Type-2 Fuzzy Single and Multi-Objective Optimisation Systems for Telecommunication Capacity Planning

Lewis H. F. Veryard

A thesis submitted for the degree of Doctor of Philosophy in

Computer Science

School of Computer Science and Electronic Engineering

University of Essex

2022

Acknowledgements

I would like to thank the following people for their contributions to this PhD and my career for whom I shall always be grateful:

Professor Hani Hagras, the academic supervisor, I would like to extend my thanks to you, without your guidance, support and tutorage over the last 4 years none of this would have been possible. He has been an outstanding mentor and role model truly inspiring my work ethic and has given me an amazing start to my carrier.

Dr Gilbert Owusu, one of the industrial supervisors, I would like to thank you for the opportunities that you presented to me and giving me the opportunity to work with British Telecom.

Anthony Conway, one of the industrial supervisors, I would like to thank you will helping me be an effective member of your team at British Telecom, granting me your wisdom and experience in Artificial Intelligence applications. Without your guidance and knowledge the real-world applications of my work would have been severely diminished.

I would like to thank British Telecom for funding and supporting my Ph.D.

Finally I would like to thank all my friends and family for their love and support throughout my Ph.D.

Publications arising from this work

Journal Papers

- L. Veryard, H. Hagras, A. Starkey, A. Conway, and G. Owusu, “NNIR: N-non-intersecting-routing algorithm for multi-path resilient routing in telecommunications applications,” *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 352–365, 2020.
- L. Veryard, H. Hagras, A. Conway, and G. Owusu, “A Heated Stack based Type-2 Fuzzy Multi-Objective Optimisation System for Telecommunications Capacity Planning” Knowledge Based Systems, 2022.

  (UNDER REVIEW)

Conference Papers

- L. Veryard, H. Hagras, A. Starkey, and G. Owusu, “A Fuzzy Genetic System for Resilient Routing in Uncertain Dynamic Telecommunication Networks,” *Proceedings of the IEEE International Conference on Fuzzy Systems*, New Orleans, USA, June, 2019.
- L. Veryard, H. Hagras, A. Conway, and G. Owusu, “A Type-2 Fuzzy Genetic Approach to Uncertain & Dynamic Resilient Routing within Telecommunications Networks,” *Proceedings of the IEEE International Conference on Fuzzy Systems*, Glasgow, UK, July, 2020.
- L. Veryard, H. Hagras, A. Conway, and G. Owusu, “A Type-2 Fuzzy Multi-Objective Multi-Chromosomal Optimisation for Capacity Planning within Telecommunication Networks,” *Proeedings of the IEEE International Conference on Fuzzy Systems*, Luxenbourg, July, 2021.

Patents

- L. Veryard, A. Starkey, H. Hagras, and G. Owusu, A33790SUSw01 “Resilient Network Routing” 2019.
- L. Veryard, A. Hardwick, and A. Conway, A35662 “Resource Capacity Management” 2021.

Abstract

Capacity planning in the telecommunications industry aims to maximise the effectiveness of implemented bandwidth equipment whilst allowing for equipment to be upgraded without a loss of service. The better implemented hardware can be configured, the better the service provided to the consumers can be. Additionally, the easier it is to rearrange that existing hardware with minimum loss of service to the consumer, the easier it is to remove older equipment and replace it with newer more effect equipment. The newer equipment can provide more bandwidth whilst consuming less power and producing less heat, lowering the overall operating costs and carbon footprint of a large scale network.

Resilient routing is the idea of providing multiple independent non-intersecting routes between two locations within a graph. For telecommunications organisations this can be used to reduce the downtime faced by consumers if there is a fault within a network. It can also be used to provide assurances to customers that rely on a network connection such as: financial institutions or government agencies.

This thesis looks at capacity planning within telecommunications with the aspiration of creating a set of optimisation systems that can rearrange data exchange hardware to maximise their performance with minimal cost and minimising downtime while allowing adaptations to an exchange's configuration in order to perform upgrades.

The proposed systems were developed with data from British Telecom (BT) and are either deployed or are planned to be in the near future. In many cases the data used is confidential, but when this is the case an equivalent open source data set has been used for transparency.

As a result of this thesis the Heated Stack (HS) algorithm was created which has been shown to outperform the popular and successful NSGA-II algorithm by up to 92 % and NSGA-III by up to 69% at general optimisation tasks. HS also outperforms NSGA-II in 100% of the physical

capacity planning experiments run and NSGA-II in 68% of the physical capacity planning experiments run. Additionally, as a result of this thesis the N-Non-Intersecting-Routing algorithm was shown to outperform Dijkstra's algorithm by up to 38% at resilient routing. Finally, a new method of performing configuration planning through backwards induction with Monte Carlo Tree Search was proposed.

Table of Contents

List of Figures

List of Tables

List of Acronyms

AI: Artificial Intelligence

BT: British Telecom

CHT: Constraint Handling Techniques

CMOP: Constrained Multi-Objective Problem

CSV: Comma Separated Value

CVLan: Customer Virtual Local Area Network

FLC: Fuzzy Logic Controller

FoU: Footprint of Uncertainty

GA: Genetic Algorithm

HA: Heated Stack Algorithm

HVI: Hyper Volume Indicator

IT2FLC: Interval Type-2 Fuzzy Logic Controller

KP: Knapsack Problem

LAN: Local Area Network

MCTS: Monte-Carlo Tree Search

NNIR: N-Non-Intersecting Routing Algorithm

SA: Simulated Annealing

SPoF: Single Point of Failure

SVLan: Service Virtual Local Area Network

TSP: Traveling Salesman Problem

UK: United Kingdom

VLan: Virtual Local Area Network

Chapter 1. Introduction

1.1 Introduction to network capacity optimisation and planning

The internet, data and interconnectivity are a fact of modern life being used as part of every aspect of people lives, from communication to commerce, from work to entertainment it is an inescapable fact of life that network connections are used for everything. This has only been compounded by the Covid-19 global pandemic with people remaining physically apart and relying more and more on network connections to maintain a normal life. In the United Kingdom this network is underpinned and maintained by the organisation known as British Telecom (BT).

BT maintains a network comprised of two divisions; the access network and the core network. The access network is used to connect consumers to the core network and utilises a variety of technologies including: copper cables, fibre optic cables, satellites and cellular networks such as 4G and 5G, and it can be extended to use any number of new technologies. The core network connects the nation together and to the rest of the world allowing access to the internet and is made up of data exchanges and high capacity fibre optic cabling. The work undertaken as part of this thesis focuses on the core network solving a variety of problems in network planning and routing.

Routing is the act of finding a path between locations with a navigable area, be this roads, a data network or the ocean. Routing is an everyday fact of life, many aspects of people's everyday using routing to achieve a goal these include: public transport [1], resource gathering [2], data networks [3], video games [4], and shipping [5]. In most cases of routing the idea is to find the shortest path between two locations, but this is not always the case sometimes there are other metrics that need to be considered. Sometimes the shortest path is not the optimal one, there could be other considerations that need to be taken into account such as in aviation,

the need to fly around a county with a closed airspace, or in the case of a bus or lorry the roads taken must be suitable for the vehicle to traverse.

As an increasing number of people connect more and more devices to the internet ever-present demand for bandwidth is set to increase [6]. To keep up with this demand networking infrastructure must be upgraded and improved in a constant never-ending cycle. Telecommunication networks are a complicated interconnected web of equipment, cables and locations making upgrades a complex, expensive and time consuming procedure.

Capacity optimisation and planning are widely used terms that are used for any number of applications including: supply chains [7], optical networks [8] bandwidth predications [9] and resource management in manufacturing [10]. In all of these cases there is a general overarching consensus that capacity management is the maximisation of resources in order to get the highest utilisation possible. In the case of supply chains this would be ensuring that each part of the supply chain is used at its most efficient be it transportation or storage. In the case of optical networks capacity management refers to the fibre optic cables and the associated hardware. For the purposes of this thesis the terms capacity management and capacity planning refer to telecommunication networks in relation to their bandwidth and routing capacities.

1.2 Research questions and contributions to science

Given that this thesis uses real-world data and solves real-world problems it is important to take into account the considerations of BT, in the form of their business objectives. Given that, a consumer is defined as someone that uses the telecommunications network for data purposes including: households, corporate entities or a government organisations. There are two important consumer focused business objectives of telecommunications capacity planning and optimisation that must be followed. First a consumer's connection must be reliable with disconnections being minimised and as infrequent as possible, but if a consumer is

disconnected they must be reconnected as quickly as possible. Secondly the bandwidth capacity given to a consumer must increase with the demands of a modern interconnected society. This thesis embodies these business objectives with its overarching goal which is to improve the speed, effectiveness and reliability of capacity planning and optimisation within large scale telecommunication networks by decreasing down time and increasing the speed at which upgrades can be performed to an exchange. In order to achieve this goal a subset of aims must achieved:

I. How to provide assurances of the steps taken to keep a reliable connection between two endpoints in a complex network.
II. How to provide those same assurances in a network when the data used for routing is uncertain or in constant flux.
III. How an ideal version of the current networking infrastructure would be configured to allow planned upgrades to be implemented.
IV. How to make changes to the network infrastructure without disconnecting consumers for extended periods of time.

In order to achieve these 4 aims 5 interconnected tasks have been achieved and will be discussed in this thesis which uphold the business objectives of telecommunication capacity planning and management:

- Perform a comparison into techniques that can perform multi-path resilient routing and determine if one can outperform the currently implemented technique of Dijkstra's algorithm.
- A comparison of type-1 and interval type-2 fuzzy logic for uncertainty handling in resilient routing as part of the selected aforementioned technique resulting in effective and consistent multi path resilient routing in uncertain environments.

- The creation of a novel evolutionary algorithm to optimise virtual groups within exchange hardware which are used for bandwidth allocation.
- Improving the aforementioned novel evolutionary algorithm to optimise existing exchange hardware allowing planned upgrades to exchange equipment.
- A step-by-step guide of how to implement changes to the exchange allowing for the optimised solutions to become reality whilst minimising consumer downtime.

The work presented within this thesis is being used to solve real world problems faced by BT and are used as improvements to pre-existing solutions. BT had an automated solution to problem I mentioned above but the introduction of the new technique described in this thesis has a 38% improvement over the pre-existing technique by decreasing costs and increasing network reliability. BT had no solution to problem II prior to the implemented solution presented in this thesis. There was no automated solution to problem III prior the work undertaken by this thesis and the optimisation was being completed by hand. Finally there was no automated solution to problem IV with the planning being completed by hand.

1.3 Thesis layout

This thesis is structured as follows: Chapter 2 gives an overview of the problems and some of the existing solutions to capacity planning and routing. Introducing what an optimisation problem is, describing some relevant generic computer science problems and discussing the composition of the communications network provided by BT. It then introduces routing whilst establishing some popular and effective routing methods, providing a detailed description of resilient routing whilst describing what makes an optimal resilient route and why they are required. Chapter 2 describes the concept of an uncertain environment and gives a detailed overview into capacity planning in telecommunications.

Chapter 3 describes fuzzy logic systems giving a detailed description of fuzzy logic in general whilst describing the difference between type-1 and interval type-2 fuzzy logic.

Chapter 4 provides an overview of simulated annealing and genetic algorithms. Beginning by describing the functionality of the single objective techniques of simulated annealing and genetic algorithms. It then moves on to the more complex and capable multi-objective genetic algorithms of NSGA-II and NSGA-III, finalising with constrain handling techniques and an introduction to the hyper volume indicator.

Chapter 5 presents the N-Non-Intersecting-Routing algorithm (NNIR) and how it is used to solve the resilient routeing problem with the experiments performed on a real-world confidential telecoms network and an open source road network.

Chapter 6 expands upon the work from the NNIR algorithm in Chapter 5. Applying fuzzy logic to the NNIR algorithm allowing it to operate in uncertain and dynamic environments.

Chapter 7 introduces the Heated Stack algorithm (HS) describing its implementation and comparing it to NSGA-II at the digital capacity planning problem.

Chapter 8 expands upon the HS introduced in Chapter 7 making improvements to its performance and applying it to the physical capacity planning problem. In this chapter HS is compared to NSGA-III at the physical capacity planning problem. Due to the confidentiality of the data used in the capacity planning problems in Chapters 7 and 8, HS is compared to a set of open source optimisation problems ensuring transparency in the results.

Chapter 9 presents Monte Carlo Tree Search with backwards induction for network configuration planning. This technique is used as a final solution sorting and representation tool for capacity planning with HS.

Finally, Chapter 10 presents the conclusion of this thesis, the real world impact and the potential future work.

Chapter 2. Overview of Capacity planning and routing

2.1 What is an optimisation problem

In the English language the word “problem” can be defined as a question to be considered, solved or answered. The word “optimisation” can be defined as the design of a system or process to make it as good as it can possibly be. Therefore, by bringing these two definitions together it can be determined that at the very least an optimisation problem is the design of a system or process that needs to be solved or answered. Optimisation problems exist throughout the everyday life of most people and they solve them without even thinking about it. These can be simple problems such as “What is the fastest way to travel from home to work”, or “How do I get all of the cups and plates into a dishwasher”. These problems increase in complexity to problems such as “How to design a car chassis” [11] or “ How to design the most efficient radio antenna” [12]. So optimisation problems can come in varying degrees of complexity with some problems being trivial to solve and others being more complex. In Computational Complexity theory a problem’s decision complexity can be classified by its NP-Hardness, where a harder problem cannot be solved by a non-deterministic Turing machine in polynomial time [13].

2.1.1 Multi armed bandit problem

An optimisation problem can be tackled from many different perspectives thus it is important to understand how to solve a problem with the correct methodology. In machine learning there is the theoretical problem of the Multi-Armed Bandit [14],[15] where a set of sequential decisions can be enacted to achieve a reward. The goal of the multi armed bandit problem is to with minimal decisions maximise the reward or pay out. The classic example of this is the casino gambler in front of a row of slot machines, where they want to win on a slot machine whilst spending as little money as possible. If the decision faced by the gambler is which of the slot machines to play and for how long for until they try a different machine, then they must

use their limited funds in the most optimal way in order to maximise the rewards. By taking the gambler's problem we can determine that there are two methods of solving methodologies to solve this problem; Exploration and Exploitation. In exploration the gambler could in-turn make one decision on each slot machine, only repeating a slot machine once they have all been visited, continuing this process until they exhaust their supply of money with the hope that at least one of the slot machines will pay out. In exploitation the gambler could pick one slot machine and only play that one using their entire money supply in the hopes that eventually the slot machine will pay out. A plan could be formulated for the gambler by using both of these techniques in varying degrees allowing of them to both explore and exploit.

This idea of exploration and exploitation is a core idea of optimisation and they are regularly seen as opposite to each other where a balance must be achieved in order to attempt to achieve a global optima. Exploration is allowing a worse state to arise in order to find better states later, whereas exploitation is only allowing better states to exist and never making a state worse.

2.1.2 Evaluation and Progressive Improvements

Problems can be solved through the use of iterative and exhaustive methods. These methods look at a set of parameters that can be manipulated and change them in order to find the best solution. The idea of exhaustively searching through all the potential solutions could work for the simplest problems, but as problems become more complex they require more intelligent optimisation strategies. Once a problem becomes increasingly complex and its NP-Hardness increases it can become impossible to exhaustively search all possible states. Therefore, computational optimisation strategies are used to make informed decisions to achieve a goal.

Given a problem has a goal and a set of decision variables representing the state of the problem and that the same decision variables can be manipulated, an optimisation strategy can be used in order to attempt to generate optimal solution [16]. An optimisation strategy should be able

to make progressive improvement upon a state until it terminates or no better solution can be found. In order for an optimisation strategy to make these progressive improvements it needs to be able to define the quality of a potential solution which is determined by evaluating a solution against a goal [16]. This goal should be represented by a mathematical function that uses the information represented in the decision variables to determine a score. This score is either a minimisation or maximisation function to determine how well a potential solution fulfils the goals of the optimisation problem.

2.2 Generic Computer Science Problems

Within the field of computer science there are many theoretical problems that can be attributed to real-world problems. These problems can used as a good basis to solve and understand many real-world problems.

2.2.1 The Knapsack Problem

In the Knapsack Problem (KP) there is a given set of items which each have a value and weight and that given sack or container can contain up to a set weight. A subset of the item must be selected to maximise the value within the container without going over the weight constraint [17]. Fig 2.1 shows a visualisation of this problem with five items of varying weight and value, and also a sack that can contain up to a maximum weight of 15kg. Which of these items should be put into the sack in order to maximise the value in the bag? What if the objective was not to just maximise the value in the sack but to also use as much of the sack's weight constant, does that change the solution?

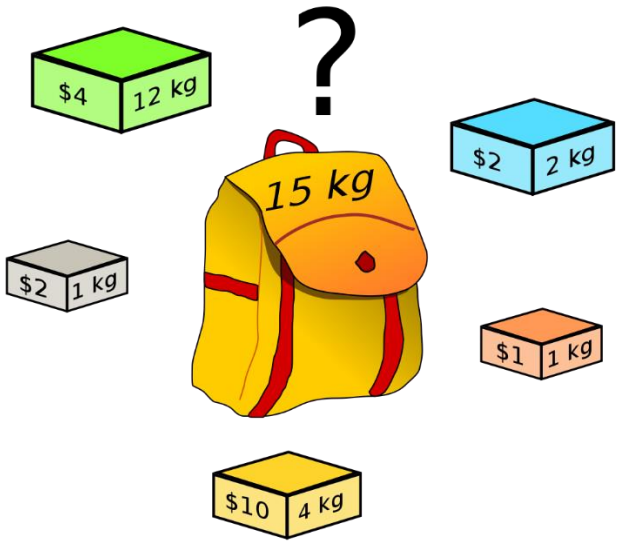


Fig 2.1 Visualisation of the knapsack problem [18]

It poses the question of which item to put into the bag and which to leave out to be the most efficient. The KP can be extended and manipulated to have many more objectives and constraints put upon it. The KP is known to be a NP-Complete optimisation and is a well-researched area of computer science [19], [20] and can be applied to a multitude of real world problems.

2.2.2 The Bin Packing Problem

The Bin Packing problem, is defined as set of objects with varying size and an infinite number of bins or containers of equal size. The objective is to put all of the objects into the minimum number of bins as possible, without exceeding the maximum size limit of the bin, thus packing the bins as efficiently as possible [21] [22]. Additional constraints can be put on each bin, such as having a minimum or maximum number of items in each bin, or not allowing two certain objects to be present in the same bin.

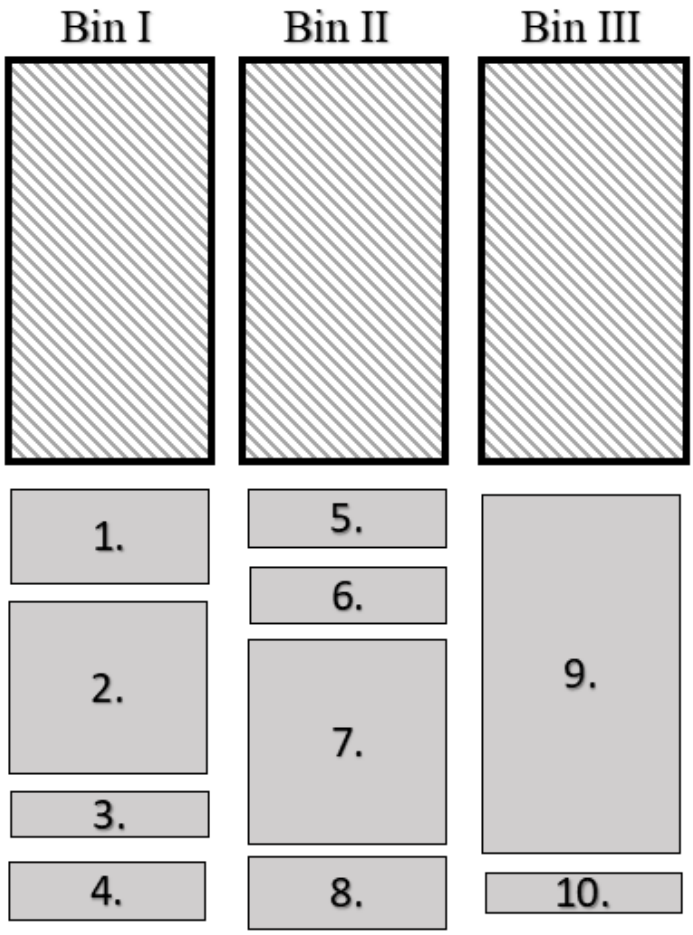


Fig. 2.2 Unsolved bin packing example.

Fig 2.2 shows an example of the Bin Packing problem with 10 items of varying size that need to fit into a number of bins with a fixed size, this example shows three bins but more could be used if they didn't fit, as the number of bins is infinite.

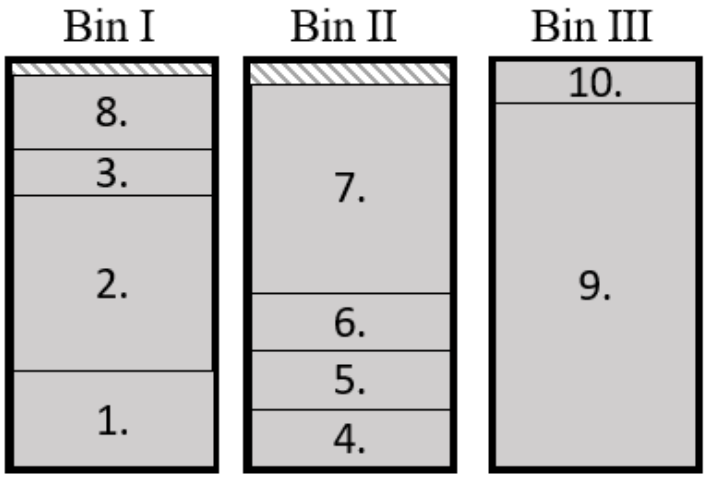


Fig 2.3 Solved bin packing example

Fig 2.3 shows the solved version of Fig 2.2 where the 10 items have been allocated to their bins. None of the bins constraints have been violated and all of the items have been packed. The items do not need to be packed in order to achieve the most efficient packing as seen in Bin I where items 1, 2, 3, 8 have been packed. Bin packing is a well-researched problem area with many different algorithms solving many different versions of the problem [23].

