvement in balancing, showing NSGA-II's results are more consistent.

	Utilisation (%)	**Balance (Hours)**	**Dist. Value**
	97.65	134.75	0.46
	96.96	131.51	0.46
	97.92	126.89	0.47
	97.42	216.29	0.37
	97.42	153.64	0.44
Average	**97.47**	**152.62**	**0.44**
Standard Dev.	**0.32**	**33.11**	**0.04**

Table 5-11 NSGA-II Optimisation Results

	Utilisation (%)	Balance (Hours)	Dist. Value
	94.63	181.33	0.39
	88.93	496.68	-0.01
	92.18	447.85	0.06
	91.48	495.13	0.00
	92.75	258.69	0.29
Average	**91.99**	**375.94**	**0.15**
Standard Dev.	**1.86**	**130.82**	**0.16**

Table 5-12 MOPSO Optimisation Results

As seen with PSO, reorganising the SDPs in geographical order, as opposed to the alphabetical order used by the GA improved the results. The results for the geographically ordered MOPSO can be found in Table 5-13. When the results are compared with Table 5-12, it can be seen again that organising the SDPs geographically improves the results. With utilisation improving by 1.48% and balancing improving by 81.54 hours or 21.69%. The standard deviation also improves by 50.53% for utilisation and 19.39% for balancing.

	Utilisation (%)	Balance (Hours)	Dist. Value
	92.14	297.11	0.24
	93.15	214.17	0.34
	93.03	461.92	0.05
	94.47	341.73	0.20
	94.54	157.07	0.42
Average	93.47	294.40	0.25
Standard Dev.	0.92	105.46	0.12

Table 5-13 MOPSO Geographic Organisation Results

Because of the multi-objective nature of the problem and the conflicts in the objective values, it is difficult to determine the overall quality of a solution. A distance metric can be used to measure how far away from the original solution a new solution is. A higher distance value would represent a better solution than the original; a negative value would represent a worse solution. The simple distance metric used for this problem is given in (5-3). This concept will be expanded upon in Section 6.2, where many-objective solutions are measured using this method.

$$DIST = \left(\frac{U_s - U_o}{U_o} - \frac{AB_s - AB_o}{AB_o}\right) \quad (5\text{-}3)$$

$$DIST = \frac{New - Original}{Original} \quad (5\text{-}4)$$

In (5-3) the utilisation for the new solution is denoted by (U_s), and the utilisation for the original is denoted by (U_O). The area balance given by the new solution is denoted by (AB_s), and the area balance given by the original is denoted by (AB_O). Each objective in equation (5-3) calculates the distance using equation (5-4).

Using the distance value as a metric of comparison we can see that the NSGA-II has a significantly higher average value when compared to the geographically organised MOPSO, as it is 0.19 or 76% stronger on average. NSGA-II's standard deviation of the distance metric is also reduced by 66.67%, reducing from a value of 0.12 to 0.04.

For completeness, the single objective GA and results from Table 5-8 and the NSGA-II results from Table 5-11 can be compared, as these were the best algorithms from each set of experiments. Both algorithms have their strengths. The single objective GA has, on average, much better patch balancing, improving over NSGA-II by 62.40%. However, the utilisation is improved by 1.99% when using NSGA-II.

The weighting of each of these improvements would be down to the user's own preference. Usually, utilisation of the workforce is more beneficial, and so a reduction in the balance of the

patches would be acceptable, especially given that the NSGA-II options still significantly improves over the current design. NSGA-II is very consistent with its utilisation, suggesting it may have hit the upper limit of utilisation of engineers for the area.

Overall, it can be said that the GA based algorithms performed better for our multi-objective problem in both the fitness function (single objective GA) and multi-objective based variations. Additionally, although the PSO based algorithms performed worse, if the problem is geographical in nature, the performance of PSO algorithms could be increased if the search space is organised geographically. This perhaps suits the underlying model of the PSO algorithm better.

5.6 Discussion

This chapter discussed the different optimisation methodologies first proposed to tackle the multi-objective mobile field workforce area optimisation problem. It discussed the need to compare both single objective optimisation algorithms and multi-objective optimisation algorithms. The chapter then introduced employing fuzzy systems to certain elements of the system, namely task allocation and patch construction. This lead to a significant improvement to both travel and patch balancing, the two objectives most affected by these fuzzy systems.

A discussion on upgrading of the type-1 fuzzy systems to type-2 systems was given, to analyse if there was further benefit to be gained.

It was explained why traditional single objective GAs cannot fully handle optimisation processes with multiple objectives, especially when those objectives are conflicting. As a result, multi-objective genetic algorithms were introduced, specifically NSGA-II. This gave the optimisation process the ability to compare the results of the individual objectives between possible solutions and rank them accordingly.

As the proposed system is designed to tackle a real-world problem with real-world data, there are many uncertainties. Thus, justifying the development of the fuzzy systems.

To fully evaluate each aspect of the proposed system, several experiments were conceived and executed. Each were designed to assess the impact of the different methodologies. The results of these experiments showed that a multi-objective system was able to optimise in more objectives than a single objective system. The results also showed that including type-1 fuzzy logic systems on the task allocation and the patch construction parts of the optimisation improved the results the system generated. With one example showing that we could have better performance in all objectives when compared to the SOGA system that employed crisp logic. With some minimisation objectives being reduced by up to 94.78%.

The results showed that upgrading the type-1 fuzzy logic systems to type-2 further improved on the results, giving up to 151% improvement over type-1 fuzzy in some instances. As this is a real-world problem being tackled, there are many aspects that could be improved upon to have a system that generates even stronger results. One area of improvement is where the parameters of the type-2 systems could be optimised.

The final section compared both a genetic algorithm-based solution against a particle swarm optimisation-based solution. This comparison has been extended to multi-objective versions of these algorithms using NSGA-II as the multi-objective GA and MOPSO as the multi-objective PSO.

The GA has, on average, increased utilisation by 2.12% when compared to PSO. GA has also improved on average in the balancing objective, improving by 14.52%, or reducing the difference by an extra 62.24 hours.

For the multi-objective variations, NSGA-II performed better than the MOPSO. NSGA-II is 5.48% better in utilisation and 59.40% better in balancing. NSGA-II also has a better standard

deviation with an 82.80% improvement in utilisation and 74.69% improvement in balancing, so its results are more consistent. Additionally, it was found that representing the search space geographically for the PSO based algorithms improved the results, however not enough to outperform the GA based algorithms.

The next chapter will discuss the proposed improvements to this system by utilising cloud resources and addressing many-objective optimisation.

Chapter 6. The Optimised Many-Objective Optimisation Cloud-Based System

The preliminary results given in section 6.4 indicated that a genetic algorithm-based system would be the most appropriate, with fuzzy logic systems in place to support some of the key decision making processes (i.e. the task allocation in the simulation and the patch construction). Both type-1 and type-2 fuzzy systems were implemented and evaluated.

Given that a foundation in which to build and develop the system has been established, it is important to continue to enhance the system's modules and complete more comprehensive research and analysis.

This chapter addresses several enhancements to the system, including its framework, scalability, advanced tuning and optimising for many-objectives.

The first two points to be addressed will be the framework and advanced tuning methods. The proposed enhancements to the system are illustrated in Figure 6.1. The two fuzzy systems used within the tool will be optimised by a separate genetic algorithm, as opposed to just being designed by an expert. The purpose of this is that there are many geographical regions, which change frequently, it is challenging to have a human expert continuously update the fuzzy systems for each area to keep them relevant. Details on the optimisation of the fuzzy systems can be found in Section 6.1.

The framework of the tool will be updated to allow the genetic algorithm to take advantage of multiple CPUs by creating multiple threads during the evaluation stage of the optimisation. The evaluation stage is the most computationally expensive part of the optimisation because it requires the simulation to run on every generated solution.

Currently, the system requires the user to enter parameters, such as the number of patches to optimise for and the various GA specific parameters (number of generations, population size etc.). Once the user then confirms the settings and starts the optimisation process, the system will check if it should optimise the fuzzy systems that will be used. If yes, the system will use a GA to optimise the membership functions. If the system has been selected to use type-2 fuzzy systems, it will then proceed to optimise the Footprint of Uncertainty (FOU) of each membership function.

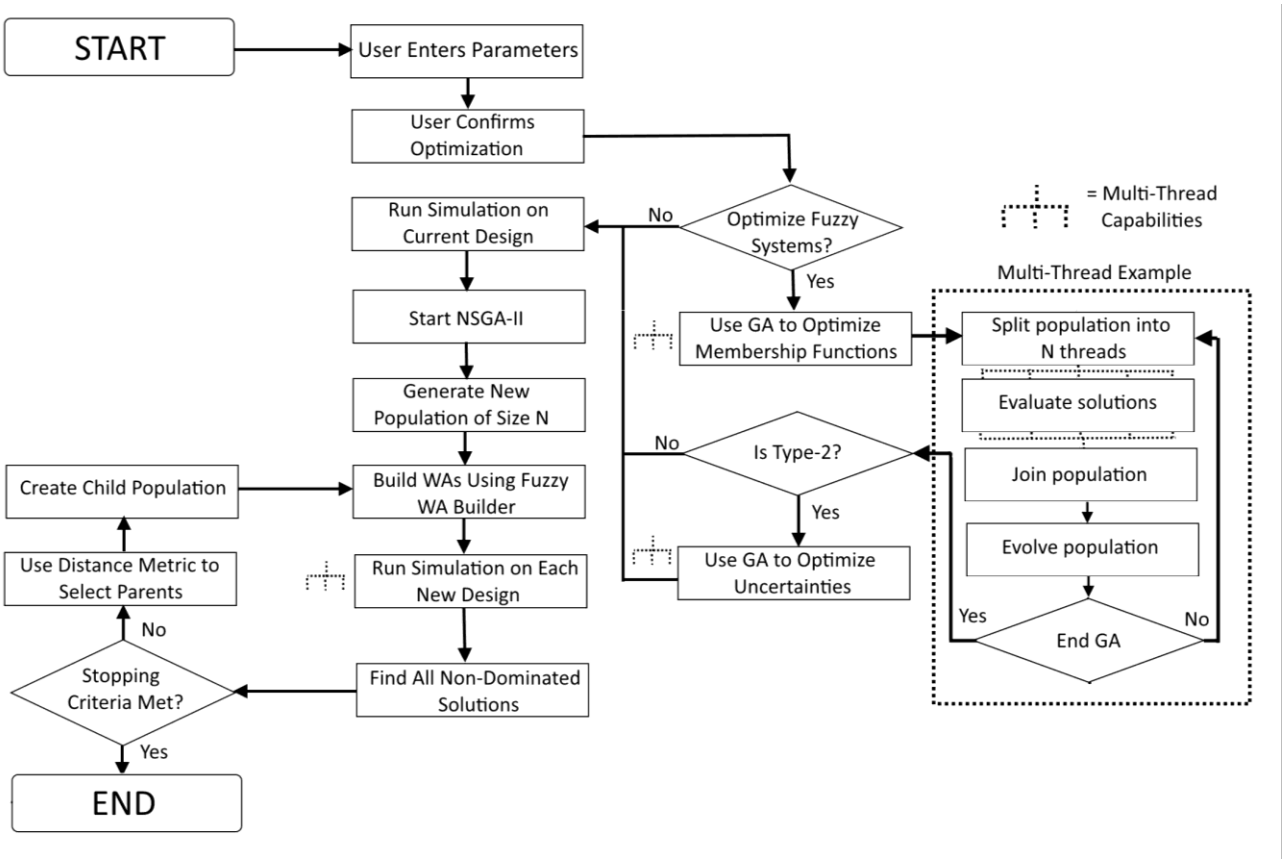

Figure 6.1: The Proposed Cloud-based Many-Objective Type-2 Fuzzy Logic Based Mobile Field Workforce Area Optimisation System

For each GA used in the proposed system, including those that optimise the fuzzy systems, multiple threads will be created at the point each solution in the population is about to be evaluated. In this way, the solutions can be evaluated in parallel, and this will have the potential to decrease the optimisation time [85]. This is where multiple threads are best placed because,

as mentioned, the evaluation of each solution takes the most time plus this step does not require all other solutions to be available, such as in the selection and crossover steps.

Once the fuzzy systems have been optimised, the system will simulate the current design. This run of the simulation is designed to get the objective values of the design currently being utilised by the mobile workforce, so that these values can be used for comparisons or benchmarking.

When the current design has been evaluated, the NSGA-II will start the optimisation process. It will create a population of solutions and evaluate each one, giving each solution, a value based on the proposed distance metric. The proposed distance metric is used to help address some of the weaknesses in NSGA-II when it comes to many-objective problems, see Section 7.2 for more details.

Multiple threads will again be created, and the population will fork into these threads, splitting the population evenly between the threads. Once all solutions have been evaluated, the population will join back up again allowing the NSGA-II to operate as normal and start calculating the dominance of each solution, creating the fronts. Because of the many-objective issues we have outlined, with all solutions ending up on the Pareto front, the distance metric is used to help with parent selection.

If the stopping criteria for the algorithm are met, then the latest Pareto front of solutions will be presented to the user with the solution that has the highest distance value being highlighted as the best, or most recommended, result.

6.1 Genetically Optimised Fuzzy Systems

Fuzzy Logic Systems have been shown to handle imprecisions and uncertainties within an environment. The majority of FLSs are type-1 based and therefore cannot fully handle the imprecisions and uncertainties presented by dynamic environments whereas type-2 systems have demonstrated they can outperform type-1 systems in these environments [86], [87], [88].

Additionally, when some fuzzy systems are created their membership functions are generated by a human expert. These membership functions could then be sub-optimal and therefore need to be tuned to perform well in a changing environment. When a type-2 system is used the uncertainty also needs to be calibrated to suit the environment the FLS will be used on.

Wagner [89] looked at this issue and proposed using a GA to tune the membership functions of a type-2 fuzzy set.

As the proposed system will be used in multiple problem environments, its membership functions cannot be tuned offline because it is unknown which set of working areas the user will be optimising. Therefore, in our proposed system the membership functions and FOUs will be tuned using a Real-Valued GA at the start of each optimisation process. The genes of each solution will represent the points each membership function has along the x-axis.

Figure 6.2 shows an example of a chromosome for the parameters of the membership functions of two type-1 fuzzy sets. Each membership function will have four points associated with it giving a total of eight genes. The first four values are for the first membership function parameters, and the last four values are for the second membership function parameters.

Figure 6.3 shows an example of a chromosome for the uncertainty associated with type-1 fuzzy sets. Each gene represents the uncertainty percentage associated with the base values of the

type-1 fuzzy sets to result in the upper and lower membership functions of the type-2 fuzzy sets.

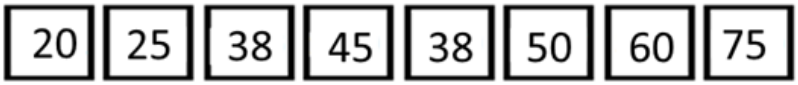

Figure 6.2: Real-Value Chromosome for the Parameters of Two Type-1 Fuzzy Sets Membership Functions.

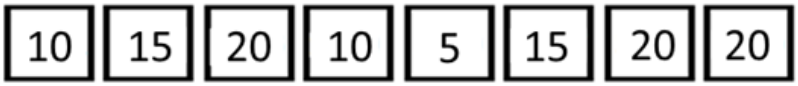

Figure 6.3: Real-Value Chromosome for Percentage Uncertainty Associated with the Type-2 Fuzzy Sets

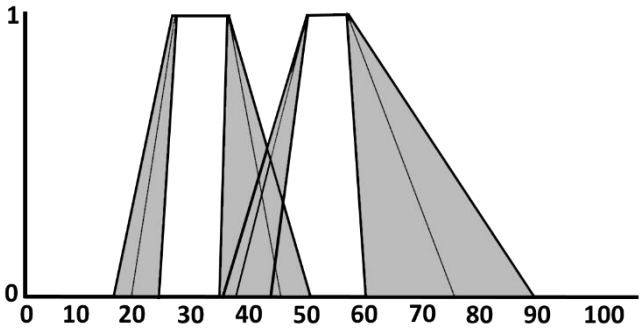

Figure 6.4: Resulting Type-2 Membership Functions from Chromosomes

Figure 6.4 shows the resulting type-2 fuzzy set, given from the genes in Figure 6.2 and Figure 6.3. This GA will evaluate the fuzzy systems on their primary purpose for ten seeded chromosomes. So, for the PCFLS it will evaluate how much the proposed membership function improves on the patch balancing objective. For the TAFLS, the system will evaluate how much improvement there is to the coverage to travel ratio. Once the ten solutions have been evaluated, the fitness of the solution is the average objective value from these ten solutions.

For the type-2 fuzzy systems, the uncertainty tuning happens after the membership function tuning has taken place.

6.2 Many-Objective Distance Metric

The proposed distance metric is used to help the parent selection process and suggest the best result to the user. The distance metric for our given objectives is shown by (6-1)

$$DIST = \left(\frac{C_S - C_O}{C_O} - \frac{T_S - T_O}{T_O} + \frac{U_S - U_O}{U_O} - \frac{AB_S - AB_O}{AB_O} - \frac{TB_S - TB_O}{TB_O}\right) \quad (6\text{-}1)$$

$$DIST = \frac{New - Original}{Original} \quad (6\text{-}2)$$

In Equation (7-1) the coverage is given by the new solution (C_s) and the coverage given by the original (C_O). The travel value given by the new solution (T_s) and the travel value given by the original (T_O). The utilisation given by the new solution (U_s) and the utilisation given by the original (U_O). The area balance given by the new solution (AB_s) and the area balance given by the original (AB_O). Finally, there is the team balance given by the new solution (TB_s) and the team balance given by the original (TB_O).

Each objective in (6-1) calculates the distance using (6-2). This change in objective value is normalised over the original value, giving the distance as a value between 0 and 1 for each objective.

Coverage and utilisation are both maximisation objectives and add to the distance value. The remaining objectives are minimisation objectives, so they subtract from the distance value. This is, for example, if the travel value in the new solution is lower than the original, it will give a negative distance for that objective, and so subtracting this negative value increases our overall distance value, giving us an indication that this solution is stronger.

It is worth noting that for this distance metric to work there needs to be original values. If we do not have a base to compare to, we do not know if we have improved over the currently implemented solution. Thus, making it difficult to assess the real-world impacts of the work.

This metric could be used if there are no original results, for example creating new patches where there weren't any before. However, this would require the new patches to be designed by an expert or by a GA process that does not use the distance metric. Such as how new patch designs were created in Section 6. Once these "original" results have been created, then this proposed system could be used to improve upon these results.

6.3 Cloud-Based Optimisation

As mentioned, one of the first enhancements to the system is to do with framework and scalability. It has been discussed how the solution evaluation of the GA will support multi-threading. This is immediately beneficial to the speed of the optimisation for desktop computers the tool is run on. However, this also means more CPU resource is allocated to the tool, and the user will have less resource to carry out other tasks while the optimisation is taking place.

One way around this is using a server, or cloud resource to run the tool on. This not only has the benefit of completely freeing up the user's personal machine, allowing them to complete tasks unhindered, but cloud resources typically have more processing power and more cores than a typical desktop machine.

The downsides to utilising cloud resources are security and accessibility. If the security of the cloud is not maintained, the sensitive data (from engineers) is at risk. If the cloud servers go down or receive too many requests (such as in a Distributed Denial of Service attack), the tool may become inaccessible. This requires there to be a reliance on a competent and responsive cloud maintenance team.

6.4 Experiments and Results for the Cloud-Based Optimised Many-Objective Optimisation System

The first set of experiments in this section are the comparison of the un-optimised, and the GA optimised fuzzy systems. These results include a comparison of both genetically optimised type-1 fuzzy systems and type-2 fuzzy systems.

The second set of results will look at the benefits brought to this system by multi-threaded cloud computing.

6.4.1 Comparison of Genetically Optimised Fuzzy Systems

An aim of one of the experiments is to compare a system that used type-1 fuzzy sets and type-2 fuzzy sets both tuned and untuned by a genetic algorithm. However, because of the problems associated with many-objective optimisation, a solution was needed to solve the problem of Pareto front saturation simultaneously. As a result, the proposed distance metric is used to help evaluate dominating solutions.

As the distance metric needs comparison scores to work from, an area in the real-world environment was selected based on its need for optimisation. The current designs for these active patches were created by experts who have local knowledge about the area.

Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)
74.6	7.00	74.03	428.74	71

Table 6-1 Current Benchmark Values

Once we had chosen a suitable area to optimise we simulated that area to see what performance levels it was currently operating at. These values can be seen in Table 6-1.

6.4.1.1 Quantitative Analysis

The initial test was to run the optimisation with untuned type-1 PCFLS and TAFLS. This configuration of optimisation was repeated for the same area 10 times. For each run, the best result based on the distance metric is shown in Table 6-2.

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	87.86	2.75	87.19	117.48	19.00	0.48
	87.20	3.33	86.53	302.15	30.00	0.36
	90.50	3.19	89.81	321.14	49.00	0.31
	94.46	2.92	93.74	253.84	38.00	0.40
	86.14	4.12	85.48	258.32	14.00	0.38
	91.10	4.55	90.41	278.15	26.00	0.36
	93.41	3.08	92.70	150.86	24.00	0.47
	90.19	4.04	89.50	221.18	20.00	0.41
	89.41	3.56	88.73	202.13	40.00	0.37
	84.72	3.16	84.07	362.07	53.00	0.25
Avg.	**89.50**	**3.47**	**88.82**	**246.73**	**31.30**	**0.36**
SD	**3.09**	**0.59**	**3.07**	**75.73**	**13.19**	**0.07**

Table 6-2 Results from Untuned Type-1 Fuzzy Systems

The average of these ten runs is shown, in bold, on row 12 (with row 1 being the header row) and the standard deviation of each objective in bold on row 13. We can then see that the distance metric used on the average of the ten runs, gives the value 0.38 for the system with untuned type-1 systems.

The experiment was repeated with the genetic tuning of the type-1 membership functions. The best solution from each of these ten runs can be seen in Table 6-3. Row 12 shows the average

of the ten solutions for each objective and gives us a distance value of 0.41. An improvement on the untuned type-1 system of 8.40%.

A comparison of the standard deviation (SD), of the un-tuned and tuned systems, can also be made. With the tuned system's SD for the patch balance and team balance objectives improving by 9.93% and 43.14% respectively.

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	89.93	3.44	89.24	203.74	26.00	0.42
	90.15	2.71	89.46	153.65	20.00	0.48
	86.46	3.25	85.80	247.37	18.00	0.40
	92.70	4.54	91.99	105.49	14.00	0.48
	85.28	3.55	84.63	267.11	20.00	0.37
	85.68	4.31	85.02	230.90	12.00	0.39
	85.10	3.57	84.44	233.91	10.00	0.42
	93.37	4.20	92.66	189.72	33.00	0.40
	88.31	3.11	87.64	237.09	14.00	0.43
	84.07	3.83	83.43	360.32	28.00	0.29
Avg.	**88.11**	**3.65**	**87.43**	**222.93**	**19.50**	**0.41**
SD	**3.30**	**0.57**	**3.28**	**68.21**	**7.50**	**0.05**

Table 6-3 Results from Tuned Type-1 Fuzzy Systems

The experiment is then repeated with the type-2 systems. Again, the optimisation is run ten times and the best result, based on the distance, is shown for each run. We have put the results for the untuned type-2 systems in Table 6-4. The untuned type-2 systems have the same membership functions as the untuned type-1 systems. However, they also have 1% uncertainty applied to them. This is based on the results in Section 6.4.3 where about 1% uncertainly performed the best.

Table 6-4 shows that the distance value, based on the average of the ten solutions, is better than the untuned type-1 system. The average of these ten runs is 0.40 compared with the untuned type-1 result of 0.36. The type-2 untuned gives an 11.11% improvement over the type-1 untuned system. This strengthens the case for type-2 systems being applied to this domain.

However, the results also show that the untuned type-2 systems performed slightly worse than the tuned type-1 system, by about 2.44%. This result suggests that tuning a type-1 system can improve the results by taking some of the uncertainty out of the membership functions. Given that type-2 fuzzy sets are designed to handle this uncertainty, it is reasonable for this to be the reason.

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	90.74	2.45	90.04	147.40	28.00	0.47
	92.01	3.72	91.30	106.31	11.00	0.51
	84.31	4.83	83.66	223.52	11.00	0.39
	89.12	3.91	88.44	180.32	17.00	0.43
	83.42	2.92	82.78	261.61	16.00	0.40
	83.57	3.97	82.93	322.10	31.00	0.30
	91.24	3.73	90.54	230.78	38.00	0.37
	89.83	3.46	89.14	285.34	42.00	0.33
	91.12	2.30	90.42	247.30	46.00	0.38
	85.60	3.19	84.95	224.93	18.00	0.41
Avg.	**88.10**	**3.45**	**87.42**	**222.96**	**25.80**	**0.40**
SD	**3.47**	**0.76**	**3.44**	**64.12**	**13.01**	**0.06**

Table 6-4 Results from Untuned Type-2 Fuzzy Systems

Finally, the tuned type-2 system was run ten times. Table 6-5 gives us the results of the tuned type-2 systems. Here we can see the tuned type-2 performed better, on average, than the untuned type-2 by 10.00% (0.40 vs 0.44) and performed better than the tuned type-1 by 7.23% (0.41 vs 0.44). Additionally, we can see that the tuned type-2 systems gave results with a smaller average standard deviation in Coverage, Utilisation and Patch Balancing than all other systems, meaning these results are more reliable and we can expect more consistency from the tuned type-2 systems.

Again, we can compare the improvement of the SD for the balance and team balance objectives in both the untuned and tuned type-2 systems. With area balance improving by 18.70% and the team balance improving by 11.99%.

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	90.72	3.25	90.03	155.48	24.00	0.45
	90.45	4.29	89.76	275.61	38.00	0.37
	92.91	3.59	92.20	131.11	8.00	0.51
	88.43	2.44	87.75	174.89	36.00	0.42
	91.05	2.68	90.35	190.63	30.00	0.44
	90.75	3.67	90.05	140.34	7.00	0.50
	87.44	3.14	86.77	206.36	31.00	0.40
	87.23	3.62	86.56	186.20	12.00	0.44
	92.19	4.50	91.48	128.64	15.00	0.46
	87.58	3.91	86.91	223.40	19.00	0.40
Avg.	**89.88**	**3.51**	**89.19**	**181.26**	**22.00**	**0.44**
SD	**2.06**	**0.65**	**2.04**	**46.05**	**11.45**	**0.05**

Table 6-5 Results from Tuned Type-2 Fuzzy Systems

6.4.1.2 Subjective Analysis

The results can be compared visually to analyse the results from a subjective view. Figure 6.5 shows the current design. Area "1" in Figure 6.5 is a large urban area. Because this large urban area is all in one patch, it results in the large imbalance of the patches given in Table 6-1.

Figure 6.6, Figure 6.7, Figure 6.8 and Figure 6.9 show a 'best' result from each of the system configurations that we ran experiments for in Section 7.4.1.1. Figure 6.6 shows the untuned type-1 system split this large urban area up into two patches, which is a reasonable proposal as this much improves the area balance over the current design. Having three patches for this large urban area will likely improve the result further.