2.3 Telecommunications Infrastructure Composition and Problems

The internet, data and interconnectivity have become integral to the modern way of life, being used as part of their personal or professional lives. People require a fast and reliable internet connection across the nation. In the United Kingdom this service is predominantly supplied by British Telecom (BT), whose services are broken down into two divisions, BT and Open Reach. BT provide the core network which connects a set of exchanges together and to the wider world. Open Reach provide access network which connects consumers and business to the core network. Open Reach provides the access network through a wide variety of technologies such as: copper cable, fibre optics, radio signals (4G/5G), satellites and it can always be upgraded to use any new technology.

2.3.1 The Core Network

The core network provided by BT is comprised of three major components; the exchange, metro nodes and the ducts connecting them all together. The exchange is where the majority of the work is done routing packets between locations. The metro nodes work as repeaters and junctions, amplifying signals that could degrade over time or joining two pathways together. A duct is a pathway containing cables of varying types, be it copper or fibre optic.

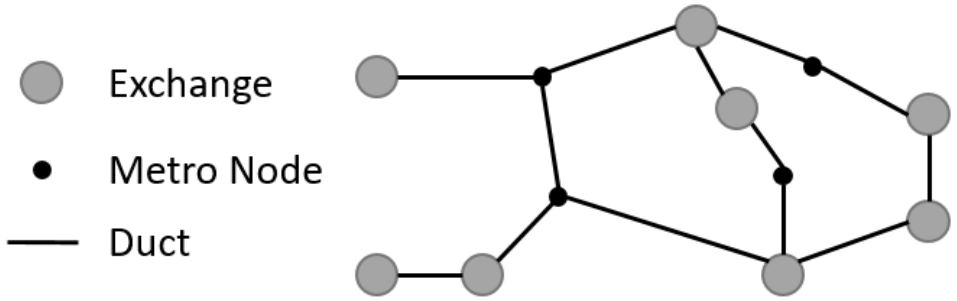


Fig 2.4 Example representation of the core network

Fig 2.4 shows an example representation of the core network in a graph with the exchanges and metro nodes represented as vertices and the ducts as edges. As shown in Fig 2.4 metro nodes are not a requirement to connect two exchanges together, they can be connected to another metro node or an exchange. They can also be used as junctions to connect to more than two ducts at one time.

2.3.2 The Exchange

The exchange is the most complex and important part of the core network and are situated across the country with every major town in the UK having at least one that services the town and surrounding area. The exchange comprises of three types of equipment; cooling, power and bandwidth. Cooling equipment is used to ensure optimal environmental conditions by managing the heat produced by the power and bandwidth equipment and consists of heatsinks, fans and thermal shielding. The power equipment provides energy for both the cooling and

bandwidth equipment and consists of transformers and cables. The bandwidth equipment is used to route data packets to the correct locations. Bandwidth equipment consists of; racks, sub-racks, cards, sub-cards and ports.

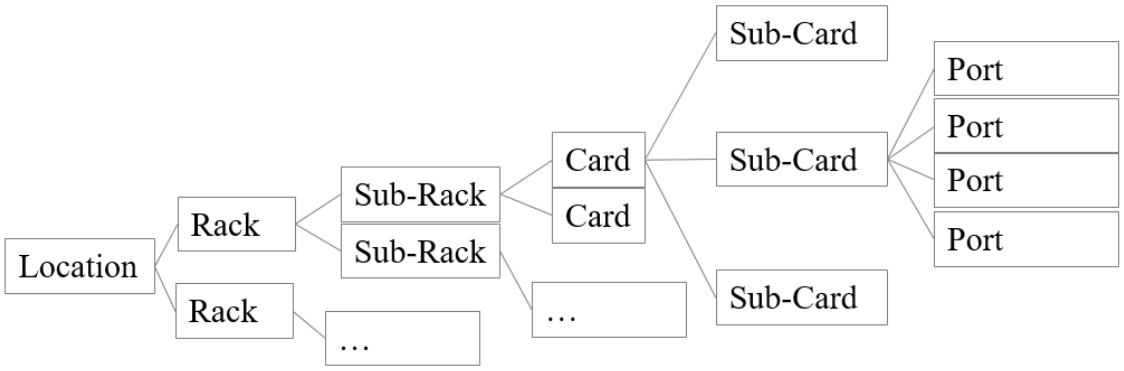


Fig 2.5 Bandwidth equipment layout within the exchange.

Fig 2.5 shows the layout of the bandwidth equipment within the exchange. Traversing down the tree, the equipment has a one-to-many or a one-to-one relationship and sub equipment is optional with some configurations requiring it and some not. The location is the physical geographical location of the exchange. The rack provides housing, power and cooling for sub-racks and cards. A card provides a position for cables to be interfaced into the ports. Each port in use has a cable in it, which in turn houses some digital infrastructure. This digital infrastructure could be a service such as voice or dedicated government or financial connects, or the infrastructure could be a Virtual Local area network (VLan).

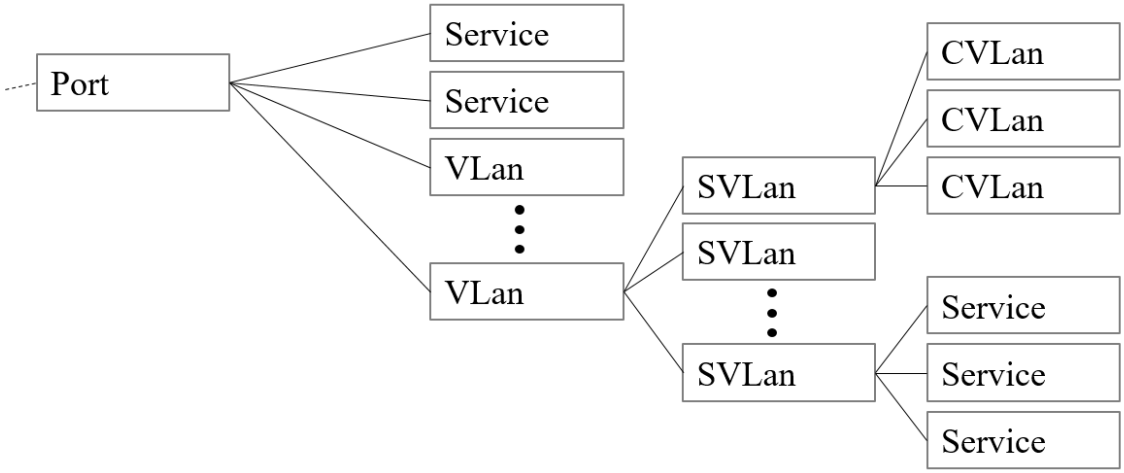


Fig 2.6 Example digital infrastructure associated with a single port

Fig 2.6 is an extension of the tree structure shown in Fig 2.5 and shows an example of the digital infrastructure that could be associated with a single port. A VLan is a management technique for groups of service and customers and one port can have many VLans but one VLan can only be associated with one port. Each VLan can have one or more Service Virtual Local Area Network (SVLan) as its child in the tree structure. Each SVLan is used to house either a set of Services or a set Customer Virtual Local Area Network (CVLan). A CVLan represents a group of access customers, such as households, business or organisations.

2.4 Routing and Resilient Routing

Routing problems are something that people face every day of their lives and is an important skill that most people have and don't even realise it. Routing can be used for a multitude of different problems; the Traveling Salesman Problem (TSP) [24] is one such example of a routing problem that has a wide variety of uses, including: sales, logistics and delivery. The goal of the TSP is to find the shortest path that visits all vertices and returns to the starting location. The most well-known routing problem is point to point routing, where the goal is to get from a start location to an end location in the shortest distance. Point to point routing is

used in a wide variety of domains including navigation, digital packet transfers and system design.

2.4.1 Dijkstra's Algorithm

Dijkstra's Algorithm [25] created by Edsger W. Dijkstra in 1965 is a greedy search algorithm used in graph theory. The algorithm was originally created for routing between computer systems within a network and is seen as one of the simplest and most effective routing algorithms. It has been used in a wide variety of domains from: robotic controllers [26] to parking lot planning and operations [27]. The algorithm is not without pitfalls, it cannot deal with negative numbers and it is slow at routing in networks with a high number of vertices and edges. Fig 2.7 shows pseudo code for Dijkstra's algorithm.

```
def Dijkstra(Graph, source, destination):
    for (vertex v in Graph.Vertices):
        distance[v] = INFINITY
        previous[v] = null
        List.append(v)

    dist[source] = 0
    while(List is not empty):
        currentVertex = vertex in List with min(dist[currentVertex])
        if (currentVertex == destination):
            return distance, previous
        List.remove(currentVertex)
        for (vertex v: List):
            currentDistance = dist[currentVertex] + Graph.Edges(currentVertex, v)
            if (currentDistance < distance[v]):
                distance[v] = currentDistance
                previous[v] = currentVertex


    return distance, previous
```

Fig 2.7 Pseudo code for Dijkstra's Algorithm

The algorithm goes through the graph one vertex at a time, exploring and expanding as it goes. From the source location it calculates the distance to all neighbouring vertices. It then picks the lowest distance from the unvisited and accessible vertices from across the graph and calculates

the distance from it to each unvisited neighbour. It does this whilst still taking into account the distance from the path already taken and updates the neighbours' distance if it is lower. It repeats this process until it reaches the destination vertex at which point it has a list of vertices that represent the route from the source to the destination.

2.4.2 A-Star Algorithm

The A-Star Algorithm [28] was first published in 1968 by Peter Hart, Nils Nilsson and Bertram Raphael and is an extension to Dijkstra's Algorithm. It achieves the same optimal results as Dijkstra's Algorithm but has a reduced time complexity at the cost of an increased space complexity [29]. A-Star Algorithm has seen widespread success in multiple domains but appears to have had the most success in the video games industry [30], [4]. It uses a modified version of the search procedure in Dijkstra's Algorithm by introducing a heuristic that indicates an estimated distance between the current vertex and the destination vertex. Just like Dijkstra's Algorithm it cannot deal with negative edge costs, but unlike Dijkstra it performs much better in larger networks, given that the heuristic used is accurate. An inaccurate heurist can ruin the graph traversal, and make it slower than Dijkstra's Algorithm in any sized network.

```
def A_Star(Graph, source, destination, heuristic):
    for (vertex v in Graph.Vertices):
        distance[v] = INFINITY
        previous[v] = null
        List.append(v)

    dist[source] = heuristic[start]
    while(List is not empty):
        currentVertex = vertex in List with min(dist[currentVertex])
        if (currentVertex == destination):
            return distance, previous
        List.remove(currentVertex)
        for (vertex v: List):
            currentDistance = dist[currentVertex] + Graph.Edges(currentVertex, v)
            if (currentDistance < distance[v]):
                distance[v] = currentDistance + heuristic[]
                previous[v] = currentVertex


    return distance, previous
```

Fig 2.8 Pseudo code for A-Star Algorithm

Fig 2.8 shows pseudo code for the A-star Algorithm. The pseudo code for the A-Star Algorithm is almost the same as that of Dijkstra in Fig 2.7 but there are three differences. Firstly the heuristic data needs to be pre-calculated. In a distance graph, the Euclidian distance between each vertex and the destination vertex could be used as the heuristic values. Secondly the distance of the source is now set to its heuristic distance not zero. Finally, anytime distance is updated, the heuristic distance is incorporated. With these changes the distance comparison will now use the distance between two locations and the heuristic distance.

2.4.3 Resilient Routing

Data routing is a complicated procedure and there are many requirements placed upon it, one of these such requirements is a reliable connection. Different services offered by BT require different levels of assurances when it comes to connection reliability. For example if the connection to your typical household consumer was interrupted for some reason it would be an inconvenience for them but not vital that it remained connected. Whereas some consumers require a connection at all times and will pay to get assurances that their connection has a

backup in place. These customers include: television entertainment companies such as the British Broadcasting Corporation (BBC) and Independent Television (ITV), government institutions such as the Ministry of Defence (MOD) and the National Health Service (NHS), or finally big financial institutions trading on the stock markets such as NatWest or Barclays Banks. This back up connection is known as a resilient route or resilient line. Resilient Routing is the act of having two or more routes between the start and finish locations of the proposed route. These resilient routes must not intersect to create a single point of failure (SPoF)[31]. This allows for system to effortless and seamlessly switch to the backup or resilient route if something happens to the primary route without a loss of service to the consumer.

There are some problems that are important not to confuse with the Resilient Routing problem. TSP is one such problem that has many wides spread practical applications but is not the same as Resilient Routing. The basic idea of TSP [24] is that an agent must travel through a graph visiting each vertex in the cheapest way possible and then returning to the starting position. Another problem that is very similar but should not be confused with Resilient Routing is the idea of Quality of Service (QoS) routing [32]. Albeit that Resilient Routing is used for QoS, QoS routing doesn't have the same set of constraints set upon it, and thus can use different techniques.

2.4.3.1 Resilience Optimality

Fig 2.9 shows an example of Resilient Routing when solved by Dijkstra's Algorithm. The route depicted by the vertices *[A, B, C, I]* is the shortest path and the primary route at a cost of *29*. The resilient route as depicted by the vertices *[A, F, G, H, I]* has a cost of *40* does not intersect with the primary route, apart from at the start and finish locations, therefore there are no SPoF across the network and it classifies as the resilient route.

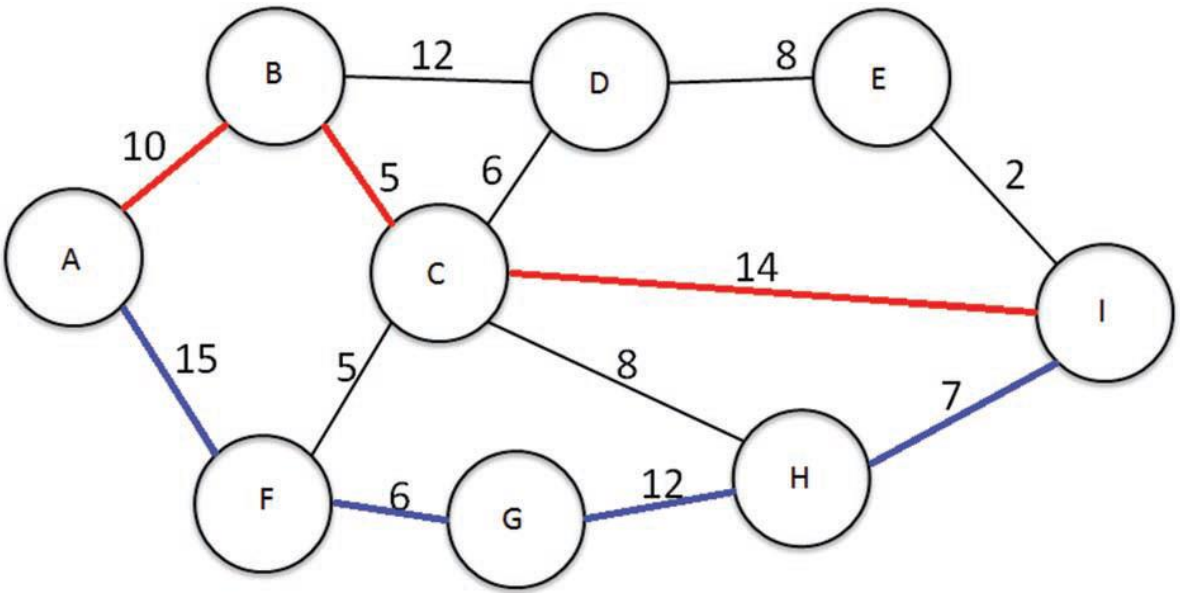


Fig 2.9 Resilient Routing example using Dijkstra's algorithm to solve the route [33]

As this is a service that BT offers the customer would be expected to pay for both lines thus more than doubling the cost to the customer. The primary route depicted by Dijkstra in Fig 2.9 is the shortest path, and the resilient route is the next shortest path. Individually Dijkstra provides the cheapest routes, but once combined into one cost they are no longer the optimal costing routes, the cost of the primary and the resilient routes combined into one solution is 69.

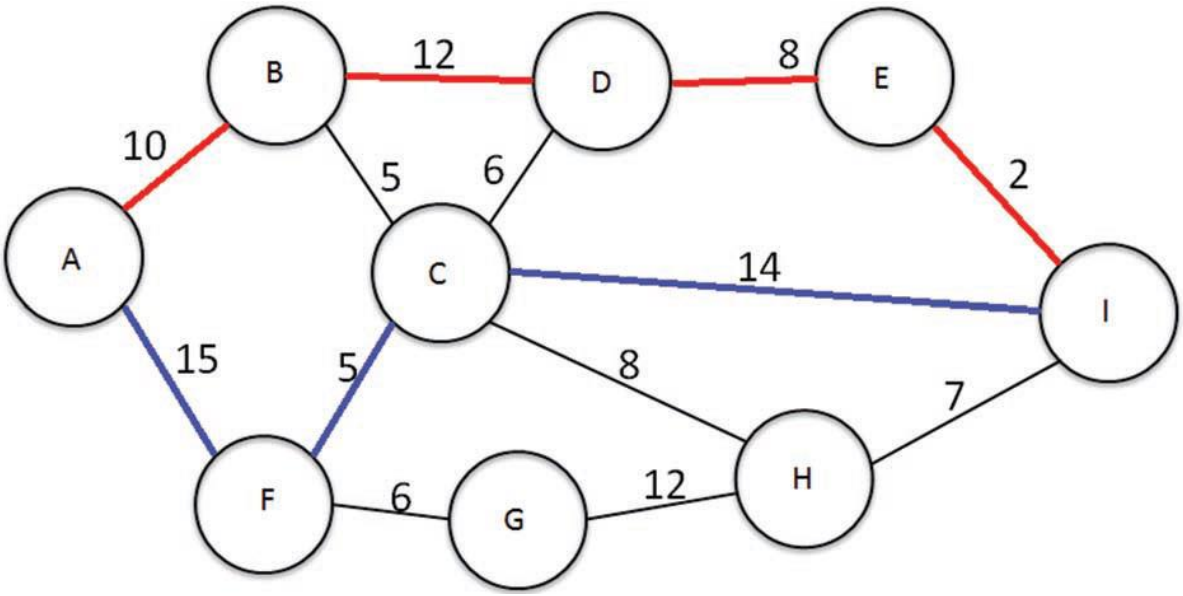


Fig 2.10 Resilient Routing example that has the optimal solution. [33]

Fig 2.10 shows an optimal solution for the same graph as shown in Fig 2.9. In Fig 2.10 the primary route is depicted by the vertices *[A, B, D, E, I]* at a cost of *32* and the resilient route is depicted by the vertices *[A, F, C, I]* at a cost of *34*. By comparing the graphs and results from the Fig 2.9 and Fig 2.10, the total cost of entire solution is reduced from *69* to *66*. This reduction is possible by forcing the primary route to take a less optimal path allowing for the resilient route to take a more optimal one. The cost of the primary route is increased by *5* but the resilient route's cost is decreased by *8*, giving us our total cost of *66*.

2.4.3.2 Resilience Blocking

Fig 2.11 shows an example of a graph where resilient route is possible but not achieved through the use of Dijkstra's Algorithm. In this example the primary route is depicted by the vertices *[A, B, D, F, G]* at a cost of *27*. There is no resilient route between the start and finish locations in Fig 2.11. The route presented between the start and finish locations is optimal for one route, but a resilience service could not be offered as the primary route blocks it.

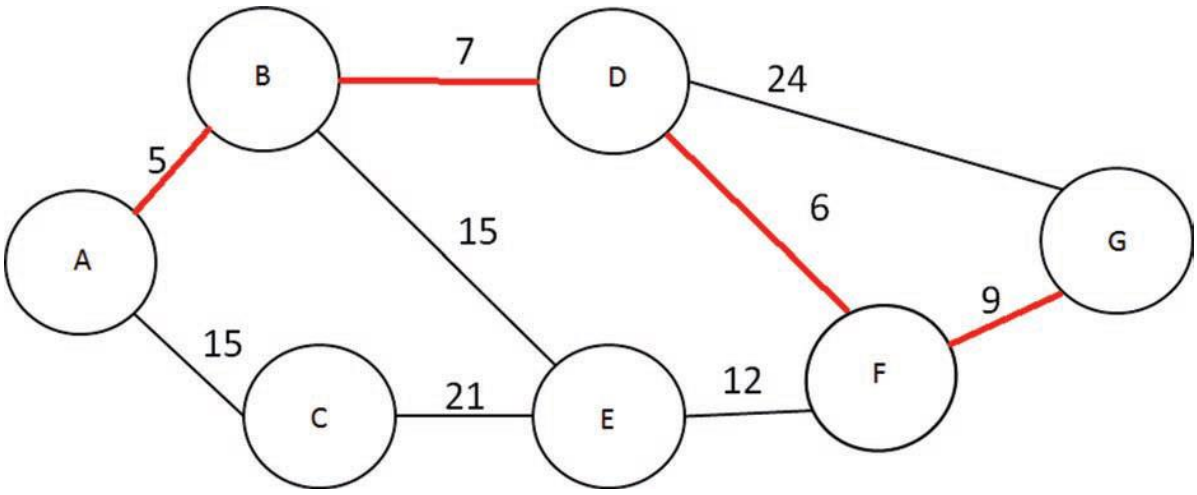


Fig 2.11 Example Graph where resilient routing is blocked with Dijkstra's Algorithm

Fig 2.12 Shows an optimal example of Fig 2.11 where a resilient route has been found. In Fig 2.12 the primary route is depicted by the vertices *[A, B, D, G]* with a cost of *35*. The resilient route in Fig 2.12 is depicted by the vertices [A, C, E, F, G] with a cost of 57. So now a resilient

solution to the graph exists with a total cost of 92. This is possible by forcing the primary route to take a sub-optimal route opening up the possibility of a resilient route.