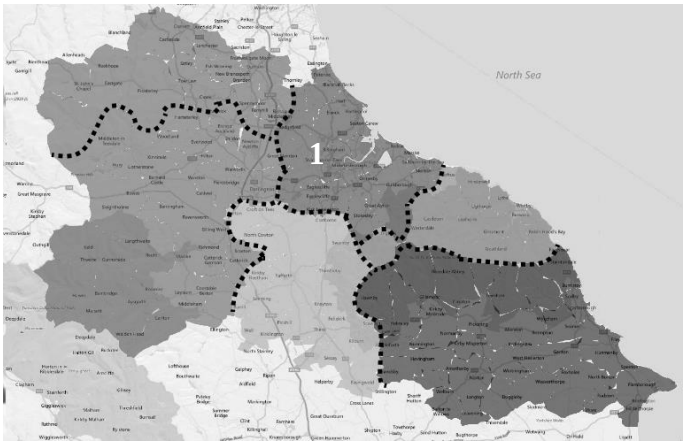

Figure 6.5: Current Patch Design

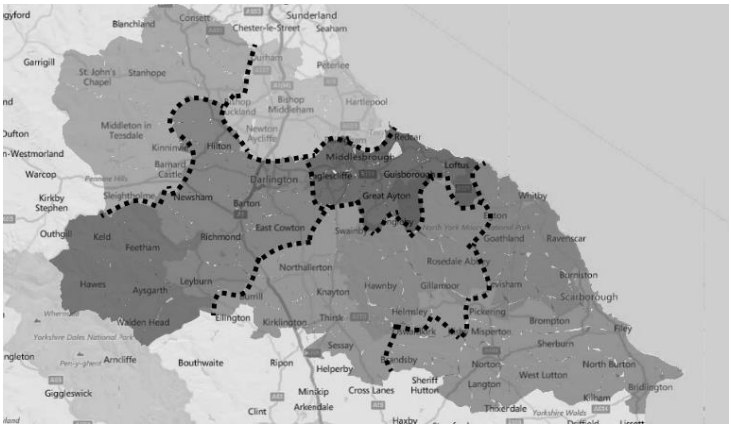

Figure 6.6: A Type-1 Un-Tuned Solution

The idea of needing three patches is supported by Figure 6.7, which shows a tuned type-1 design that has splits the area into three and improved on the balancing objective. However, because area "1" in Figure 6.7 is small and area "2" is so large, it impacts on the travel, and subsequently the coverage.

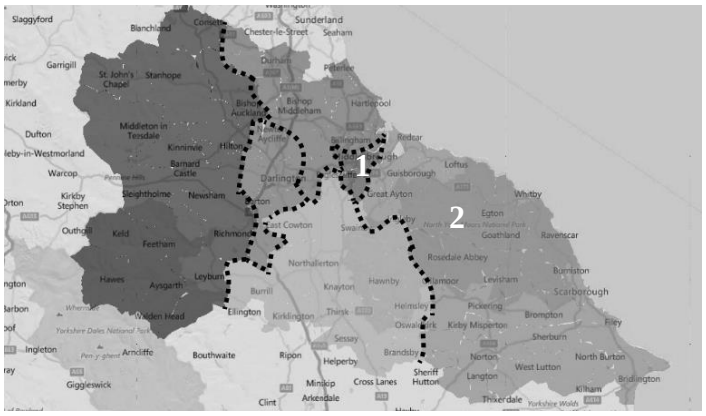

Figure 6.7: A Type-1 Tuned Solution

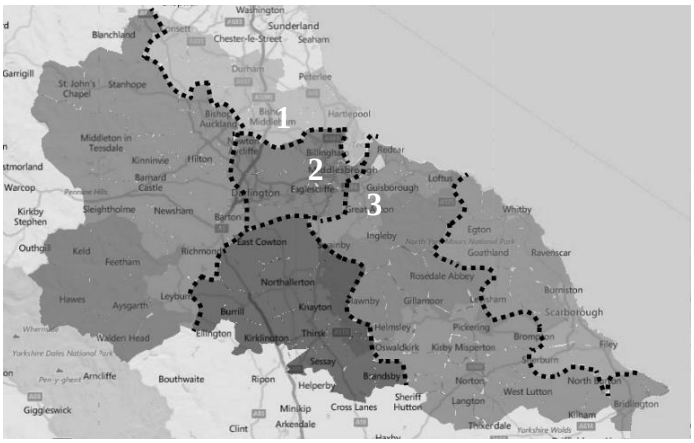

Figure 6.8: A Type-2 Un-Tuned Solution

Figure 6.8 shows us an untuned type-2 result. It splits up the urban area into three patches which is good, but area "3" extends far away from the urban area. Similar to Figure 6.7. The similarities of Figure 6.7 and Figure 6.8 are backed up by the similar results of the type-1 tuned and the type-2 untuned results in Table 6-3 and Table 6-4.

Visually, it is clear from Figure 6.9 that the tuned type-2 result is more logical. The urban area is split into three equal patches (1-3) with the rural patches outside and much larger.

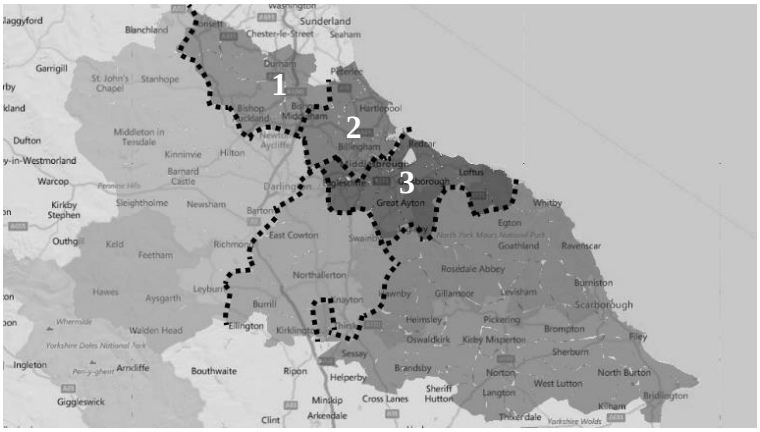

Figure 6.9: A Type-2 Tuned Solution

6.4.2 The Speed of Optimisation Results

The first experiments, for improving the speed of the optimisation process, will consist of evaluating the time difference of running the system on the current hardware, compared to running the system in the cloud. These tests include splitting the population into multiple threads as well as comparing just the single thread option.

The model of CPU in the standard laptop that runs the desktop application version of the tool, is an Intel Core i5-4300U, whereas the model of CPU in the Cloud is stated to be an Intel Xeon E5-2680. A comparison of the specification of the laptop and the cloud is given in Table 6-6. Clearly, the cloud has much more processing resources available. The use of the cloud helps solve the problem of resource scarcity with personal devices such as laptops.

Hardware Comparison	Laptop	Cloud
CPU Clock Speed	2 x 1.9GHz	8 x 2.7GHz
CPU Threads	4	16
RAM	4GB	32GB

Table 6-6 Optimisation Hardware Comparison

The experiments in Section 6.4.1 have established that the type-2 tuned fuzzy logic version of the optimisation system is the strongest. We can look at the potential benefits of utilising cloud resources. Figure 6.10 shows the comparison of how long a GA (and MOGA) would take. This is important as if we want to use the type-2 tuned fuzzy systems we add two additional GAs

into the optimisation process (one for the membership functions and one for the FOUs). Figure 6.10 gives us an indication of the level of improvement we would expect, before moving onto GAs with larger populations.

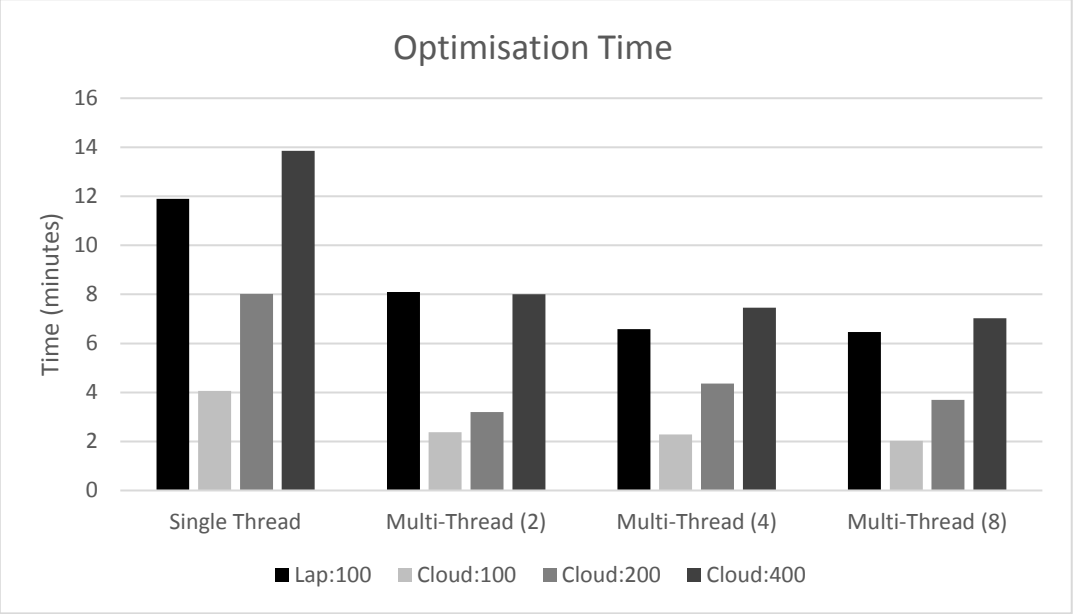

Figure 6.10: Optimisation Times

In Figure 6.10, we can see that on the laptop for a population size of 100 and the old single-threaded model it takes approximately 12 minutes to complete the optimisation. However, if the system is moved into the cloud and run the same optimisation, with a population of 100, the time taken to optimise can be dramatically reduced. The overall time is reduced by approximately 66.66% to about 4 minutes. This is clearly just due to the extra CPU resources available in the cloud. We can then increase the population and measure the increase in time in the cloud. The optimisation was then run with a population of 200, giving an average optimisation time of 8 minutes. Doubling the population size again, to 400, and the optimisation takes 14 minutes. This tells us that we can quadruple the population size in the optimisation, and on the cloud and it only takes 16.67% longer.

However, this is just the single threaded model. If the CPU power is utilised as much as possible the optimisation time can be further reduced. By increasing the number of threads to two, we

can reduce the time taken to optimise on the laptop by about 33.33% to 8 minutes. However, the reduction in optimisation time is greater in the cloud, as the optimisation time is reduced to approximately 2 minutes 23 seconds. By adding multi-threading capabilities to the system and moving the system into the cloud, we can reduce the optimisation time form approximately 12 minutes to 2 minutes 23 seconds, give a reduction in time of about 9 minutes and 37 seconds, or about 80.14 %.

By increasing the threads, the optimisation time can be further reduced. However, there is evidence of diminishing returns having a significant effect. Increasing the number of threads to four reduces the average time to 2 minutes 17 seconds. Increasing the number of threads to 8 reduces the time to approximately 2 minutes for a population size of 100. Giving a total reduction in time of 10 minutes, or 83.33%

Due to this significant time reduction in the multi-threaded model, the population size can be increased as was done with the single threaded model. If we increase the population size to 200 we get times of 3 minutes 12 seconds for two threads, 4 minutes 21 seconds for four threads and 3 minutes 42 seconds for eight threads. The minor fluctuations in time can be attributed to a few causes. It could be that there were a different number of processes taking place in the cloud at the time of optimisation, thus affecting the time to optimise. This is one of the minor drawbacks, as there may not be total control over the available resources in the cloud. Additionally, it could be that there needs to be a minimum number of solutions per thread to have a practical benefit. For example, if a population of 200 is split into eight threads then that is only 25 solutions per thread.

If the population is increased to 400, we get times of 8 minutes for two threads, 7 minutes 27 seconds for four threads and 7 minutes for eight threads. The continued reduction in time seems

to support the theory of a minimum number of solutions per thread to have maximum time benefit.

Overall, for the time experiments, we can conclude that moving the system to the cloud and adding multi-threading capabilities significantly improve the time. However, some tuning may be required to optimise the number of threads to be used, to gain the most time benefit. With this reduced time to optimise we can then increase the population size in the optimisation to 400. This gives us a similar time to optimise in the cloud when compared to the time to optimise on the laptop with a population of 100 and two threads.

6.4.3 The Increased Population Results

Now that significant time benefit has been gained, because the system now runs in the cloud, the population size can now be increased, thus covering more of the search space. However, the aim here is to see if increasing the population size gives improved results. As if there is minimal benefit in the results of the optimisation, then it may be that the most benefit from moving the system to the cloud is just time. Thus the population should stay at 100 to gain the most time benefit.

The optimisation with the type-2 genetically optimised fuzzy systems selected is used in the following experiments due to the results from section 7.4.1. We increased the population to 200, and the results of this experiment are given in Table 6-7

The optimisation was run five times, smaller than the 10 for the other experiments. However, the standard deviation (SD) is significantly reduced due to the increase in population size. It has been reduced from 0.05 to 0.02 or by 60%. In addition to more consistent results, the results give improved objective values and result in an increased average distance value of 0.06 or 13.64%. Perhaps more significantly this increased population size has helped the NSGA-II and the many-objective problem, as all five objectives are improved over the average results of the

type-2 tune system with a smaller population given in Table 6-5

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	90.60	2.76	90.10	153.48	20.00	0.48
	89.66	3.38	89.16	93.84	4.00	0.53
	90.96	3.13	90.45	92.66	17.00	0.51
	92.77	4.59	92.26	84.71	8.00	0.50
	90.91	3.58	90.40	105.00	16.00	0.49
Avg.	**90.98**	**3.49**	**90.48**	**105.94**	**13.00**	**0.50**
SD	**1.13**	**0.69**	**1.12**	**27.54**	**6.71**	**0.02**

Table 6-7 Results from Increasing Population to 200

The next experiment involved running the optimisation with a population of 400. These results can be found in Table 6-8. The standard deviation is the same as a population of 200. However, the average distance value has increased to 0.52, an increase of 4%. As with the population of 200, all objectives have been improved over the average results given in is Table 6-5. Additionally, these results improve in 4 out of 5 objectives when compared to the population of 200 results.

A summary of the average results can be found in Table 6-9. Where T1 means type-1 fuzzy systems and T2 means type-2 fuzzy systems. T2_POP200 and T2_POP400 are the tuned type 2 systems with populations of 200 and 400 respectively. All the results improve over the original, in all objectives. This is a result of using the fuzzy systems with a multi-objective genetic algorithm. We have also shown that tuning any fuzzy system that is to be used will improve the results and showing that the tuned type-2 systems improve the results the most.

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val
	92.10	3.55	91.59	102.89	14.00	0.51
	87.26	2.87	86.78	137.65	8.00	0.50
	92.45	4.17	91.94	100.40	9.00	0.50
	92.74	2.72	92.23	102.55	10.00	0.54
	92.49	2.98	91.98	115.19	8.00	0.54
Avg.	**91.41**	**3.26**	**90.90**	**111.73**	**9.80**	**0.52**
SD	**2.33**	**0.60**	**2.32**	**15.61**	**2.49**	**0.02**

Table 6-8 Results from Increasing Population to 400

	Coverage (%)	Travel (km)	Utilisation (%)	Balance (Hours)	Team Balance (People)	Dist. Val	SD
Current	74.60	7.00	74.03	428.74	71.00	-	-
T1	89.50	3.47	88.82	246.73	31.30	0.38	0.07
T2	88.11	3.65	87.43	222.93	19.50	0.41	0.05
T1_Tuned	88.10	3.45	87.42	222.96	25.80	0.40	0.06
T2_Tuned	89.88	3.51	89.19	181.26	22.00	0.44	0.05
T2_POP200	90.98	3.49	90.48	105.94	13.00	0.50	0.02
T2_POP400	91.41	3.26	90.90	111.73	9.80	0.52	0.02

Table 6-9 Cloud Optimisation Results Summary

Due to the availability of cloud resources and the modification of the software to support multi-threaded genetic algorithms, we can improve the optimisation process in 2 ways. The system can either run the optimisation in a greatly reduced time or we can run it for the same time, but with a greatly increased population size. The increase in population has produced even stronger and more consistent results. Improving by as much as 18.18% if the population is increased to 400.

6.4.4 Comments on the Experiments and Results

In this section of the results, a cloud-based many-objective type-2 fuzzy logic based mobile field workforce area optimisation system has been presented. The cloud in this context was secure on-site hardware with more CPU capacity and more multi-threading capabilities. Due to high-levels of security, the data had to remain on BT premises and thus BT's own internal 'Cloud' systems were used. These results have demonstrated the need to optimise any fuzzy logic system used in the optimisation process. The optimisation of our type-2 fuzzy logic system improved the results by 10.00%. Additionally, the potential practical benefits have been explained.

Potential improvements in results can be gained from moving the system from personal hardware to the cloud. This allows the optimisation process to run much faster, by as much as 83.33%, allowing the population size of the genetic algorithm to be increased by 300%. This increase in population resulted in better results. These results improved, on average, in all objectives when compared to the smaller population tests by as much as 18.18%.

These improvements allow the system to be effectively used on a daily basis. The users of the system will still run it for the same amount of time but are presented with better results. In addition to this, their CPU's are freed up, and they can run the optimisation as long as they want, without having to worry about shutting off their laptops for travel purposes. One of the key benefits of cloud computing.

6.5 Discussion

This chapter presented a many-objective fuzzy logic system for the optimal design of patches. This system includes a distance metric for analysing the solutions that are generated by a multi-objective optimisation algorithm, and ultimately resulting in a single recommended solution that can be given to the user. I tested this metric on a system where we changed the type of fuzzy logic used and compared the effect tuning of these systems had on the results.

The chapter demonstrated that a genetically optimised type-2 fuzzy logic system would produce better results than an un-optimised type-1 system. The results I obtained showed that the optimised type-2 system improved over the un-optimised type-1 system by 15.27% and also improved on the standard deviation of the results by almost 35%. Additionally, our results also showed the optimised type-2 system improved over the optimised type-1 systems by 6.34% and reduced the standard deviation by 24.88%.

The next chapter will discuss the proposed genetic algorithm-based approach for the simultaneous optimisation of workforce skill sets and team allocation. This next chapter focuses on the resource optimisation problem.

Chapter 7. A Genetic Algorithm Based Approach for the Optimisation of Workforce Skill Sets and Team Allocation

This chapter describes the work around skill optimisation, team optimisation and the approach for optimising both objectives together. This chapter aims at addressing the weaknesses in organisational design related to individuals and resource, as opposed to the geographical or management hierarchy addressed in previous chapters.

7.1 Initial Workforce Skill Set Optimisation System

Initially, the system created to optimise the workforce skill sets is primarily a real-valued genetic algorithm (RVGA). The genes in each of the solutions represent an engineer ID. The solution length (number of genes) is related to the number of upskills, where an upskill in the next logical skill set for any given engineer.

The next logical skill set is an important aspect that should have already been decided based on the type of engineers an organisation has. These next logical skill sets are designed to build upon the skill set the engineer already has. So, for example, if an engineer already has the skills of server installation the next logical skill to give this engineer might be server repair rather than air conditioning installation. The next logical skill set may also be tailored by technical managers who see engineers have an aptitude (or ineptitude) for a particular type of task.

As with the previous chapters' method for evaluating good organisational design decisions, this system also uses a daily simulation. This simulation is able to estimate the coverage, travel and utilisation values for any given solution, i.e. the team and their proposed skills including any new skills. This is extremely important as the values given from this simulation are fed into the fitness function.

The crucial variable for this problem, with respect to the simulation's task assignment, is the skill compatibility of an engineer to a task. From the GA we get the ID's of engineers to be upskilled, so if the simulation comes to one of these engineers, their skill set will be different and will contain more skills than if they were not in their upskilled state.

As a result, the engineer has more tasks to choose from, and their route may be different as a result. The way this simulation system is designed means that utilisation and coverage should always increase regardless of the skill configurations from solution to solution. This is because once the N number of upskilled engineers have been chosen, the order in which engineers are selected from the list to simulated will change. The list will choose the engineers with the least amount of skills first and leave the engineers with the most amount of skills last. This means that there won't be a situation where an engineer is chosen to be upskilled and are then given some of the tasks that a lower skilled engineer could have done.

In this situation utilisation is likely to be reduced as the low-skilled engineer has fewer tasks to choose from, meaning either travelling more to find work (reducing utilisation) or not matching with enough compatible tasks to fill all their available hours.

However, because the upskilled engineer will be further down the list, the minimum the engineer will do is exactly the same as if he/she were not upskilled. This ultimately means that this engineer is a poor choice to spend time and money on training. This solution is then more likely to be lost as the GA evolves.

7.2 Simultaneous Optimisation of Skill Sets and Teams

As described, the proposed optimisation system is a real-valued genetic algorithm based solution. The genes in each of the solutions represent an engineer ID. The solution length (number of genes) is related to the number of upskills, where an upskill is the next logical skill set for any given engineer. Once a team has more skills available, the team dynamic will change, and the members of the team may need changing to for the most optimal resource set-up. This would require team members moving across to neighbouring teams. However, just moving engineers may be sufficient. The benefit of this option is that it has no cost attached. Choosing the right people is crucial because there is a knock-on effect as to how this will the distribution of tasks to the remaining engineers.

Figure 7.1 shows an example of the real-valued chromosome, where an ID of an engineer is stored within each gene. This then tells the simulation that this set of engineers needs to use their upskilled skill set. It can also be used to decide which engineers to move to a different team.

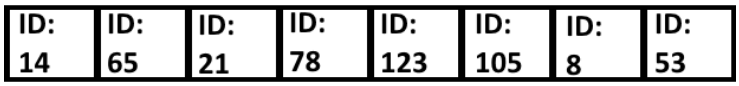

Figure 7.1: Upskilling Chromosome

The reason why we do not use a binary valued GA here is that each gene would have to represent an engineer that could be upskilled. The GA would then switch on/off engineers to be trained, but this would be uncontrolled. The GA could select any number of engineers to train and not optimise for the number we have specified. In most situations, the binary GA will switch on all engineers to be trained as this gives the most benefit. However, this is not practical from a business point of view for several reasons, not least the cost of training all engineers, as well as the opportunity cost of the lost time while the engineers are on training courses.

One problem with the RVGA that we have controlled for is the chromosome containing duplicates of the same engineer ID. As if there is a duplicate the GA will then give a result of one engineer less than we wanted. This solution will be penalised and given a zero-fitness value, as the solution does not meet the optimisation criteria.

To simplify the optimisation, not all engineers are eligible to be upkilled. Either they are at the maximum level of their skill path, or they do not yet have enough experience to be given another higher-level skill. These engineers will be filtered out to avoid redundant selections by the GA.

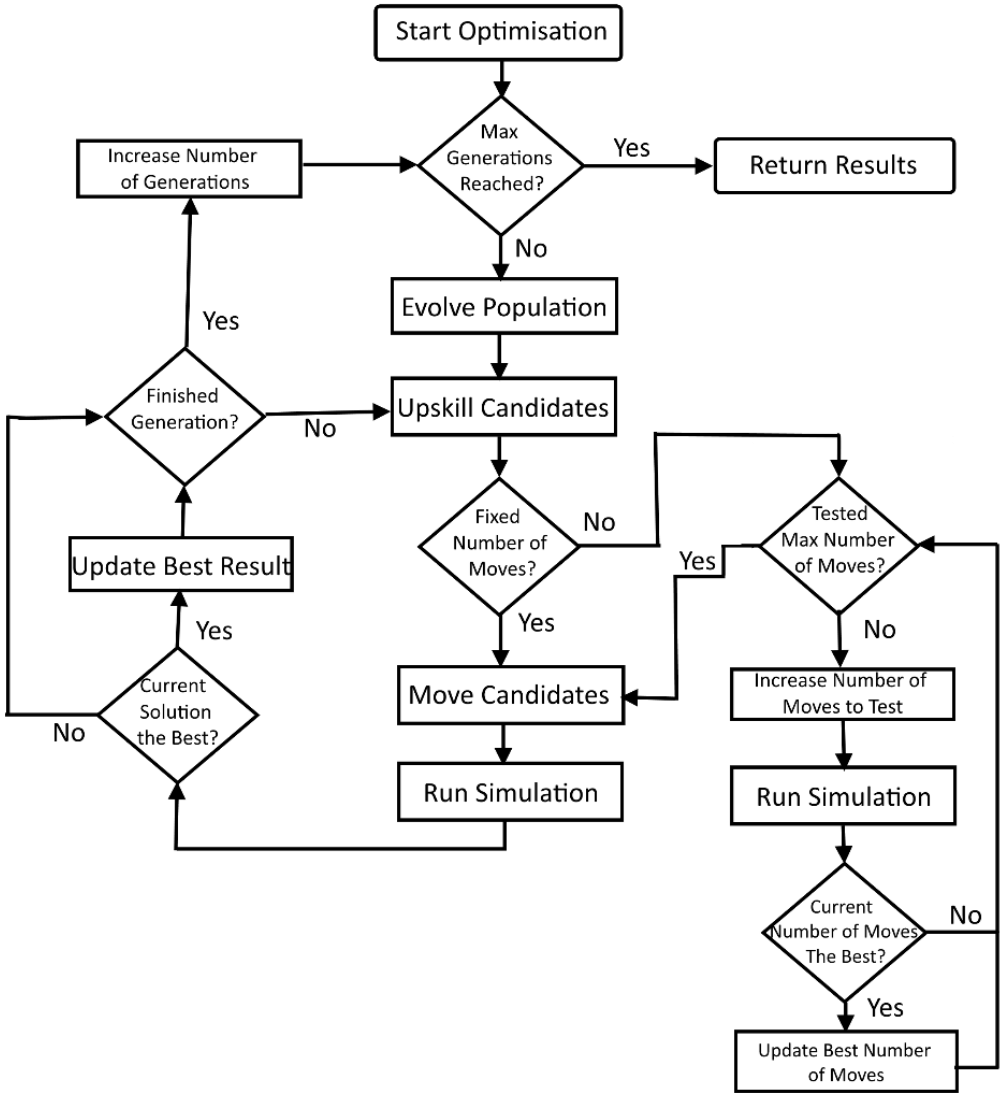

Figure 7.2: Solution creation and evaluation

During the solution evaluation section of the GA, shown in Figure 7.2, each solution will take the workforce to be optimised and give engineers (that have been selected by the genes) their next set of skills. This is the upskill candidates step. The daily simulation will run at the end of this step, and the effect of the upskills will be measured. After the upskilling, we have two options for moving engineers. We either set a fixed number of engineers to move (N) or try to evaluate how many would be the best to move.

If we have a fixed number, we move onto the next step and move N number of engineers. If we want the system to decide how many engineers to move, we start at 0 moves and evaluate the effect of increasing the number of moves up to the maximum number we want to test for. As a result, N number of moves will be equal to the number of moves that gave the best result.

Once we have determined N, we select N number of engineers to move to their closest alternative team. The alternative team is the one that is geographically closest to them. We do not want engineers to travel too far to their first task. The engineers selected here are those deemed least utilised, based on the simulation.

The system then runs the simulation again, once the teams have been altered, and the results of this second simulation are put into the fitness function to score the solution. This way the selection of the engineers to be upskilled will affect the engineers that will move teams, with the hope that both aspects will be considered during the optimisation process, Thus, producing better results than upskilling or moving engineers in two separate optimisation systems.

7.3 Real-World Background

The developed system has been deployed as part of BT's iPatch software. Initially, a business problem with resource management was highlighted, so I looked at how we could develop the iPatch tool to help solve this problem using similar techniques we have had success with before, with regards to the geographic optimisation techniques described in Chapters 6 and 7. The

resource optimisation problem was originally proposed by BT's Field Engineering Division where they wanted to know which engineers would give the greatest benefit, after they were trained in more skills. Although they were the primary stakeholder, one of the focuses was to keep the solution to the problem generic, so that it could be applied to other areas of the business. We then developed the GA to select any given number of engineers, then simulated the effect they would have with new skills.

We had to model the problem well to give a realistic view of the effect of training the selected engineers. The model to evaluate any of the proposed solutions involved simulating an average day's work. This began with setting up the engineers to be as close to reality as possible. The engineers were grouped into their current teams, placed at their known starting location, given the skills they currently have listed. BT provided this data.

The simulation then involved allocating the closest tasks to the engineers based on the skills they have (and how much time they had left for the day). Further feedback from the stakeholders led us to reorder the task allocation so those with few skills would be allocated tasks first. A greedy logic was then implemented here so that the list will choose the engineers with the least amount of skills first and leave the engineers with the most amount of skill last. This meant that highly skilled engineers were not taking jobs from the lower skilled engineers. If this happened, those engineers would be poorly utilised, and the higher skilled tasks would not be completed because the relevant engineers would be doing something else.

The simulation was then run on areas and teams where the objective values were roughly known so that the simulation results could compare against these. The comparison of the real coverage, travel and utilisation were close enough that the stakeholders were happy with the simulation as a means of testing solutions to the resource problem.

The focus between the stakeholders and us after this point was to discuss how the optimisation of upskilling was performing and if the suggestions were logical. After the upskills were seen to be logical another problem was highlighted, with under-resourced areas and underutilised resources in neighbouring areas. The solution here would be to move engineers between teams, but the most optimal solution to this problem was not known. Thus, we set out the discussed experiments to investigate this issue. The results presented gave the stakeholders confidence in the best methods presented for getting the most out of each engineer.

7.4 Experiments and Results

7.4.1 Workforce Skill Optimisation

In the data there is a list of engineers and their current skill sets and the task data that would be presented to them on an average day. This enables us to simulate the overall utilisation of the engineers. This initial utilisation value gives us a base value to compare our optimisation results. This is important because this initial utilisation value has been created from the current system of choosing engineers to upskill, i.e. by managers picking who they think is suitable for more training. The original results for the area being tested shown in Table 7-1.

Coverage (%)	Travel per Engineer (km)	Average Utilisation (%)
90.20	24.84	77.03

Table 7-1 Benchmark Results for Resource Optimisation

Our first set of experiments aim to tune the GA. These experiments test whether Tournament Selection or Roulette Selection is better for the problem. It also tests if a crossover value of 0.4 or 0.2 is better for this problem. Tables Table 7-2 to Table 7-5 outline these results. The following results are for five upskills. This means the system will try to pick the five best possible candidates to be trained to their next logical skill set.

	Coverage (%)	Travel (km)	Utilisation (%)
	94.12	25.61	80.38
	94.01	26.02	80.29
	94.06	25.48	80.33
	93.94	25.23	80.23
	94.12	25.10	80.38
Average	**94.05**	**25.49**	**80.32**

Table 7-2 Tournament Selection with Crossover of 0.4

	Coverage (%)	Travel (km)	Utilisation (%)
	94.01	25.61	80.29
	94.01	25.88	80.29
	94.12	25.48	80.38
	93.94	25.23	80.23
	94.05	25.57	80.32
Average	**94.03**	**25.55**	**80.30**

Table 7-3 Tournament Selection with Crossover of 0.2

	Coverage (%)	Travel (km)	Utilisation (%)
	93.90	26.07	80.20
	93.90	25.75	80.20
	93.63	25.70	79.97
	94.01	25.63	80.29
	93.73	25.64	80.05
Average	**93.83**	**25.76**	**80.14**

Table 7-4 Roulette Selection with Crossover of 0.4

	Coverage (%)	Travel (km)	Utilisation (%)
	93.44	25.43	79.80
	93.42	25.62	79.79
	93.96	25.53	80.25
	93.90	26.16	80.20
	93.76	25.40	80.08
Average	**93.70**	**25.63**	**80.02**

Table 7-5 Roulette Selection with Crossover of 0.2

From the results in Table 7-2 to Table 7-5 there is no statistical significance between a crossover rate of 0.2 and 0.4 for either method. However, for the coverage and utilisation objectives, Tournament selection performed statistically significantly better. Taking the 0.4 crossover rates results as an example (Table 7-2 and Table 7-4), the one-way ANOVA P-Values are 0.022, 0.171 and 0.023 for Coverage, Travel and Utilisation respectively. The travel value is difficult to assess from a P-Value perspective because travel should increase the better the result, so a P-Value comparison would not be able to tell if the values weren't significantly different because of a bad result, or a good result that required a bit higher travelling. On this point, Tournament Selection also produced a lower average travel rate than Roulette Selection. As mentioned it is typical for travel to increase the more tasks that are covered. This increase in coverage and reduced increase in travel explains the overall increase in utilisation. Based on these results Tournament Selection is the selection method we will proceed with. The crossover rate will be 0.4, based on this being our default value and 0.2 makes no difference.

Our second set of results focus on the number of engineers to be upskilled vs the benefit from the upskill. Table 7-6 and Table 7-7 add to the results we already have from Table 7-2 as the following results used Tournament selection with a crossover rate of 0.4.

Table 8-2 gives us the results for 5 upskill, Table 8-6 gives the results for 10 upskills and table 8-7 gives the results for 15 upskills.

	Coverage (%)	Travel (km)	Utilisation (%)
	95.19	26.33	81.30
	95.02	25.72	81.16
	95.19	26.21	81.30
	95.00	25.62	81.13
	94.94	26.64	81.09
Average	**95.07**	**26.10**	**81.20**

Table 7-6 Optimisation with 10 Upskills

	Coverage (%)	Travel (km)	Utilisation (%)
	95.33	25.86	81.42
	95.28	26.47	81.37
	95.09	25.52	81.21
	95.33	25.92	81.42
	95.27	26.39	81.36
Average	**95.26**	**26.03**	**81.35**

Table 7-7 Optimisation with 15 upskills

The maximum number of engineers who can be upskilled, for the area the experiments are being run on, is 107 out of 141. The remaining 34 engineers already hold the maximum amount of skills available to them. The results for the maximum number of possible upskills are shown in Table 7-8.

Coverage (%)	Travel per Engineer (km)	Average Utilisation (%)
95.36	24.75	81.45

Table 7-8 Maximum Number of Upskills for Test Area

Given that we now have the original results, results for 5, 10, 15 and maximum upskills we can plot them to see the level of diminishing returns for each upskill. This is important as the number of upskills directly correlates to training costs. Figure 7.3, Figure 7.4 and Figure 7.5 show the graphs for the number of engineer upskilling vs the coverage benefit, utilisation benefit and travel cost respectively.

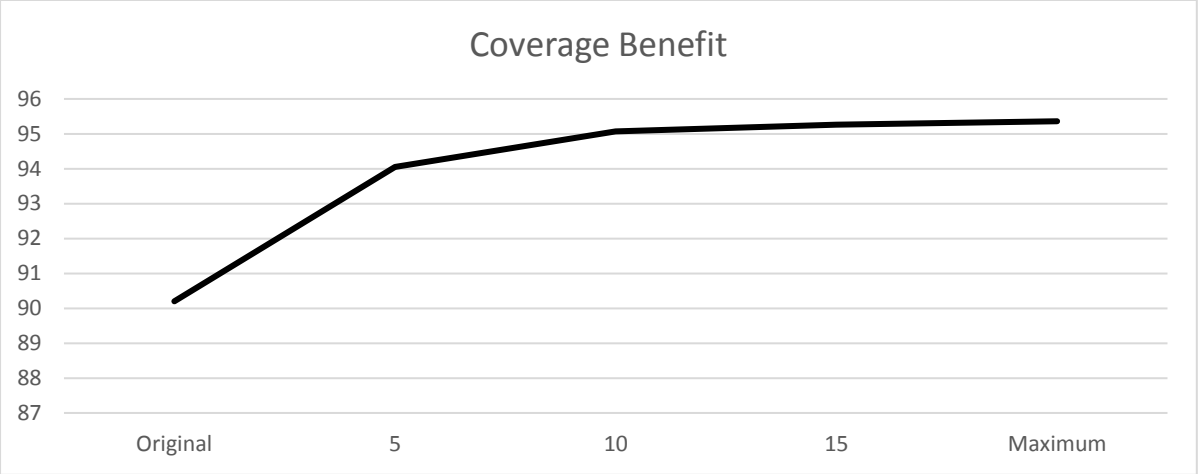

Figure 7.3: Coverage Benefit

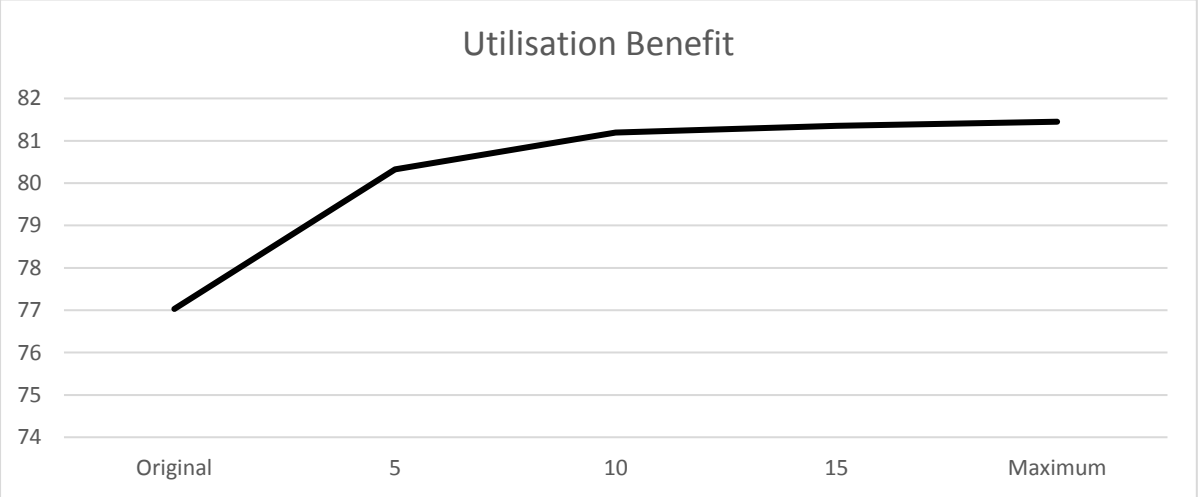

Figure 7.4: Utilisation Benefit

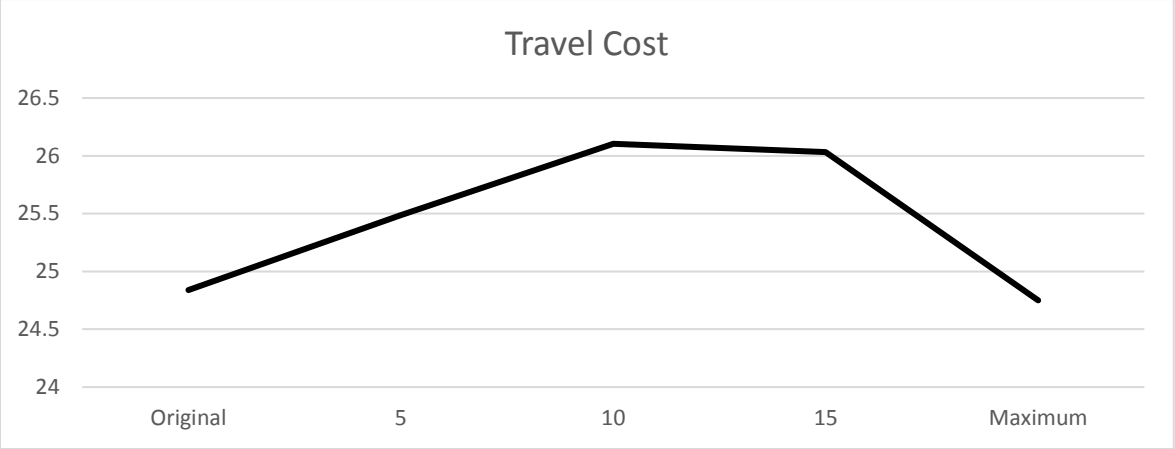

Figure 7.5: Travel Cost

Figure 7.3 and Figure 7.4 show that the most benefit per upskill is gained within the first five. After this, the benefit of both coverage and utilisation is greatly reduced. The benefit to coverage and utilisation becomes almost negligible after 15 upskills.

Figure 7.5 shows the average travel per engineer, which for the first 10 upskills increases linearly. However, after this point travel starts to be reduced with a significant drop at maximum upskills. This is likely because engineers have many more tasks to choose from that are closer to their current location, allowing them to always choose the closest task.

From this, we can say the only significant benefit gained after ten upskills comes from the reduction in travel.

If we compare the genetic algorithm based system with the current manual system, (where managers choose engineers they think are suitable for more training, results shown in Table 8-1), the results show that using this system to select employees for training has a 4.27% increase in overall employee utilisation, with only 3.52% of the workforce being trained (5 out of 141 engineers in the workforce). It also shows that there is a 5.41% increase in overall engineer utilisation when 7.04% is selected to be trained (10 out of 141 engineers in the workforce). This shows that the first few employees to be selected for training can produce the most benefit so selecting the right people is crucial and hence the proposed use of genetic algorithms for

this problem should be used. At this point, there is a potentially an exponential level of diminishing returns on employee utilisation

7.4.2 Simulaneous Optimisation of Skills and Teams

In the following experiments, which investigate the impact moving engineers has on the optimisation of the teams, we selected a region to optimise. This region contained eight sub-regions. Each sub-region contains patches. The teams are allocated to the patches, and any team reorganisation at the sub-region level involves moving engineers between the patches. These experiments build on from the experiments in the previous section and used the system laid out in Figure 7-2 of section 7.2.

Table 7-9 to Table 7-16 shows the optimisation results for each sub-region. The tables show the original results from the current teams with their current skill sets. Then each column shows the results from a different experiment with the aim of improving in the three objectives. The results shown from these experiments are the average of five runs of each of the experiments. The full results for the simultaneous optimisation methods can be found in our hypervolume analysis, section 7.5.

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	89.47	89.47	91.92	91.95	92.24	92.34
Travel	9.15	9.15	8.07	8.29	8.12	8.09
Utilisation	80.69	80.69	82.90	82.93	83.20	83.29

Table 7-9 Resource Optimisation Results for Sub-Region 1

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	94.18	94.96	94.18	94.18	94.82	94.18
Travel	7.28	7.32	5.87	5.92	5.93	5.88
Utilisation	84.30	84.99	84.30	84.30	84.87	84.30

Table 7-10 Resource Optimisation Results for Sub-Region 2

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	86.10	86.89	88.24	87.80	89.70	88.34
Travel	5.81	6.24	4.96	5.59	6.32	5.33
Utilisation	68.80	69.43	70.51	70.16	71.68	70.59

Table 7-11 Resource Optimisation Results for Sub-Region 3

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Moves & Upskills*	*Upskills Dynamic Moves*
Coverage	95.52	95.52	95.57	95.57	95.59	95.59
Travel	8.15	8.15	6.85	7.07	6.83	6.84
Utilisation	81.39	81.39	81.44	81.44	81.46	81.46

Table 7-12 Resource Optimisation Results for Sub-Region 4

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	89.84	89.84	90.06	90.06	89.79	90.06
Travel	6.66	6.66	6.52	6.61	6.90	6.54
Utilisation	80.57	80.57	80.76	80.76	80.51	80.76

Table 7-13 Resource Optimisation Results for Sub-Region 5

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	89.36	89.36	89.20	89.14	89.17	89.11
Travel	9.06	9.06	7.73	7.78	7.68	7.69
Utilisation	88.87	88.87	88.71	88.65	88.68	88.62

Table 7-14 Resource Optimisation Results for Sub-Region 6

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	81.56	82.30	81.87	82.08	84.47	82.06
Travel	6.27	6.29	5.54	5.59	5.66	5.47
Utilisation	75.86	76.55	76.15	76.35	78.57	76.30

Table 7-15 Resource Optimisation Results for Sub-Region 7

Objective	*Original*	*Moves Only*	*Upskill Only*	*Upskill then Move*	*Upskills & Move*	*Upskills Dynamic Moves*
Coverage	93.32	93.32	93.33	93.33	93.16	93.34
Travel	11.65	12.00	10.54	10.67	10.71	10.53
Utilisation	81.57	81.57	81.58	81.58	81.43	81.59

Table 7-16 Resource Optimisation Results for Sub-Region 8

For the purpose of our experiments, we keep the number of upskills at 10; this is based on our previous results from Section 7.4.1. The number of fixed moves we set at five. Our GA settings are 0.4 for crossover, 0.05 for mutation, a population of 100 and max generations is set at 30 (Convergence of results with these settings is easily obtained by 30 generations).

Fixed moves without any upskilling is our first experiment. We want to know the effect of just moving the five least utilised workers in each sub-region.

From Table 7-9 to Table 7-16 we can see that just by moving the least utilised we will increase the coverage in three of the eight sub-regions with the remaining five having no effect. Also, travel increases as a result of moving engineers in four of the eight sub-regions.

This is most likely for two reasons. Firstly, the move only process is not part of the GA system, so it does not use the fitness function and is not constrained by travel. Secondly, it simply looks at the least utilised engineers and assigns them to a different team. Thus, the engineer will then have to travel further to their new patch, as they will now be assigned to one that is further away.

Our second set of experiments look at just running the upskilling optimisation (the same process completed in section 7.4.1). With ten engineers throughout the sub-region being selected for training. This has the effect of increasing coverage in six of the eight areas. Of these six areas, four of them performed better in coverage than just moving engineers. In sub-region 7 (Table 7-15), *moving only* performed better than *upskill only*. However, both improved on the original.

Our third set of experiments looked at a step process in which we first upskill ten engineers via the GA process; then once the GA process is complete, we move the five least utilised workers based on the new upskills. The method produces suboptimal results as the GA has attempted

to find the best solution for upskilling engineers, then the solution is impacted (usually made worse) by moving engineers between teams.

Although this method produced coverage result that were better than just moving the engineers (sub-regions 1, 3, 4, 5 and 8) in the majority of cases, 75%, this method performed worse than upskill only when travel was also taken into account (sub-regions 2, 3, 4, 5, 6 and 8). This led us to the last two experiments.

The fourth set of experiments looked at combining the upskilling optimisation with moving the least utilised engineers within the GA process. This experiment now has the advantage of applying the fitness function to new team configurations. This process improves in coverage in five of the eight sub-regions and six of the eight in travel.

The final set of experiments looks at allowing the system to run simulation tests to alter the number of moves to find the best number of engineers to move within each sub-region. The system could choose up to 10 engineers to move. This resulted in six of the sub-regions being improved in coverage, with only sub-region 6 (Table 7-14) performing worse. All sub-regions have improved travel distances when compared to upskill *then* move. The reason for sub-region 6 performing worse in coverage could be because of the fitness function. The reduction in travel of 15.12% may be why the solutions produced for sub-region 6 has a small 0.28% reduction in coverage, given the equal weighting of these objectives.

In all cases for the combined optimisation methods, either fixed moves or dynamic moves outperformed the step process of upskilling *then* moving engineers. With a fixed number of moves being the process that outperforms the step process most often, in 75% of cases (Sub-Regions: 1, 2, 3, 4, 6 and 7). This shows us that combining these methods produces better results.

Table 7-17 gives an overview of the coverage improvements with the different experiments. On average for the eight sub-regions, we can see that only moving engineers gives a 0.29% improvement. Only upskilling gives a 0.63% improvement. Upskilling *then* moving give a 0.58% improvement (smaller than just upskilling). Moving and upskilling simultaneously produces the best improvement, with the dynamic number of moves giving a 0.71% improvement and the fixed number of moves giving a 1.20% improvement. When this is applied to the 7571 hours' worth of work across the region, this 1.20% improvement is equal to 90.85 hours work per day.

Sub-Region	Moves Only	Upskill Only	Upskill then Move	Upskills & Move	Upskills & Dynamic Moves
1	0.00%	2.45%	2.48%	2.77%	2.87%
2	0.78%	0.00%	0.00%	0.64%	0.00%
3	0.79%	2.14%	1.70%	3.60%	2.24%
4	0.00%	0.05%	0.05%	0.07%	0.07%
5	0.00%	0.22%	0.22%	-0.05%	0.22%
6	0.00%	-0.16%	-0.25%	-0.19%	-0.25%
7	0.74%	0.31%	0.52%	2.91%	0.50%
8	0.00%	0.01%	0.01%	-0.16%	0.02%
AVG.	**0.29%**	**0.63%**	**0.59%**	**1.20%**	**0.71%**

Table 7-17 Coverage Results Evaluation from Resource Optimisation Sub-Regions

Table 7-18 gives an overview of the travel improvements. If coverage increases, we expect travel to increase also. Because the engineers are travelling and completing more tasks. So, these are directly conflicting objectives. If the travel is also being reduced at the same time as increased coverage, then the task allocation for the engineers has become much more efficient.

Table 7-18 also shows the extra kilometres travelled per engineer as a result of the optimisation attempts. Moving only will increase travel on average for the sub-regions, this is expected given we are forcing the engineers to travel to a patch further away. Upskilling only produces the most travel benefit. This is logical as the same teams are given a wider selection of jobs to choose from, meaning they are more likely to choose jobs that are closer to them. Upskill *then* move produces a good reduction in travel, but again not as good as upskilling only. A fixed number of moves and upskills, simultaneously, increases travel the most on average. This makes sense as it is also the optimisation technique that increases coverage the most.

Sub-Region	Moves Only	Upskill Only	Upskill then Move	Upskills & Move	Upskills & Dynamic Moves
1	0.00km	-1.08km	-0.86km	0.05km	-0.03km
2	0.04km	-1.45km	-1.36km	0.06km	-0.05km
3	0.43km	-1.28km	-0.22km	1.36km	-0.99km
4	0.00km	-1.30km	-1.08km	-0.01km	0.00km
5	0.00km	-0.14km	-0.05km	0.38km	-0.36km
6	0.00km	-1.33km	-1.37km	-0.05km	0.01km
7	0.02km	-0.75km	-0.68km	0.12km	-0.18km
8	0.35km	-1.46km	-0.98km	0.17km	-0.18km
AVG.	**0.11km**	**-1.10km**	**-0.82km**	**0.26km**	**-0.22km**

Table 7-18 Travel Results Evaluation from Resource Optimisation Sub-Regions

Upskilling with a dynamic number of moves reduces travel despite having the second highest coverage increase. The difference between fixed moves and dynamic moves is 0.48km per engineer. Which is significant as in this region there are 1481 engineers. Resulting in a difference of about 710km per day.

Given this, for the simultaneous optimisation techniques, fixed moves increase travel 385km per day, while dynamic moves give a reduction of 325km per day. If the regional manager is looking to reduce fuel consumption cost and CO_2 emissions, using the dynamic system looks far more attractive, because not only will fuel costs be reduced, more work will be completed.

Alternatively, if the regional manager has the goal of completing more tasks, which results in increased customer satisfaction and a reduced reliance on contractor work, then the fixed option looks better, just from a maximise job completion perspective.

Whichever of these options is chosen, it will be with the simultaneous optimisation, as these combined optimisation techniques outperform either move only, upskill only and upskill *then* move methods in coverage. If travel is of concern, the dynamic moves option may be the best, given that it is the second best at increasing coverage but also has the benefit of reducing travel.

7.4.3 Hypervolume Analysis

In section 7.4.2 we established that the simultaneous optimisation methods, either a fixed number of moves or a dynamic number of moves, were the strongest options depending on the goals of the managers. This was based on our simple analysis of the average improvement in the objective values for each area. To definitively prove if these methods are better, we will compare the hypervolumes created by the five runs of these methods in each area. Then statistical analysis is performed on these hypervolumes.

Table 7-19 to Table 7-26 show the results of the hypervolumes. In many cases, the GA produced the same results in either method. This helps to prove the GA is producing strong and consistent results. This also helps the real-world users to have more confidence in the results. The Root Mean Square (RMS) of all hypervolumes against their respective original solutions is 0.025. Thus we present the hypervolumes to two digits of precision.

Sub-Region 1	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	89.47	9.15	80.69	0.39
Upskills & Fixed Move	91.79	8.08	82.78	
	92.63	8.12	83.54	
	92.45	8.00	83.39	0.47
	91.97	8.11	82.95	
	92.38	8.30	83.32	
Upskills & Dynamic Moves	92.83	8.17	83.72	
	92.80	8.06	83.70	
	91.80	8.07	82.80	0.46
	91.98	8.06	82.96	
	92.30	8.08	83.25	

Table 7-19 Resource Optimisation Hypervolume Analysis for Sub-Region 1

Sub-Region 2	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	94.18	7.28	84.30	0.52
Upskills & Fixed Move	94.96	5.90	84.99	
	94.69	6.00	84.75	
	94.62	6.01	84.69	0.61
	94.96	5.92	84.99	
	94.96	5.91	84.99	
Upskills & Dynamic Moves	94.18	5.91	84.30	
	94.18	5.88	84.30	
	94.18	5.90	84.30	0.61
	94.18	5.84	84.30	
	94.18	5.86	84.30	

Table 7-20 Resource Optimisation Hypervolume Analysis for Sub-Region 2

Sub-Region 3	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	86.10	5.81	68.80	0.61
Upskills & Fixed Move	90.38	6.30	72.22	
	90.47	6.36	72.29	
	86.10	5.75	68.80	0.62
	90.79	6.28	72.55	
	90.78	6.90	72.54	
Upskills & Dynamic Moves	87.70	5.24	70.07	
	88.58	5.39	70.78	
	88.26	5.27	70.52	0.65
	88.58	5.43	70.78	
	88.58	5.32	70.78	

Table 7-21 Resource Optimisation Hypervolume Analysis for Sub-Region 3

Sub-Region 4	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	95.52	8.15	81.39	0.46
Upskills & Fixed Move	95.10	7.03	81.03	
	95.07	6.99	81.01	
	95.12	7.02	81.05	0.53
	95.07	7.16	81.01	
	95.42	7.03	81.31	
Upskills & Dynamic Moves	95.59	6.84	81.46	
	95.59	6.83	81.46	
	95.59	6.84	81.46	0.55
	95.59	6.85	81.46	
	95.59	6.82	81.46	

Table 7-22 Resource Optimisation Hypervolume Analysis for Sub-Region 4

Sub-Region 5	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	89.84	6.66	80.57	0.56
Upskills & Fixed Move	89.84	6.66	80.57	
	89.93	7.26	80.64	
	89.71	7.27	80.44	0.56
	89.84	6.88	80.57	
	90.10	6.81	80.79	
Upskills & Dynamic Moves	90.06	6.50	80.76	
	89.94	6.51	80.65	
	90.10	6.57	80.79	0.56
	90.10	6.57	80.79	
	90.10	6.57	80.79	

Table 7-23 Resource Optimisation Hypervolume Analysis for Sub-Region 5

Sub-Region 6	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	89.36	9.06	88.87	0.40
Upskills & Fixed Move	89.27	7.65	88.78	
	89.12	7.71	88.63	
	89.27	7.76	88.78	0.49
	89.27	7.73	88.78	
	89.27	7.65	88.78	
Upskills & Dynamic Moves	89.12	7.63	88.63	
	89.12	7.63	88.63	
	89.12	7.69	88.63	0.49
	89.12	7.69	88.63	
	89.08	7.82	88.59	

Table 7-24 Resource Optimisation Hypervolume Analysis for Sub-Region 6

Sub-Region 7	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	81.56	6.27	75.86	0.58
Upskills & Fixed Move	84.37	5.66	78.47	
	84.37	5.73	78.47	
	84.49	5.69	78.59	0.62
	83.75	5.71	77.90	
	84.37	5.75	78.47	
Upskills & Dynamic Moves	82.27	5.45	76.52	
	82.07	5.48	76.34	
	81.95	5.46	76.22	0.64
	81.95	5.47	76.22	
	82.07	5.51	76.22	

Table 7-25 Resource Optimisation Hypervolume Analysis for Sub-Region 7

Sub-Region 8	*Coverage*	*Travel*	*Utilisation*	*Hypervolume*
Original Solution	93.32	11.65	81.57	0.22
Upskills & Fixed Move	84.37	5.66	78.47	
	84.37	5.73	78.47	
	84.49	5.69	78.59	0.62
	83.75	5.71	77.90	
	84.37	5.75	78.47	
Upskills & Dynamic Moves	82.27	5.45	76.52	
	82.07	5.48	76.34	
	81.95	5.46	76.22	0.64
	81.95	5.47	76.22	
	82.07	5.51	76.22	

Table 7-26 Resource Optimisation Hypervolume Analysis for Sub-Region 8

What we can see from tables Table 7-19 to Table 7-26 is that in all cases, except sub-region 5, the hypervolume of the simultaneous optimisation methods are greater than the original solution's hypervolume. When comparing the fixed upskill & move method with the dynamic move method, the dynamic version usually outperforms the static method.