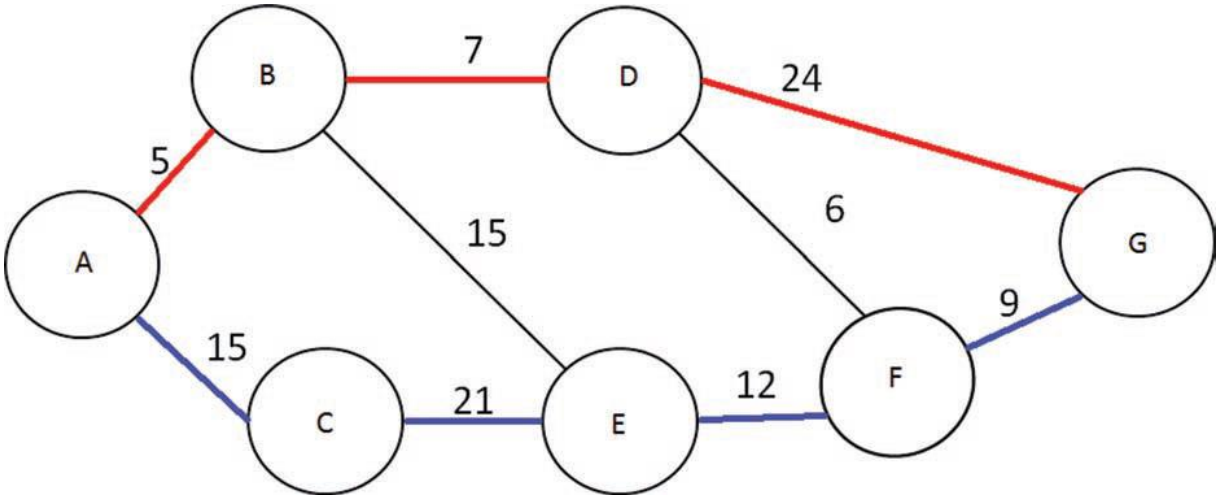


Fig 2.12 Example Graph where both the primary and resilient routes have been found

2.5 Uncertain environment

The Collins English dictionary defines uncertain as “not able to be accurately known or predicated” or as “not precisely determined, established, or decided” [34]. The Collins English dictionary defines environment as “external conditions or surroundings” [34]. If we combined these two definitions together an uncertain environment would mean an external surrounding that could not be precisely determined. Applying this to the field for Artificial Intelligence (AI) it can be seen as an unknown or imprecise data set.

AI requires data, be this pre-collected data stored into a file such as a database or a comma separated value file (CSV). Alternatively the information could be collected in real-time by a sensor such as a thermostat or a radar. But in all scenarios data is required and at a high quality, but in some cases this is not possible. This can be for several different reasons. In the case of pre-collected data it could be that there are several conflicting data sources. These data sources are usually very similar but there is no way to determine that one of them is correct, so all data sources must be used. In the case of sensors, it is generally good practice to ensure that multiple

sources are taken to ensure a more accurate result, these data sources will not always match up and give one clear result.

Evaluation is a core part of the AI process and requires accurate metrics in order to function properly. In this context evaluation means a way to score or determine the quality of something by using a mathematical function. With an unknown or unreliable set of numbers a mathematical function cannot give a reliable result. Therefore as routing and optimisation need to evaluate potential solutions they require good data from a certain environment to be able to give a high quality result.

2.5 Overview of Capacity Planning

Capacity planning is the idea of planning for the usage or storage of some product or service over a period of time. Capacity planning can be applied to many different fields. Multi agent systems have been used in manufacturing [35], specifically in production capacities and scheduling of a machine-building enterprise. Statistical modelling has been used in virtual machine deployment [36] where they are modelling the amount of physical hardware required to contain virtualised operating servers. System dynamics have been used within closed loop supply chains [37] for understanding and adjustments to a closed loop system, to allow prices to change as elements of the loop change. Multi-objective optimisation have been shown to be effective in supply chain management [7] where they try to maximise profits whilst minimising risks.

There are many different types of capacity planning but they all have a somewhat similar set of goals: to increase the capacity of a service (supply chain, virtual machine or production capabilities) whilst keeping the costs as low as possible. Capacity planning in telecommunications is no different with some overarching goals: to reduce costs whilst

increasing the available capacity or utilisation. Within BT there are two types of bandwidth capacity planning within an exchange; Digital and Physical.

2.5.1 Digital Capacity Planning in Telecommunications

In the telecoms industry, products and services are assigned to physical ports on cards within data exchanges. These products and services arranged across multiple digital layers as shown in Section 2.3.2 The Exchange and more specifically Fig 2.6 Example Digital Infrastructure associated with a single port. Each of these services have their own set of requirements and constraints, which are broken down into technical and business constraints. Technical requirements and constraints are hard and must be met and cannot be broken, otherwise the software would not function. Business requirements and constraints are soft and therefore can be broken, but it is extremely preferable to meet them as they line up with overarching goals of BT.

The Knap Sack problem and the Bin Packing problems are classic computer science problems covered in Sections 2.2.1 and 2.2.2 respectively. Digital capacity planning closely resembles something in between these two problems but there are some differences. In Bin Packing there are an unlimited number of bins that equally sized objects must all fit within. In the Knap Sack problem there is one container that the maximum number of the maximum valued objects must fit within. Whereas in the digital capacity planning problem there are a fixed number of bins that all the objects must fit within and those objects are themselves containers that must fit another subset of objects within. In Section 2.3.2 VLans, SVLans and CVLans are introduced, these three elements together create three layers of capacity bins. CVLans have a utilisation value that represents how much bandwidth they consume. SVLans contain a number of CVLans and have a limit to their capacity, and VLans contain a number of SVLans and also have a limit to their capacity.

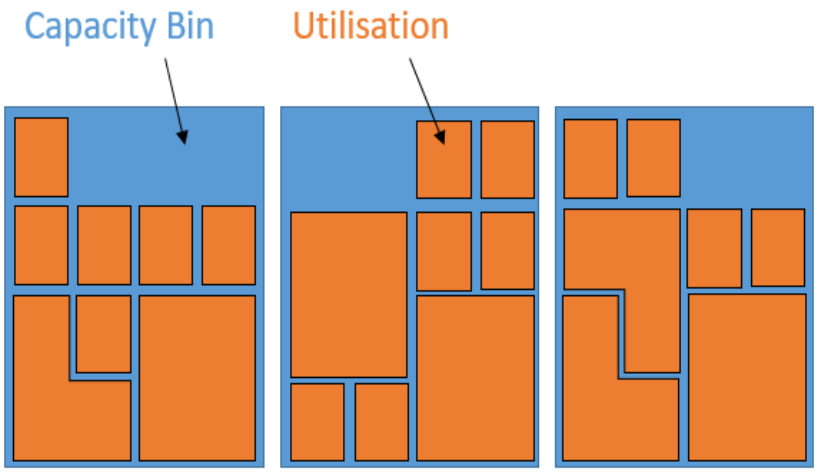


Fig 2.13 Demonstration of Utilisation in capacity bins [38]

Fig 2.13 shows an example of how utilisation of varying sizes can be placed within the allotted capacity without overflowing the available space within a capacity bin. In the example the capacity bin is the SVLan and the utilisation in the CVLan. Now this is a simple example with three bins and 23 objects, but in actuality the number of SVLans (bins) and CVLans (utilisation) is much greater. One physical port can contain multiple VLans, one VLan can contain anywhere up to 100 SVLans and each SVLan can contain up to 2500 CVLans. Very quickly a concern becomes the problem of scale, to add to this utilisation is based upon customer bandwidth usage, which changes by the minute. The objectives of digital capacity planning are to ensure that the VLans and by extension the SVLans do not breach the limit of a physical port's bandwidth capability.

2.5.2 Physical Capacity Planning in Telecommunications

A data exchange is a vital part of the telecoms industry and it serves as core part of a nations networking infrastructure. In Section 2.3.2 it was established that an exchange is made up of three types of equipment; power, cooling and bandwidth. Physical capacity planning in telecommunication is regarding the physical equipment within the exchange. The capacity of the cooling system is the simplest and cheapest to upgrade. The power system is by far the

most expensive system to upgrade and requires the complete shutdown of a data exchange for an extended period of time to upgrade which is unacceptable. The bandwidth systems consume the most power and generate most of the heat, but newer bandwidth equipment has higher capacity whilst being more power and thermally efficient. The bandwidth infrastructure is therefore the priority as it reduces the requirements on the other two types of equipment whilst also increasing the bandwidth available.

The bandwidth system is by far the most complicated to upgrade for a multitude of reasons: Firstly, the system is highly interconnected and any disruption to one set networking card can have effects to other cards either at the same physical location or elsewhere in the network. Secondly, different services on each card have their own set of requirements that must be met, some of these include: digital capacity planning, resilient routeing, card and service incompatibilities and bandwidth restrictions. Thirdly most exchanges have been incrementally improved overtime, so they are at their power and cooling capacities so new equipment can't be added and then service migrated. Finally it's a live system so any work undertaken to upgrade it must be done with minimal disruption to the network, as consumers must not be left without service for any extend period of time. In order to achieve this minimal disruption to consumer connectivity upgrades must be completed in a two hour time window.

To solve these problems the algorithmic concept of divide and conquer [39] is employed to split the process down into smaller processes and then solve them individually. In an ideal world that process would go something like this; Identify the parts of the system that need to upgraded first, then remove all the services from it and unplug all the cables, and replace it with the new piece of equipment so it can be used by those same services and more. Then move onto the next piece of bandwidth equipment and repeat the process, but this time there is more bandwidth capacity as the new piece of equipment is in place and should have a greater ability to satisfy the bandwidth demand.

Fig 2.14, Fig 2.15 & Fig 2.16 show an example of how ports can be moved in order to remove a card from a networking device such as a rack. In the example shown in Fig 2.14 Card F is selected to be removed and thus all the filled ports on that card must be moved. Fig 2.15 shows the location that the ports have been moved too, with two moved to *Card A*, three to *Card B*, one moved to *Card D*, one moved to *Card E*, one moved to *Card G*, three moved to *Card H* and one moved to *Card J*. Fig 2.16 shows that all ports on *Card F* have been emptied and that card can now be removed without causing any issues. Looking at Fig 2.16 you can see that cables within the ports have been moved to different ports across the device and don't need to all be on the same card after the move.

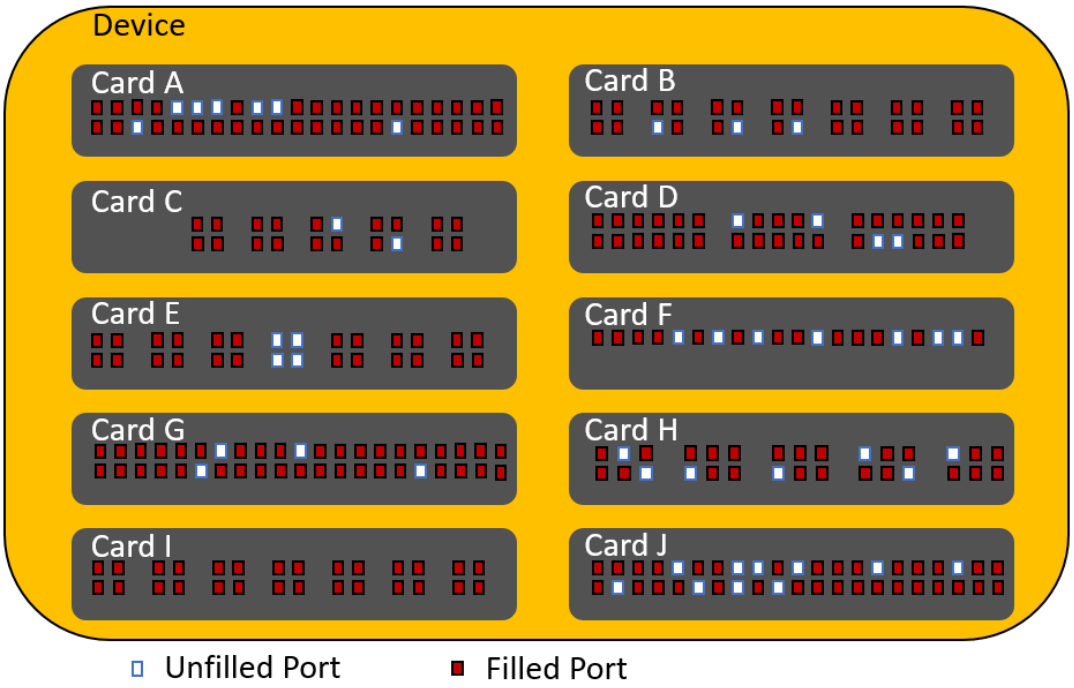


Fig 2.14 Physical Capacity Planning Example before port move

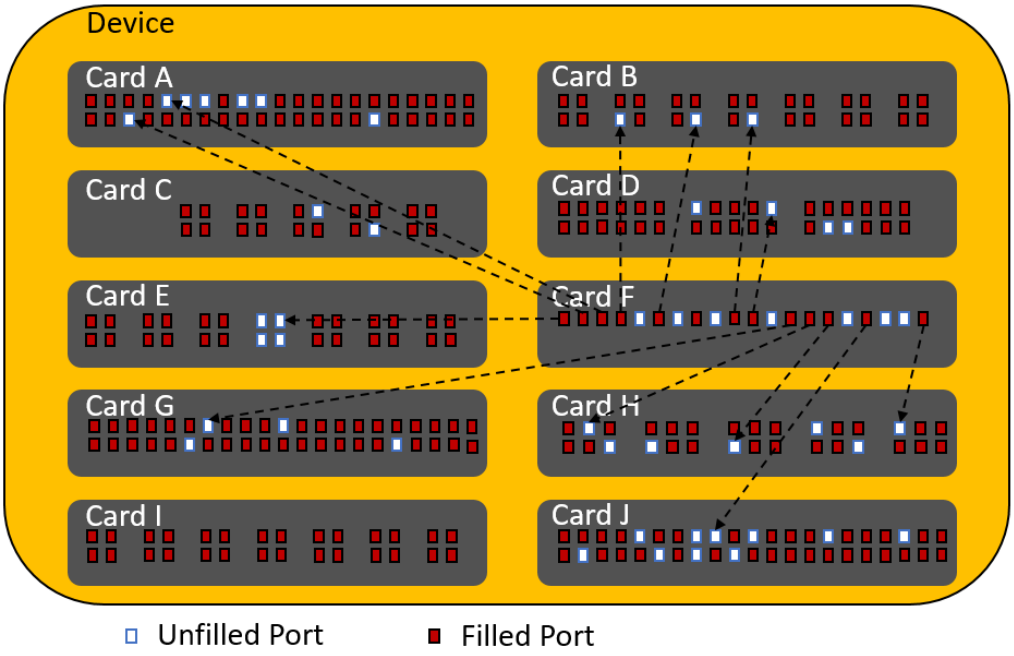


Fig 2.15 Physical Capacity Planning Example during move

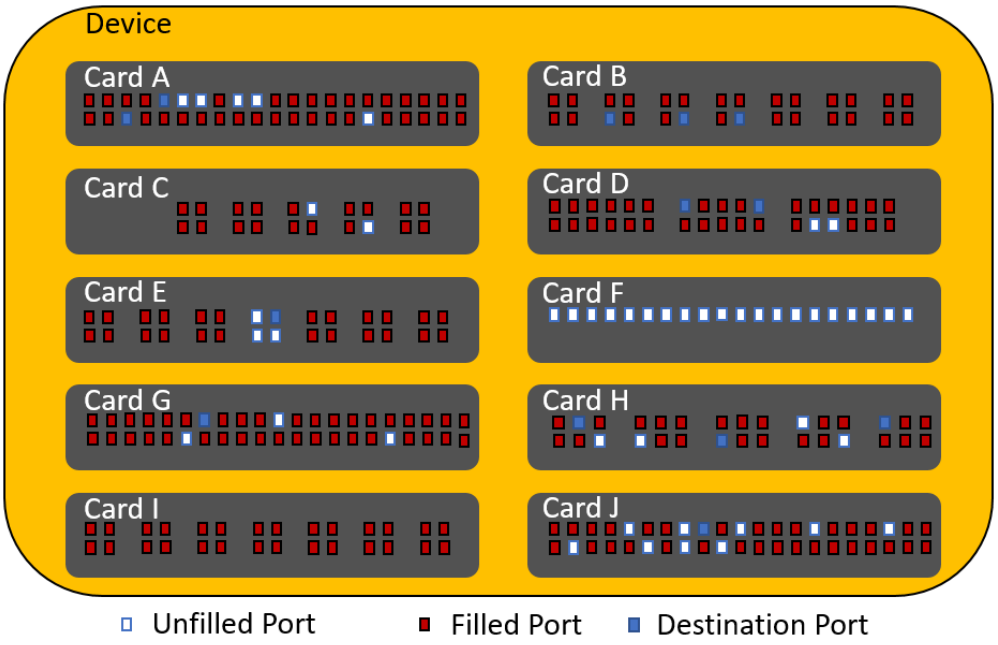


Fig 2.16 Physical Capacity Planning example after move

Unlike the example, the real-world problem is much more complex with a multitude of constants set upon each port and there is not always enough space to move the ports around like this, but the example in as an ideal scenario and works to demonstrate the problem.

Chapter 3. An Overview of Fuzzy Logic Systems

Fuzzy Logic was presented by Lotfi Zadeh who first published in his paper “Fuzzy Sets” [40] where he introduced the core ideas of Fuzzy Logic which did set the stage for a myriad of extensions to the theory and applications of that theory. In traditional logic something can exist as one of two states “True and False” but that is not how humans understand the real world, we don’t understand everything in a binary state, it’s understood as a continuous set of states. Additionally when people communicate they don’t convey information in a precise manor they tend to use a word or label that can convey that information in a high enough level of detail. A prime example of this is how people discuss temperature. If you asked someone if it was warm outside, it is unlikely that they would respond with the answer “its 22.6° C outside” it is much more likely they would say “it’s nice out” or “it’s ok outside”. Fuzzy Logic takes this idea of the imprecisions of human linguistic language and converts it into a mathematical function, allowing information to exist in an imprecise non binary fashion, known as a linguistic variable [41]. Most commonly the Mamdani methods of Fuzzy Logic implications are used within software and they used for the Fuzzy Logic process in this thesis [42].

Fuzzy Logic excels at modelling the uncertainty of inputs, with Type-1 and Type-2 modelling different levels of uncertainty. Type-1 and Type-2 Fuzzy Logic has been hugely successful across a multitude of domains, showing that it can perform well on its own as a control system [43] [44] or classifier [45][46]. It can work well in conjunction with other techniques such as evolutionary algorithms [47], image processing [48] and neural networks [49].

3.1 Type-1 Fuzzy Logic System

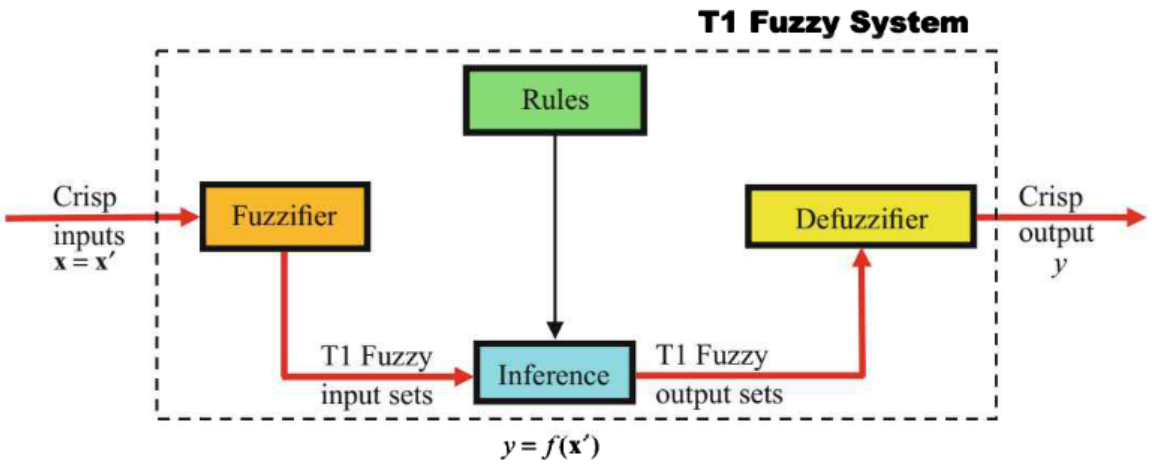


Fig 3.1 Type-1 Fuzzy Logic System [50]

Fig 3.1 shows the outline of a Type-1 Fuzzy Logic Controller (FLC), consisting of four parts; fuzzifier, rules, inference and the output processor also commonly known as the defuzzifier [50]. The system takes crisp real numbers and passes them to the fuzzifier which uses Fuzzy Sets to transform those values into fuzzy values. These values are passed into the inference which uses the rules to transform the values into output fuzzy values using output Fuzzy Sets. Finally, the output processor transforms them into back into crisp numbers that can be used outside of the Fuzzy Logic system.

3.1.1 Type-1 Membership Functions

In Fuzzy Logic membership functions are represented as $\mu_F(x)$ and have a linguistic variable associated with them which is used to bind the firing strength of a particular membership function to its rule, allowing the sets to be interpreted by the rules in the inference stage. Membership functions can take any number of geometric shapes the most common being; Triangles, Trapezoids, Gaussians, and Singletons. Fig 3.2 shows these shapes.

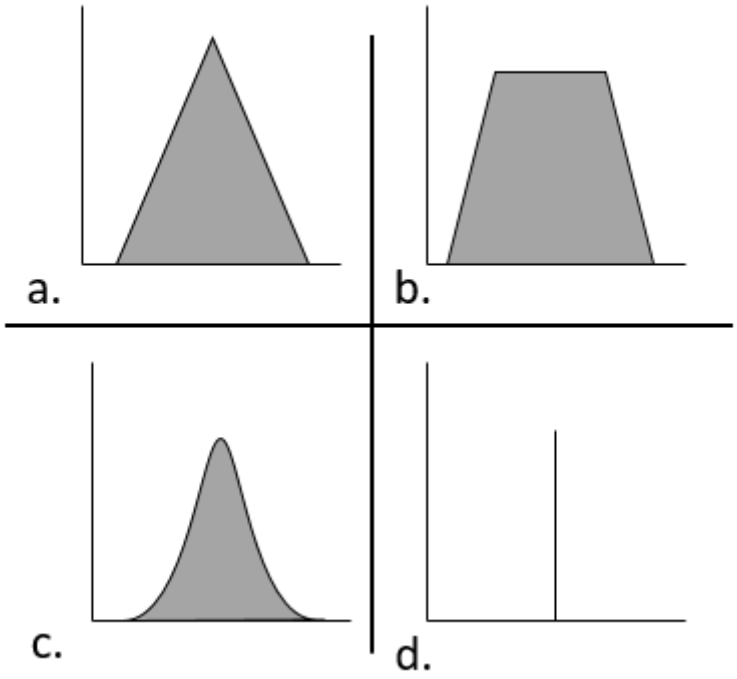


Fig 3.2 (a) Triangle. (b) Trapezoid. (c) Gaussian. (d) Singleton.

The *y* axis of a membership function indicates the membership degree and the *x* axis indicates the crisp inputs of a given problem domain. The membership degree is calculated differently for each shape. With the singleton being either 0 or 1 as it can only be on the singleton point or not on the point as shown in equation 3.1.

$$\mu_F(x) = m \tag{3.1}$$

The firing strength of the other three shapes are determined by the intersection of the *y* axis given an *x* axis point. Equation 3.2 shows the calculation for a triangle given that *a*, *b*, and *c* are the points of the triangle in order [50].

$$\begin{aligned} \mu_F(X) &= \frac{x-a}{b-x} \quad a \leq x \leq b \\ \mu_F(X) &= \frac{c-x}{c-b} \quad b < x \leq c \\ \mu_F(X) &= 0 \quad Otherwise \end{aligned} \tag{3.2}$$

The membership value of a trapezoid is found using equation 3.3 given than a, b, c and d are the points of the trapezoid in order [50].

$$\begin{aligned} \mu_F(X) &= \frac{x-a}{b-x} \quad a \leq x \leq b \\ \mu_F(X) &= 1 \quad b < x \leq c \\ \mu_F(X) &= \frac{d-x}{d-c} \quad c < x \leq d \\ \mu_F(X) &= 0 \quad Otherwise \end{aligned} \tag{3.3}$$

The membership value of a Gaussian membership function is calculated using equation 3.4.

$$\mu_F(X) = e^{(-0.5\left(\frac{x-a}{\theta}\right)^2)} \tag{3.4}$$

Input membership functions perform at their best when they overlap, allowing a crisp input to exist as part of multiple input classes. The crisp value *x* is input into a fuzzy system allowing the input to be in both the Low and Medium membership functions at the same time to varying degrees.

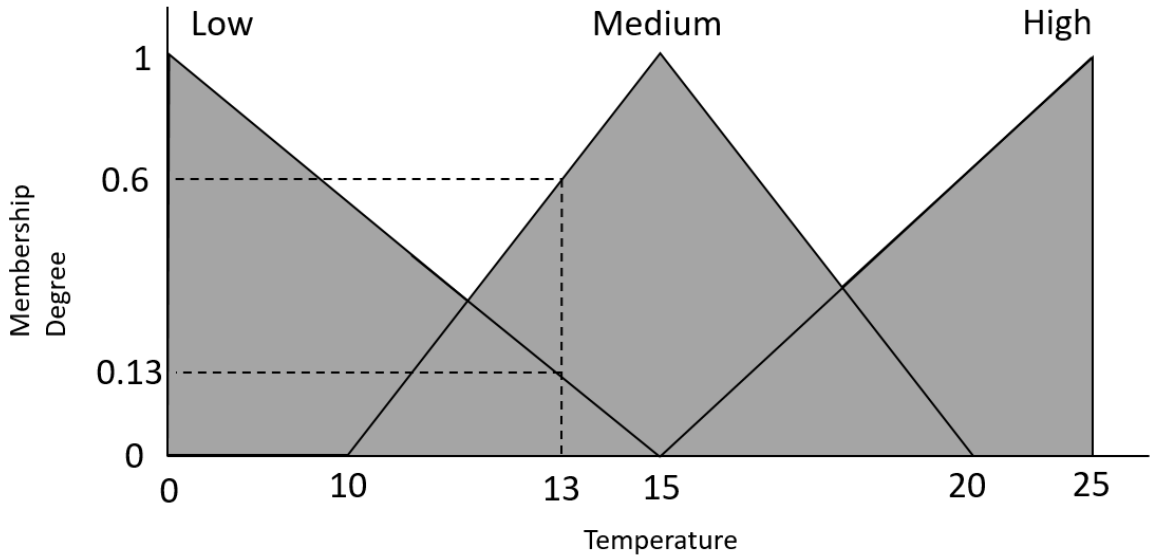


Fig 3.3 Example of Type-1 Fuzzy Set for a heating problem.

Fig 3.3 shows an example input membership function with three input sets Low, Medium and High. In this example all three input sets are triangles and the crisp input set of 13 the membership values are 0.6 Low and 0.13 Medium.

3.1.2 Rules and Inference

Once the input membership functions have determined the membership degree of each of the inputs, the fuzzified inputs are passed to inference engine which utilises the rules. The rules are used to map inputs to desired outputs working as a mechanism to control the systems behaviour in linguistic terminology. Rules are logic IF-THEN statements, with the IF being the antecedent and the THEN being the consequent [50]. There are two major types of rule in a FLC with AND rules shown in equation 3.5 and OR rules shown in equation 3.6.

$$R: IF\ x_1\ is\ F_1^l\ and\ \dots\ and\ x_p\ is\ F_p^l\ THEN\ G^l \quad l = 1, \dots, M \tag{3.5}$$

$$R: IF\ x_1\ is\ F_1^l\ or\ \dots\ or\ x_p\ is\ F_p^l\ THEN\ G^l \quad l = 1, \dots, M \tag{3.6}$$

Where *x* denotes the input and F denotes the membership function, these two together make the antecedent and G is the consequent denoting output membership function.

TABLE 3.1 Example Rules for a Heating Problem

Rule Number	Thermometer One	Thermometer Two	Heating Level
R1	Low	Low	Low
R2	Low	Medium	Low
R3	Low	High	Medium
R4	Medium	Low	Low
R5	Medium	Medium	Medium
R6	Medium	High	High
R7	High	Low	Medium
R8	High	Medium	High
R9	High	High	High

Table 3.1 shows the rules base for the Heating Problem example. In this example imagine that there are two thermostats in two rooms and the system is designed to control the temperature in these two rooms. Using the membership values from Fig 3.3 of 0.6 Low and 0.13 Medium for thermometer one, and introducing some new values of 0.8 Medium and 0.1 High given an input value of 16, which rules are fired can be determined. In this example the rules are AND so the rules R1, R2, R5 and R6 fire, giving the consequence for the fuzzy system for defuzzification.

Inference is the process of determining which rules fire and by how much. The amount a rule fires is known as fuzzy implication based upon two parts: the membership values of a rules antecedent and its t-norm operator [50]. There are lots of t-norm operators and selecting the correct one is based upon what the conditional of the IF-THEN rule is. The two most common

t-norms for AND logical rules are minimum and product shown in equation 3.7 and 3.8 respectively.

$$\mu_{A\to B}(x,y) \equiv \min[\mu_A(x), \mu_B(y)] \tag{3.7}$$

$$\mu_{A\to B}(x,y) \equiv \mu_A(x)\,\mu_B(y) \tag{3.8}$$

Continuing the heating example from Fig 3.3 and Table 3.1 using minimum as the t-norm the inference value for the rules are shown in Table 3.2. The four rules have their values ready for defuzzification and the consequent of each rule is known.

TABLE 3.2 Inferred rules with their minimum values for the heating example

Rule	Thermometer One	Thermometer Two	Heating Level	Minimum Value
R1	Low (0.6)	Low (0.8)	Low	0.6
R2	Low (0.6)	Medium (0.1)	Low	0.1
R5	Medium (0.13)	Medium (0.8)	Medium	0.13
R6	Medium (0.13)	High (0.1)	High	0.1

3.1.3 Defuzzification

There are several different methods of defuzzification some of which include: centre of sets, centroid, height and modified height [50]. For defuzzification to take place the output sets must have a set of membership functions much like our input. In the case of our heating problem (Fig 3.3, Table 3.1, Table 3.2) there would be three output sets: Low, Medium and High as seen in Fig 3.4. Each membership function indicates a temperature that would be preferable for the heating system to reach.

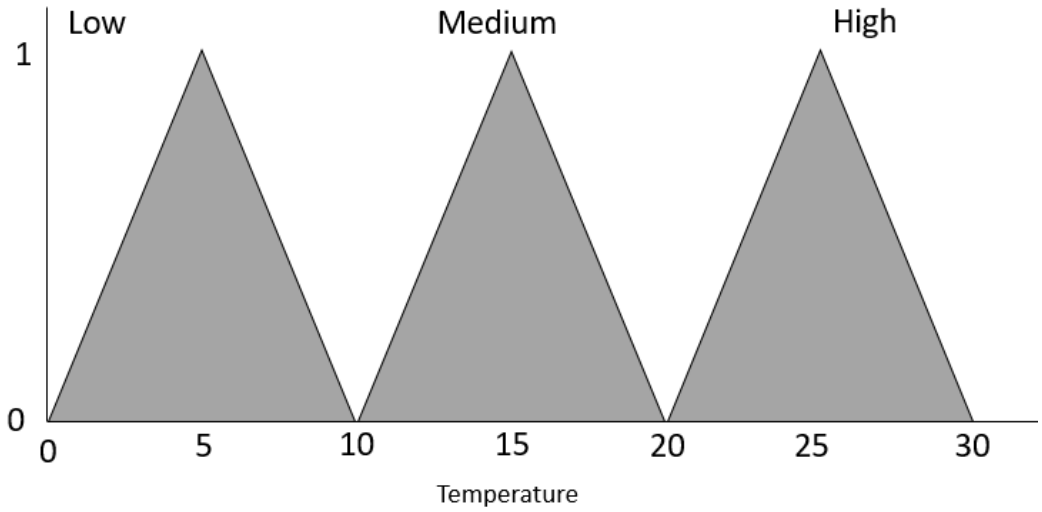


Fig 3.4 Output Membership Functions for the Heating Example

Centre of Sets (CoS) defuzzification is the simplest method, by taking the centre sum of all the output set *c* multiplied by their firing strengths *f* taken from inference and divided by the sum of the firing strengths as shown in equation 3.9 [50].

$$Y_{CoS}(X) = \frac{\Sigma\, c * f}{\Sigma\, f} \tag{3.9}$$

Centroid defuzzification is very similar to CoS but instead of using the centre of the set, the centre of mass or centroid of the shape is used. This adds the computational complexity of calculating the centroid of each output membership function, but if these functions never change this can be pre-calculated before runtime execution. Equation 3.10 [50] shows the centroid defuzzification method, with *Com* denoting the centre of mass of a given output shape and *f* denoting the firing strength taken form inference.

$$Y_{Centroid(X)} = \frac{\Sigma\, Com * f}{\Sigma\, f} \tag{3.10}$$

3.2 Interval Type-2 Fuzzy Logic System

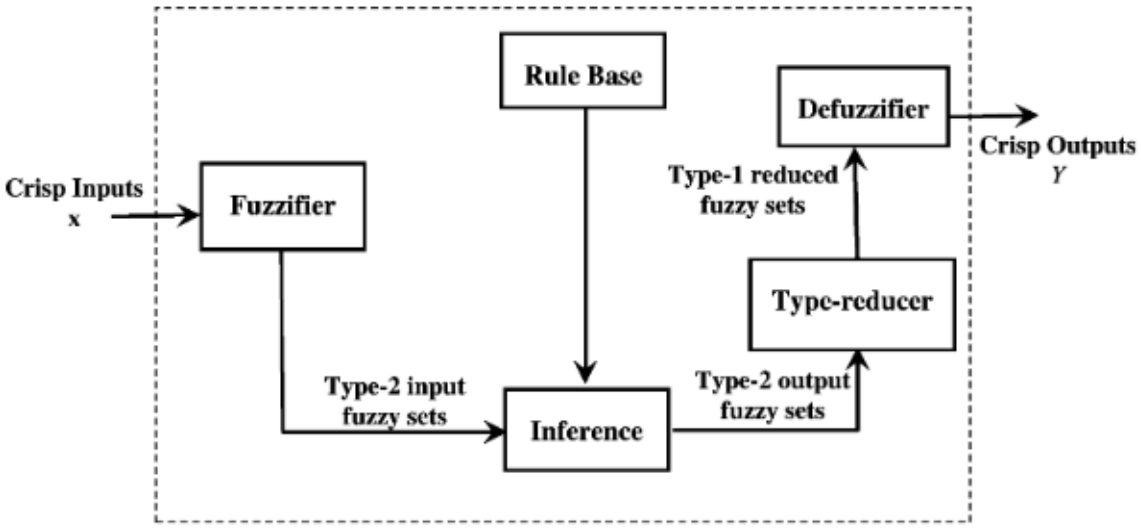


Fig 3.5 Type-2 Fuzzy Logic System.[43]

An Interval Type-2 Fuzzy Logic Controller (IT2FLC) is an extension of a Type-1 FLC. Fig 3.5 shows the IT2FLC. Notice the new step, type-reducer. This is used to convert a Type-2 fuzzy output sets into a Type-1 fuzzy sets in order to defuzzify them into a crisp number. It uses the same ideas of fuzzification of uncertain information, rules inference and defuzzification, but some of these steps are extended. It has advantages over a Type-1 FLC with its ability to represent a larger array of information in an uncertain environment [51].

3.2.1 Interval Type-2 Membership Functions

Whilst the IT2FLC is more computationally expensive it has a better capability to model uncertainty than its Type-1 counterpart. This increased capability comes from how it models fuzzy sets, with Type-2 membership functions using a footprint of uncertainty (FOU) [52] [51]. An interval Type-2 set can be seen as a collection of embedded Type-1 sets [53]. The firing strength of a Type-2 set is depicted as an interval set with an upper and lower firing strength as shown in equation 3.11.

$$\mu_F(x) = \left[\overline{f}(x), \underline{f}(x)\right] \equiv [\overline{f}, \underline{f}] \tag{3.11}$$

Fig 3.6 shows a Type-2 membership function with its shaded FOU. In this example "12.5" is the crisp input value, the type-2 membership function has membership values bounded by an upper membership value 0.5 and a lower membership value 0.16.

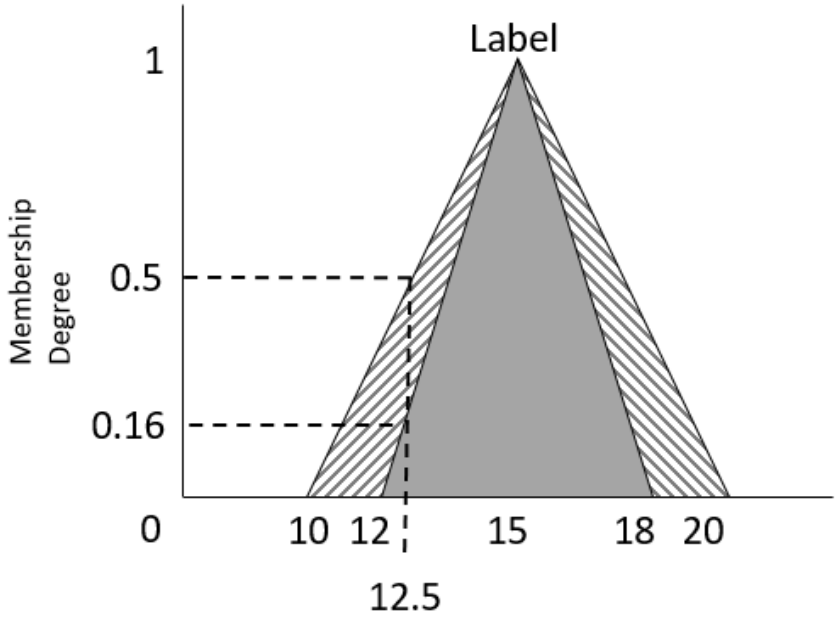


Fig 3.6 Interval Type-2 Fuzzy Set.

3.2.2 Type Reduction and Defuzzification

Type reduction is the act of converting the information of Type-2 sets into Type-1 sets for defuzzification. There are many different techniques of type reduction including: centre of sets, centroid, height and modified height. In all of these techniques a type reduced set is represented as an interval set as seen in equation 3.12 [50].

$$Y_{TR} = [y_l, y_r] \tag{3.12}$$

Centre of set type reduction is the simplest and computationally the least expensive of all the listed techniques and is computed with equation 3.13 [50] where Y_{cos} is an interval set determined by its two points as seen in equation 3.12.

$$Y_{\cos}(x) = [y_l, y_r] = \int_{y^1 \in [y_l, y_r]} \cdots \int_{y^m \in [y_l^m, y_r^m]} \int_{f^1 \in \left[\underline{f}^1, \overline{f}^1\right]} \cdots \int_{f^1 \in \left[\underline{f}^M, \overline{f}^M\right]} 1 \Big/ \frac{\sum_{i=1}^{M} f^i y^i}{\sum_{i=1}^{M} f^i} \quad (3.13)$$

The centroids of the interval Type-2 outputs set must computed prior calculating Y_{cos}, through the use of the interval consequence equation shown in equation 3.14 [50].

$$C_{\tilde{G}^i} = \int_{\theta_1 \in J_{y1}} \cdots \int_{\theta_1 \in J_{yN}} 1 \Big/ \frac{\sum_{i=1}^{N} y_i \theta_i}{\sum_{i=1}^{N} \theta_i} = \left[y_l^i, y_r^i\right] \quad (3.14)$$

The type reduced set can now be defuzzified as seen in equation 3.15. As the type reduced set is an interval set it can be defuzzified by taking the average of the two interval values.

$$Y(x) = \frac{y_l + y_r}{2} \quad (3.15)$$

Chapter 4. Overview of Simulated Annealing & Single and Multi-Objective Genetic Algorithms

Optimisation is a vast and varied field of computational intelligence and many different types of techniques exist. Evolutionary Algorithms is one such type of technique whereby one or more solutions are progressively improved to produce a better solution, some examples include; Genetic Algorithms[54], Big Bang Big Crunch [55] and Simulated Annealing [56]. Another type of optimisation algorithm is swarm intelligence optimisers, in which a group of agents work together to move towards a common goal, some examples include; Ant Colony Optimisation [57] and Particle Swarm Optimisation [58]. Routing algorithms such as Dijkstra's Algorithm [25] and A-Star [28] can be seen as types of optimisation algorithm as they must find the optimal path through a graph.

One of the simplest optimisation algorithms is the hill climbing algorithm which performs an iterative search in a local area. Starting with a random solution within the search space it evaluates that solution giving it a score. From the initial solution it proposes minor changes only accepting them if they produce an improvement to the solution, rejecting changes that would reduce the score of a solution.

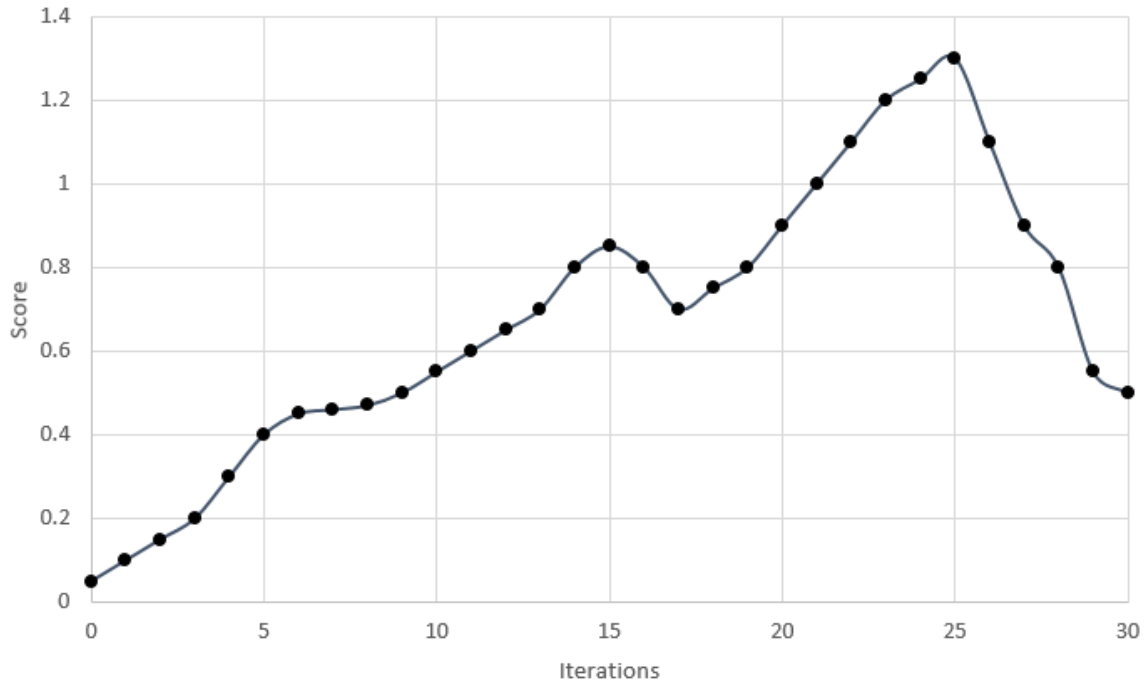


Fig 4.1 Hill climbing score example indicting the local and global minima problem

Fig 4.1 shows an example of how the hill climbing algorithm's score would increase over time, whilst also showing the biggest drawback of this technique. Between iterations 0 and 15 there is a continuous improvement of the score, but at iteration 16 the score decreases, so the hill climbing algorithm would stop at iteration 15 indicating that it was the best possible solution. By looking at Fig 4.1 it is clear that this isn't the case, the best score and thus solution is to be found at iteration 25. In this example iteration 15 is the local minima or the best solution that the optimisation algorithm can find, whereas iteration 25 is the global optima or the best solution possible for the given problem. The hill climbing algorithm has this drawback due to the fact it is solely exploitative and not explorative (see Section 2.1) in its search, only accepting solutions that improve its score.

4.1 Single Objective Optimisation

In optimisation an object is used to guide the search towards the desirable outcome. An objective can come in many different forms but mostly appear as minimisation or maximisation. A minimisation objective could be something such as minimising costs, power usage or wastage. A maximisation objective could be something such as maximising output, tensile strength or signal reception.

4.1.1 Simulated Annealing

Simulated Annealing was designed by S. Kirkpatrick in the 1980s [56]. It is based upon the idea of annealing which is the method of forging metals such as iron in smithing. Taking the idea of annealing and applying it as a statistical model, with its influences taken from the Metropolis Algorithm [59]. The basic idea of annealing is to introduce energy or heat into a material allowing it to be more easily manipulated, but as material cools overtime it can become increasingly more difficult to make changes. Larger changes are easy to make early on in the process whilst the material is in a high energy state and only smaller changes are possible as the material cools, eventually it would be impossible to make large changes to the material without comprising it integrity [60].