If we create three hypervolume sets for the original, fixed methods and dynamic methods, we can perform statistical analysis (specifically the Kruskal–Wallis test) and attain p-values for the comparisons.

If we compare the original hypervolume set and to the fixed moves hypervolume set we get a p-value of 0.074, which is good, but not below the 0.05 threshold to show significance. If we

compare the original hypervolume set with the dynamic moves hypervolume set, we get a p-value of 0.046, which is below the 0.05 thresholds and indicates that the results from this method are a statistically significant improvement over the original results.

7.5 Discussion

This chapter presented a real value GA system for engineer upskilling recommendations. The results showed that for this particular problem Tournament selection with a crossover probability of 0.4 performed better.

Once the system was tuned, the optimisation for the area found that the most benefits gained from the system were in the first five engineers who were upskilled. However, it also found that upskilling all possible engineers is the best way to reduce travel costs.

After this initial investigation, a real-value GA system for engineer upskilling and move recommendations was presented. The first experiment was to only move the least utilised engineers; the second was to only upskill ten engineers across the sub-region. The third was to combine both moves and upskilling sequentially. The final two experiments were to combine both the moves and the upskills in a simultaneous optimisation method, with either a fixed number of moves or a dynamic number of moves.

The results showed that combining team moves and engineer upskilling in the same optimisation process lead to an overall 1.20% increase in coverage across the region with the fixed moves option and a 0.71% increase with a dynamic number of moves. Both of these results produced better coverage than only moving engineers between teams, just upskilling the engineers or upskilling then moving the engineers in a sequential process.

Finally, the hypervolumes created by the results of the simultaneous optimisation methods were evaluated. From this it was seen that these algorithms outperformed the original solutions.

When performing the Kruskal–Wallis test to calculate a p-value we saw that the dynamic moves simultaneous optimisation method gave us a p-value of 0.046, below the threshold to show statistical significance. This test indicated that this method is clearly better than the method to create the original solutions (which was primarily manual).

The next chapter will discuss fuzzy dominance in real-world many-objective optimisation problems.

Chapter 8. Fuzzy Dominance Rules in Real-World Many-Objective Optimisation Problems.

In the previous chapters the proposed cloud-based type-2 fuzzy logic many-objective optimisation system has been developed and enhanced; additional workforce optimisation functionality has been introduced to the tool in the form of optimisation methods for workforce skill sets and team allocation.

However, at its core, the system is a multi-objective optimisation algorithm. A distance metric was introduced in Section 6.2 to help with the problem of solutions saturating the Pareto front. This method, while providing some benefit, doesn't address the main reason why Pareto based algorithms fail to effectively navigate a many-objective search space.

In this chapter, the problem with dominance and the Pareto front will be described. Then a solution to this problem will be introduced in the form of Fuzzy Dominance Rules.

8.1 Dominance in Many-Objective Problems

As mentioned in Section 3.5, Many-objective problems are described as those with four or more objectives [69] [70]. The more objectives there are, the more likely that the mentioned dominance rules will not be sufficient to distinguish between good solutions. Thus, the Pareto front will become saturated with solutions (potentially containing all solutions in the population) making it very difficult to choose parents in the selection stage of the GA.

The problem stems from the first rule; that no objective can be worse. Consider the results in Table 8-1. Table 8-1 shows five solutions to a problem that has five objectives, where each of the five objectives should be minimised.

Solution No.	Objective 1 (min)	Objective 2 (min)	Objective 3 (min)	Objective 4 (min)	Objective 5 (min)	Solutions Dominated
1	3.0	6.0	8.0	4.0	7.0	0
2	2.0	5.0	5.0	4.0	8.0	0
3	2.0	6.0	1.0	5.0	1.0	0
4	1.0	1.0	1.0	5.0	2.0	0
5	8.0	1.0	1.0	1.0	1.0	0

Table 8-1 Dominance in Many-Objective Problems: Example I

In Table 8-1, solution '4' does a very good job of minimising all objective, except objective 4. This objective has been sacrificed for all others. This is an expected outcome with conflicting objectives. The same could be said of solution 5. These are clearly two good solutions, however because of the rule stating no objective can be worse, these solutions fail to dominate the clearly weaker ones. Selection pressure (when selecting the parents for the next generation) does not consider the stronger solutions because of this; it has to rely on weaker or secondary selection pressures such as crowding distance, or the distance metric which has been proposed. The problem is exaggerated in Table 8-2.

Solution No.	Objective 1 (min)	Objective 2 (min)	Objective 3 (min)	Objective 4 (min)	Objective 5 (min)	Solutions Dominated
1	3.0	100.0	800.0	4.0	70.0	0
2	2.0	100.0	50.0	4.0	80.0	0
3	2.0	410.0	1.0	50.0	1.0	0
4	1.0	1.0	1.0	4.1	1.1	0
5	3.1	1.0	1.0	1.0	1.0	0

Table 8-2 Dominance in Many-Objective Problems: Example II

Table 8-2 shows another situation where we have five solutions that do not dominate each other. However, to any human solutions, 4 and 5 are clearly better. Solutions 1, 2 and 3 have

failed in the majority of the objectives, but under dominance, they are good candidates for selection stage in the GA.

To address this problem, we will use the proposed Fuzzy Dominance Rules (FDRs). This is the introduction of a fuzzy logic system in place of the standard dominance rule check. Each objective value is fuzzified and then compared. The membership functions for this FLS are proportional to the values being compared. For example, a 10% tolerance value on objective 4 when comparing solutions 2 and 4 from Table 8-2 would mean solution 4 could have a value of 4.4 and the condition of 'no objective worse' would instead be satisfied.

The need to make the dominance rule less strict by fuzzifying what it sees as 'Worse', 'Equal' or 'Better' can be illustrated in Table 8-3. In Table 4 we have allowed a tolerance of 10%, meaning when comparing X to Y, Y can be up to 10% of Xs value larger and still not be considered as worse. With this fuzzification of the rules, we can now see that solutions 4 and 5 dominate the 3 other solutions (solutions 1, 2 and 3).

Solution No	Objective 1 (min)	Objective 2 (min)	Objective 3 (min)	Objective 4 (min)	Objective 5 (min)	Solutions Dominated
1	3.0	100.0	800.0	4.0	70.0	0
2	2.9	100.0	50.0	4.0	80.0	0
3	2.9	410.0	1.0	50.0	1.0	0
4	1.0	1.0	1.0	4.1	1.1	3
5	3.1	1.0	1.0	1.0	1.0	3

Table 8-3 Dominance in Many-Objective Problems: Example III

The context of the objective values can determine the design of the membership functions. The most critical membership function in our system is what values can be considered equal. For example, most people would consider an outside temperature of 20 and 22 degrees Celsius 'the

same' i.e. they would not make any changes to their clothing or behaviour. However, people may consider 20 and 25 degrees different enough to change their behaviour. Similarly, if the price of coffee increases from £1.85 to £1.90 there may be little change in behaviour so that most people would view that as the 'same' price. However, if the price increased to £2.00, this could affect sales as the price change increase from 2.70% to 8.11%.

8.2 Proposed Fuzzy Dominance Rules

The proposed fuzzy dominance rules work like any traditional type-1 FLS where the inputs are the objective values of the solutions being compared. The FLS will then decide if the relative objective value is worse, equal or better for each objective for solution A and solution B.

Once the FLS has processed each objective, these outcomes will be used to work out dominance using the standard dominance rules described in Section 4.4.1, except now these rules are not comparing the raw crisp values for each objective. They will use the output of the FLS. Thus, the dominance rules become Fuzzy Dominance Rules (FDR).

There are three fuzzy sets representing the inputs and outputs as shown in Figure 8.1, Figure 8.2 and Figure 8.3. Figure 8.1 shows the input fuzzy set of solution A's objective values being compared to solution B's. In this example set, we allow a tolerance of 10%. So, if the objective value for A is 10.5, then B can have an objective value of between 9.45 and 11.55 and be seen as equal. In an example where B is 9.5, B still falls into the range of 'Equal' with a membership value of 0.048.

The reverse comparison is then made between B and A, shown in Figure 8.2. In this case, if the B value is 9.5 and the A value is 10.5. A will be seen as 'Better' to the degree of 1.0 and won't be seen as 'Equal' to any degree (as the maximum value for 'Equal' would be 10.45). This two-way validation strengthens the dominance decision. This is reflected in the A to B and B to A fuzzy sets.

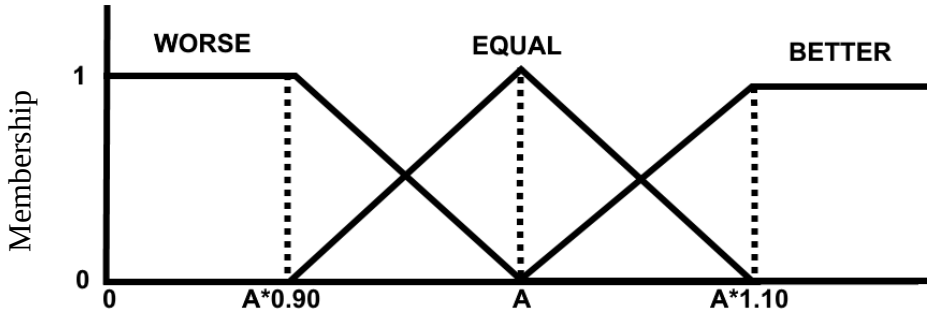

Figure 8.1: Fuzzy Set comparing A objectives to B objectives

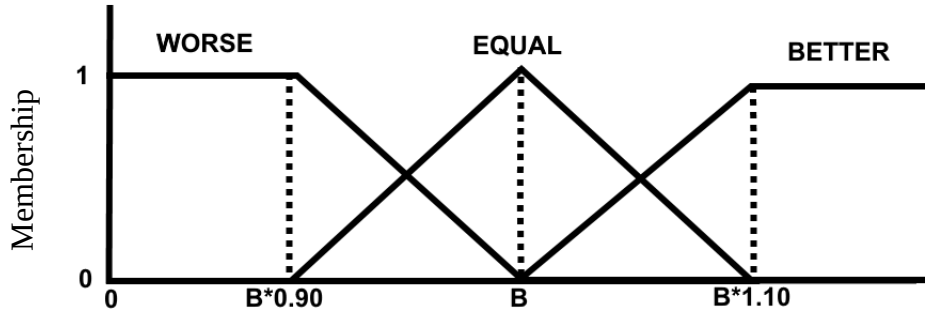

Figure 8.2: Fuzzy Set Comparing B Objectives to A Objectives

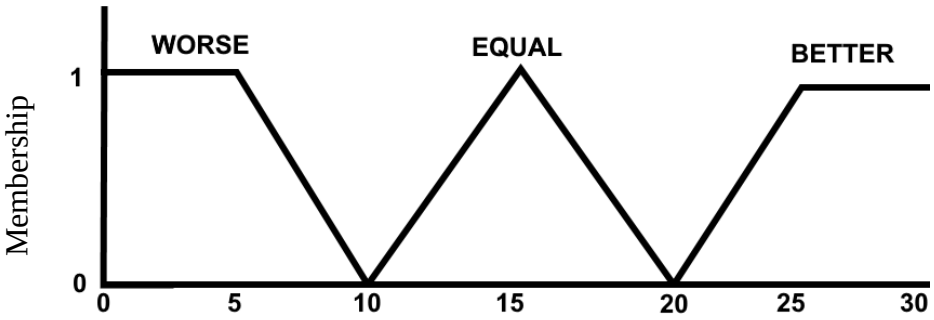

Figure 8.3: Output Fuzzy Set for Comparing Two Objective Values

The rules in this fuzzy system are given in Table 8-4. So, for the example where 'Equal and Better' would fire B is 'Worse', this is also the case for 'Worse and Better'. Meaning, of the two rules, that fire we are certain that B is worse than A. The rules here are also designed to accommodate non-uniform membership functions. The output set shown in Figure 8.3 will dictate the final outcome of the dominance. A crisp output will then be given using the centre of sets defuzzification. The output value is then used to determine dominance. If a crisp value of less than 10 is given, then B is worse. If the defuzzification gives a value of

between 10 and 20, then B and A are equal, and if the value if greater than 20, then B is better. These values are determined by the output fuzzy set.

Compare A to B	Compare B to A	Final Output Comparing A to B
Worse	Worse	**Equal**
Worse	Equal	**Worse**
Worse	Better	**Worse**
Equal	Worse	**Better**
Equal	Equal	**Equal**
Equal	Better	**Worse**
Better	Worse	**Better**
Better	Equal	**Better**
Better	Better	**Equal**

Table 8-4 Fuzzy Dominance Rule Base

8.3 Experiments and Results

8.3.1 Black Box Optimisation

Our experiments involve applying the NSGA-II algorithm to some many-objective problems then comparing the difference in the Pareto fronts against an NSGA-II algorithm that utilised the described FDR in place of the crisp domination evaluation described in section 3.4.1. To compare the Pareto fronts of the two algorithms the hypervolume, mentioned in section 3.6, will be used

The first set of experiments involves the Black Box Optimization Competition (BBComp) [90]. The BBComp allows competitors to test their optimisation algorithms on a number of black box problems, with the winning algorithm being the one that optimises the best in the most problems. There are single-objective and multi-objective tracks for this competition.

There are specific optimisation rules imposed, such as; competitors cannot run their algorithm on a problem more than once, and there is a budget for each problem. Where the budget is described as, the number of times the evaluation of a solution can be called. Meaning higher populations would lead to a reduced number of generations.

The first rule is a problem if enough runs of the same problem are to be collected, for the purpose of performing statistical analysis on the two algorithms. Fortunately, there is a 'Test Track' within the competition which can be run any number of times, so that track will be used. With regards to the budget, it will be divided equally between population and generations. For example, if the budget is 100, the population size will be set to 10, and the number of generations will be set to 10. This would lead to 100 solution evaluation calls. For multi or many-objective problems in the competition, the value returned at the end of each problem will be 1-hypervolume value. Smaller values are stronger Pareto fronts.

The system was run on 120 of the available problems in the multi-objective track; each problem was run 30 times for both the NSGA-II and the NSGA-II with FDR (NSGAIIFDR), leading to a total of 7,200 hypervolume values.

Within the first 50 problems, there wasn't any significant improvement. However, given the earlier problems have fewer problem dimensions (i.e. two dimensions for problems 0-49), this falls in line with what is expected, this is because the FDRs are designed to tackle the issues with many-objective optimisation, and two objectives do not fall within this. Instead, the remaining 70 problems will be analysed.

Table 8-5 shows that problems where NSGAIIFDR gives a statistically significant smaller average 1-hypervolume values. The Kruskal–Wallis test was performed on the two sets given for each problem to obtain the P-Value in column five. These two sets are the 30 runs without the FDR, and the 30 runs with the FDR for each problem, in columns three and four

respectively. Significance when comparing the hypervolume sets for the listed problems of the two algorithms is <0.05 (the threshold for significance). The Root Mean Square (RMS) for the P-Value is 0.038, thus these values are given to 3 decimal places.

Problem	Problem Dimensions	Average NSGA-II 1-Hypervolume	Average NSGAIIFDR 1-Hypervolume	P Value
50	4	0.58	0.57	0.047
52	4	0.81	0.80	0.042
59	4	0.85	0.85	0.037
63	4	0.79	0.79	0.038
73	4	0.96	0.96	0.042
81	4	0.95	0.95	0.044
84	4	0.58	0.57	0.048
91	4	0.95	0.95	0.017
116	5	0.84	0.84	0.035
119	5	0.87	0.86	0.007

Table 8-5 Results from BBCOMP Problems

There is a trend towards more significant improvements as the number of objectives increase in the problems, shown in Figure 8.4. The number in brackets in Figure 8.4 represents the number of dimensions in each problem. Problem 119 has a P value of just 0.0068 (or <1%). In total, there were ten statistically significant improvements from the remaining 70 problems. Leading to at least 14.29% of problems being improved when using NSGAIIFDR. Importantly this is just using a generic model, with no tuning on any of the problems. There has yet to be specifically optimised aspects for each problem.

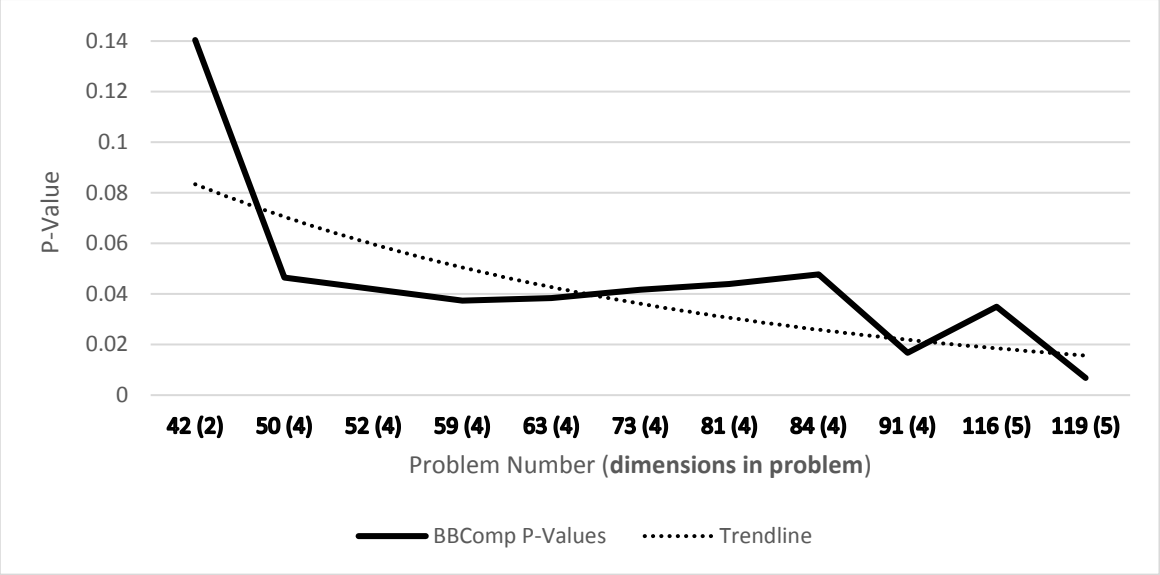

Figure 8.4: Value Plot and Trendline of BBComp Results

Some minor tuning of the algorithm for each test problem could also achieve greater significance, rather than having a general setup for all problems. This could also help a greater number of problems being improved to fall under the 0.05 significance threshold. As those slightly above the threshold have not been listed.

As the optimisation is blind and restricted on the budget, we can say with a degree of confidence that we can improve on the results given by NSGA-II by adding our proposed Fuzzy Dominance Rules.

8.3.2 Real-World implementation

To further test the hypothesis of whether implementing FDR to multi-objective algorithms help solve many-objective problems, and to validate its usefulness, the next step was to apply NSGAIIFDR to the described real-world many-objective problem.

There are a number of experiments that aim to show improvements to the modified NSGA-II (NSGAIIFDR) algorithm. The first experiment utilises the type-1 fuzzy versions of the PCFLS and TAFLS as outlined in Sections 5.1 and 5.2. The second experiment replaces the fuzzy

systems with type-2 FLSs. The third experiments utilise the type-2 FLSs but have a short optimisation with a GA; this is the genetically optimised system from Chapter 6.

Coverage (%)	Travel (km)	Utilisation (%)	Area Balance (Hours)	Team Balance (people)
76.12	26.50	68.15	354.65	71

Table 8-6 Benchmark Results for Fuzzy Dominance Rules

The experiments start by choosing a single geographical area to optimise. The current design is evaluated to get our benchmark objective values for this area. These values can be found in Table 8-6.

Our first aim is to show that the introduction of the FLSs improves our system, like the experiments from Chapter 6. However, only three objectives will be chosen so the benefit of these FLS can be separated from the many-objective environment. Thus, the optimisation is not hindered by the problems associated with many-objective optimisation and shows the distinct contribution the FLSs have on the ability to improve the results. Hence, the results from Chapter 6 cannot be compared as they include five objectives.

For the first four sets of results only Coverage, Travel and Patch Balancing will be used as objectives. Each experiment will run the optimisation 30 times and will be given 30 unique seed values each time. Each experiment will use the same 30 unique seed values to reduce the elements of randomness further. Each run will give a Pareto front where we will use the discussed hypervolume metric, from Section 4.6, to evaluate the Pareto fronts. Each hypervolume value will be given to two decimal places. The reference points for the three objectives will be 0, 100 and 1000 for Coverage, Travel and Patch Balance respectively.

Maximisation objectives are multiplied by -1 to make sure the hypervolume forms a convex shape. This is so that any improvement in any objective value will cause a point on the Pareto front to trend in the same direction in the objective space. i.e. if travel reduces from 10 to 5,

this is an improvement, so if we multiply coverage by -1, an improvement of 5 to 10 will be shown in the hypervolume shape as an improvement from -5 to -10.

All the hypervolumes from the experiments are shown in the Hypervolume Summary Table, Table 8-7. Table 8-7 shows the hypervolume set for the NSGA-II algorithm as **N**, the introduction of type-1 fuzzy systems gives the hypervolume set noted by **T1**. The upgrade to type-2 systems gives the hypervolume set noted by **T2**. Finally, the hypervolume set given by the NSGA-II algorithm with genetically optimised type-2 fuzzy logic systems is denoted by **OT2**. We can plot a Pareto front result from each of the hypervolume sets for a visual comparison. Figure 8.5 to Figure 8.7 show different perspectives of the same four Pareto fronts

											Avg.
N	0.63	0.73	0.69	0.62	0.65	0.65	0.68	0.71	0.68	0.63	0.67
	0.68	0.67	0.69	0.67	0.74	0.70	0.71	0.64	0.63	0.64	
	0.67	0.65	0.64	0.70	0.71	0.69	0.62	0.70	0.71	0.72	
T1	0.72	0.67	0.67	0.70	0.72	0.59	0.67	0.67	0.69	0.66	0.68
	0.69	0.70	0.67	0.69	0.68	0.66	0.69	0.67	0.74	0.66	
	0.65	0.67	0.69	0.71	0.63	0.67	0.68	0.65	0.66	0.69	
T2	0.62	0.66	0.70	0.71	0.71	0.71	0.69	0.70	0.71	0.71	0.68
	0.64	0.66	0.66	0.67	0.65	0.75	0.70	0.68	0.68	0.66	
	0.65	0.65	0.67	0.69	0.70	0.65	0.64	0.69	0.69	0.70	
OT2	0.65	0.72	0.67	0.68	0.66	0.72	0.73	0.75	0.72	0.68	0.70
	0.67	0.71	0.74	0.70	0.73	0.74	0.72	0.73	0.74	0.69	
	0.67	0.66	0.75	0.70	0.69	0.68	0.74	0.70	0.67	0.70	

Table 8-7 Hypervolume Summary Table for Integrated Fuzzy Logic Systems in iPatch

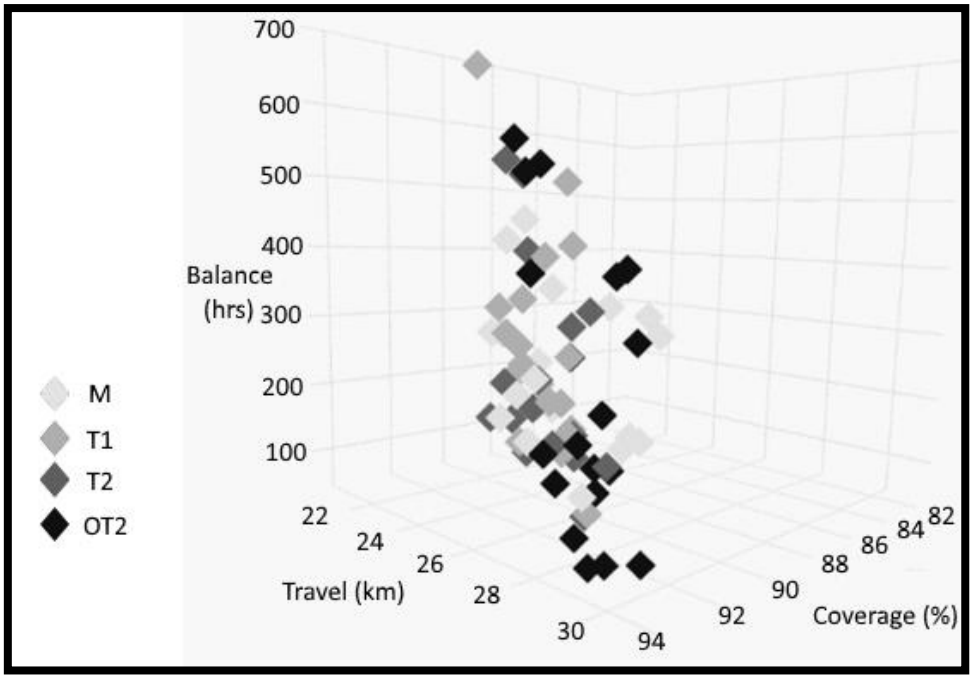

Figure 8.5: 3D plot of Pareto fronts (1)

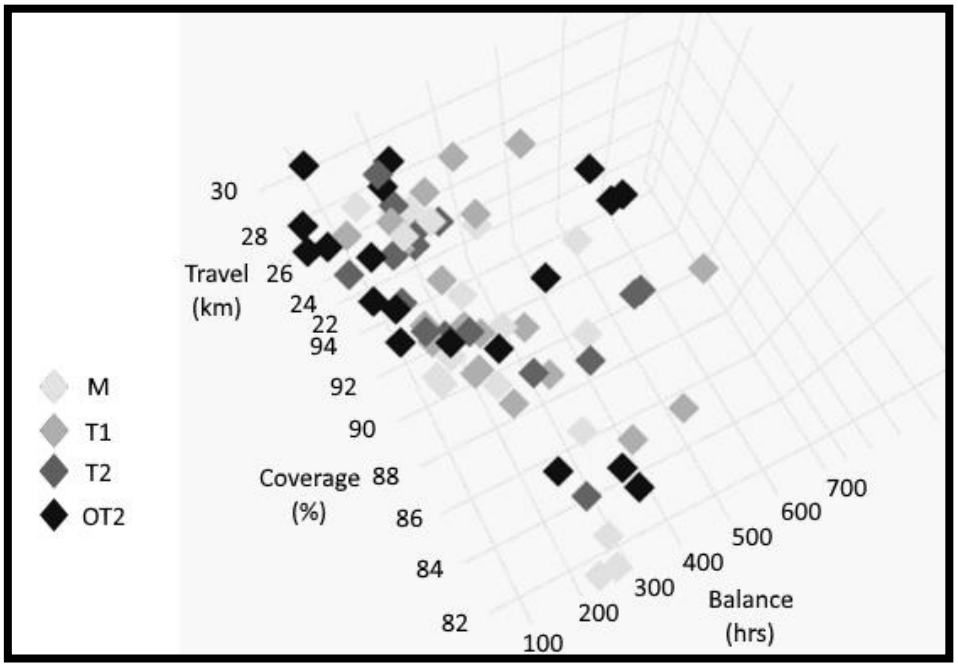

Figure 8.6: 3D plot of Pareto fronts (2)

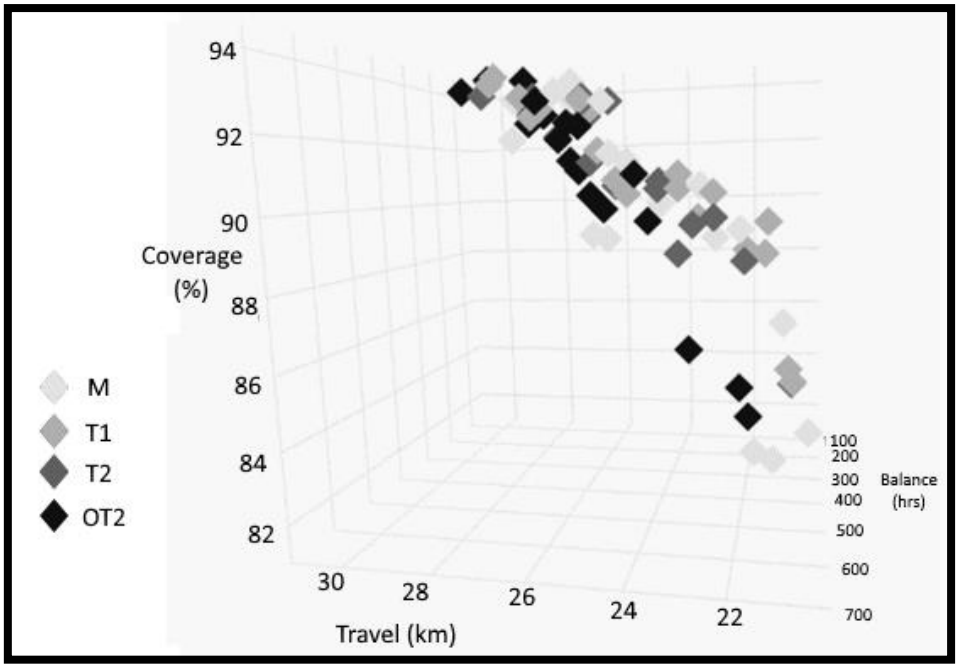

Figure 8.7: 3D plot of Pareto fronts (3)

These Pareto fronts were taken from each method's final result from the same seed. These graphs clearly show the conflicting relationship between coverage and travel. They also highlight a positive correlation showing more balanced patch designs lead to higher levels of task coverage.