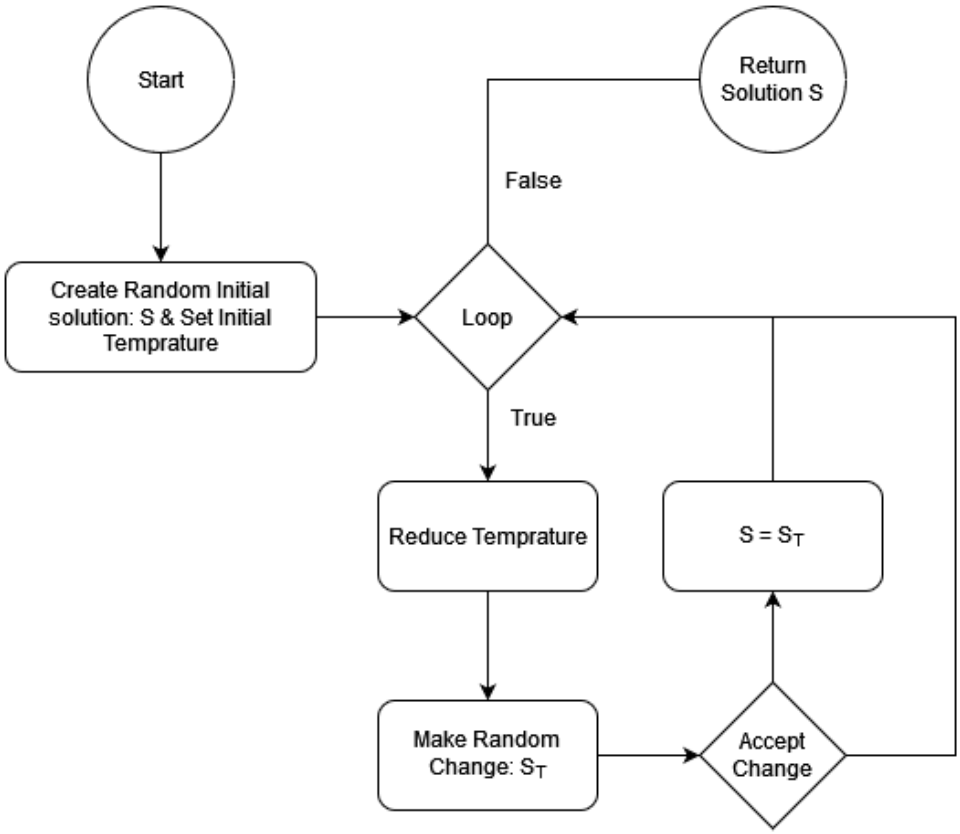


Fig 4.2 Flowchart of simulated annealing

Fig 4.2 shows a flowchart for simulated annealing. It begins by creating an initial random state for the solution and setting an initial temperature for the system. A solution *S* is represented by a set of decision variables, or in other words a set of variables that can be manipulated to affect the objective function. Fig 4.3 shows how a solution could be represented visually with the letters A-H representing decision variables.

Fig 4.3 Simulated annealing solution made up of decision variable.

The flowchart in Fig 4.2 then moves onto the main loop in which there are three stages, temperature reduction, propose random change and determine acceptance. The temperature within simulated annealing is used to govern how likely a proposed change is to be accepted.

By reducing temperature it becomes less likely that undesirable changes will be accepted, pushing the algorithm from an explorative state to an exploitative state. After temperature reduction a random change is made to the solution and stored as a copy *ST* preserving the original. The copy *ST* is then evaluated for acceptance and if it passes the evaluation *ST* becomes *S* replacing the solution *S*. If it fails the evaluation *ST* is discarded and the next iteration of the loop continues. The acceptance criteria of the evaluation uses the Boltzmann probability factor [61], incorporating temperature and the fitness score of *ST*. Fitness score is a measure of how effectively a given solution fulfils the objective of the optimisation.

4.1.2 Genetic Algorithms

Genetic Algorithms (GA) [54] draw inspiration from nature using the core ideas of evolution and survival of the fittest to produce optimised solutions. In nature animals mostly reproduce in pairs producing a number of offspring with genetic material from each parent. Sometimes mutations occur within that mating process and one more of the offspring has some new genetic material that is not present in either parent. In some cases this new genetic material can provide an advantage over other member of the same species, asserting dominance in the gene pool, this is known as survival of the fittest where only the best members of a population thrive. Once the offspring are older they too will produce offspring introducing a new generation into the world continuing the cycle of evolution and survival of the fittest. GA's have been successful in a multitude of different applications from Antenna Design [12], Access Point Design [62], and Fuzzy Logic Controller Design [63] . Fig 4.4 shows a flowchart indicting the different stages of a GA, over the following subsections each part of this flowchart will be explained in detail.

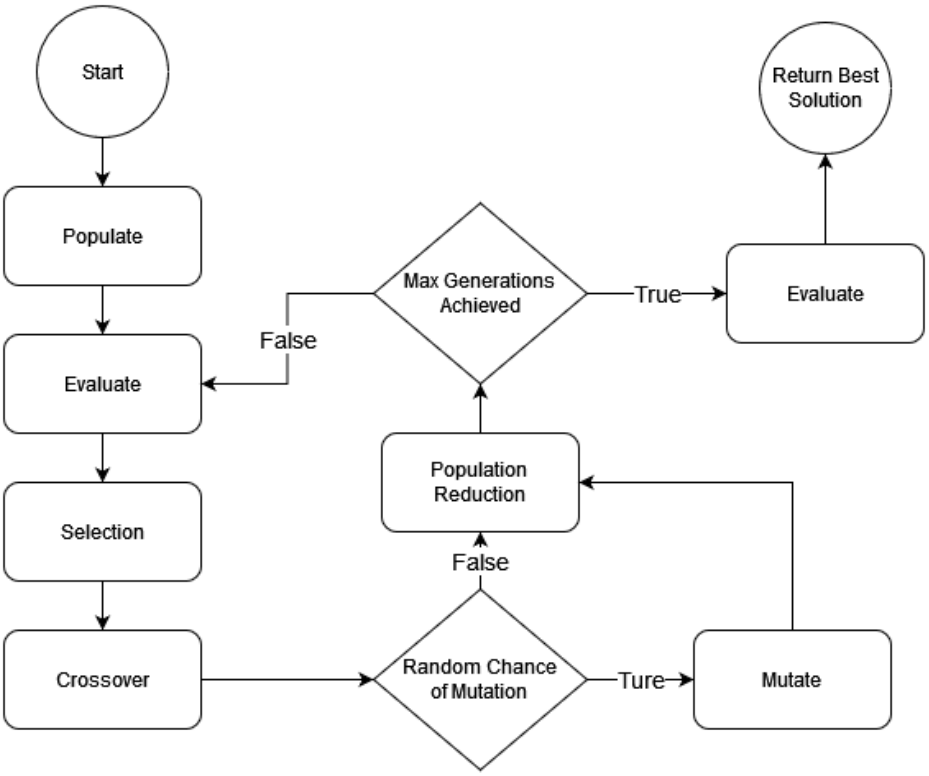


Fig 4.4 Flowchart of a Genetic Algorithm

4.1.2.1 Population

A potential solution to a problem is represented as a member of the population [54]. Each member of the population is represented as a chromosome comprised of a set of genes. Fig 4.5 shows a visualisation of three different chromosomes each containing 8 genes and different encoding styles.

(a)	A	B	C	D	E	F	G	H
(b)	0	0	1	1	0	0	1	1
(c)	1.6	0.5	22	11	1.5	3	0.1	4

Fig 4.5 Chromosome and gene representation with different gene encoding

(a) Problem Specific. (b) Binary. (c) Real Value.

Each gene represents a single decisions variable, with decisions variables being the mutable information that effects evaluation. The number of genes in a GA is problem dependant as each problem would have a differing number of decision variables. In nature groups of the same species make a gene pool, allowing for genetic diversity and a competition for the best mate to exist. These ideas have been takes for a GA so multiple chromosomes exist in a gene pool allowing for genetic diversity and competition for the best mate.

In order for the cycle of evolution to exist and progress within a GA there must be an initial population to mate and evolve. The initial population is built in the population function prior to the looping mechanism. Due to the chromosome being problem dependant so is the creation of the initial population, but there are two overarching strategies to the creation of an initial population. Firstly they could be built randomly by just putting genes together in any order. Secondly it could be built by an expert in the domain knowledge as a good starting point for optimisation. In some cases it can be best to combine these two strategies together and produce a random assortment of chromosome based upon an expert designed chromosome.

4.1.2.2 Evaluation & Selection

For optimisation to exist there must be an overarching goal guiding the search space known as the objective. Each chromosome in the population is rated against this evaluation function for fitness with fitness representing how well a member of the population fulfils the objectives. With each member of the population given a fitness score they can now be put into an order or ranking, making it easy to determine how each member of the population compares to one and other.

Selection is the process of determining which members of the population will be picked to create offspring and thus increase the members of the gene pool. There are two popular styles of selection known as roulette wheel and tournament selection [64]. In roulette wheel two

members of the population are picked randomly for crossover, with no member being picked more than once. In tournament selection the population is split into two groups randomly and then sorted based upon their evaluated fitness score, then in order one member of each group is selected for crossover until all population members have a partner for crossover.

4.1.2.3 Crossover & Mutation

In nature two members of a species must mate in order to produce offspring and increase the size of the population. In a GA this process is known as crossover as information from each parent is taken and crossed together to produce a number of offspring. There are many different methods of crossover including single point crossover, multi-point crossover and uniform binary crossover [65]. In single point crossover the chromosomes are split at one point and half of each parent is passed onto the offspring chromosomes. This is shown in Fig 4.6 where two children are produced from two parents. In the example the split point is in the middle splitting the genes in the chromosome equally, but the split point could be anywhere in the chromosome.

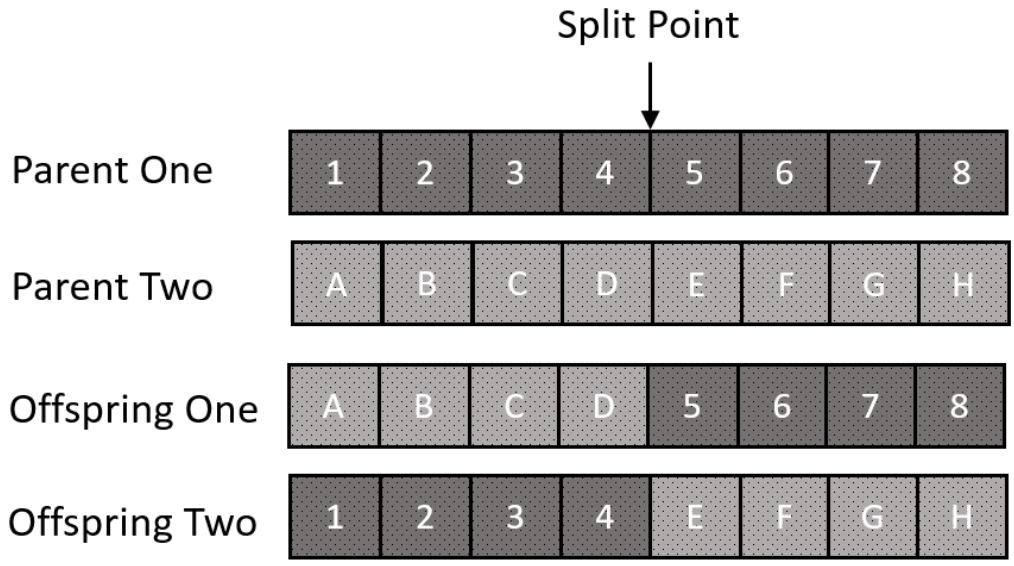


Fig 4.6 Visualisation of single point crossover

Multi point crossover is similar to single point crossover but this time there is more than one split point allowing for different sections of the chromosomes to be used. An advantage of using this is that if a specific set of genes must be kept together they can be. Fig 4.7 shows a visualisation of how multi split crossover can occur with two split points.

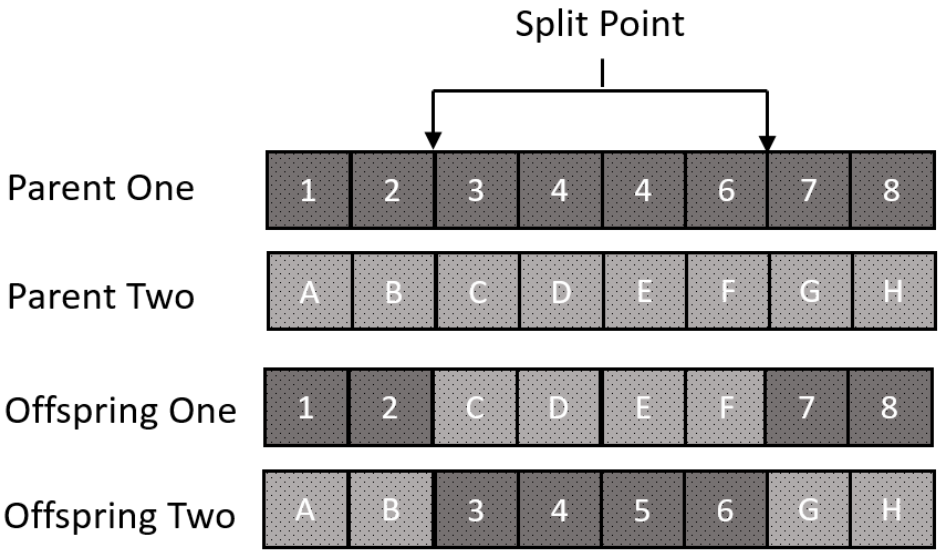


Fig 4.7 Visualisation of a multi-point crossover

The third crossover technique is the uniform binary crossover which can only be used for chromosomes that are represented by binary. In this technique each child is a copy of one of the parents then the crossover mark filter is applied to each child. The crossover mark filter also known as the crossover mark indicates which bit is to be flipped. The crossover mark is randomly generated every time crossover is initiated. Fig 4.8 shows a visualisation of this technique with the "0" bit indicating that the bit is not flipped and the "1" bit indicating the crossover bit is to be flipped.

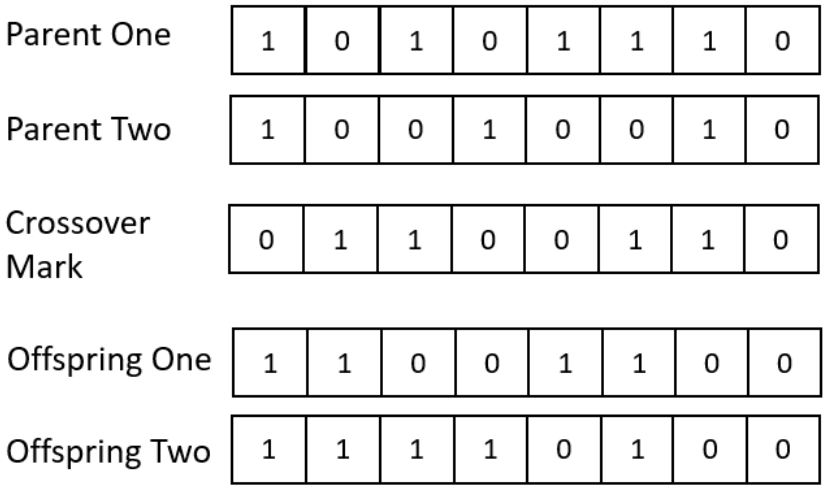


Fig 4.8 Visualisation of binary uniform crossover

Sometimes in nature during reproduction the offspring will have minor changes in their genetic material that where not inherited from either parent this is known as mutation. This idea has been applied to GAs allowing the offspring to have new unseen genes, known as mutation [66]. After any of these crossover techniques there is a random chance that the mutation operator will activate, this is a tuneable value for a GA, with more frequent changes pushing the algorithm to be more explorative, whereas less frequent changes make the algorithm more exploitative. Once these offspring have been created and had the chance for mutation they are added to the general population in preparation for the next generation.

4.1.2.4 Population Reduction and Generations

In nature animals can die in any number of ways controlling the size of the population. In a GA the members of the population cannot die so they must be removed. At the end of every generation the population is evaluated and ranked based upon their fitness scores, then the population is reduced to a predetermined size with the worst performing members being removed first, this includes the newly created members of the population from crossover [67]. A GA takes place over a number of generations, halting when a predefined number of

generations have passed. At which point the best performing member of the population is returned as the solution and the algorithm finishes.

A GA exploits the solution space through the combined mechanism of population reduction and to a lesser extent selection if tournament selection is used. It explores the solution space through the use of the mutation operator introducing new genes randomly to see if they have an advantageous effect. It can use the roulette wheel selection method to help with this exploration process by allowing any two members to crossover. The tuneable parameters of a GA include: the number of generations, the number of population members and the likelihood of mutation.

4.2 Multi Objective Optimisation

Multi objective optimisation is much like single objective optimisation but as the name suggests there is more than one objective. For comparison; in single objective optimisation there could be an objective to maximise the output of an assembly line, but in this example it would never take into account the cost of doing so, effectively saying there is infinite money to solve this issue. This is where multi objective optimisation steps in, looking at the same example there are now two objectives, maximise the output of an assembly line and minimize costs. This case is much more sensible as the desired objective exists whilst ensuing the costs don't get out of hand. There are several successful multi objective optimisation algorithms including; cARMOEA[68], AnD [69], CMMO [70], NSGA-II [71] and NSGA-III [72].

4.2.1 Dominance & Pareto Front

Ranking and evaluation in optimisation is core to determining the worth of a potential solution to a problem. When there is more than one objective evaluation in the traditional sense (comparing the scores of solutions) is no longer possible. This is where domination [73] is used to compare solutions and determine an order for those solutions. Domination is achieved

through the use of 2 rules [73][74]. Given there are two solutions *A* and *B* that both have more than one objective. *A* is dominant over *B* if all of *A's* objectives are no worse performing than *B's* objectives and at least one of *A*'s objectives is better performing than one of *B*'s objectives, otherwise *A* is not dominant over *B.*

In a population-based optimisation algorithm, dominance rules will sort the population into groups of equally evaluated solutions but those groups will be ranked into order. Each group represents a front of solutions Fig 4.9 shows an example of these fronts, with every solution on front 1 better than every solution on front 2, and so on.

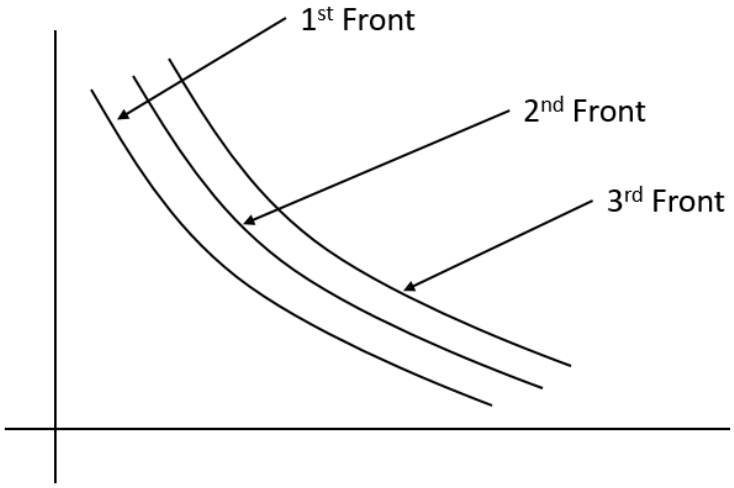


Fig 4.9 Example of fronts in domination

In multi objective optimisation there is this concept of Pareto optimality [75] [76]. A Pareto optimal solutions are a set of solutions that cannot be improved upon without compromising one of the objectives. So a Pareto front represents the best possible front that could be found for a problem. In most but not all cases of multi objective optimisation the objectives are conflicting, producing more than one solution in each front, but if the objectives are complimentary then there will be only one solution in each set.

4.2.3 NSGA-II

Non-Dominant Sorting Genetic Algorithm II (NSGA-II) [77] is a well-used optimisation algorithm across a wide array of domains including: workforce optimisation [78], strategic network design [79] and furnace scheduling on rolling production [80]. As an extension of the single objective GA it uses very similar functionality of selection, crossover, mutation and population control. The biggest differences occur during sorting for selection and population reduction being sorted in a two stages, first with the domination rules (Section 4.2.1) and second with the crowding distance metric being applied to each front. Crowding distance functions as a density metric calculating how closely grouped members of a front are together. Crowding distance is measured by taking the Euclidean distance between two members of the population. Equation 4.1 indicates how to calculate Euclidian distance, where p_1 and q_1 are objective values from one solution and p_2 and q_2 are objective values from a second solution.

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2} \tag{4.1}$$

Tournament selection is used as part of NSGA-II with the population being randomly split into two groups of equal size and sorted for crossover, with the members of each group mating in descending order until the worst performing solution from each group have performed crossover. During population reduction the worst performing members of the population are removed. Fig 4.10 depicts the sorting and reduction process, showing the old population and the new population for the next generation.

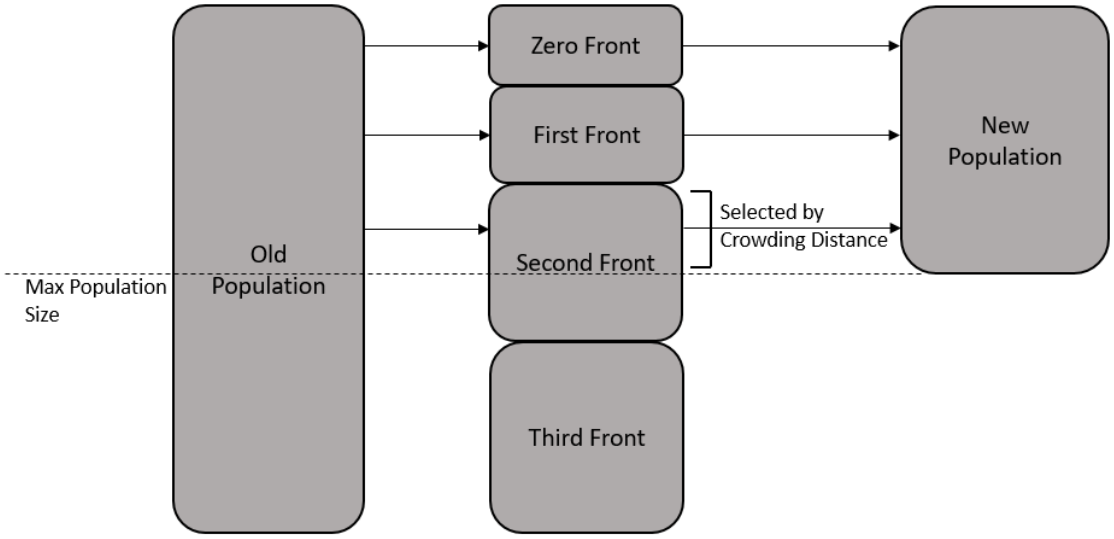


Fig 4.10 NSGA-II population reduction sorting

4.2.4 NSGA-III

Non-Dominant Sorting Genetic Algorithm-III (NSGA-III) [72] is an iteration of NSGA-II again following the same GA ideas of selection, crossover, mutation and reduction. Once again using the domination rules (Section 4.2.1) for sorting, but the crowding distance metric has been done away with and replaced with the Niche-Preservation operator to sort the contents of each front. The Niche-Preservation operator has three stages: Normalise, Associate and Niche-Preservation. To normalise the maximum and minimum values each of the objective function must be known, the minimum value is simply the smallest value possible for each objective and can be pre-cached prior to exertion. The maximum value of each objective is more computationally expensive as it must be calculated dynamically during execution every generation. Equation 4.2 shows the normalisation equation, with the min and max values calculated as described above and *fOi* being the specific objective value that is being normalised.

$$norm(fO_i) = \frac{Max - fO_i}{Max - Min} \quad (4.2)$$

Every solution now has a set N-dimensional normalised coordinate associated with it where N represents the number of objectives. The next stage is to associate these coordinates with a point on a pre-calculated reference plane. The reference plane can either be created using a systematic approach or designed by the programmer. In the NSGA-III paper, it is suggested that the Das and Dennis's [81] systemic approach can be used to apply the points onto the plane. Fig 4.11 shows a 3-dimensional normalised reference plane. The reference points are equidistantly spaced across it, with the points *f1(0,0,1), f2(1,0,0)* and *f3(1,0,0)* at the corners of the normalised plane.

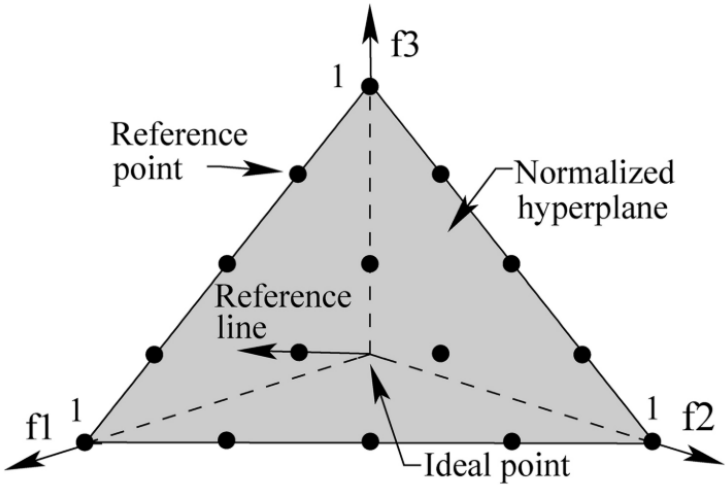


Fig 4.11 Visualisation of a 3-dimensional reference plan [72]

The association operator works slightly differently from this point onwards for selection and population reduction, the following explanation is for population reduction. The association operator places N-dimensional points on the reference plane in two stages, firstly in batches and then individually. The batches are placed from each of the fronts in descending order, with the first front placed first then the second front next and so on, until the placement of an additional front would infringe upon the maximum population limit. At which point they are

placed individually based upon a set of criteria as part of the Niche-Preservation operation. Fig 4.12 shows a visualization of the placement of front members and how they are broken down into the batch placement and the individual placement.

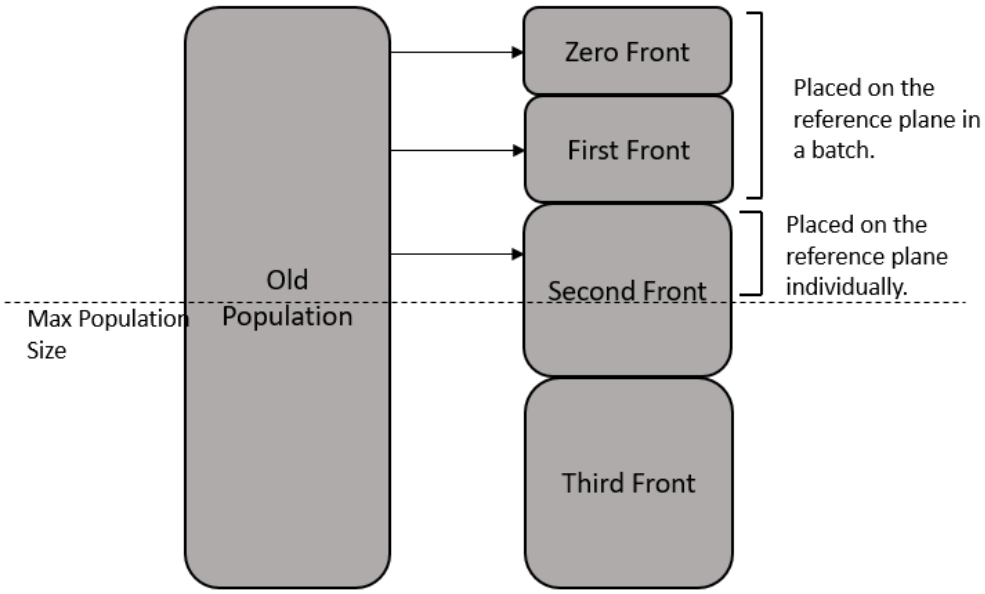


Fig 4.12 Visualization of which members of the population are placed on the reference plane

Once the members of the batch placement are on the reference plane they each need to be associated with one of the reference points. A point is associated to the closet reference point, when the distance calculated between is the perpendicular distance between a point and the reference line. The reference line is defined as a line from the origin passing through a reference point. Fig 4.13 shows a visualisation of a set of points and how they are associated with reference lines.

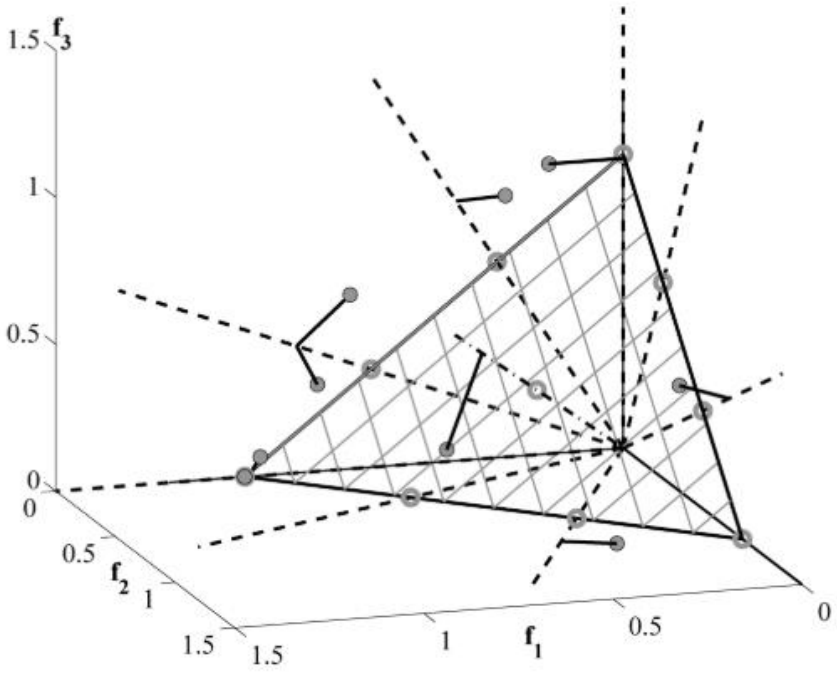


Fig 4.13 A visualisation of points and the distances to the associated reference lines and therefore associated reference points

The remaining points due to be placed individually are placed with the Niche-Preservation operation. There are two reference planes for this operation and first is the reference plane with all the points added in the batches, and second a new plane that shall be referred to as the niche plane in order to avoid confusion between the two. This new plane has the same reference points as the reference plane with all of the point from the next front (shown as the second front in Fig 4.12) associated to those reference points.

Niche-Preservation happens in an iterative loop until the number of points on the reference plane is equal to the max population size, as each point represents a member of the population after all. Prior to this loop a set of all the reference points is collected called *S*. First a reference point in set *S* is identified with the minimum number of associated points on the reference plane, we shall call this the S_p. If there is more than one point that fulfils the requirements to become S_p, then one is picked at random. If S_p has points associated with it on the niche plane,

then the one with the smallest distance to S_p reference line is selected. Once a point is selected it is removed from the niche plane, and added to the reference plane. If there are no points associated with S_p on the niche plane, then S_p is removed from the set S and the next iteration of the loop begins if appropriate.

As said earlier this operation is different for selection, as all members of the population can be added in the batch stage. Members of the population are sorted for selection at two levels, first they are sorted by their fronts, and then by their perpendicular distances to the reference points. NSGA-III is seen to be an improvement to NSGA-II at the cost of computational complexity [72].

4.3 Constraint Handling

As the complexity of problems increase they begin to incorporate constraints along with objectives, which is known to be too difficult for Evolutionary Algorithms to handle [82]. A constraint is a rule that works alongside the objectives in order to achieve the desired result of optimisation. During optimisation, solutions that break the constraints are allowed to exist as an explorative step during the search. Any solutions that break one or more of the constraints are known as infeasible solutions, whereas a solution that doesn't break any constraints is known as a feasible solution. Two solutions can break a different number of constraints making them both infeasible but to different degrees, therefore there are degrees of infeasibility. There are two major types of constraints: a single constraint commonly named simple constraint and a bound constraint. A single constraint such as the sum of two numbers cannot be greater than a set value, and a bound constraint where a number must fall within a range. The search space can be divided up into a number of regions based upon how they are broken down. Constrained problems can be broken down into four types each having a different search space. [83]:

Type I: The constrained Pareto front is the same as the bound-constrained Pareto front.

Type II: The constrained Pareto front is a subset of the bound-constrained Pareto set.

Type III: Some portions of the constrained Pareto front are the same as the bound constrained Pareto front.

Type IV: The intersection of both Pareto sets is null and they have no common region.

Knowing how these types of problems exist allow a mapping of the search space to exists prior to optimisation in order to see if a feasible solution is possible from the space. Each problem type has different mapped area in the search space.

In order to optimise with constraints, Constraint Handling Techniques (CHT) need to be incorporated into a given optimisation algorithm. Three popular CHT include: the constrained dominance principle [71] [84], self-adaptive penalty function [85] and stochastic ranking [86]. NSGA-II and NSGA-III use the dominance principle as their CHT which extends the idea of the dominance rules to include constraint dominance. This extension to the rules happens prior to the dominance rules. Given there are two solutions and both have a set of objective scores and a set of constraints violation values, the rules can be used to split the population into fronts. Solution A constraint dominates solution B if A is feasible and B is infeasible. Solution A constraint dominates solution B if both solution A and B are both infeasible but A has a lower constricts violation count than B. If Solution A and Solution B are both feasible then the normal domination rules apply (Section 4.2.1).

4.4 Hyper Volume Indicator

As multi-objective optimisation algorithms have become more complex and better at solving multi-objective problems, it has become harder to determine the quality of an algorithm. This difficulty comes from the fact that by their nature multi objective problems rarely get a single solution as the result. As sets of solutions from the same front are returned from an algorithm,

this makes it very difficult to determine if one front is better than another if they don't dominate one another.

The Hyper-Volume Indicator (HVI) [87] [88] has come to the forefront as a methodology of evaluating the quality of a solution fronts. The HVI is an extension of the Lebesgue measure [89] invented by Henri Lebesgue. The Lebesgue measure is a standard way of measuring the subset of an N-dimensional Euclidian Space.

A solution front exists within an N-dimensional space where N represents the number of objectives. The HVI is used to determine an approximation of the area explored by an algorithm and the density of solution across the solution front. A larger value indicates the area has been well explored across with solutions on the extreme edge of the possible solution front, it also indicates that members of the population are well spaced across the solution front. A smaller value shows that the problem space has not been well searched and that solutions on the solution front are densely grouped together.

Chapter 5. NNIR: N-Non-Intersecting-Routing Algorithm for Multi-Path Resilient Routing in Telecommunications Applications.

Large telecoms companies nowadays have a global presence with locations across continents and it is important that these locations stay connected at all times. These connections are maintained through the utilisation of networking infrastructure as described in Section 2.3. Locations across Europe are connected through the use of metro nodes, ducts and data exchanges as described in Section 2.3.1. With locations connected together it is important that data packets can be transferred across a network as efficiently and quickly as possible. This efficiency can be assured by the use of a routing algorithm such as Dijkstra's algorithm or the A-Star algorithm, as described in Section 2.4.1 and Section 2.4.2 respectively. These algorithms work perfectly but only when the networks infrastructure is operating as intended, but there are many reasons why a network ceases to work as intended. This could be something as simple as a power failure in the network, or it could be due to an increase in network activity slowing the network to a crawl, alternately it could be something more malicious such as network attack.

Traditional routing algorithms are particularly vulnerable to single points of failure (SPoF) [31] [90]. When a portion of a network graph fails, be it an edge or a vertex, and that portion is being used by a route the route is no longer valid. Traditional routing methods are not capable of being prepared to deal with these problems instantaneously, they would require the recalculation of thousands of routes. This is where resilient routing as described in Section 2.4.3 becomes useful. With the primary route being unusable, having a secondary pre-calculated route that uses none of the same infrastructure apart from the start and finish locations, allows the network to instantaneously switch over to this back up route with minimal loss of service.

In resilient routing it is important to know that the cost of the N routes is the total cost of the solution. This is due to the objective of resilient routing is to provide N routes between two locations as the lowest total cost without the routes ever sharing infrastructure apart from the start and finish locations.

In this chapter the N-Non-Intersecting-Routing-Algorithm (NNIR) will be introduced with a detailed description of its methodology, experiments and results which was originally presented in a journal paper in International Journal of Computational Intelligence Systems published in 2020 [33].

5.1 The NNIR Algorithm

Fig 5.1 shows an overview of the proposed N-Non-Intersecting-Routing-Algorithm (NNIR) which consists of two major parts, Dijkstra's algorithm, and a Genetic Algorithm (GA). The GA runs for a number of generations returning the best ranked member of the population. For simplicity all descriptions in this chapter will be under the assumption that only two routes are required, but the algorithm can easily be scaled up to use more routes.

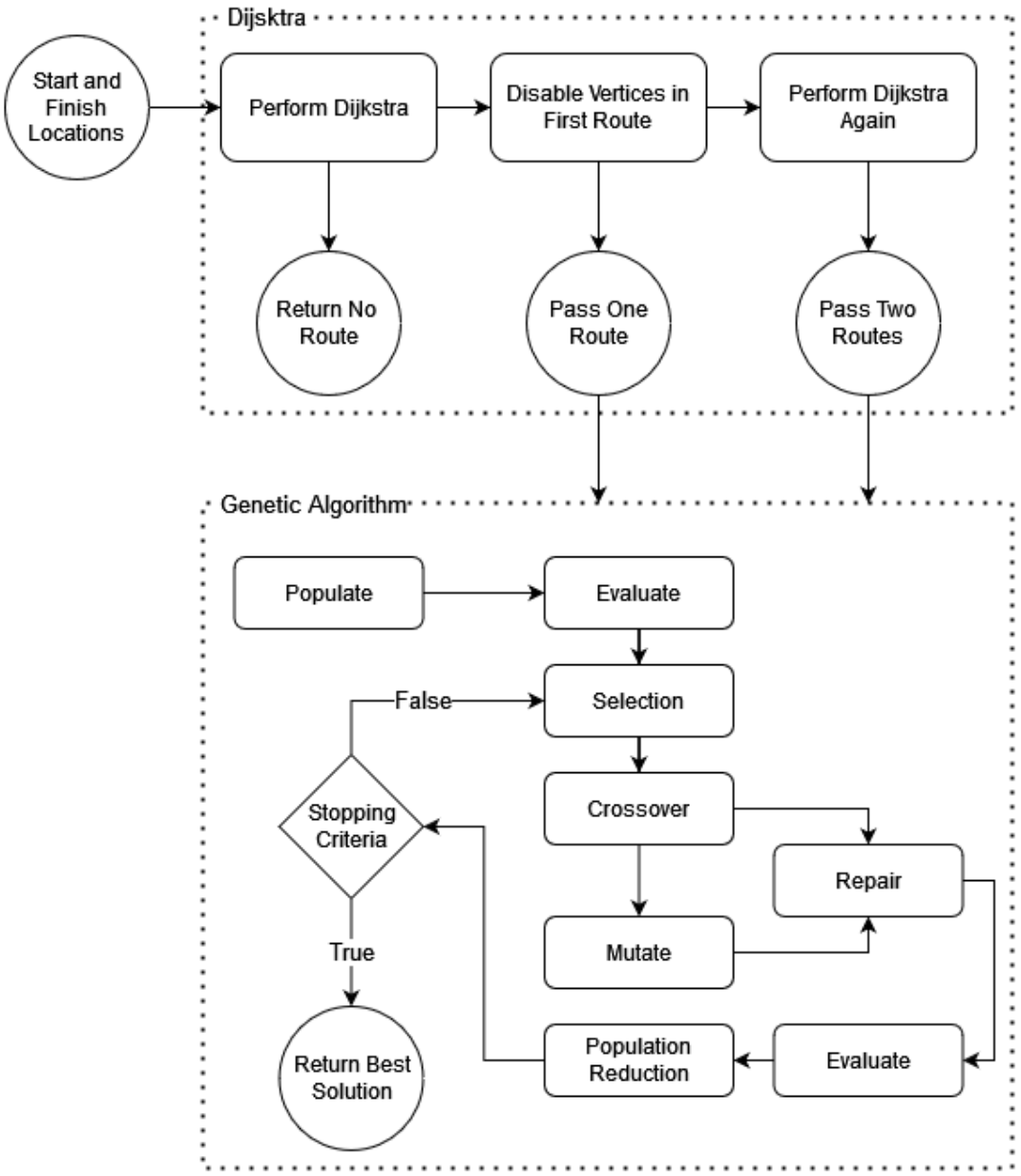


Fig 5.1 An overview of the N-Non-Intersecting-Routing Algorithm (NNIR)

The algorithm begins by calculating a route between the start and finish locations using Dijkstra's algorithm to calculate a primary route. If a route can't be found the algorithm terminates. If a route can be found it removes all vertices and edges in the primary route from the graph with exception to the start and finish locations as these locations are required for the calculation of the secondary route. It then moves on to calculate another route using Dijkstra's

algorithm to produce the secondary route. A primary and secondary route cannot always be found this way, but if they are found this way these two routes are the worst-case solution.

5.1.1 Solution Representation and Population Initialisation

The routes found by Dijkstra's algorithm are passed into the Genetic Algorithm (GA) to be used as part of the initial population. Each member of the population is comprised of one or more chromosomes. The number of chromosomes is based upon the number of desired routes, with *N-1* chromosomes if *N* is the number of desired routes. Each chromosome is comprised of a set of genes with each gene representing a location in the network, known as vertex in graph theory. Fig 5.2 shows an example of how a chromosome is represented in NNIR with the start and finish locations.

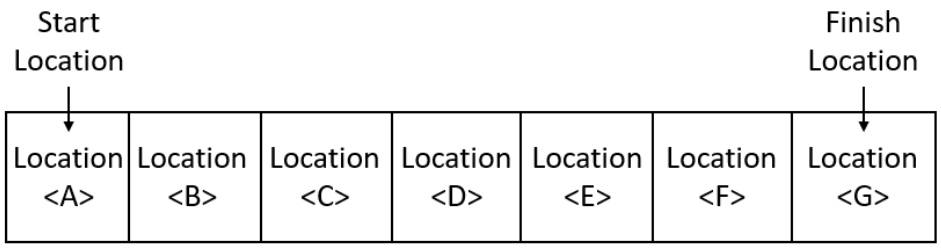


Fig 5.2 Example Chromosome in NNIR with the start and finish locations labelled

The start and finish locations are bound to the first and last position in the chromosome, they can never occupy another location in the chromosome without making it invalid. The order of the genes in the chromosome denotes the route taken to get between two locations. In Fig 5.2 the route is shown as *A, B, C, D, E, F, G*. Due to the nature of routing and that not all routes are of the same length, a chromosomes length is variable and can be changed as required. A solution is represented by N-1 chromosomes and one final route that is always calculated with Dijkstra's algorithm.

In a GA it is important to start with a high enough genetic diversity in order to explore the problem space effectively whilst ensuring that all solutions are still valid. To that end the population initialisation process uses the primary and secondary routes from Dijkstra's algorithm as part of the initial population. The rest of the initial population is calculated in an iterative process by randomly removing a small portion of the vertices in the graph and using Dijkstra's to calculate N routes between the starts and finish locations with N-1 of them becoming chromosomes as part of one solution, and the final route being stored. Fig 5.3 shows the iterative process used to create the initial population.