If we look at the average (Avg.) of the 30 runs in Table 8-7 for each hypervolume set, we can see that best average hypervolume was achieved by OT2, followed by T2 and T1 and finally N. This is a similar pattern seen in Chapter 7 where the distance metric was used in place of the hypervolume. This experiment helps to justify the use of the distance metric as an effective measure to distinguish between solutions and to compare fronts generated by different optimisation techniques. Also, we have shown that we can improve NSGA-II even further by including the type-2 FLSs and pre-optimising the membership functions and footprints of uncertainty before the primary patch optimisation takes place. To conclusively say this is the case, we can perform statistical analysis on the two sets of hypervolume values given by NSGA-II and the NSGA-II system with optimised type-2 FLSs.

The P-value given if we compare these two sets of hypervolume values is **0.0016**, or 0.16**%** significantly below the alpha value of 0.05 (or 5%) to show a statistically significant difference between the sets.

8.3.3 Results for Fuzzy Dominance Rules in Real-World Many-Objective Problems

The first set of experiments described in Section 9.3.2 builds upon the experiments of Chapter 7 and concludes with a strong degree of certainty that the use of optimised type-2 FLSs improve the results for our multi-objective problem. However, we detailed that there are five total objectives, making this a many-objective problem. We talked about the issues surrounding parent selection for many-objective problems in 9.1. We discussed that it was believed to be a problem with the crisp value comparison in the dominance rules. Hence, the results were presented for our experiments using Fuzzy Dominance Rules (FDRs) described in Section 9.2 and tested on black-box problems in 9.3.1.

These results showed the that NSGAIIFDR could improve the optimisation of many-objective problems. To take this to a real case the following experiments will take the area for this section (benchmark results of this area were given in Table 9-6) and optimise it with all five objectives, firstly with NSGA-II, to get the multi-objective algorithm results. Then with NSGAIIFDR to measure any improvement of the implementation of the fuzzy dominance rules. Then the optimised type-2 fuzzy logic systems will we be switched on for the final set of results to get a complete view of how the full experimental system will improve over the standard NSGA-II.

We will use a 10% tolerance for the objective values when we calculate the dominance. As we are using five objectives, we cannot compare the hypervolume values from Table 8-7.

Now there are more objectives, there are also more reference points for the hypervolume. Once again, we multiply our maximisation objective by -1 when calculating the hypervolume. Our

reference points are now 0, 100, 0, 850 and 150 for coverage, travel, utilisation, patch balancing and team balancing respectively.

											Avg.
N	0.41	0.41	0.42	0.43	0.44	0.44	0.45	0.45	0.46	0.46	**0.48**
	0.46	0.47	0.47	0.47	0.47	0.47	0.47	0.48	0.49	0.49	
	0.49	0.49	0.51	0.52	0.53	0.53	0.54	0.56	0.57	0.60	
FDR	0.42	0.43	0.43	0.44	0.45	0.45	0.47	0.47	0.47	0.47	**0.51**
	0.48	0.48	0.49	0.50	0.51	0.51	0.51	0.52	0.52	0.52	
	0.54	0.54	0.55	0.56	0.57	0.57	0.57	0.57	0.59	0.63	
OT2-FDR	0.54	0.54	0.54	0.55	0.56	0.57	0.58	0.60	0.60	0.61	**0.63**
	0.61	0.62	0.62	0.63	0.63	0.64	0.64	0.65	0.65	0.66	
	0.66	0.67	0.67	0.67	0.69	0.69	0.70	0.70	0.73	0.73	

Table 8-8 Hypervolume Summary Table for Fuzzy Dominance Rules in iPatch

Table 8-8 shows the hypervolume values for NSGA-II using crisp dominance, given by **N.** The average of these runs is 0.48. **FDR** gives the hypervolume values for NSGA-II with FDRs implemented, with an average hypervolume of 0.51. If Kruskal–Wallis statistical analysis is performed on these hypervolume sets, we get a P-value of **0.049**, which is less than the required alpha value of 0.05 to prove the difference in the results are statistically significant.

Figure 8.8 illustrates these results and shows a standard boxplot where all values (lower acceptance limit, interquartile ranges, median and upper acceptance limit) are stronger in the NSGAIIFDR results than they are in the NSGA-II results. When the hypervolume sets of NSGA-II and NSGAIIFDR are compared, this gives an average improvement of 5.46%.

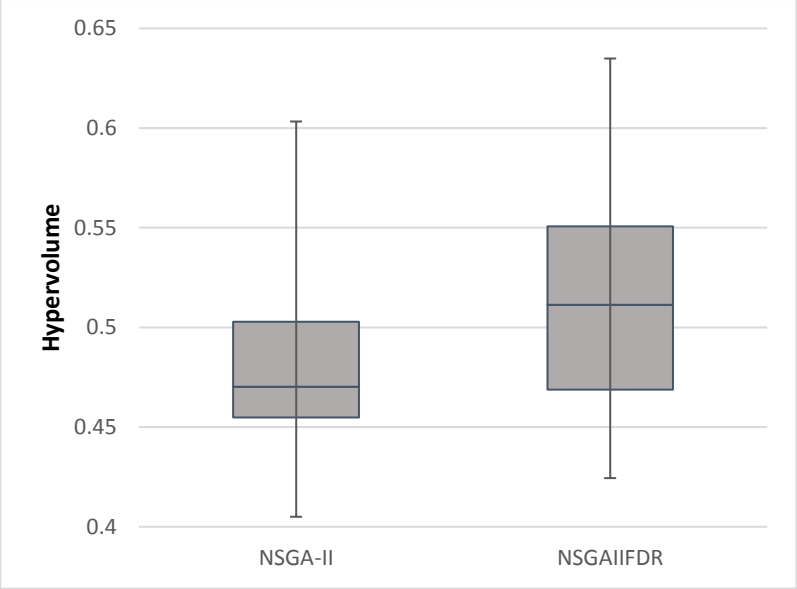

Figure 8.8: Box plot of NSGA-II vs NSGAIIFDR for our real world many objective problem

So far it has been proven that using genetically optimised type-2 systems *or* the introduction of FDRs statistically improves the Pareto front results independently. The final step is to measure the impact combining these two methods of improvement together. The results for this are shown in Table 8-8 as the **OT2FDR** hypervolume set. From the average hypervolume values, it can be seen that OT2FDR has improved the average hypervolume by 24.29% if this set is compared to the FDR results set. The Kruskal–Wallis test is performed, and this results in a P-value of $\mathbf{4.47^{-9}}$ when comparing FDR to OT2FDR, for completeness, we also get a P value of $\mathbf{1.86^{-10}}$ if we compare the N with OT2FDR.

These results make a strong case for both types of fuzzy system to be introduced in our optimisation algorithm. As part of solution generation and evaluation (with the optimised type-2 systems) and identifying dominant solutions (as with the Fuzzy Dominance Rules). Indeed, there is a significantly stronger case for these fuzzy methods to be implemented together.

8.4 Discussion

This chapter presented a solution to the help improve the weaknesses associated with many-objective optimisation, specifically the saturation of the Pareto front. The chapter described the

need to fuzzify the objective values in the dominance comparison of the NSGA-II algorithm. Fuzzy dominance rules were applied to a set of budget restricted black-box optimisation problems. The results showed that some problems were statistically significantly improved, achieving a P-value of <1%.

The chapter went on to discuss the emerging trend of the more dimensions the problem contained, the more effective NSGAIIFDR was able to improve over the standard NSGA-II. Because of the black-box nature of the competition, it is difficult to determine the particular strengths of the fuzzy dominance rules, in relation to the black-box problems.

The NSGAIIFDR algorithm was applied to the problem of organisational structure optimisation. For this, it was presented that the results were statistically significant in their improvement in the solutions created, with a P-value of 0.048. The hypervolumes were on average 5.46% better for the real-world problem when Fuzzy Dominance Rules were applied.

It was shown that genetically optimising the type-2 FLSs gave us a real improvement when comparing the hypervolumes of the NSGA-II and the genetically optimised type-2 FLSs. The P-value here was 0.0016 significantly below the required 0.05 to prove statistical significance.

The work was extended by looking at how we could solve the many-objective issues given by standard crisp dominance rules. This showed that by including FDRs to the implemented NSGA-II algorithm improved on the hypervolumes given by the Pareto fronts. The P-value attained here was 0.048, again lower than the required 0.05. Combining the fuzzy systems and FDRs resulted in a significant improvement to the many-objective algorithm, with a P-value of 1.86^{-10} when compared to the standard MOGA we previously used.

The next chapter will present the conclusions of this thesis, the real-world impact and the future work.

Chapter 9. Conclusions and Future Work

In this thesis, a novel many-objective type-2 fuzzy logic system for the optimisation of large-scale organisational design problems has been present. Further discussions about how the system can be improved through cloud resources have also been discussing. Additionally, a method for handling the saturation of the Pareto front in many-objective problems using a multi-objective algorithm has been presented. This method uses a simple fuzzy logic system in place of the crisp dominance rules to allow tolerance and flexibility between solutions.

9.1 Conclusions

The aims of the thesis were as follows:

- To investigate the most suitable optimisation methods for organisational design.

This was achieved by investigating meta-heuristic methods used to find near-optimal solutions in a vast search space. These methods were simulated annealing, genetic algorithms (GAs) and particle swarm optimisation (PSO). A GA and PSO algorithm was implemented while simulated annealing was not because of its slower traversal of the search space and potentially weaker overall results (as discussed in section 5.3). In the experiments, it was found that in both single-objective and multi-objective variations the GA outperformed the PSO. Thus, the recommended meta-heuristic for large-scale organisational design is a GA, with the NSGA-II algorithm being a suitable base for multi-objective optimisation. The NSGA-II algorithm was then adapted to handle many-objective problems by fuzzifying the dominance comparison.

- To examine the potential benefits of implementing fuzzy logic to handle the uncertainties in the data.

This was achieved by implementing fuzzy logic systems in the simulation to improve the results generated by the GA. The implemented fuzzy systems aimed to handle the uncertainties

in the task completion time and the travel time. When the type-1 systems were implemented and showed potential improvement, an interval type-2 variants of the fuzzy logic systems were implemented and this showed how changing to type-2 fuzzy systems can improve the optimisation further for this problem. The fuzzy systems' performance was then improved by tuning the membership functions and footprints of uncertainty with a genetic algorithm. So not only is the conclusion here to fuzzify measures, that are widely known to be uncertain, but the fuzzy system should be tuned using an optimisation algorithm. For large-scale organisational design optimisation problems, optimised type-2 fuzzy systems can give the most benefit when handling uncertainties in the data used to measure performance.

- To develop a system in which each proposed organisational design should take into account the wide range of complex real-world constraints, to give results that can easily be implemented into the real-world environment on which it is based.

To take into account the wide range of complex constraints a neighbourhood-based clustering algorithm was implemented, that avoided generating geographical regions that would break the required constraints (such as not crossing rivers). This clustering algorithm was one part of the system that benefited from the fuzzy logic that handled the uncertainties in travel times and task times. This meant the proposed designs could be acceptable to area managers in the organisation. For highly constrained construction of geographies in large-scale organisational design problems, bespoke clustering algorithms are necessary to avoid key business constraints being broken.

- To develop a system which should produce near-optimal geography and team designs, to reduce the amount the mobile workforce travels and increase the number of tasks the workforce completes.

This near-optimality was achieved through the many separate parts of this work that were effectively brought together into one system. The underlying meta-heuristic is a GA because it outperformed PSO. Uncertainties were well handled due to the GA tuned type-2 fuzzy logic systems. Finally, all objectives were able to be included into the optimisation due to the fuzzy dominance improvements developed for the NSGA-II algorithm. The fuzzification of the dominance comparison is a key improvement to the systems optimisation process. The near-optimality of the system can be measured due to the real-world implementation of this tool. The organisation's productivity increased 0.5% and their travel reduced by 7.7 million miles. A more in-depth discussion on the real-world benefits of the implemented system see section 10.3.

One key aspect of the success of the system is the incremental process of development. Starting with a discussion with the users and stakeholders in the organisation, understanding the business problem, researching how the problem can be tackled with state of the art techniques and develop novel methods where appropriate. Then developing prototype functionality, testing this new functionality and building up the confidence of the newly implemented methods and features, to allow the system to tackle the real-world problems.

Working with industry meant there was a constant flow of problems to solve and a feedback process that helped to tune the implemented methods. As a result, the final version of the system, which has been produced as a result of my work, is now a comprehensive tool developed to produce optimal, large-scale, organisational designs for geographical, skill and team design questions. As such this tool has been recognised as a cutting-edge, industry-leading tool by a number of external organisations, such as the British Computer Society, Institute of Engineering and Technology and the Global Telecoms Business Awards.

9.2 Real-World Impact of iPatch

As mentioned in the developed iPatch tool (shown in Figure 9.1) was implemented with the goal of improving the organisational design of a large mobile workforce. Specifically, this was British Telecoms' (BT's) mobile engineering workforce. The work presented looks at the geographical optimisation and the resource optimisation functionality.

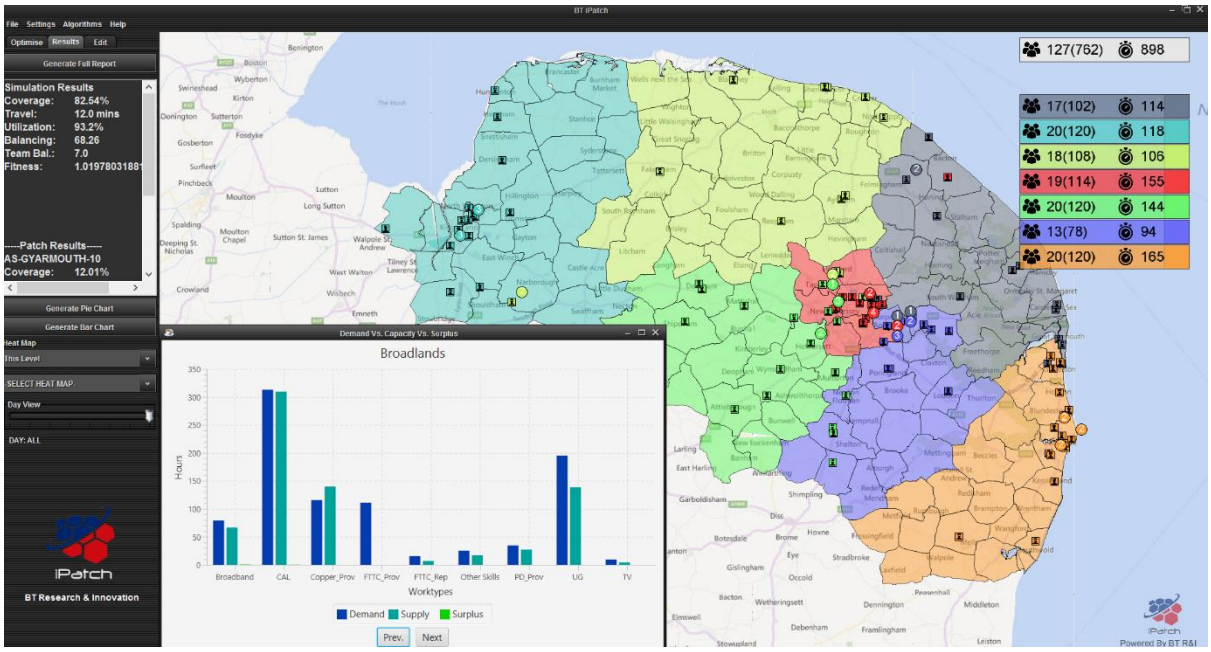

Figure 9.1: Final Version of BT's iPatch Tool

The application was developed with a strong communication and relationships with our users. This, in turn, allowed detailed feedback on problems that came to light throughout development, which allowed the results produced by iPatch to translate into the real-world effectively.

iPatch has generated an increase in productivity of 0.5% saving an estimated £1million a year over the first two years. iPatch has also cut fuel consumption by 2.9%, leading to an additional saving of over £200k a year. In addition to the financial benefits, customer commitments are now more effectively met, improving the service quality, and due to less fuel consumption, the

company can promote sustainability targets with less CO_2 emitted. Over the period of deployment, iPatch has reduced CO_2 emissions by more than 2,500 metric tonnes.

Furthermore, a report by the UK's Department of Transport found that for every billion vehicle miles travelled there were 15,409 serious injuries or deaths, or 1 per 64,900 miles [91].

As iPatch has saved an estimated 7.7 million miles of travelling, this equates to preventing 118 casualties and fatalities. The system won the 2015 Global Telecommunications Business award for best business innovation of the year in its first year of use [92], was highly commended at the IET Innovation Awards 2016 [93] and won A BCS Best Application paper award at the 36th International Conference of the BCS SGAI International Conference on Artificial Intelligence.

These outcomes show the real world impact these AI technologies, including advanced fuzzy logic systems, are having on a large, nationwide, mobile engineering workforce.

9.3 Future Work

The future work will explore the tuning of the FDR along with the exploring different membership functions for the sets used in the FDR FLS. There is also the prospect of expanding this work into the type-2 fuzzy logic domain. Where a type-2 fuzzy dominance rules could improve the performance due to type-2's ability to more effectively handle uncertainty. Uncertainty in this context would stem from the solution's strengths and weaknesses. In our real-world problem, there are high levels of uncertainty when evaluating each objective. Improving on the system's ability to distinguish between stronger solutions would improve the transfer of results from the simulation to the real world.

Additionally, addressing some of the potentially weaknesses of the optimisation which have thus far not been addressed due to scope will be looked at. This includes seeding the population

of the GA with SDPs considered as idea candidates to be centre points. The list of ideal SDPs will first come from the human patch optimisation team. Then, a neural network will take all the data available on each SDP and try to learn the most optimal exchanges for this seeding task. This will create a human vs machine comparison to benchmark the results against.

Lastly, building on the deep neural network work will be looked at by implementing an iPatch assistant to tell the human patch optimisation team which patches are underperforming and how this can be addressed with the functionality available within the iPatch tool.

References

[1] M. Cimitile, M. Gaeta and V. Loia, "An ontological multi-criteria optimization system for Workforce Management," in *FUZZIEEE 2012*, Brisbane, 2012 pp. 1-7

[2] B. Guido, G. Roberto, P. D. Tria and R. Bisio, "Workforce management (WFM) issues," in *Network Operations and Management Symposium*, New Orleans, 1998 pp. 473-482

[3] C. H. Papadimitriou and K. Steiglitz, Combinatorial Optimization – Algorithms and Complexity, New York: Dover Publications, 1982.

[4] W. Fan, Z. Gurmu and E. Haile, "A Bi-Level Metaheuristic Approach to designing Optimal Bus Transit Route Network," in *IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, Nanjing, 2013 pp. 208-313

[5] R. Domberger, L. Frey and T. Hanne, "Single and multiobjective optimization of the train staff planning problem using genetic algorithms," in *IEEE Congress on Evolutionary Computation*, Hong Kong, 2008 pp. 970-977

[6] Y. Liu, S.-l. Zhao, X.-k. Du and S.-q. Li, "Optimization of resource allocation in construction using genetic algorithms," in *International Conference on Machine Learning and Cybernetics*, Guangzhou, 2005 pp. 3428-3432.

[7] J. Tanomaru, "Staff scheduling by a genetic algorithm with heuristic operators," in *IEEE International Conference on Evolutionary Computation*, Perth, 1995 pp. 456-461

[8] N. K. Sharma, D. S. Babu and S. C. Choube, “Application of Particle Swarm optimization Technique for Reactive Power Optimization,” in *IEEE International Conference on Advances in Engineering, Science and Management*, Nagapattinam, 2012 pp. 88-93

[9] K.-B. Lee and J.-H. Kim, “Multi-Objective Particle Swarm Optimization with Preference Based Sort and its Application to Path Following Footstep Optimization for Humanoid Robots,” *IEEE Transactions on Evolutionary Computation,* vol. 17, no. 6, pp. 755 - 766, 2013.

[10] B. Qi and F. Shen, “Performance Comparison of Particle Swarm Optimization Variant Models,” in *International Conference on Information Technology: New Generations*, Las Vegas, NV, 2014 pp. 575-580

[11] D. L. Applegate, R. E. Bixby, V. Chvátal and W. J. Cook, The Traveling Salesman Problem, Princeton University Press, 2007.

[12] M. R. Garey and D. S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, New York: W. H. Freeman and Company, 1979.

[13] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan and D. B. Shmoys, The Traveling Salesman Problem: A guided Tour of Combinatorial Optimization, Chichester: John Wiley & Sons, 1985.

[14] “Rules for the Millennium Prizes,” Clay Mathematics Institute, 25 September 2012. [Online]. Available: http://www.claymath.org/millennium-problems/rules-millennium-prizes. [Accessed 12 February 2018].

[15] L. V. Snyder and M. S. Daskinb, “A random-key genetic algorithm for the generalized traveling salesman problem,” *European Journal of Operational Research,* vol. 174, no. 1, pp. 38-53, 2006.

[16] T. P. Bagchi, J. N. Gupta and C. Sriskandarajah, “A review of TSP based approaches for flowshop scheduling,” *European Journal of Operational Research,* vol. 169, no. 3, p. 816–854, 2006.

[17] C. E. Noon and J. C. Bean, “An Efficient Transformation Of The Generalized Traveling Salesman Problem,” *INFOR: Information Systems and Operational Research,* vol. 31, no. 1, pp. 39-44, 1993.

[18] G. Laporte and U. Palekar, “Some applications of the clustered travelling salesman problem,” *Journal of the Operational Research Society*, vol. 53, no. 9, p. 972–976, 2002.

[19] C. Ding, Y. Cheng and M. He, “Two-Level Genetic Algorithm for Clustered Traveling Salesman Problem with Application in Large-Scale TSPs,” *Tsinghua Science & Technology,* vol. 12, no. 4, pp. 459-465, 2007.

[20] S. Shakya, S. Kassem, A. Mohamed, H. Hagras and G. Owusu, “Enhancing Field Service Operations via Fuzzy Automation of Tactical Supply Plan,” in *Transforming Field and Service Operations*, Berlin, Springer, 2013, pp. 101-114.

[21] A. Mohamed, H. Hagras, S. Shakya and G. Owusu, “Tactical Resource Planner for Workforce Allocation in Telecommunications,” in *International Conference on Autonomous and Intelligent Systems,* Aviero, 2012 pp. 98-105

[22] D. Applegate and W. Cook, “A computational study of the job-shop scheduling problem,” *ORSA Journal on computing*, vol. 3, no. 2, pp. 149-156, 1991.

[23] D. S. Mankowska, F. Meisel and C. Bierwirth, “The home health care routing and scheduling problem with interdependent services,” *Health care management science*, vol. 17, no. 1, pp. 15-30, 2014.

[24] D. Lesaint, C. Voudouris and N. Azarmi, “Dynamic Workforce Scheduling for British Telecommunications plc,” *Interfaces*, vol. 30, no. 1, pp. 45-56, 2000.

[25] H. Algethami, R. Lankaites-Pinheiro and D. Landa-Silva, “A Genetic Algorithm for a Workforce Scheduling and Routing Problem,” in *2016 IEEE Congress on Evolutionary Computation (CEC)*, Vancouver, 2016 pp. 927-934

[26] J. A. Castillo-Salazar, D. Landa-Silva and R. Qu, “Workforce scheduling and routing problems: literature survey and computational study,” *Annals of Operations Research*, vol. 239, no. 1, pp. 39-67, 2016.

[27] P. Toth and D. Vigo, The Vehicle Routing Problem, Philadelphia: SIAM: Society for Industrial and Applied Mathmatics, 2002.

[28] M. Widmer, A. Hertz and D. Costa, Production Scheduling, Wiley, 2008.

[29] J. E. Gomar, C. T. Haas and D. P. Morton, “Assignment and Allocation Optimization of Partially Multiskilled Workforce,” *Journal of Construction Engineering and Management*, vol. 128, no. 2, pp. 103-109, 2002.

[30] K. Deb, Multi-Objective Optimization using Evolutionary Algorithms, Chichester: John Wiley & Sons, 2001.

[31] M. Affenzeller, S. Wagner, S. Winkler and A. Beham, Genetic Algorithms and Genetic Programming: Modern Concepts and Practical Applications, Boca Raton: Chapman & Hall / CRC, 2009.

[32] C. R. Darwin, The Origin of Species: By Means of Natural Selection or the Preservation of Favoured Races in the Struggle for Life, London: John Murray, 1859.

[33] T. Back, "Selective pressure in evolutionary algorithms: a characterization of selection mechanisms," in *IEEE Conference on Evolutionary Computation*, Orlando, FL, 1994, pp. 57-62

[34] K. Y. Lee and P. S. Mohamed, "A real-coded genetic algorithm involving a hybrid crossover method for power plant control system design," in *IEEE Congress on Evolutionary Computation*, Honolulu, HI, 2002, pp. 1069-1074

[35] L. Wang, H. J. Siegel, V. P. Roychowdhury and A. A. Maciejewski, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," *Journal of Parallel and Distributed Computing*, vol. 47, no. 1, pp. 8-22, 1997.

[36] U. Mehboob, J. Qadir, S. Ali and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, no. 6, pp. 2467-2501, 2016.

[37] C. Reeves, Modern Heuristic Techniques for Combinatorial Optimization, McGraw-Hill International Ltd, 1995.

[38] R. Hongliang, W. Guo, S. S. Ge and W. Lim, "Coverage planning in computer-assisted ablation based on genetic algorithm," *Computers in biology and medicine*, vol. 49, no. 1, pp. 36-45, 2014.

[39] Z. Michalewicz, Genetic Algorithms + Data Structures = Evolution Programs, Springer, 1992.

[40] D. Dumitrescu, B. Lazzerini, L. C. Jain and A. Dumitrescu, Evolutionary Computation, CRC Press, 2000.

[41] H. Cartwright, "Getting the timing right - the use of genetic algorithms in scheduling," in *Adaptive Computing and Information Processing Conference*, 1994 pp. 393-411

[42] J. H. Holland, Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence, London: The MIT Press, 1975.

[43] R. Ghanea-Hercock, Applied Evolutionary Algorithms in Java, New York: Springer-Verlag, 2003.

[44] M. Mitchell, An Introduction to Genetic Algorithms, London: MIT Press, 1998.

[45] K.-F. Man, K.-S. Tang and S. Kwong, "Genetic algorithms: concepts and applications," *IEEE transactions on Industrial Electronics*, vol. 43, no. 5, pp. 519-534, 1996.

[46] J. Kennedy and R. C. Eberhart, Swarm Intelligence, Morgan Kaufmann Publishers, 2001.

[47] Y. Shi and R. Eberhart, “A modified particle swarm optimizer,” in *IEEE World Congress on Computational Intelligence*, Anchorage, AL, 1998 pp. 69-73

[48] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” in *IEEE International Conference on Neural Networks*, Perth, 1995 pp. 1942-1948

[49] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, “Optimization by Simulated Annealing,” *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[50] R. A. Rutenbar, “Simulated Annealing Algorithms: An Overview,” *IEEE Circuits and Devices Magazine*, vol. 5, no. 1, pp. 19-26, 1989.

[51] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, “Equation of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087-1092, 1953.

[52] A. Abraham, L. Jain and R. Goldberg, Evolutionary Multiobjective Optimization: Theoretical Advances and Applications, London: Springer-Verlag, 2005.

[53] A. Kurapati and S. Azarm, “Immune Network Simulation with Multiobjective Genetic Algorithms for Multidisciplinary Design Optimization,” *Engineering Optimization*, vol. 33, no. 1, pp. 245-260, 2000.

[54] C. A. Coello Coello and A. D. Christiansan, “Multiobjective Optimization of Trusses using Genetic Algorithms,” *Computers and Structures*, vol. 75, no. 6, pp. 647-660, 2000.

[55] A. Osyczka, S. Krenich and K. Karas, "Optimum design of robot grippers using genetic algorithms," in *Congress of Structural and Multidisciplinary Optimization*, Buffalo, NY, 1999 pp. 139-146

[56] J. Teo and H. A. Abbass, "Is a Self-Adaptive Pareto Approach Beneficial for Controlling Embodied Virtual Robots," in *Genetic and Evolutionary Computation (GECCO)*, Chicago, 2003 pp. 1612-1613

[57] R. Kumar and N. Banerjee, "Multicriteria Network Design Using Evolutionary Algorithm," in *Genetic and Evolutionary Computation (GECCO)*, Chicago, 2003 pp. 2179-2190

[58] W. Pullan, "Optimising Multiple Aspects of Network Survivability," in *Congress on Evolutionary Computation*, Piscataway, NJ, 2002 pp. 115-120

[59] H. Ishibuchi, T. Yoshida and T. Murata, "Balance Between Genetic Search and Local Search in Memetic Algorithms for Multiobjective Permutation Flowshop Scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204-223, 2003.

[60] C. Brizuela, N. Sannomiya and Y. Zhao, "Multi-Objective Flow-Shop: Preliminary Results," in *First International Conference in Evolutionary Multi-Criterion Optimization*, London, Springer-Verlag, 2001, pp. 443-457.

[61] R. M. Ramos, R. R. Saldanha, R. H. Takahashi and F. J. Moreira, "The Real-Biased Multiobjective Genetic Algorithm and its Application to the Design of Wire Antennas," *IEEE Transactions on Magnetics*, vol. 39, no. 3, pp. 1329-1332, 2003.

[62] M. Krause and V. Nissen, “On using penalty functions and multicriteria optimisation techniques in facility layout,” in *Evolutionary Algorithms in Management Applications*, Berlin, Springer-Verlag, 1995 pp. 153-166

[63] A. Ekart and S. Z. Nemeth, “Selection Based on the Pareto Nondomination Criterion for Controlling Code Growth in Genetic Programming,” *Genetic Programming and Evolvable Machines,* vol. 2, no. 1, pp. 61-73, 2001.

[64] X. Llora and D. E. Goldberg, “Bounding the Effect of Noise in Multiobjective Learning Classifier Systems,” *Evolutionary Computation,* vol. 11, no. 3, pp. 279-298, 2003.

[65] A. Cunha, P. Oliveira and J. A. Covas, “Genetic Algorithms in multiobjective optimization problems: An application to polymer extrusion,” in *Genetic and Evolutionary Computation Conference*, Orlando, FL, 1999 pp. 129-130

[66] G. T. Parks, “Multiobjective Pressurized Water Reactor Reload Core Design by Nondominated Genetic Algorithm Search,” *Nuclear Science and Engineering,* vol. 124, no. 1, pp. 178-187, 1996.

[67] F. de Torro, E. Ros, S. Mota and J. Ortega, “Non-invasive Atrial Disease Diagnosis Using Detection Rules: A Multi-objective Optimization Approach,” in *International Confrence on Evolutionary Multi-Criterion Optimization*, Faro, 2003 pp.638-647

[68] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II,” *IEEE Transactions on Evolutionary Computation,* vol. 6, no. 2, pp. 182-197, 2002.

[69] H. Ishibuchi, N. Tsukamoto and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *IEEE Congress on Evolutionary Computation*, Hong Kong, 2008 pp.2419-2426

[70] J. Y. Liu, D. Gong, J. Sun and Y. Jin, "A Many-Objective Evolutionary Algorithm Using A One-by-One Selection Strategy," *IEEE Transactions on Cybernetics,* vol. 47, no. 9, pp. 2689 - 2702, 2017.

[71] M. Farina and P. Amato, "On the optimal solution definition for many-criteria optimization problems," in *Annual Meeting of the North American Fuzzy Information Processing Society NAFIPS*, New Orleans, LA, 2002 pp.233-238

[72] E. Zitzler, L. Thiele, M. Laumanns and C. M. Fonseca, , "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation,* vol. 7, no. 2, pp. 117-132, 2003.

[73] A. L. Custódio, M. Emmerich and J. F. A. Madeira, "Recent Developments in Derivative-Free Multiobjective Optimisation," *Computational Technology Reviews*, vol. 5, no. 1, pp. 1-31, 2012.

[74] R. D. White, "Organizational design and ethics: The effects of rigid hierarchy on moral reasoning," *International Journal of Organization Theory and Behavior 2 (1999): 431-456.*, vol. 2, no. 3, pp. 431-456, 1999.

[75] P. Thannimalai, M. M. Kadhum, C. J. Feng and S. Ramadass, "A glimpse of cross training models and workforce scheduling optimization," in *IEEE Symposium on Computers & Informatics*, Langkawi, 2013 pp. 98-103

[76] G. Koole, A. Pot and J. Talim, “Routing Heuristics for Multi-Skill Call Centers,” in *Winter Simulation Conference*, New Orleans, LA, 2003 pp. 1813-1816

[77] F. F. Easton, “Staffing, cross-training, and scheduling with cross-trained workers in extended-hour service operations,” Robert H. Brethen Operations Management Institute, 2011 pp. 1-33

[78] A. Lin and A. Ahmad, “SilTerra's experience in developing multi-skills technician,” in *IEEE International Conference on Semiconductor Electronics*, Kuala Lumpur, 2004 pp. 508 - 511

[79] C. T. Haas, J. D. Borcherding, R. W. Glover, R. L. Tucker, A. Rodriguez and J. Gomar, “Planning and scheduling a multiskilled workforce,” 1999, Center for Construction Industry Studies pp. 1-32

[80] H. Hu, Zhengbing, R. Moh'd and A. Shboul, “The application of ant colony optimization technique (acot) for employees selection and training,” in *IEEE Database Technology and Applications*, Wuhan, 2009 pp. 497-502

[81] T. W. Manikas and J. T. Cain, “Genetic Algorithms vs. Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem,” Computer Science and Engineering Research, Dallas, 1996 pp. 1-14

[82] T. Nair and K. Sooda, “Comparison of Genetic Algorithm and Simulated Annealing Technique for Optimal Path Selection In Network Routing,” in *National Conference on VLSI and Networks*, Chennai, 2009 pp. 36-41

[83] V. Poorjafari, W. L. Yue and N. Holyoak, “A Comparison between Genetic Algorithms and Simulated Annealing for Minimizing Transfer Waiting Time in Transit

Systems," *IACSIT International Journal of Engineering and Technology*, vol. 8, no. 3, pp. 216-221, 2016.

[84] A. Sadegheih, "Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance," *Applied Mathematical Modelling*, vol. 30, no. 1, pp. 147-154, 2006.

[85] J. Wan, D. Zhang, Y. Sun, K. Lin, C. Zou and H. Cai, "A Novel Architecture for Integrating Vehicular Cyber-Physical Systems and Mobile Cloud Computing," *Mobile Networks and Applications*, vol. 19, no. 2, p. 153–160, 2014.

[86] H. Hagras, "A Hierarchical Type-2 Fuzzy Logic Control Architecture for Autonomous Mobile Robots," *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 4, pp. 524-539, 2004.

[87] H. Hagras and C. Wagner, "Towards the Widespread Use of Type-2 Fuzzy Logic Systems in Real World Applications," *IEEE Computational Intellgence Magazine*, vol. 7, no. 3, pp. 14-24, 2012.

[88] C. Lynch, H. Hagras and V. Callaghan, "Embedded interval type-2 neuro-fuzzy speed controller for marine diesel engines," in *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, Paris, 2006 pp. 1446-1453

[89] C. Wagner and H. Hagras, "A Genetic Algorithm Based Architecture for Evolving Type-2 Fuzzy Logic Controllers for Real World Autonomous Mobile Robots," in *IEEE International Conference on Fuzzy Systems*, London, 2007 pp. 1-6

[90] I. Loshchilov and T. Glasmachers, “Black Box Optimization Competition - BBComp,” [Online]. Available: https://bbcomp.ini.rub.de/. [Accessed 12 February 2018].

[91] D. Lloyd, “Reported Road Casualties in Great Britain: Main Results 2014,” Department of Transport, 25 June 2015. [Online]. Available: www.gov.uk/government/uploads/system/uploads/attachment_data/file/463045/rrcgb2014-01.pdf. [Accessed 12 February 2018].

[92] “School of Computer Science and Electronic Engineering: News,” University of Essex, 18 May 2015. [Online]. Available: https://www1.essex.ac.uk/news/event.aspx?e_id=7695. [Accessed 12 February 2018].

[93] “School of Computer Science and Electronic Engineering: News,” University of Essex, 10 February 2017. [Online]. Available: https://www1.essex.ac.uk/news/event.aspx?e_id=11882. [Accessed 12 February 2018].

[94] L. A. Zadeh, “Fuzzy Sets,” *Information and Control*, vol. 8, no. 3, pp. 338-353, 1965.

[95] C. Lynch, H. Hagras and V. Callaghan, “Embedded Type-2 FLC for the Speed Control of Marine and Traction Diesel Engines,” in *International Conference on Fuzzy Systems*, Reno NV, 2005 pp. 347-352

[96] J. Maiers and Y. S. Sherif, “Applications of fuzzy set theory,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 15, no. 1, pp. 175 - 189, 1985.

[97] C. C. Lee, “Fuzzy Logic in Control Systems: Fuzzy Logic Controller, Part II,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 20, no. 2, pp. 419 - 435, 1990.

[98] M. Antonelli, D. Bernardo, H. Hagras and F. Marcelloni , “Multiobjective Evolutionary Optimization of Type-2 Fuzzy Rule-Based Systems for Financial Data Classification,” *IEEE Transactions on Fuzzy Systems,* vol. 25, no. 2, pp. 249 - 264, 2017.

[99] G. J. Klir and M. J. Wierman, Uncertainty-Based Information, Heidelberg: Physica-Verlag, 1998.

[100] J. M. Mendel, Uncertain Rule-based Fuzzy Logic Systems: Introduction and New Directions, Prentice Hall, 2001.

[101] H. R. Berenji, “Treatment of Uncertainty in Artificial Intelligence,” in *Machine Intelligence and Autonomy for Aerospace Systems*, Washington DC, AIAA, 1988, pp. 233-247.

[102] J. M. Mendel, “Fuzzy logic systems for engineering: a tutorial,” *Proceedings of the IEEE* , vol. 83, no. 3, pp. 345-377, 1995.

[103] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning - I,” *Information Sciences,* vol. 8, no. 3, pp. 199-249, 1975.

[104] E. Cox, “Fuzzy Fundamentals,” *IEEE Spectrum,* vol. 29, no. 10, pp. 58-61, 1992.

[105] S. Horikawa, T. Furuhashi and Y. Uchikawa, “On fuzzy modeling using fuzzy neural networks with the back-propagation algorithm,” *IEEE transactions on Neural Networks,* vol. 3, no. 5, pp. 801-806, 1992.

[106] J. Jang, “Self-learning fuzzy controllers based on temporal backpropagation,” *IEEE Transactions on Neural Networks,* vol. 3, no. 5, pp. 714-723, 1992.

[107] L.-X. Wang and J. M. Mendel, *Generating fuzzy rules from numerical data, with Applications,* Signal and Image Processing Institute, University of Southern California, Department of Electrical Engineering-Systems, 1991.

[108] L.-X. Wang and J. M. Mendel, "Back-propagation fuzzy system as nonlinear dynamic system identifiers," in *IEEE International Conference on Fuzzy Systems*, San Diego, 1992 pp. 1409-1418

[109] A. Kaufman and M. M. Gupta, Introduction to Fuzzy Arithmetic: Theory and Applications, New York: Van Nostrand Reinhold, 1991.

[110] C. Fu, A. Sarabakha , E. Kayacan , C. Wagner, R. John and J. M. Garibaldi, "A comparative study on the control of quadcopter UAVs by using singleton and non-singleton fuzzy logic controllers," in *IEEE International Conference on Fuzzy Systems*, Vancouver, 2016 pp. 1023-1030

[111] L.-X. Wang, Adaptive fuzzy systems and control: design and stability analysis, Englewood Cliffs, NJ: Prentice-Hall, 1994.

[112] N. Vadiee and M. Jamshidi, "A Tutorial on Fuzzy Rule-Based Expert Systems (FRBES) Models 1: Mathmatical Foundations," *Journal of Intelligent and Fuzzy Systems,* vol. 1, no. 1, pp. 171-188, 1993.

[113] E. H. Mamdani, "Applications of Fuzzy Algorithms for Simple Dynamic Plant," *Proceedings of the IEEE,* vol. 121, no. 12, pp. 1585-1588, 1974.

[114] D. Driankov, H. Hellendoorn and M. Reinfrank, An Introduction to Fuzzy Control (2nd ed.), Springer-Verlag, 1996.

[115] L.-X. Wang, A course in Fuzzy Systems and Control, Upper Saddle River, NJ: Prentice-Hall, 1997.

[116] M. Sugeno and T. Yasukawa, "A Fuzzy-Logic-Based Approach to Qualitative Modeling," *IEEE Transactions on Fuzzy Systems*, vol. 1, no. 1, pp. 7-31, 1993.

[117] J. M. Mendel, "Type-2 Fuzzy Sets and Systems: An Overview," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 20-29, 2007.

[118] H. Hagras, "Type-2 FLCs: A New Generation of Fuzzy Controllers," *IEEE Computational Intelligence Magazine*, vol. 2, no. 1, pp. 30-43, 2007.

[119] J. M. Mendel, R. I. John and F. Liu, "Interval Type-2 Fuzzy Logic Systems Made Simple," *IEEE Transactions on Fuzzy Systems*, vol. 14, no. 6, pp. 808-821, 2006.

[120] N. N. Karnik and J. M. Mendel, "An introduction to type-2 fuzzy logic systems," in *IEEE World Congress on Computational Intelligence*, Anchorage, AK, 1998 pp. 915-920

[121] D. Wu and W.-W. Tan, "Type-2 FLS modeling capability analysis," in *IEEE International Conference on Fuzzy Systems*, Reno, NV, 2005. pp. 241-247

[122] Q. Liang and J. M. Mendel, "Interval Type-2 Fuzzy Logic Systems: Theory and Design," *IEEE Transactions on Fuzzy Systems*, vol. 8, no. 5, pp. 535-549, 2000.

[123] D. Dubois and H. Prade, Fundementals of Fuzzy Sets, Springer Science & Business Media, 2000.

[124] H. Hersh and A. Carmazza , “A Fuzzy Set Approach to Modifiers and Vagueness in Natural Language,” *Journal of Experimental Psychology: General,* vol. 105, no. 3, pp. 254-276, 1976.

[125] J. L. Chameau and J. C. Santamarina , “Membership Functions Part I: Comparing Method of Measurement,” *International Journal of Approximate Reasoning* , vol. 1, no. 3, pp. 287-301, 1987.

[126] W. Pedrycz, A. Skowron and V. Kreinovich, Handbook of Granular Computing, Wiley-Blackwell, 2008.

[127] L.-X. Wang and J. M. Mendel, “Generating Fuzzy Rules by Learning from Examples,” *IEEE Transactions on Systems, Man and Cybernetics,* vol. 22, no. 6, pp. 1414-1427, 1992.

[128] E. E. Omizegba and G. E. Adebayo, “Optimizing Fuzzy Membership Functions Using particle swarm algorithm,” in *IEEE International Conference on Systems, Man, and Cybernetics*, San Antonio, 2009 pp. 3866-3870

[129] A. A. Esmin, A. R. Aoki and G. Lambert-Torres, “Particle swarm optimization for fuzzy membership functions optimization,” in *IEEE International Conference on Systems, Man, and Cybernetics*, Yasmine Hammamet, 2002 pp. 1-6

[130] E. Cox, “Adaptive Fuzzy Systems,” *IEEE Spectrum,* vol. 30, no. 2, pp. 27-31, 1993.

Appendix A

A.1 A Brief Introduction to Fuzzy Logic

Fuzzy Logic (FL) was first introduced by Lotfi Zadeh in his 1965 paper 'Fuzzy Sets' [94]. Zadeh describes the non-binary classification of elements to classes; that is, instead of classifying if an element belongs to a class as either true or false, there is instead a degree of membership to that class.

An example of these imprecisely defined classes might be height. If there are three classes, *short*, *average* and *tall* how would people be classified? Certainly, there are many contextual factors that should be considered in such a classification like gender and age, but there would certainly be consensus that anyone over 1.9 metres (6 foot 3 inches) could only be classified as tall, however someone who is 1.7 meters (5 foot 7 inches) is more difficult to classify and may belong to both the *average* and *tall* classes, but with less certainty about both classifications. This is when fuzziness is introduced as the degree of membership to each class may be less than 1 (100% true) but greater than 0 (100% false).

Zadeh goes on to state that such imprecisely defined classes exist throughout the real world and play an important part in human reasoning and decision-making. Particularly when it comes to pattern recognition, communication and abstraction.

Since its conception, the field of fuzzy logic has expanded and has been applied to numerous real-world applications. [95] [86] [96] [97] [98]

There are now sub-fields within the discipline of fuzzy logic including type-1 fuzzy logic systems (using the originally proposed fuzzy logic methods) and type-2 fuzzy logic.

A.2 Uncertainty

Fuzzy logic has been designed to handle uncertainty in many forms. In general, uncertainty comes in many guises and is independent of what kind of fuzzy logic, or any kind of methodology, one uses to handle it. One of the best sources for general discussions about uncertainty is by Professor Klir [99] [100].

Klir and his students have focused on uncertainty since the 1980s. Regarding the *occurrence of uncertainty*, they state [100]:

When dealing with real-world problems, we can rarely avoid uncertainty. At the empirical level, uncertainty is an inseparable companion of almost any measurement, resulting from a combination of inevitable measurement errors and resolution limits of measuring instruments. At the cognitive level, it emerges from the vagueness and ambiguity inherent in natural language. At the social level, uncertainty has even strategic uses, and it is often created and maintained by people for different purposes (privacy, secrecy, propriety) [99]

Regarding the *causes of uncertainty*, they state:

The uncertainty involved in any problem-solving situation is a result of some information deficiency. Information (pertaining to the model within which the situation is conceptualised) may be incomplete, fragmented, not fully reliable, vague, contradictory, or deficient in some other way. In general, these various information deficiencies may result in different types of uncertainty [99].

Regarding the *nature of uncertainty*, they state:

Three types of uncertainty are now recognised…*fuzziness* (or vagueness), which results from imprecise boundaries of fuzzy sets; *nonspecificity* (or imprecision), which is concerned with

sizes (cardinalities) of relevant sets of alternatives; and *strife* (or discord) which expresses conflict among the various sets of alternatives [99].

The types of uncertainty stated above are divided into two major classes, fuzziness and ambiguity, where ambiguity (one to many relationships) include non-specificity and strife. Another source for some general discussion of uncertainty is Berenji [101]. Who state, in agreement with Klir, that "uncertainty stems from the lack of complete information". He also states that "uncertainty may also reflect incompleteness, imprecision, missing information, or randomness in data and a process".

A.3 Type-1 Fuzzy Logic Systems

A fuzzy logic system (FLS) can be defined as a nonlinear mapping of an input data (feature) vector into a scalar output (the vector output case decomposes into a collection of independent multi-input/single-output systems). The richness of FL is that there are enormous numbers of possibilities that lead to lots of different mappings. This richness does require a careful understanding of FL and the elements that comprise FLS [102].

A fuzzy logic system has multiple components to it, the fuzzifier, the inference engine, the rule base and the defuzzifier. Figure A.1 illustrates these components and shows the process of taking in crisp input values and giving out crisp output values. Crisp values are real numbers and the uncertainty associated with the value is not represented. However real values are necessary for control and decision-making systems. For example, height in metres, the temperature in Celsius and speed in km/h are all examples of crisp values needed as either inputs or outputs to the fuzzy logic system.

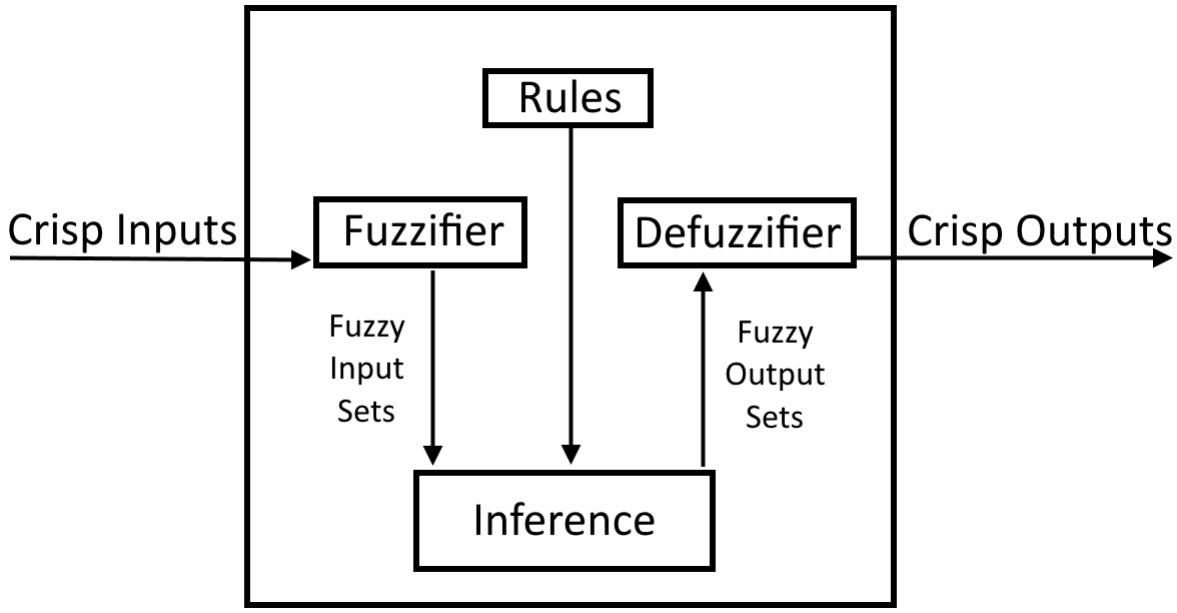

Figure A.1: Type-1 Fuzzy Logic System [102]

A.3.1 Linguistic Variables

Zadeh describes linguistic variables as "variables whose values are not numbers but words or sentences in natural or artificial language". This is because in general linguistic characterisations are less specific than numerical ones [103].

This means that numerical values can be classified under certain linguistic variables and still retain its contextual meaning. Given that fuzzy logic allows classification [0, 1] rather than true or false, a numerical value may fall into two or more linguistic labels, but to varying degrees of membership.

For fuzzy logic systems, linguistic variables are used to name the classes attributed to any input within the system. In turn, these linguistic variables can be classified or grouped, into sets, where the set also has a label. Let u denote the name of the set (e.g. temperature). Numerical values of a linguistic variable u are denoted x, where $x \in U$. Sometimes x and u are used interchangeably, especially when a linguistic variable is a letter, as in sometimes the case in engineering applications. A linguistic variable is usually decomposed into a set of terms, T(u),

which cover its universe of discourse [102]. The universe of discourse is defined as the complete range of values to be expressed within the discussion

We can use an example to illustrate: Let temperature (*u*) be interpreted as a linguistic variable. It can be decomposed into the following terms: T(*temperature*) = {cold, cool, okay, warm, hot} each term in T(*temperature*) is characterised by a set in the universe of discourse X = [0°C, 50°C]. We might interpret cold as a temperature below 10°C, cool to a temperature close to 15°C, okay as a temperature close to 23°C, warm as a temperature close to 28°C and hot as a temperature above 32°C [104].