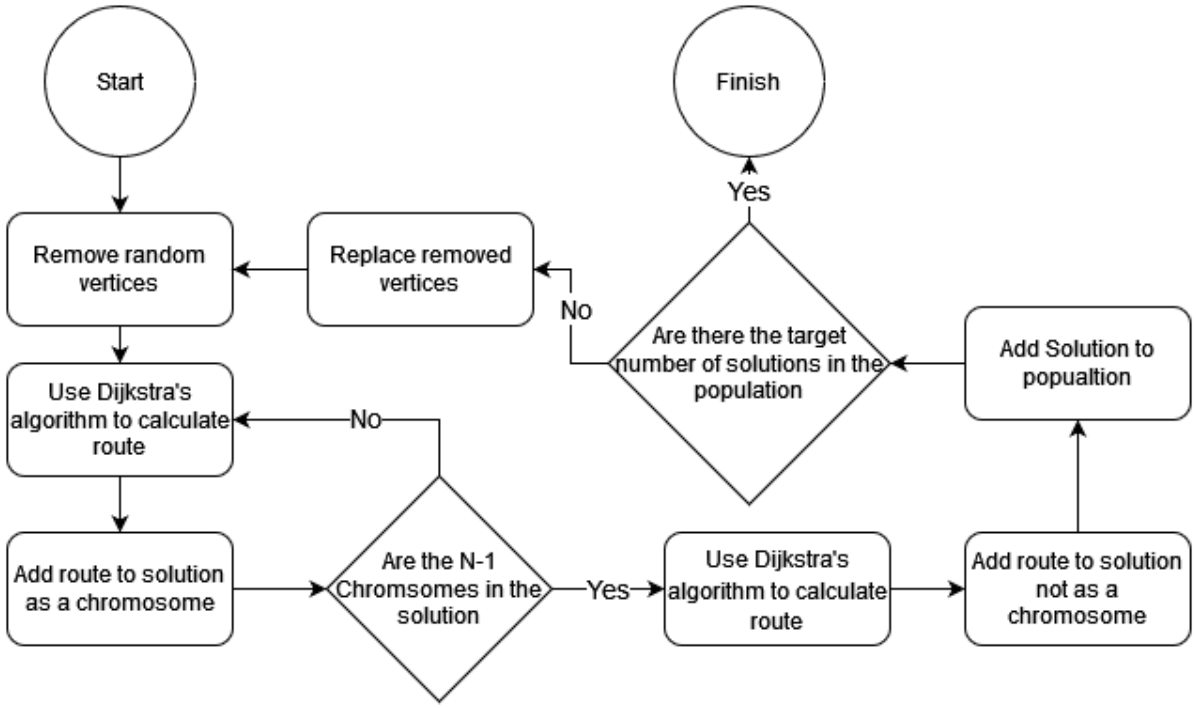


Fig 5.3 Flow chart for the population initialisation process in the GA of NNIR

There are three conditions used during vertices removal that guide the randomness. First the start and finish locations cannot be removed as this would not allow Dijkstra's algorithm to produce a route. Second at least 10% of the network must be removed so enough of the network is missing every iteration to allow for differences in the initial population. Finally, at least 10% of the shortest path must be removed so that the Dijkstra's Algorithm doesn't create a set of chromosomes with the same shortest path. There is one final constraint to be considered when

producing the initial population; no duplicate solutions are allowed to be added to the population. Using these rules will help guide the GA into having a diverse initial population to search the problem space.

5.1.2 Evaluation and Selection

A GA requires an evaluation function to fulfil the objectives of the optimisation. The objective of resilient routing is to produce a set of routes between start and finish locations with the lowest total cost that doesn't intersect at any point, ensuring there is no SPoF. The evaluation function uses the edge costs to calculate the fastness of each population member. The equations 5.1 and 5.2 denote how this cost is calculated. The total cost of a solution is comprised of two parts: the cumulative cost (CC) (the cost of all vertices of the routes represented by a solution) and the route discount (RD). RD is calculated in equation 5.1 where RD is equal to the desired number of routes (N) minus the number of solution routes (SR).

$$\mathrm{RD} = N - \mathrm{SR} \tag{5.1}$$

$$Total\ Cost = \frac{1}{\mathrm{CC}^{\mathrm{RD}}} \tag{5.2}$$

The total cost as shown in equation 5.2 is represented as a fraction so that all fitness values are represented as an interval value [0, 1], 1 over CC to power of RD. The denominator of equation 5.2 is the CC to the power of RD so that route discount can make solutions that do not reach the desired number of routes score exponentially worse by the number of routes that they are missing. As routing in most cases is looking to minimise the costs, NNIR treats fitness function as a maximisation function, in any cases where the cost is to be maximised so the fitness function should be minimised.

The GA in NNIR uses the tournament selection process [64] to pick which members of the population are chosen to be crossed over. The population is randomly split into two equal sets

and then crossover is performed on the best of each set produce new members of the population. Once the best members have been crossed over the next best members do the same to produce new population members, this continues iteratively until all solutions have been used to create offspring. This strategy helps the idea of survival of the fittest propagate through the population over the course of several generation.

5.1.3 Offspring Creation

In a traditional GA, new members of the population are added through the combined use of the crossover and mutation operators. In NNIR this is still the case with the addition of the repair operator, which ensures the validity of offspring prior to them being added to the population. The crossover operator is key to exploiting the best information within the chromosomes, to this end single point crossover is used within NNIR. The single point is randomly selected in each parent solution independently. An example of crossover is shown in Fig 5.4. The example shows the range that the parents can have the split point, parent one has a range between *E*-*G* and parent two has a range between *B*-*F*. Parent one's split point is picked between *K*-*T* and parent two's is between *D*-*E*. Then the offspring are created in the standard split point way, with half of each parent used to make up each offspring.

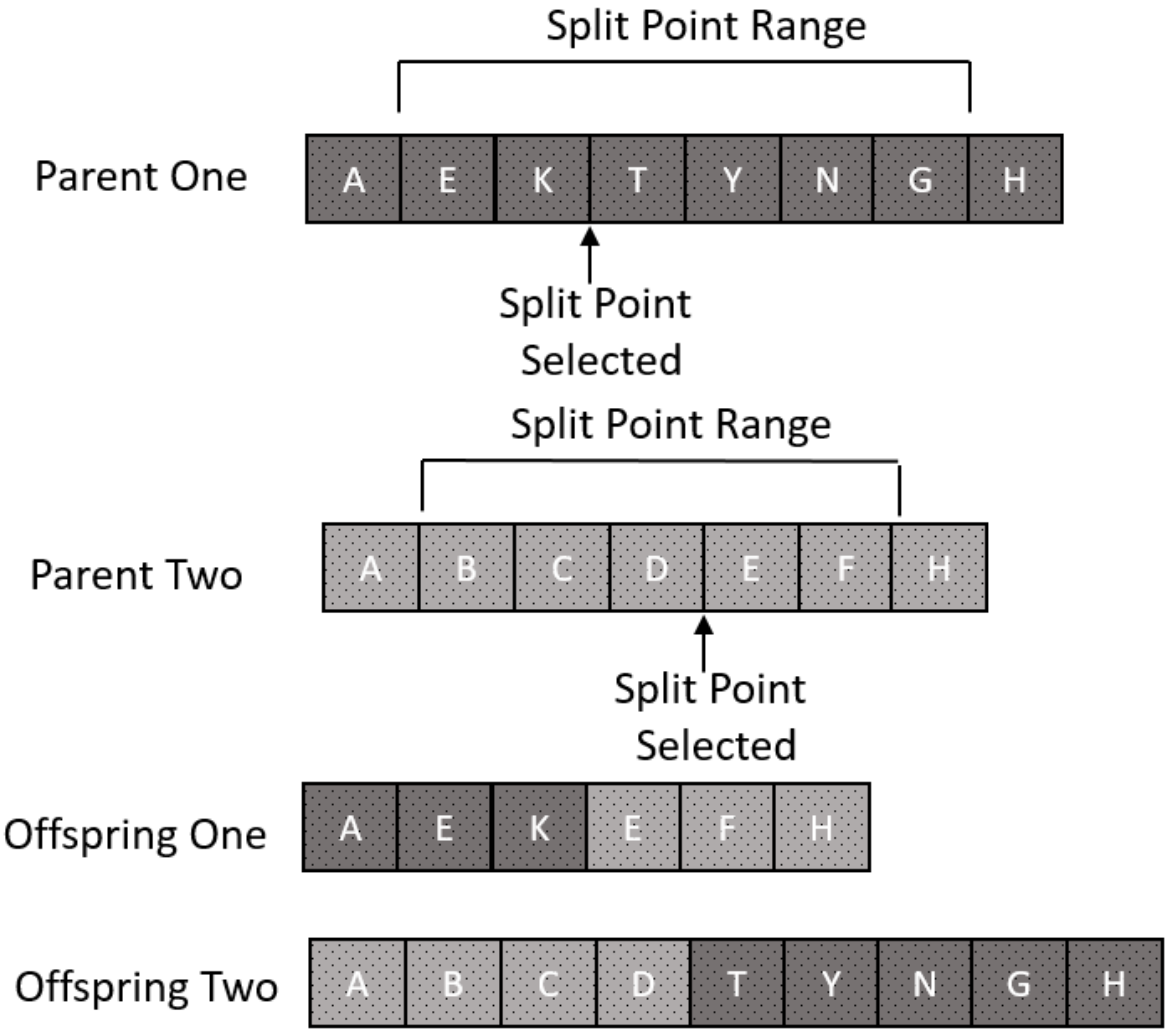


Fig 5.4 Crossover example in NNIR with the randomly split point and dynamically sized chromosomes

Crossover is great at exploiting the current search space when combined with the tournament selection but it has a limited ability to explore new solutions. This is why there is a need for the mutation operator to create minor changes within the chromosome to introduce new information into the search space. Mutation occurs by random chance so that not all members of the population are mutated, otherwise crossover would not be able to exploit for potentially beneficial solutions. A gene (vertex) from a newly created child is selected at random, the first and last genes are excluded from this random selection. The gene selected is then removed

from the chromosome and replaced with a different vertex (every vertex is represented by a gene). The vertices that can be selected as a replacement follow one simple rule. The new vertex to be inserted into the chromosome must be a different vertex connected to the previous vertex. If there are no such vertices to connect then put the originally connected vertex back into the chromosome and try a different vertex randomly excluding any vertices that have previously been selected for mutation. If none of the genes (vertices) can fulfil this rule then mutation is not possible and the operator exits without performing mutation.

The repair operator is an integral part of NNIR, without it search times are much larger. The repair operator is used to reduce the incorrect number of solutions generated by crossover & mutation. There are two sub operations used to repair a chromosome namely loop removal and reconnection.

Loop removal is a simple but effective way of reducing invalid chromosomes. Fig 5.5 shows loop removal, if a loop is detected by a vertex being visited more than once, every element of the loop is removed. One of the constraints of resilient routing problem is that no edge of vertex can be visited more than once. Due to the nature of the crossover and mutation operators it is common for loops or cycles to occur in the graph.

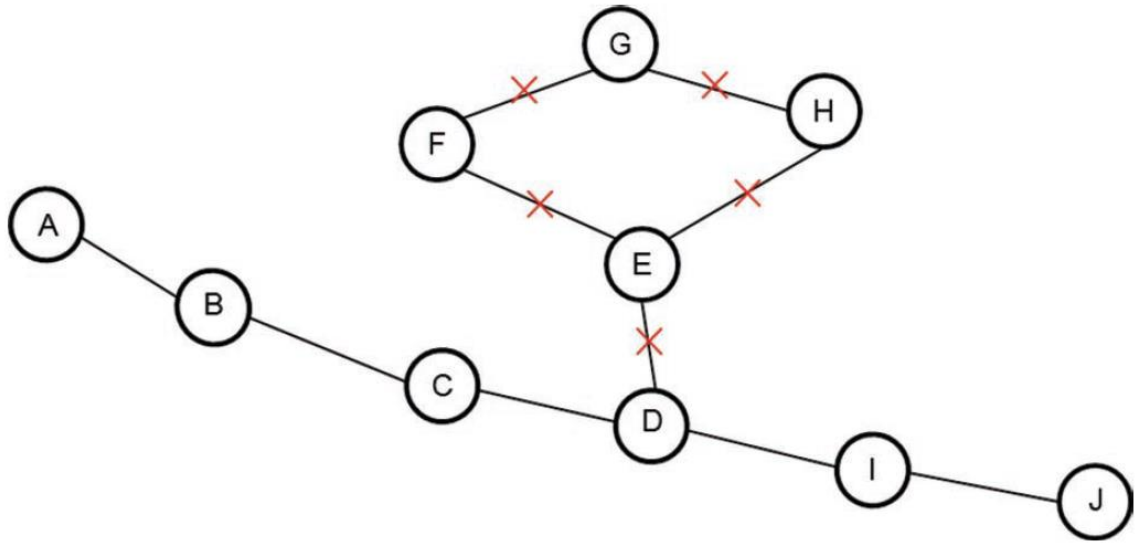


Fig 5.5 Visualisation of loop removal

Reconnection is the more computationally expensive of the two repair methods, but it repairs chromosomes far more effectively. The method iterates through a chromosome to find if any genes are not connected to the next. If two genes are found to not be connected then Dijkstra's algorithm is used to find a route between them in an attempt to reconnect them. These new genes are then inserted into the chromosome in between the disconnect genes. The key constraints of the problem still apply and if a chromosome uses the same edge or vertex more than once loop removal is run again.

Due to nature of the information being represented by the chromosomes they are extremely fragile and prone to inaccurate solutions. The repair operator is an effective tool to combat these issues and help the GA to perform a search of the problem space.

Once the crossover, mutation and repair have taken place for all the chromosomes in a single solution the final route is calculated with Dijkstra's algorithm. This is an easy way to find the last route in a solution which is guaranteed to be the shortest route left available without breaking any of the problem constraints.

5.2 Experiments & Results

In order to prove the validity of NNIR a set of experiments across two data sets has taken place. Firstly, as a baseline comparison between NNIR and traditional routing methods, this is so that any possible improvements that NNIR has over traditional routing methods can be established. Secondly, in order to establish a comparison between the GA present in NNIR with another Evolutionary Algorithm.

During the experiments there are two metrics to take note of: first is a total cumulative cost in other words the fitness function, second is the existence of a resilient route where one was not present prior. This second metric is more important that the first, as the idea of resilient routing is to provide multiple non intersecting routes between two locations.

The comparison to traditional routing methods is between NNIR, Dijkstra's Algorithm and the A-Star Algorithm. The comparison of Evolutionary Algorithms is between NNIR with its GA, and NNIR with Simulated Annealing. Dijkstra's Algorithm and the A-Star Algorithms have been selected due to their success in a wide array of route applications [2], [91]. Simulated Annealing has been selected for comparison as it's a very successful optimisation algorithm [60] and has shown success in the Traveling Salesman problem [24] which is a different routing problem. For this problem to ensure a fair comparison Simulated Annealing has been given access to the repair operator in an attempt to make more solutions presented by Simulated Annealing valid. Simulated Annealing also uses the mutation operator from the NNIR GA to make changes to its solution. Just as within NNIR the final route is found through the use of Dijkstra's algorithm.

All the experiments presented are for a two route scenario (where $N = 2$) which shall be referred to as the primary and secondary routes. The experiments undertaken are across two data sets: a Telecoms Data set, provided by British Telecom. The other data set is an open source data

set taken from the Ordinate Survey of the road network for the town of Exeter, UK [92]. The road data is openly available for use whereas the BT data is confidential data. The telecoms data set is the origin of the resilient routing problem and initially illuminated the need for this type of algorithm. Whereas the Ordinate Survey Road data set has been selected for two reasons: first to show the algorithm can be applied to a differing domain and secondly to allow anyone to reproduce the experiments presented.

During an execution of one experiment start and finish locations are selected at random, with no start and finish location picked more than once in order to stop experiment duplication. The Telecoms data set has 250 discrete routing scenarios and the Road data set has 500 discrete routing scenarios. The results of the experiments are presented in three areas: blocking where Dijkstra's algorithm cannot find the resilient route, sub-optimal where Dijkstra's algorithm finds both the primary route and the resilient route but the total cost is not the lowest possible, and finally the combined results which shows the other two scenarios in one result.

5.2.1 The Blocking Results

In Section 2.4.3.2 an explanation of resilience blocking is given, where we describe that resilience blocking exists when the shortest path uses key vertices that would allow for multiple paths to exist between two locations. It is not always possible to find a secondary route between two locations as there may be only one route to a location.

In Table 5.1, we can see that NNIR gives the largest number solutions in both data sets with 70% in the Telecoms data set and 36.8% in the Road data set. Simulated Annealing is the next best performing with 64% in the Telecoms data set and 24.2 % in the Road data set. Finally Dijkstra's algorithm and the A-Star algorithm have the same results with 50% in the Telecoms data set and 24.2% in the Road data set. Curiously Simulated Annealing, Dijkstra's Algorithm and A-Star all found the same quantity of resilient routes in the Road data set.

TABLE 5.1 The number of solutions where the given algorithm finds a resilient route across both data sets

	Percentage of cases where resilience was found	
	Telecoms	**Road**
Dijkstra	50% (125 out of 250)	24.2% (121 out of 500)
NNIR	70% (175 out of 50)	36.8% (184 out of 500)
A*	50% (125 out of 250)	24.2% (121 out of 500)
Simulated Annealing	64% (160 out of 250)	24.2% (121 out of 500)

Due to the nature of the roads in the UK and particularly in urban areas such as Exeter, the road data set has many dead ends (a road that has the same entrance and exit), which explains why the number of resilient routes is much lower. This nature also explains why all algorithms perform much worse on the road data set than the telecoms data set.

In order to achieve resilience in cases that have been blocked NNIR increases the cost of the primary route, thus freeing up critical infrastructure and allowing it to be used for the resilient route. Fig 5.6 shows the percentage increase in the primary route cost to allow a resilient route to exist when using the NNIR algorithm. Both the Telecoms and the Road data sets are represented in the graph with the Telecoms being the left most of each range represented in blue, the Road data is the right most of each range represented in orange. The majority of the Road data set cases are within *0-1%* increase range, therefore by increasing the primary route cost by a very small margin a resilient route can be found. The Telecommunications data set on the other hand is much more spread across the possible ranges, but notable in 7 cases a *2-3%* increase and in 6 cases a *10-11%* increase allows for a resilient route to be found.

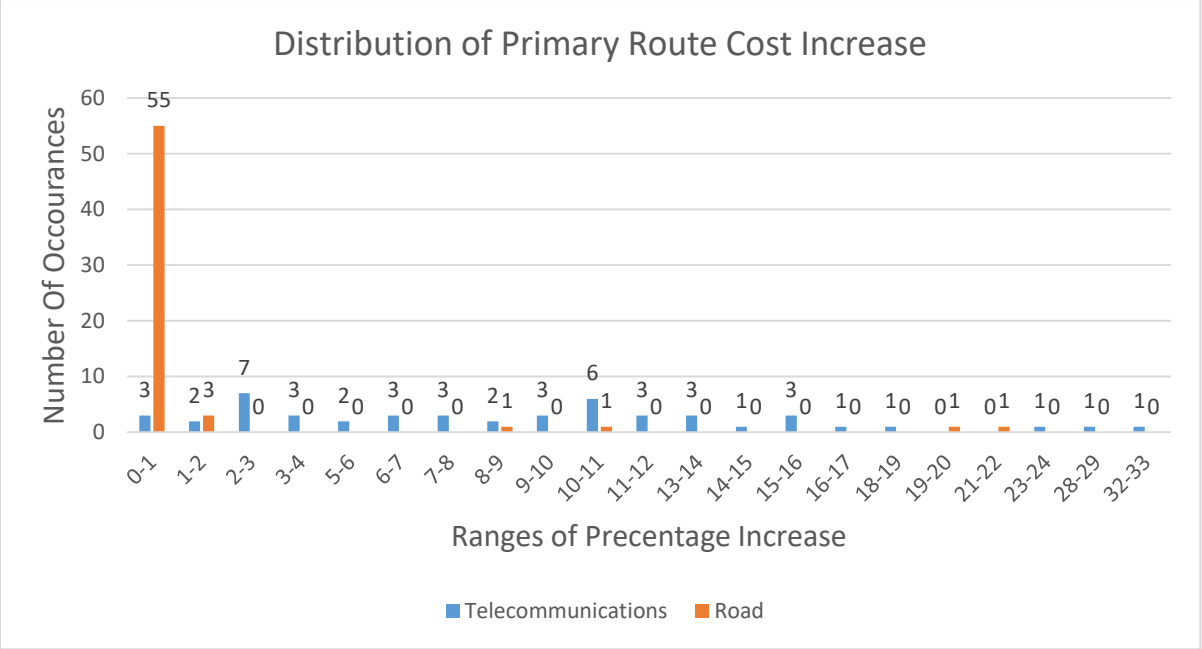


Fig 5.6 Distribution of primary route cost increase as part of the blocking example

5.2.2 Cost Reduction Results

The cost reduction scenario is described in Section 2.4.3.1, where we describe that the total cost of all routes determines optimality not just the cost of a single route. For this reason we say that traditional routing algorithms do not find the optimally shortest routes when combined together, but by increasing the cost of the primary route it allows for a resilient route to take a more optimal path, reducing the total cost of both routes combined. The following experiment compares NNIR, Simulated Annealing and A-Star to a baseline result from Dijkstra's algorithm.

In Table 5.2 the number of cases where the total cost of the two routes is less than Dijkstra's algorithm. NNIR has the best reduction in both data sets with an *18%* reduction in the Telecoms data set and *17.8%* reduction in the Road data set. The A-Star Algorithm doesn't present an improvement in either of the data sets. This is due to the similarities between Dijkstra's Algorithm and the A-Star Algorithm with them both being inherently greedy in their search

strategy. Simulated Annealing does not perform well in either of the data sets with no improvement in the Telecoms data set and a small improvement of 0.6% in the Road data set.

TABLE 5.2 Percentage of cases with a reduction in the total cost when compared to Dijkstra's Algorithm in the cost reduction example

	Percentage of cases with a reduction in total cost vs Dijkstra	
	Telecoms	Road
NNIR	18.0% (45 out of 250)	17.8% (89 out of 500)
A*	0.0% (0 out of 250)	0.0% (0 out of 500)
Simulated Annealing	0.0% (0 out of 250)	0.6% (3 out of 500)

Just like in the blocking scenario in order to produce a more optimal result the cost of the primary route must be increased, that increase is shown in Fig 5.7. Fig 5.7 shows the increase in the primary route cost found by NNIR when compared to the baseline primary route cost found by Dijkstra's algorithm. The majority of the Road data increases the cost of the primary route are between *0-1%* and it's mostly clustered to the lower end of the scale. Whereas the Telecoms data is spread across the scale with one case having an increase of between *80-81%*. In all of these cases the total cost of both routes combined is now lower than the routes found by Dijkstra's algorithm.