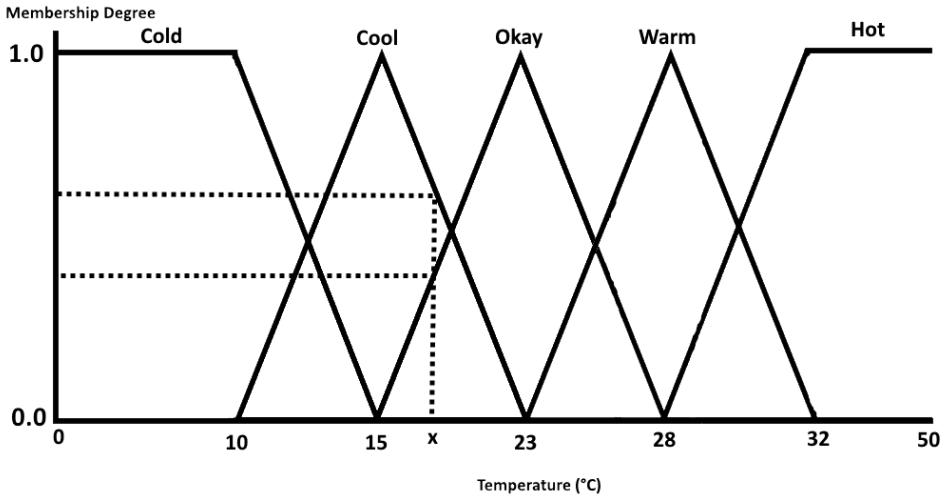

Figure A.2 Membership Functions for T(*temperature*)

These terms can be characterised as fuzzy sets whose membership functions are shown in Figure A.2. Measured values of temperature (*x*) lie along the temperature axis. In this example, a vertical line from any measured value intersects at most, two linguistic classes, also known as membership functions (see Membership Functions). For example, let x = 18°C and resides in the linguistic classes cool and okay, but to different degrees of similarity [100].

A.3.2 Membership Functions

In fuzzy logic systems membership functions (MFs) are associated with the linguistic labels and help to define the range of values that can be associated with that linguistic label and the degree in which it should be associated. Membership functions has the mathematical notation $\mu_F(x)$ [100].

The most common geometric shapes used for membership functions are triangular, trapezoidal, Gaussian and singleton, shown in Figure A.3. Membership functions are sometimes chosen by the user arbitrarily, based on the user's experience; hence, the membership functions for two users could be quite different depending on their experiences, perspectives, cultures, etc. Membership functions can be designed using optimisation procedures (for example, [105] [106] [107] and [108].

One common method for designing the membership functions for a fuzzy set is to have an expert design them, where the term expert is loosely used here. The expert could be the person developing the system. Other methods of designing fuzzy systems can be found in Section 3.4.

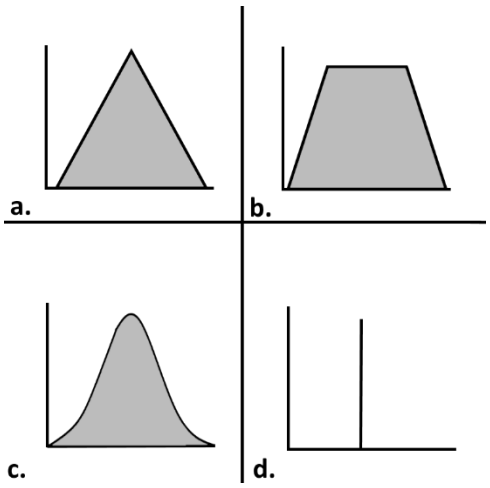

Figure A.3: Types of Membership Function a) Triangular
b) Trapezoidal c) Gaussian d) Singleton [100]

The definition of the triangular membership function is [100]

$$\begin{cases} \dfrac{x-a}{b-a} & a \le x \le b \\ \frac{c-x}{c-b} & b < x \le c \\ 0 & Otherwise \end{cases} \tag{A-1}$$

The definition of the trapezoid membership function is [100]

$$\begin{cases} \dfrac{x-a}{b-a} & a \le x \le b \\ 1 & b < x \le c \\ \dfrac{d-x}{d-c} & c < x \le d \\ 0 & Otherwise \end{cases} \tag{A-2}$$

The definition of the Gaussian membership function is [100]

$$_e(-0.5((x-a)/\theta)^2) \tag{A-3}$$

The definition of the singleton membership function is [100]

$$\mu_F(x) = m \tag{A-4}$$

Greater resolution is achieved by using more membership functions at the price of greater computational complexity. Membership functions must overlap. This expresses the fact that "The glass can be partially full and partially empty at the same time." In this way, we are able to distribute our decisions over more than one input class, which helps to make FL systems robust. Although membership functions do not have to be scaled between zero and unity, most people do this so that variables are normalised [102].

A.3.3 Fuzzy Set Theoretic Operations

Now that we have defined fuzzy sets, what can we do with them? Let us describe the set operations of union, intersection and complement [100].

Let A and B be two subsets of X. The *union* of A and B, denoted by $A \cup B$, contains all the elements in either A or B, i.e [100].

$$\mu_{A\cup B}(x) = \begin{cases} 1 \; if \; x \in A \; or \; x \in B \\ 0 \; if \; x \notin A \; and \; x \notin B \end{cases} \tag{A-5}$$

The *intersection* of A and B denoted $A \cap B$, contains all the elements that are simultaneously in A and B, i.e [100].

$$\mu_{A\cap B}(x) = \begin{cases} 1 \; if \; x \in A \; and \; x \in B \\ 0 \; if \; x \notin A \; or \; x \notin B \end{cases} \tag{A-6}$$

Let $\bar{A}$ denote the complement of A; it contains all the elements not in A, i.e. [100],

$$\mu_{\bar{A}}(x) = \begin{cases} 1 \; if \; x \notin A \\ 0 \; if \; x \in A \end{cases} \tag{A-7}$$

From these facts, it is easy to show that [100]:

$$A \cup B \Rightarrow \mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)] \tag{A-8}$$

$$A \cap B \Rightarrow \mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)] \tag{A-9}$$

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{A-10}$$

In fuzzy logic, union, intersection and complement are defined in terms of their membership functions. Let fuzzy sets A and B be described by their membership functions $\mu_A(x)$ and $\mu_B(x)$. One definition of fuzzy union leads to the membership function [100]:

$$\mu_{A\cup B}(x) = \max[\mu_A(x), \mu_B(x)] \tag{A-11}$$

Moreover, one definition of the fuzzy intersection leads to the membership function [100]:

$$\mu_{A\cap B}(x) = \min[\mu_A(x), \mu_B(x)] \tag{A-12}$$

Additionally, the membership function of fuzzy compliment is [100]:

$$\mu_{\bar{A}}(x) = 1 - \mu_A(x) \tag{A-13}$$

Although equations (A-8)-(A-10) and (A-11)-(A13) look exactly alike, we must remember that:

1. Sets A and B in (3-8)-(3-10) are fuzzy, whereas in (3-11)-(313) they are crisp.
2. Fuzzy sets can *only* be characterised by their membership functions, whereas crisp users can be characterised by either their membership functions, a description of their elements, or a listing of their elements. [100]

A.3.4 Fuzzifier

The fuzzier maps a crisp point x = col($x_1, \dots, x_n$) $\in U$ into a fuzzy set A* in U. The most widely used fuzzier is the singleton fuzzier which is nothing more than a fuzzy singleton [102].

Singleton fuzzification may not always be adequate, especially when data is corrupted by measurement noise. Nonsingleton fuzzification provides a means for handling such uncertainties totally within the framework of FLS's [102].

In non-singleton fuzzification, measurement $x_i = x_i'$ is mapped into a fuzzy number [109]. i.e. a membership function is associated with it. More specifically:

A non-singleton fuzzifier is one for which $\mu_{x_i}(x_i') = 1\ (i = 1, \dots, p) and\ \mu_{x_i}(x_i)$ decrases from unity as x_i moves away from x_i'.

Conceptually, the non-singleton fuzzifier implies that the given input value x_i' is the most likely value to be the correct one from all the values in its immediate neighbourhood; however,

because the input is corrupted by noise, neighbouring points are also likely to be the correct value, but to a lesser degree. Figure A.4 illustrates singleton and non-singleton fuzzification.

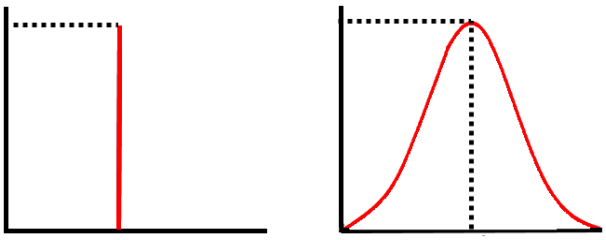

Figure A.4 a) Singleton Fuzzification b) Non-singleton Fuzzification [110]

A.3.5 Rules

Rules are at the heart of any fuzzy logic system. Rules can be provided by experts or can be extracted from numerical data. In either case, the rules that we are interested in can be expressed as a collection of IF-THEN statements. The IF-part of the rule is its *antecedent*, and the THEN-part of a rule is its *consequent* [100]

Consider a fuzzy logic system having p inputs $x_1 \in X_1, \ldots, x_p \in X_p$ and one output $y \in Y$. Let us suppose it has M rules, where the lth rule has the form [100]:

$$R^1: IF\ x_1\ is\ F_1^l\ and \ldots and\ x_p\ is\ F_p^l, THEN\ y\ is\ G^l \quad l = 1, \ldots, M \tag{A-14}$$

This rule represents a fuzzy relation between the input space $X_1 \times \ldots \times X_p$ and the output space, Y of the fuzzy logic system [100].

Multi-antecedent multi-consequent rules can be expressed as a group of multi-input single-output rules. Six such rules are summarised next, with the first five being adapted from [111]. Of course, in practical applications, it is possible to have rules that combine non-obvious IF-THEN rules in all sorts of interesting ways.

A.3.5.1 Incomplete IF Rules

Suppose we have created a rule base where there are p inputs, e.g [100].

$$IF\ x_1 is\ F_1 and \dots and\ x_m\ is\ F_m, THEN\ y\ is\ G \tag{A-15}$$

Such rules are called incomplete IF rules and apply regardless of $x_{m+1}, \dots, x_p$. They can be put into the format of the complete IF rule by treating the unnamed antecedents (e.g., $x_{m+1}, \dots, x_p$) as elements of the fuzzy set IN-COMPLETE (IN for short) where, by definition $\mu_{IN\,(x)=1}$ for all $x \in X$, i.e [100].

$$(IF\ x_1 is\ F_1 and \dots and\ x_m\ if\ F_m, THEN\ y\ is\ G) \tag{A-16}$$

$$\Leftrightarrow\ (IF\ x_1 is\ F_1 and \dots and\ x_m\ if\ F_m\ and\ x_{m+1}\ is\ IN \dots and\ x_p\ is\ IN, THEN\ y\ is\ G)$$

A.3.5.2 Mixed Rules

Not all rules use the "and" connective; some use the "or" connective, and some use a mixture of both. The latter rules are called *mixed rules*. These rules can be decomposed into a collection of equivalent rules, using standard techniques from crisp logic. Suppose, for example; we have the rule [100]:

$IF(x_1 is\ F_1\ and \dots and\ x_m\ is\ F_m) or (x_{m+1} is\ F_{m+1}\ and \dots and\ x_p\ is\ F_p)$ THEN y is G (A-17)

This rule can be expressed as the following two rules [100]:

$$R^1\text{: } IF x_1 is\ F_1\ and \dots and\ x_m\ is\ F_m\ THEN\ y\ is\ G \tag{A-18}$$

$$R^2\text{: } x_{m+1} is\ F_{m+1}\ and \dots and \dots\ x_p\ is\ F_p\ THEN\ y\ is\ G$$

Observe that both of these rules are Incomplete IF rules. See [112] for a related discussion on nesting of rules.

A.3.5.3 Fuzzy Statement Rules

Some rules do not appear to have antecedents; they are statements involving fuzzy sets. Hence, they are called *fuzzy statement rules*. For example, $y\ is\ G$ is such a rule. Clearly this is an extreme case of an incomplete IF rule, and can therefore be formulated as [100]:

$$IF\ x_1 is\ IN\ and\ \dots and\ x_p\ is\ IN, THEN\ y\ is\ G \tag{A-19}$$

A.3.5.4 Comparative Rules

Some rules are comparative, e.g. $The\ Smaller\ the\ x\ the\ bigger\ the\ y$. Such rules must first be reformulated as IF-THEN rules. This rule should then be expressed as $IF\ x\ is\ S, THEN\ y\ is\ B$. Where S is a fuzzy set representing *smaller* and B is a fuzzy set representing *bigger* [100].

A.3.5.5 Unless Rules

Rules are sometimes stated using the connective "unless"; such rules are called *unless rules* and can be put into the required format by using logical operators. For example, the rule [100]:

$$y\ is\ G\ unless\ x_1 is\ F_1\ and\ \dots and\ x_p is\ F_p \tag{A-20}$$

can be expressed as [100]:

$$IF\ not\ \left(x_1 is\ F_1 and\ \dots and\ x_p is\ F_p\right), THEN\ y\ is\ G \tag{A-21}$$

A.3.5.6 Quantifier Rules

Rules sometimes include the quantifiers "some" or "all"; such rules are called *quantifier rules*. Because of the duality between propositional logic and set theory, rules with the quantifier "some" means that we have to apply the union operator to the antecedents or consequents to which the "some" applies, whereas rules with the quantifier "all" means we have to apply the intersection operator to the antecedents or consequents to which the "all" applies [100].

A.3.6 Inference Engine

In the fuzzy inference engine (which is labelled fuzzy inference engine in Figure A.1), fuzzy logic principles are used to combine fuzzy IF-THEN rules from the fuzzy rule base into a mapping from fuzzy input sets in $X_1 \times ... \times X_p$ to fuzzy output sets in Y. Each rule is interpreted as a fuzzy implication. With reference to Figure A.1 [102]. Mamdani implications are the most commonly used in engineering applications. We treat the fuzzy inference engine as a system, one that maps fuzzy set into fuzzy sets by means of $\mu_{A\to B}(x, y)$ [100]

Mamdani [113] simplified the computations associated with calculating weights associated with each rule. The weights of the rules are more commonly referred to as the firing strengths of the rules.

There are three widely used implications to calculate firing strength. If all connectives in a rule are "And" then the minimum membership degree can be used (A-22) or the product of the membership degrees (A-23) (TNORMS) [100]:

$$\mu_{A\to B}(x, y) \equiv \min[\mu_A(x), \mu_B(y)] \tag{A-22}$$

$$\mu_{A\to B}(x, y) \equiv \mu_A(x)\, \mu_B(y) \tag{A-23}$$

If all the rule connectives are "Or" then the maximum membership degree can be used (A-22) (TCONORMS) [100]:

$$\mu_{A\to B}(x, y) \equiv \max[\mu_A(x), \mu_B(y)] \tag{A-24}$$

A.3.7 Defuzzifier

Defuzzification produces a crisp output for FLS from the fuzzy set that is the output of the inference engine. Because we are interested in practical applications of FL, one criterion for the choice of a defuzzifier is computational simplicity. The case for computational simplicity is strengthened because we plan to use FLSs within population-based optimisation algorithms.

In this type of application, the calls to the FLS will be frequent and demanding. [100] Some defuzzification methods are as follows:

A.3.7.1 Centroid Defuzzifier

The centroid defuzzifier combines the output fuzzy sets using union (i.e. a t-cornorm, e.g. maximum) and then find the centroid of this set. If the composite fuzzy output set is B is [100]:

$$B = \cup_{l=1}^{M} B^l \tag{A-25}$$

With associated membership function $\mu_B(y)$, and $\mu_{B^l}(y)$ is the membership function of the *l*th rule, then the centroid defuzzification is [100]:

$$y_c(x) = \frac{\sum_{i=1}^{N} y_i \mu_B(y_i)}{\sum_{i=1}^{N} \mu_B(y_i)} \tag{A-26}$$

Unfortunately, the centroid defuzzification is usually difficult to compute because of first having to compute the union (in A-25). However, in practice we can get around this by pre-computing the centroids of the output sets, assuming they are fixed for the FLS. This would negate the performance impact of this defuzzification method [100].

A.3.7.2 Height Defuzzifier

The height defuzzifier [114] , also called the centre average defuzzifier [111] [115], replaces each rule output fuzzy set with a singleton at the point of having maximum membership in the output set, then calculating the centroid of the type-1 set comprised of these singletons. The output of a height defuzzifier is given as [100]:

$$y_h(x) = \frac{\sum_{l=1}^{M} \bar{y}^l \mu_{B^l}(\bar{y}^l)}{\sum_{l=1}^{M} \mu_{B^l}(\bar{y}^l)} \tag{A-27}$$

(A-27) is very easy to use because the centres of gravity of commonly used membership functions are known ahead of time. For example, regardless of whether minimum or product inference are used, the centre of gravity of $B^l$for:

1. A symmetric triangular consequent membership function is at the apex of the triangle.
2. A Gaussian consequent membership function is at the centre value of the Gaussian function.
3. A symmetric trapezoidal membership function is at the midpoint of its support.

A.3.7.3 Modified Height Defuzzifier

The modified height defuzzifier, also called the modified centre average defuzzifier [102] [111], is very similar to the height defuzzifier, the only difference being that the modified height defuzzifier scales each $\mu_{B^l}(\bar{y}^l)$ by the inverse of the square of the spread (or some measure of the spread) of the *l*th consequent set. Its output can be expressed as [100]:

$$y_{mh}(x) = \frac{\sum_{l=1}^{M} \bar{y}^l \mu_{B^l}(\bar{y}^l)/\delta^{l^2}}{\sum_{l=1}^{M} \mu_{B^l}(\bar{y}^l)/\delta^{l^2}} \tag{A-28}$$

A.3.7.4 Centre-Of-Sets Defuzzifier

In centre-of-sets defuzzification [116], we replace each rule consequent set by a singleton situated at its centroid, whose amplitude equals the firing level, and then the centroid of the type-1 set comprised of these singletons. The expression the output is given as [100] :

$$y_{cos}(x) = \frac{\sum_{l=1}^{M} c^l T_{i=1}^{p} \mu_{F_i^l}(x_i)}{\sum_{l=1}^{M} T_{i=1}^{p} \mu_{F_i^l}(x_i)} \tag{A-29}$$

A.4 Type-2 Fuzzy Logic Systems

Type-1 fuzzy logic systems have limited capabilities to directly handle data uncertainties, where handle means to model and minimise the effect of. As discussed, uncertainty comes in many guises and is independent of the kind of fuzzy system or methodology one uses to handle it. Two important aspects of uncertainties are linguistic and random. The former is associated with words, and the fact that *words can mean different things to different people,* and the latter

is associated with unpredictability. Probability theory is used to handle random uncertainty, and fuzzy systems are used to handle linguistic uncertainty, and sometimes FLSs can also be used to handle both kinds of uncertainty, because a fuzzy system may use noisy measurements or operate under random disturbances [117]

Adding uncertainty to the type-1 membership functions means that the membership grade is no longer a crisp number, it is its own set in the range [0, 1]. Calculating all $x \in X$ creates a three-dimensional membership function, a type-2 membership function that characterises a type-2 fuzzy set.

A.4.1 Interval Type-2 Fuzzy Logic Systems

The interval type-2 FLS uses interval type-2 fuzzy sets to represent the inputs and/or outputs. The interval type-2 FLS is depicted in Figure A.5 and it consists of a Fuzzifier, Inference Engine, Rule Base, Type-reducer and Defuzzifier.

Only interval type-2 FLS will be implemented in this thesis.

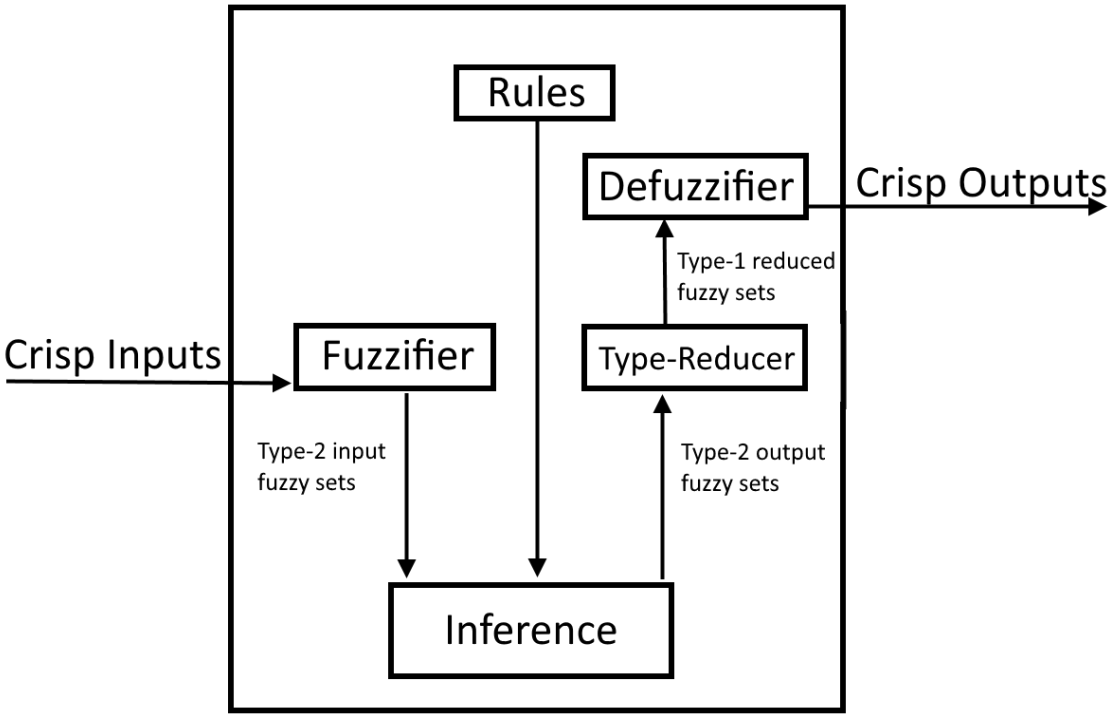

Figure A.5 Type-2 Fuzzy Logic System [100]

The interval type-2 FLS works as follows: the crisp inputs are first fuzzified into input type-2 fuzzy sets; singleton fuzzification is usually used in interval type-2 FLC applications due to its simplicity and suitability for embedded processors and real-time applications. The input type-2 fuzzy sets then activate the inference engine and the rule base to produce output type-2 fuzzy sets. The type-2 FLC rules will remain the same as in type-1 FLC, but the antecedents and/or the consequents will be represented by interval type-2 fuzzy sets. The inference engine combines the fired rules and gives a mapping from input type-2 fuzzy sets to output type-2 fuzzy sets. The type-2 fuzzy outputs of the inference engine are then processed by the type-reducer, which combines the output sets and performs a centroid calculation that leads to type-1 fuzzy sets called the type reduced sets. After the type-reduction process, the type-reduced sets (or approximate type-reduced sets) are then defuzzified (by taking the average of the type reduced/approximated type-reduced set) to obtain crisp outputs [118].

A.4.2 Interval Type-2 Fuzzy Sets

Consider the transition from ordinary sets to fuzzy sets. When we cannot determine the membership of an element in a set as 0 or 1, we use fuzzy sets of type-1. Similarly, When the circumstances are so fuzzy, we have trouble determining the membership grade even as a crisp number [0,1] we use fuzzy sets of type-2, a concept that was first introduced by Zadeh in 1975 [103].

A type-2 set can also be described as the blurring of a type-1. Figure A.6 *a* shows a type-1 membership function, we can ‘blur’ in by shifting the points on the triangle to the left or right, but not necessarily by the same amount, this would generate Figure A.6 *b*. This means that at a specific value of x say x', there no longer is a single value for the membership function; instead, the membership function takes on values wherever the vertical line intersects the blurs.

Calculating all $x \in X$ creates a three-dimensional membership function, a type-2 membership function that characterises a type-2 fuzzy set [100].

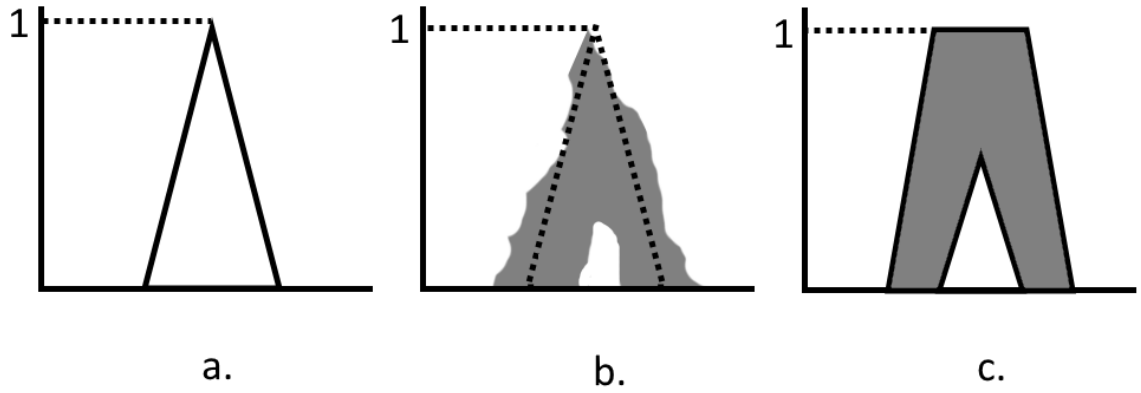

Figure A.6 a) Type-1 Membership Function b) Blurred Type-1 Membership Function c) Footprint of Uncertainty [100]

A type-2 fuzzy set denoted $\tilde{A}$, is chracterised by a type-2 membership function $\mu_{\tilde{A}}(x,u)$ where $x \in X$ and $u \in J_x \subseteq [0,1]$, i.e [100].

$$\tilde{A} = \{(x,u), \mu_{\tilde{A}}(x,u)\} \mid \forall x \in X, \forall u \in J_x \subseteq [0,1] \qquad \text{(A-30)}$$

In which $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$. $\tilde{A}$ can also be expressed as [100]

$$\tilde{A} = \int_{x \in X} \int_{u \in J_x} \mu_{\tilde{A}}(x,u)/(x,u) \quad J_x \subseteq [0,1] \qquad \text{(A-31)}$$

Where $\iint$ denotes union over all admissible x and u [100].

For the discrete universe of discourse, $\int$ is replaced by $\sum$.

In equation (A-30) the first restriction that $\forall u \in J_x \subseteq [0,1]$ is consistent with the type-1 constraint $\leq \mu_A(x) \leq 1$, i.e. when uncertainties disappear, a type-2 membership function must reduce to a type-1 membership function. The second restriction that $0 \leq \mu_{\tilde{A}}(x,u) \leq 1$ is consistent with the fact that the amplitudes of a membership function, should lie, or be equal to 0 and 1. When all $\mu_{\tilde{A}}(x,u) = 1$ then $\tilde{A}$ is an interval type-2 fuzzy set [119].

It has been argued that using interval type-2 fuzzy sets to represent the inputs and/or outputs of FLS has many advantages when compared to the type-1 fuzzy sets; some of these advantages are as follows [118]:

- As the type-2 fuzzy sets membership functions are fuzzy and contain a footprint of uncertainty, then they can model and handle the linguistic and numerical uncertainties associated with the inputs and outputs of the FLS. Therefore, FLSs that are based on type-2 fuzzy sets will have the potential to produce a better performance than the type-1 FLCs when dealing with uncertainties [86].
- Using type-2 fuzzy sets to represent the FLS inputs and outputs will result in the reduction of the FLS rule base when compared to using type-1 fuzzy sets, as the uncertainty represented in the footprint of uncertainty in type-2 fuzzy sets lets us cover the same range as type-1 fuzzy sets with a smaller number of labels and the rule reduction will be greater when the number of the FLS inputs increases [100].
- Each input and output will be represented by a large number of type-1 fuzzy sets, which are embedded in the type-2 fuzzy sets [100] [119]. The use of such a large number of type-1 fuzzy sets to describe the input and output variables allows for a detailed description of the analytical control surface as the addition of the extra levels of classification give a much smoother control surface and response. In addition, according to Karnik and Mendel [120], the type-2 FLS can be thought of as a collection of many different embedded type-1 FLSs.
- It has been shown in [121] that the extra degrees of freedom provided by the footprint of uncertainty enables a type-2 FLS to produce outputs that cannot be achieved by type-1 FLSs with the same number of membership functions. It has been shown that a type-2 fuzzy set may give rise to an equivalent type-1 membership grade that is negative or

larger than unity. Thus, a type-2 FLS can model more complex input-output relationships than its type-1 counterpart and, thus, can give better control response.

A.4.3 Type-Reduction

Many defuzzification methods have been described in section 3.2.7; they involve computing the centroid of a type-1 fuzzy set. An important calculation for type-2 fuzzy logic systems is type-reduction. Type-Reduction represents a mapping of a type-2 fuzzy set into a type-1 fuzzy set [100].

There exist many types of type-reduction, such as centroid, centre-of-sets, height, modified height. However, to illustrate the concept, and the type-reduction method used in later chapters, Centre-of-Sets type reduction is described [122]. Regardless of which type-reduction method is used, the type-reduced set is also an interval set and has the following structure [100]:

$$Y_{TR} = [y_l, y_r] \tag{A-32}$$

Center-of Sets type reduction,Y_{cos}, which can be expressed as [100]:

$$Y_{\cos}(x) = [y_l, y_r] = \int_{y^1 \in [y_l, y_r]} \cdots \int_{y^m \in [y_l^m, y_r^m]} \int_{f^1 \in \left[\underline{f}^1, \overline{f}^1\right]} \cdots \int_{f^1 \in \left[\underline{f}^M, \overline{f}^M\right]} 1 \Big/ \frac{\sum_{i=1}^{M} f^i y^i}{\sum_{i=1}^{M} f^i} \tag{A-33}$$

Where: $Y_{\cos}(x)$ is an interval set determined by its two end-points, y_l and y_r; and $\left[y_l^i, y_r^i\right]$ corresponds to the centroid of the type-2 interval consequence set $\tilde{G}^i$, which can be obtained from [100]:

$$C_{\tilde{G}^i} = \int_{\theta_1 \in J_{y1}} \cdots \int_{\theta_1 \in J_{yN}} 1 \Big/ \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i} = \left[y_l^i, y_r^i\right] \tag{A-34}$$

Note that $\left[y_l^i, y_r^i\right]$ *(i = 1,… M)* must be computed before the computation of $Y_{\cos}(x)$ [100].

A.4.5 Deffuzzification

As $Y_{\cos}$ is an interval set, we defuzzify it using the average of y_l and y_r; hence the defuzzified output of an interval singleton type-2 fuzzy logic system is [100]:

$$Y(x) = f_{s2}(x) = \frac{y_l + y_r}{2} \quad \text{(A-35)}$$

A.5 Design Methods for Fuzzy Logic Systems

As FLSs are expert systems (a system that uses expert knowledge to make decisions) one of the more difficult tasks when designing FLSs is the injection of this knowledge into the fuzzy sets and rules. To assist with this task there are a number of methods used to help produce the most effective systems.

A.5.1 Surveys, Polls and Questionnaires

There are six methods of elicitation to extract the required expert knowledge from relevant experts or groups of individuals [123] (if one is building a fuzzy system using the 'Wisdom of Crowds' principle).

- **Polling**: Do you agree that John is 'Tall' (Yes/No)
- **Direct Rating (Point Estimation)**: Classify colour A according to its darkness, classify John according to his tallness, in general the question is; "How F is a?"
- **Reverse Rating**: Identify the person who is tall to the degree 0.6? In general, identify a who is F to the degree $\mu_F(a)$
- **Interval Estimation (Set Value Statistics):** Give an interval in which you think colour A lies, give an interval in which you think the height of John lies.
- **Membership Function Exemplification:** What is the degree of belonging of the colour A to the (fuzzy) set of dark colours? What is the degree of belonging of John to the set of tall people? In general, to what degree a is F?

- **Pairwise Comparison:** Which colour, A or B, is darker (and by how much?).

A.5.1.1 Polling

In polling, one subscribes to the point of view that fuzziness arises from *interpersonal disagreements*. The question "do you agree that a is F?" is asked to different individuals, the answers are polled, and the average is taken to construct the fuzzy sets. Polling is one of the natural ways of eliciting membership functions for the likelihood interpretation [123].

A.5.1.2 Direct Rating

Direct rating seems to be the most straightforward way to come up with a membership function; this approach subscribes to the point of view that fuzziness arises from *individual subjective vagueness*. The subject is required to classify a with respect to F over and over again in time. The experiment has to be carefully designed so that it will be harder for the subject to remember past answers [123].

A.5.1.3 Reverse Rating

In this method, the subject is given a membership degree and then asked to identify the object for which the degree corresponds to the fuzzy term in question. This method can be used for individuals repeating the same question for the same membership function as well as for a group of individuals.

Once the subject's (or subjects') responses are recorded the conditional distribution can be taken to be normally distributed, and the unknown parameters (mean and variance) can be estimated as usual. This method also requires evaluations to be made on at least interval scales [123].

A.5.1.4 Interval Estimation

Interval estimation subscribes to the *random-set* view of the membership function. The subject is asked to give an interval that describes *a.* Let I_i be the set-valued observation (the interval) and m_i the frequency with which I_i is observed. The $R = (I_i, m_i)$ defines a random set. Notice that this method is more appropriate to situations where there is a clear linear ordering in the measurement of the fuzzy concept, like in tallness, heat, time, etc [123].

A.5.1.5 Membership Function Exemplification

Regarding membership function exemplification, Hersh & Carmazza [124]performed a test for the direct elicitation of the membership function. In the test, they ordered 12 squares in ascending order and indicated each square with an ordinal number. They asked the subjects "Write the number(s) which is appropriate for 'large', 'very large', 'small' etc. The results are at variance with direct rating and polling most likely because there is no repetition in this elicitation method to normalise the effects of error or 'noise' [123].

The use of computer graphics to give an example membership function to be modified by the subject greatly enhanced the procedure as is usually witnessed in commercial applications of "fuzzy expert system shells" [123].

A.5.1.6 Pairwise Comparison

Chameau and Santamarina [125] use pairwise comparison technique and report it to be as robust as polling and direct rating. They require the subjects to provide pairwise comparisons and the strength of preference. This yields a non-symmetrical full matrix of relative weights. The membership function is found by taking the components of the eigenvector (a vector which when operated on by a given operator gives a scalar multiple of itself) corresponding to the maximum eigenvalue (any number such that a given matrix minus that number times the

identity matrix has zero determinant). The values are also normalised. Chameau and Santamarina also find the requirement that evaluations on a ratio scale to be unnatural.

However, they espouse a 'comparison-based point estimation' which determines the position of a set of stimuli on the reference axis by pairwise comparison and the membership is calculated by aggregating provided by several subjects. Although the subjects of Chameau and Santamarina experiments ranked the method almost as good as the interval estimation method (which was ranked as the best method) this method also needs the unfortunate assumption of a ration scale. Furthermore, pairwise comparison requires many comparison experiments in a relatively simple domain [126].

A.5.2 Fuzzy Systems from Examples

Wang and Mendel developed a well-known method for developing fuzzy systems from examples, combining both expert knowledge and numerical data examples [127]. They proposed a five-step generalised method for constructing these fuzzy systems, with emphasis on generating fuzzy rules by learning from examples.

Suppose we are given a set of desired input-output data pairs [127]:

$$\left(x_1^{(1)}, x_2^{(1)}; y^{(1)}\right), \left(x_1^{(2)}, x_2^{(2)}; y^{(2)}\right), \ldots \quad \text{(A-36)}$$

Where x_1 and x_2 are inputs and y is an output. This simple two-input one-output case is chosen in order to emphasize and to clarify the basic ideas of the Wang and Mendel approach; extensions to general multi-input multi-output cases are straightforward. The task here is to generate a set of fuzzy rules from the desired input-output pairs of (1), and use these fuzzy rules to determine a mapping $f : (x_1, x_2) \rightarrow y$. The Wang and Mendel approach consists of the five following steps [127]:

Step 1 – Divide the input and output spaces into fuzzy regions.

Assume that the domain intervals x_1, x_2 and y are $[x_1^-, x_1^+]$, $[x_2^-, x_2^+]$ and $[y^-, y^+]$, respectively, where "domain interval" of a variable means that most probably this variable will lie in this interval (the values of a variable are allowed to lie outside its domain interval). Divide each domain interval into 2N + 1 regions (N can be different for different variables, and the lengths of these regions can be equal or unequal), denoted by SN (Small N), …,S1 (Small 1), CE (Centre), B1 (Big 1), …, BN (Big N), and assign each region a fuzzy membership function [127].

Figure A.7 shows an example where the domain interval $x_1$is divided into five regions (N = 2), the domain region of $x_2$is divided into seven regions (N = 3), and the domain interval of y is divided into five regions (N = 2). The shape of each membership function is triangular; one vertex lies at the centre of the region and has membership value unity; the other two vertices lie at the centres of the two neighbouring regions, respectively, and have membership value equal to zero. Of course, other divisions of the domain regions and other shapes of membership functions are possible [127].

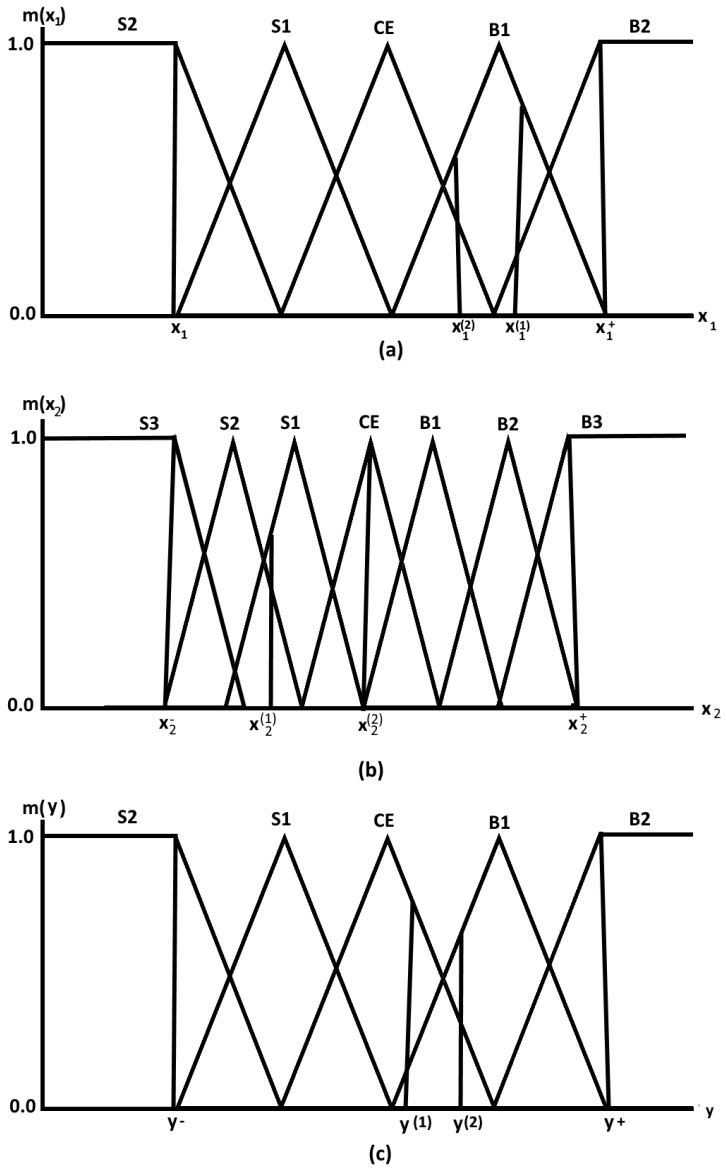

Figure A.7: Division of Domain Intervals [127]

Step 2 – Generate fuzzy rules from given data pairs.

First, determine the degrees of given $x_1^{(i)}, x_2^{(i)}$ and $y^{(i)}$ in different regions. For example, $x_1^{(1)}$ in Figure A.7 has degree 0.8 in B1, degree 0.2 in B2 and zero degrees in all other regions. Similarly, $x_2^{(2)}$ in Figure A.7 has degree 1 in CE, and zero degrees in all other regions [127].

Second, assign a given $x_1^{(i)}$, $x_2^{(i)}$ or $y^{(i)}$ to the region with maximum degree. For example, $x_1^{(1)}$in Figure A.7 is considered to be B1, and $x_2^{(2)}$ is considered to be CE [127].

Finally, obtain one rule from one pair of desired input-output data, e.g [127].,

$$\left(x_1^{(1)}, x_2^{(1)}; y^{(1)}\right) \Rightarrow \begin{bmatrix} x_1^{(1)}(0.1\ in\ B1, max), x_2^{(1)}(0.7\ in\ S1, \max); \\ y^{(1)}(0.9\ in\ CE, max) \end{bmatrix} \Rightarrow Rule\ 1\text{: (0-37)}$$

IF x_1 is B1 and x_2 is S1, THEN y is CE [127];

$$\left(x_1^{(2)}, x_2^{(2)}; y^{(2)}\right) \Rightarrow \begin{bmatrix} x_1^{(2)}(0.6\ in\ B1, max), x_2^{(2)}(1\ in\ CE, \max); \\ y^{(2)}(0.7\ in\ B1, max) \end{bmatrix} \Rightarrow Rule\ 2\text{: (0-38)}$$

IF x_1 is B1 and x_2 is CE, THEN y is B1;

The rules generated in this way are "and" rules, i.e., rules in which conditions of the IF part must be met simultaneously in order for the result of the THEN part to occur [127].

Step 3 – Assign a degree to each rule

Since there are usually lots of data pairs, and each data pair generates one rule, it is highly probable that there will be some conflicting rules, i.e., rules that have the same IF part but a different THEN part. One way to resolve this conflict is to assign a degree to each rule generated from data pairs, and accept only the rule from a conflict group that has maximum degree. In this way not only is the conflict problem resolved, but also the number of rules is greatly reduced [127].

We use the following product strategy to assign a degree to each rule: for the rule: "IF x_1 is A and x_2 is B, THEN y is C" the degree of this rule, denoted by D(Rule), is defined as [127]:

$$D(Rule) = m_A(x_1)m_B(x_2)m_C(y) \quad \text{(A-39)}$$

As examples, Rule 1 has degree [127]

$$\begin{aligned} D(Rule\ 1) &= m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y) \\ &= 0.8 \times 0.7 \times 0.9 = 0.504 \end{aligned} \quad \text{(A-40)}$$

(see Figure A.7) and Rule 2 has degree [127]

$$D(Rule\ 2) = m_{B1}(x_1)m_{CE}(x_2)m_{B1}(y) \quad \text{(A-41)}$$
$$= 0.6 \times 1.0 \times 0.7 = 0.42$$

In practice, we often have some prior information about the data pair. For example, if we let an expert check given data pairs, the expert may suggest that some are very useful and crucial, but others are very unlikely and may be caused just by measurement errors. Therefore, we can assign a degree to each data pair that represents our belief of its usefulness. In this sense, the data pairs constitute a fuzzy set, i.e. the fuzzy set is defined as the useful measurements; a data pair belongs to this set to a degree assigned by a human expert [127].

Suppose the data pair $\left(x_1^{(1)}, x_2^{(1)}; y^{(1)}\right)$ has degree $m^{(1)}$, then we redefine the degree of Rule 1 as [127].

$$D(Rule\ 1) = m_{B1}(x_1)m_{S1}(x_2)m_{CE}(y)m^{(1)} \quad \text{(A-42)}$$

i.e., the degree of a rule is defined as the product of the degrees of its components and the degree of the data pair that generates this rule. This is important in practical applications because real numerical data have different reliabilities, e.g., some real data can be very bad ("wild data"). For good data, we assign higher degrees, and for bad data, we assign lower degrees. In this way, human experience about the data is used in a common base as other information. If one emphasises objectivity and does not want a human to judge the numerical data, our strategy still works by setting all the degrees of the data pairs equal to unity [127].

Step 4 – Create a combined fuzzy rule base

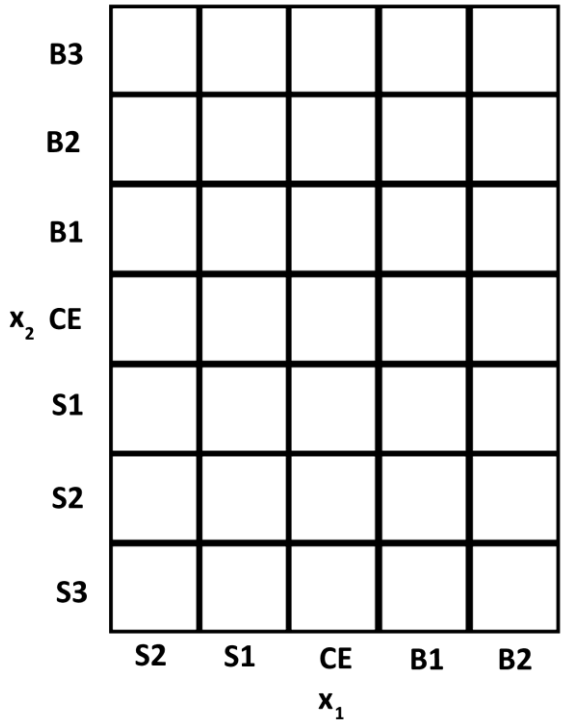

Figure A.8: Form of Fuzzy Rule Base [127]

The form of a fuzzy rule base is illustrated in Figure A.8. We will fill the boxes of the base with fuzzy rules according to the following strategy: a combined fuzzy rule base is assigned rules from either those generated from the numerical or linguistic rules (we assume that a linguistic rule also has a degree that is assigned by the human expert and reflect the experts belief of the importance of the rule); if there is more than one rule in one box of the fuzzy rule base, use the rule that has maximum degree.

In this way, both numerical and linguistic information is codified into a common framework – the combine fuzzy rule base. If a linguistic rule is an "and" rule, it fills only one box of the fuzzy rule base; but, if a linguistic rule is an "or" rule (i.e., a rule for which the THEN part follows if any condition of the IF part is satisfied), it fills all the boxes in the rows or columns corresponding to the regions of the IF part. For example, suppose we have the linguistic rule: "IF $x_1$is S1 or x_2 is CE, THEN y is B2" for the fuzzy rule base of Figure A.8; then we fill

the seven boxes in the column of S1 and the five boxes in the row of CE with B2. The degree of all the B2's in these boxes equal the degree of this "or" rule [127].

Step 5 – Determine a mapping based on the combined fuzzy rule base.

We use the following defuzzification strategy to determine the output control y for given inputs (x_1, x_2): first, for given inputs (x_1, x_2), we combine the antecedents of the ith fuzzy rule using product operation to determine the degree, m_{Oi}^{i}, of the output control corresponding to (x_1, x_2), i.e. [127],

$$m_{Oi}^{i} = m_{I_1^i}(x_1) m_{I_2^i}(x_2) \tag{A-43}$$

Where O^i denotes the output region of Rule i, and I_j^i denotes the input region of Rule i for the jth component, e.g., Rule 1 gives [127]

$$m_{CE}^{1} = m_{B!}(x_1) m_{S1}(x_2) \tag{A-44}$$

Then we use the following centroid defuzzification formula to determine the output [127].

$$y = \frac{\sum_{i=1}^{K} m_{O^i}^{i} \bar{y}^i}{\sum_{i=1}^{K} m_{O^i}^{i}} \tag{A-45}$$

Where $\bar{y}^i$ denotes the centre value of region O^i (the centre of a fuzzy region is defined as the point that has the smallest absolute value among all the points which the membership function for this region has membership value equal to one), and K is the number of fuzzy rules in the combined fuzzy rule base [127].

A.5.3 Genetic Algorithm Optimised Fuzzy Logic Systems

Wagner and Hagras developed an architecture for evolving the parameters of a fuzzy logic system (both Type-1 and Type-2 FLSs) using a genetic algorithm [89]. The purpose of using a genetic algorithm is because they do not require a priori knowledge such as a model or data but perform a search through the solution space based on natural selection using a specified fitness function. A more in-depth discussion of genetic algorithms and other evolutionary techniques can be found in Chapter 4.

Wagner and Hagras demonstrate their technique on interval type-2 fuzzy sets, which use Gaussian primary membership functions, with uncertain standard deviation. Their genetic algorithm system uses real value encoding to encode each gene in the chromosome. The genetic algorithm based system procedure can be summarised as follows [89]:

Step 1: 30 chromosomes are generated randomly while taking into account the grammatical correctness of the chromosome (for example the inner standard deviation σ_1 is less than the outer standard deviation σ_2). The "Chromosome Counter" is set to 1 (the first chromosome). The "Generation Counter" is set to 1 (the first generation) [89].

Step 2: A type-2 FLS is constructed using the chromosome identified by the "Chromosome Counter". The environment in which the FLS is tested is set up (which could be a simulation), and the fitness of the current controller is evaluated, based on how the chromosome performed in the environment. Any chromosome that fails any of the primary test conditions is automatically given a disastrous fitness [89].

Step 3: If "Chromosome Counter" < 30, increment "Chromosome Counter" by 1 and go to Step 2, else proceed to Step 4 [89].

Step 4: The best individual-so-far's performance is preserved separately [89].

Step 5: If "Generation Counter = 1 then store current population, copy it to a new population *P* and proceed to Step 6. Else, select 30 best chromosomes from population "Generation Counter" and population "Generation Counter"-1 and create a new population *P* [89].

Step 6: Use roulette wheel selection (See Section 3.1.2.1) on population P to populate the new breeding pool [89].

Step 7: Crossover (See Section 3.1.2.2) is applied to chromosomes in the breeding pool and "chromosome consistency" is checked (*) [89].

Step 8: "Generation Counter" is incremented. If "Generation Counter" < the number of maximum generations, or if the desired performance is achieved, reset "Chromosome Counter" to 1 and go to Step 2, else go to Step 9 [89].

Step 9: The chromosome which resulted in the best fitness is kept, and the solution has been achieved; END [89].

(*) The crossover operator used here computes the arithmetic average between two genes. With chromosome consistency, it is refereeing to the correctness of the genes of the chromosomes in relation to their function in the FLS. Chromosomes are completely eliminated if they violate this criterion or if the problem is restricted to the means of the MFs (for example the mean of the membership function "Far" < mean of the membership function "near") [89]. Figure A.9 shows an example of how a chromosome would look for a four input four output Gaussian membership function based FLS.

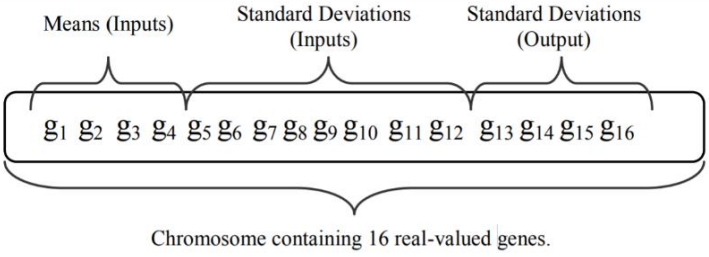

Figure A.9: Chromosome Structure for Optimising Fuzzy Sets [89]

A.5.4 Particle Swarm Optimised Fuzzy Logic Systems

Another method of optimisation by Evolutionary Algorithms (EAs) is to use particle swarm optimisation. A more in-depth discussion on PSO can be found in Chapter 4.2.

Several researchers have shown that PSO can also be used to tune the membership functions of a fuzzy set [128] [129]. In [128] it is shown that by correlating the fitness of the PSO with the Route Mean Squared Error (RMSE) of the fuzzy systems output, the performance of the fuzzy systems can be improved. One important find from [128] is that the PSO search process is given an advantage if the fuzzy system is first designed by an expert. This ensures that the start of the PSO is not completely random. It was also found that optimising the output membership functions had more influence on the performance of the fuzzy model.

The process of integrating the PSO algorithm with fuzzy control is as follows [129]:

1. The subpopulation is defined as a link of the membership functions adjustment values.
2. The parameters are the centers and widths of each fuzzy set. These parameters compose the particle (agent)
3. To check the performance of the fuzzy system it is rolled up from an initial set of possible parameters
4. This information is used to set up each sub-population adjustment (adaptability and the making of the evolution of the population.
5. The cycle repetition is made up for completion of the defined PSO iteration number made by the user. To each PSO iteration is found the best value set for the membership function parameters.

A.5.6 Adaptive Fuzzy Logic Systems

As described by Cox [130] an adaptive fuzzy logic system adjusts to time and processed phased conditions, and also changes the supporting system controls. This means that an adaptive system modifies the characteristics of the rules, the topology of the fuzzy sets, and the method of defuzzification based on predictive convergence metrics (or more simply, how quickly it is approaching or leaving a goal state). In the way, they work adaptive fuzzy systems resemble neural networks. Both systems are trained through a performance metric usually a set of cases indicating an input and desired output; and both act as classifiers, where the classification space is intensified by changes to weights that are adjusted according to how much the system is in error [130].

An adaptive fuzzy system, however, is much more sophisticated and has a higher degree of adaptive parameters. Such systems are able to deal with their human partners since they can, in effect, explain their reasoning – a task that neural networks do rather poorly, or not at all [130].

Adaptive systems usually work like back-propagation in neural networks, by examining a solution with a target result. Like their neural network counterparts, fuzzy systems can run in both supervised and unsupervised (or autoadaptive) mode. However, unlike neural systems, fuzzy systems are more likely to run unsupervised by the very nature of their internal organisation and a priori knowledge base [130].

Regardless of the adaptation method used, there are several interconnected means of allowing fuzzy systems to adapt. These include the management of weights attached to the rules (a concept spoken about in 0), the dynamic hedging of the fuzzy regions, the structural modification of the fuzzy sets, the redefinition of truth in the fuzzy model and the selection of alternative methods of defuzzification [130].

In contrast to neural networks, the weights in an adaptive fuzzy system are associated with the rule nodes, not with the connecting edges of the network. Training, however, is conducted in a manner analogous to neural backpropagation. The error discriminant is propagated back to the rules. In general, the weight modification algorithm is fairly simple: it consists of multiplying the various W_is by an error factor. In supervised learning the error factor is the ratio of the actual system output to the correct output. In an autoadaptive mode, it is the mean squared distance from the centre of the optimal control region to the center of the system response. An additional factor the fuzzy attenuation control, is sometimes included in the multiplication. If included this parameter attenuates, or controls, the strength of the training applies to each rule's contribution weight. It is analogous to the training rate parameter that is to be found in certain neural networks [130].