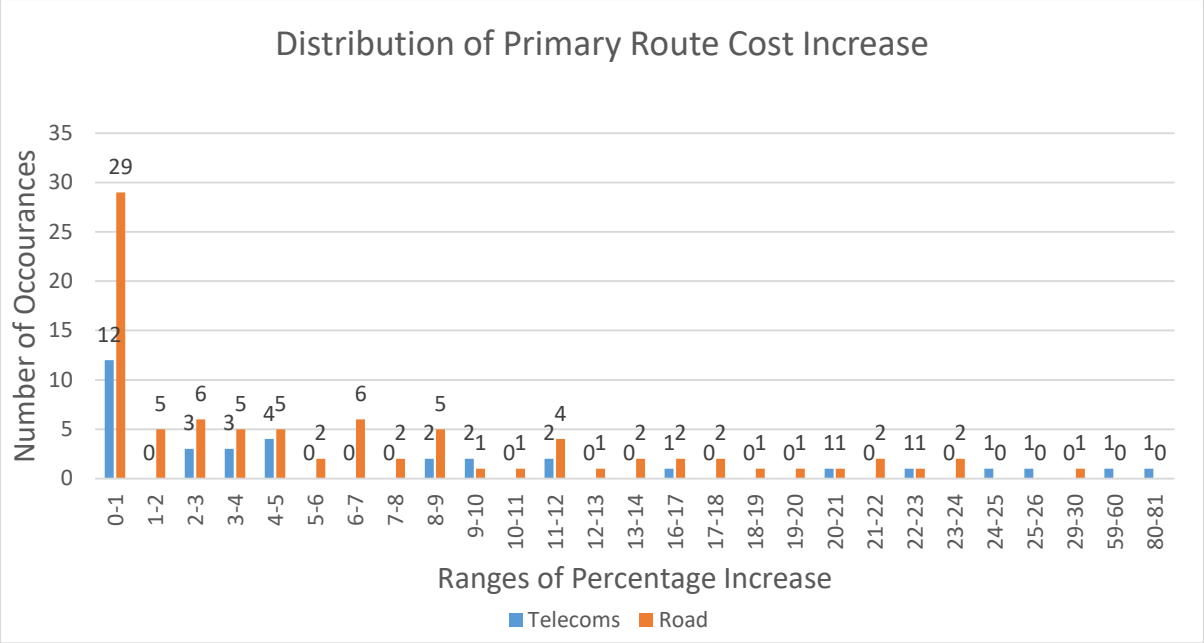


Fig 5.7 Distribution of Primary Route Cost Increase as part of the solution to the cost reduction scenario

Fig 5.8 shows the distribution of the decrease in the total cost of both routes combined found by NNIR when compared to Dijkstra's algorithm. The majority case in the Telecoms data sets have a small reduction of between *0-1%* with two cases having a reduction between *24-25%*. On the Road data set the reduction is spread more across the range of reductions with some being smaller and one being *20-21%*.

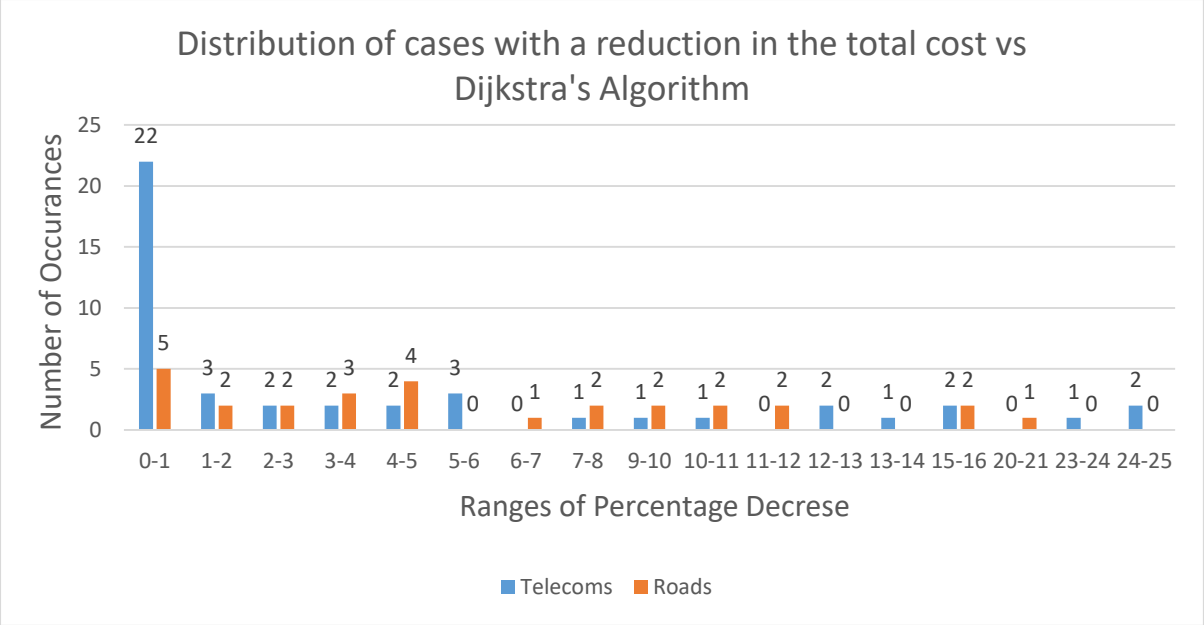


Fig 5.8. Distribution of Cases with a Reduction in Total Cost NNIR VS Dijkstra's Algorithm for the cost reduction scenario

5.2.3 Combined Results

Section 5.2.1 & Section 5.2.2 gave a more granular breakdown of the results whereas this section aims to give a boarder representation of the results. Table 5.3 demonstrates the combined results to that end, using both the blocking and cost reduction results to show the improvement presented by NNIR over Dijkstra's algorithm.

TABLE 5.3 Total of improvement of NNIR over Dijkstra's algorithm on the resilient routing problem when a total improvement is defined as a combination of the blocking scenario and the cost reduction scenario

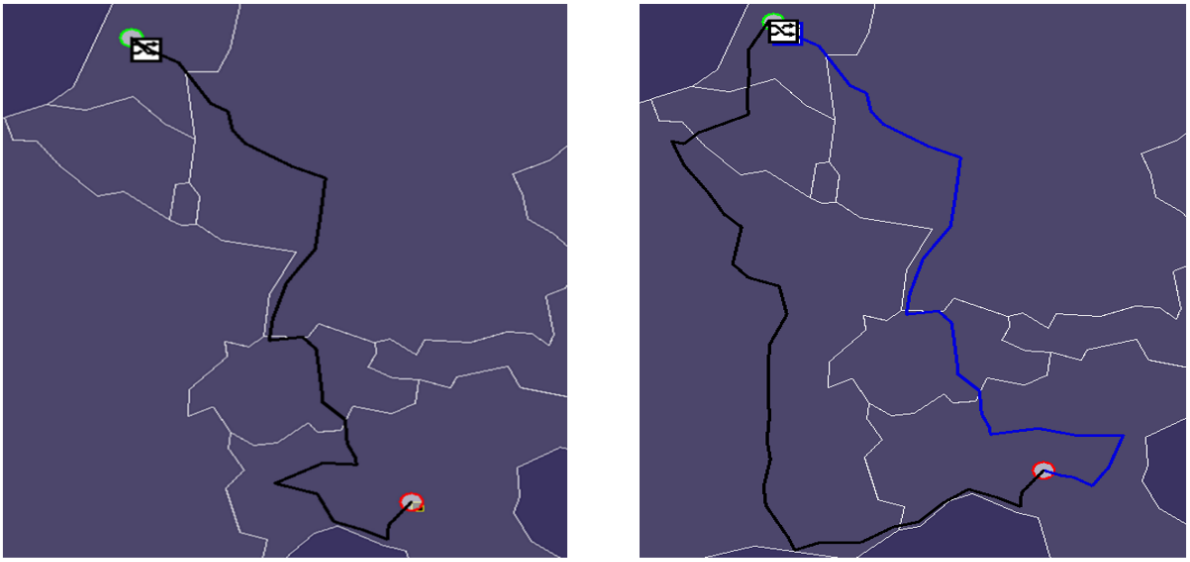
	Telecoms	Road
Total Improvement	38% (95 out of 250)	30.4% (152 out of 500)

By combining both scenarios NNIR outperforms Dijkstra's Algorithm on the Telecoms data set by in a total of 95 out of 250 cases, this is an improvement in 38% of cases. On the Road data set NNIR outperforms Dijkstra's Algorithm in 152 out of 500 cases which is an

improvement of 30.4%. NNIR was designed with the intention of being used on the Telecoms data set which shows in the results, although the Road data set indicates that it is applicable in other domains that can be represented as a graph. Additionally, due to the highly constrained nature of this problem, it seems reasonable to suggest that population base evolutionary algorithms perform much better. The primary purpose of the NNIR algorithm was to reduce costs, but these experiments show that it is also very capable at finding new resilient routes where Dijkstra couldn't.

5.3 Discussion

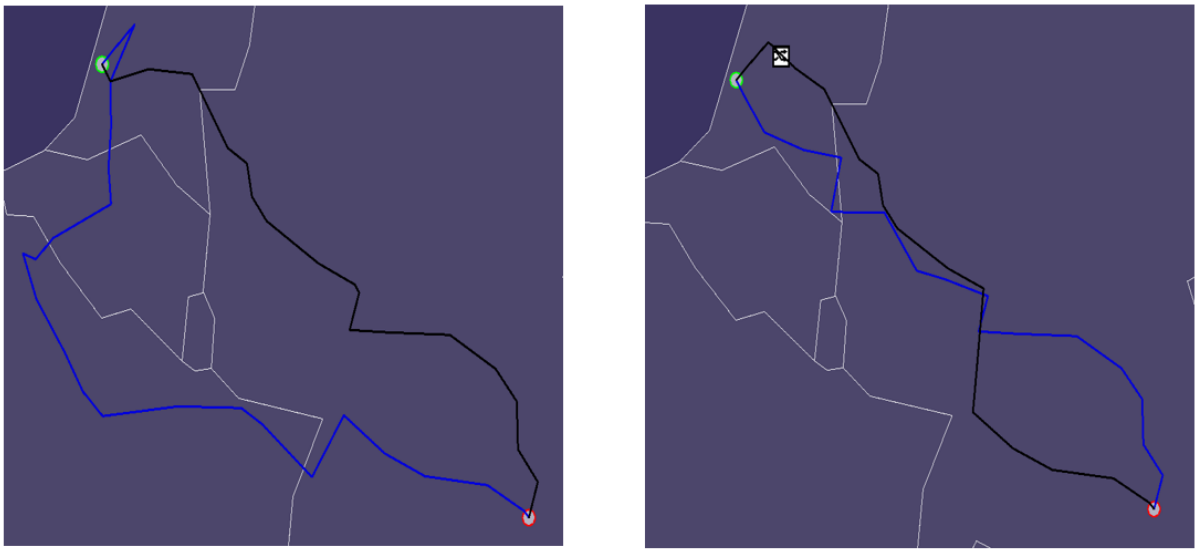
This chapter introduced a solution to the resilient routing problem namely the NNIR algorithm whilst also showing the need for such an algorithm by comparing it to Dijkstra's Algorithm, A-Star and Simulated Annealing. The resilient routing problem is a real-world problem faced by BT and has been implemented into tools currently in use. Fig 5.9 and Fig 5.10 show screen shots from one such tool. This tool is used for pricing for corporate customers that require multiple dedicated lines between two locations, or as it's known within the literature resilient routing. Fig 5.9 and 5.10 show routes in central Europe, the primary route is shown in black and the resilient route is shown in blue.



(a) (b)

Fig 5.9. (a) Shortest path available when using Dijkstra's algorithm (the blocking Scenario)

(b) Shortest path available when using NNIR (the blocking Scenario)



(a) (b)

Fig 5.10. (a) Shortest path available when using Dijkstra's algorithm

(the cost reduction example)

(b) Shortest path available when using NNIR (the cost reduction example)

In the results sections of this chapter the distance reductions are rather small percentages, but when considering that some connections in the Telecoms data sets are routes across continents small percentages added up to reasonable sums of money. With the NNIR algorithm integrated into the pricing tool BT have been able to offer more competitive pricing outperforming competitors for contracts, thus proving the commercial value of the NNIR algorithm to BT.

This chapter showed that by increasing the cost of a primary route a more optimal resilient route can be found, making the entire solution more optimal in *38%* of cases of the Telecoms data set and *30.4%* of case in the Road data set. Amazingly the Telecoms data set with an increase in the primary route of up to *81%* can still result in a more optimal solution, with the total cost of both routes lower than the solution presented by Dijkstra's algorithm. Not every possible route has been evaluated in these data sets but a large sample has been taken from both. The Road data set was selected for two reasons first to show that the resilient routing problem and by extension NNIR can be applied to differing domains. Secondly to ensure there was transparency in the experiments as the Road data is open source whereas the Telecoms data is confidential.

Everything outlined in this chapter has be published in the journal paper NNIR: N-Non-Intersecting-Routing Algorithm for Multi-Path Resilient Routing in Telecommunications Applications [33]. The next chapter will discuss how NNIR deals with uncertain environments through the use of a Fuzzy Logic System.

Chapter 6. A Fuzzy Genetic System for Resilient Routing in Uncertain Dynamic Telecommunication Networks

Large telecoms companies have a global presence with thousands of locations in a single nation but sometimes even this is not enough for all of their needs so they will use infrastructure from their competitors. In Section 2.4.3 the idea of resilient routing was introduced, then in Chapter 5 the N-Non-Intersecting-Routing-Algorithm (NNIR) was introduced and shown to be an effective solution to the resilient routing problem. In Section 2.5 the idea of uncertain environments was introduced. In resilient routing, an uncertain environment can arise from three scenarios. First the data internally to the telecoms company about routing costs can be inconsistent as the information is sourced from multiple sources. These routing costs can include; distances, latency or service volume. Secondly, if a competitor's infrastructure is used then there are no assurances of the accuracy of the data provided, this is not through malice but just an unfortunate fact about telecoms networks. Finally, sometimes routing is performed on data that is dynamic or in an ever-changing state such as latency or network bandwidth so it can be difficult to determine a shortest path through it.

In this chapter the work was originally presented across two conference papers in FUZZ-IEEE 2019 [93] and WCCI 2020 [94]. This chapter presents a set of modifications to the NNIR algorithm in order to allow it to function effectively in uncertain environments. These changes take in the form of Fuzzy Logic controller. In Section 6.1, the type-1 extensions are presented. In Section 6.2 the type-2 extensions are presented. In Section 6.3 the experiments and results from both extensions are presented. Finally in Section 6.4, there is a discussion about the work from the two conference papers and what has been presented in the preceding sections of this chapter.

6.1 NNIR in uncertain environments with type-1 fuzzy logic

NNIR is used the same as described in Chapter 5 with some modification to the evaluation function. The evaluation function is critical to the operation of the GA in NNIR ranking the population for selection and population reduction. The modifications to the evaluation function allow for uncertainties in the data to be interpreted by a Type-1 Fuzzy Logic system. The evaluation function uses the same principle of Route Discount known as *RD* shown in equation 6.1, where N is the desired number of routes and *SR* is the number of routes in a solution and FS is the Fuzzy Score produced by the Fuzzy Logic system. RD works on the same principle as in Chapter 5 but the equation is different to ensure that it still has the desired effect on the total cost.

$$RD = FS^{N-SR} \tag{6.1}$$

The total cost on the other hand has been changed to incorporate the crisp value from the Fuzzy Logic system *FS,* and a new conflict resolution metric *NLC.* NLC is a count of the number of edges in the solution, but to ensure it has a minimal effect on the overall fitness function and is only used when it is needed it has been scaled by *0.25*. Equation 6.2 shows this new evaluation equation.

$$Total\ cost\ = \frac{1}{RD * FS * (0.25 * NLC)} \tag{6.2}$$

The Fuzzy Score or *FS* is a cumulative cost calculated on every edge in a route prior to being used in the evaluation equation. The Type-1 Fuzzy Logic controller has two inputs and one output. The inputs are named “Local Input” and “Global Input”, named this due to where the information comes from for the generation of the input sets. The local input is dynamically generated for every vertex. Fig 6.1 shows the membership functions for local input, with its

three inputs “LOW”, “MEDIUM” and “HIGH”. The local input membership function has three points *P1, P2* and *P3* which are extracted from the edges of a given vertex. *P1* is the minimum cost from the given vertex, *P2* is the mean cost from a given vertex and finally *P3* is the maximum cost from a given vertex. To increase execution speed each of these input member functions can be pre-calculated prior to execution.

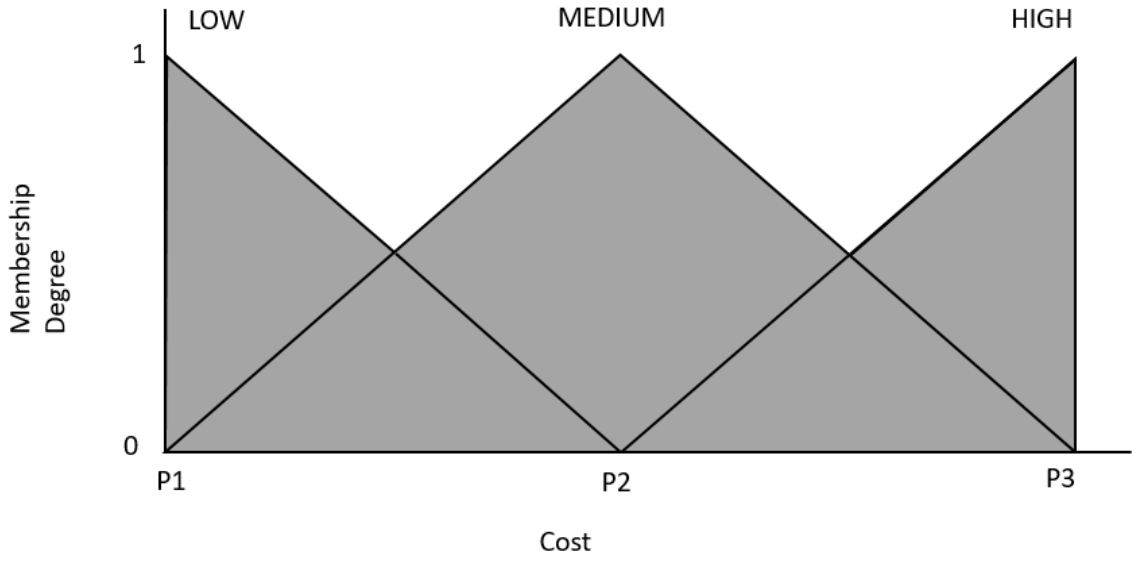


Fig 6.1 Local input of a Type-1 input Membership function, with three inputs LOW, MEDIUM, HIGH

There is only one global input membership function for the entire network and it is generated from the edge cost values from across the network. Fig 6.2 shows the global input membership function, which has five inputs named “VLOW”, “LOW”, “MEDIUM”, “HIGH” and “VHIGH”. The global input membership function has five points taken from the graph *P1, P2, P3, P4* and *P5*. These points are mapped to the points of a box plot [95], with P1 as the minimum, P2 as the lower quartile, P3 as the median, P4 as the upper quartile and P5 as the maximum value. The output membership function named output uses the same values as the global input member ship function with five outputs “VLOW”, “LOW”, “MEDIUM”, “HIGH”

and “VHIGH”. The membership functions for the output set uses the same values of P1, P2, P3, P4 and P5 so all information is extracted from the data.

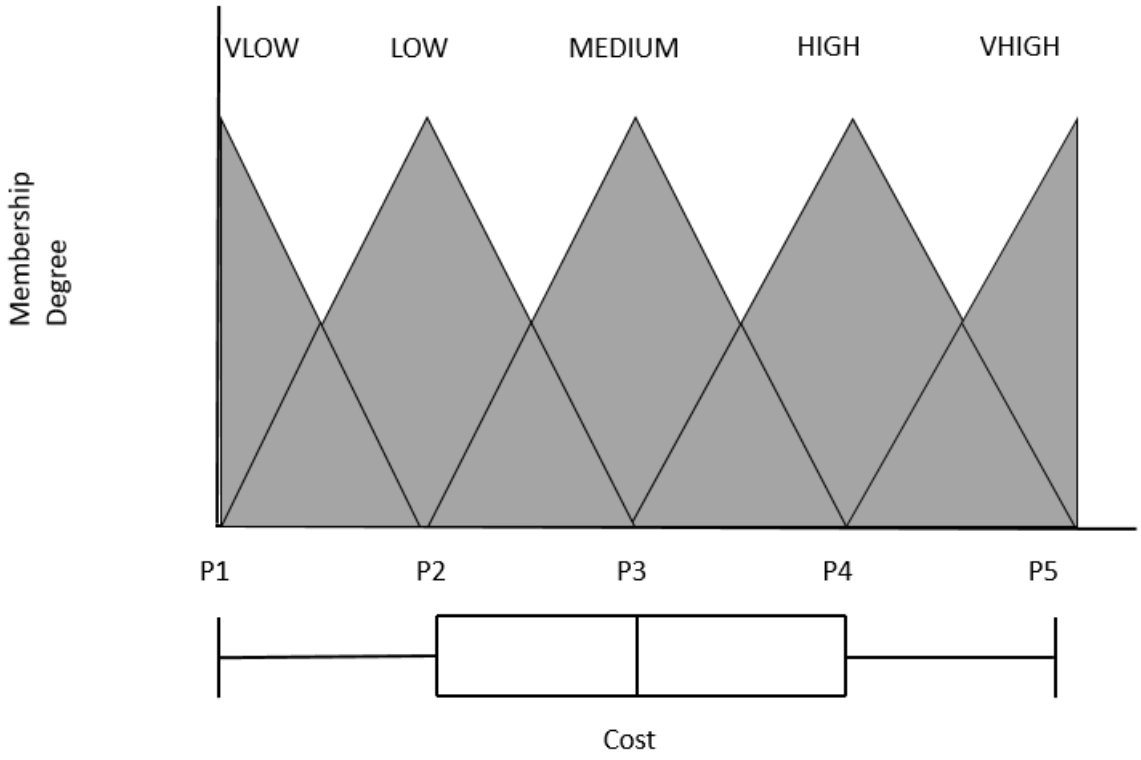


Fig 6.2 Global input of a Type-1 input membership function, with five inputs VLOW, LOW, MEDIUM, HIGH, VHIGH and a boxplot showing how the membership functions are generated.

The local input membership function works as a comparison between the edges in the route to the other edges available at the same vertex, whereas the global input membership function works as comparison of the given edge cost to the cost of all the vertices in the graph. The output membership functions are based on the same information as the global input membership functions, using the cost information from the edges in the network to ensure that all the outputs are correctly scaled to one and other. In order to connect the input to the outputs a rule base is required for inference, Table 6.1 shows such a rule base.

TABLE 6.1 The rules used in the Type-1 fuzzy logic system for evaluation within NNIR

Global input	*Local input*	*Output*
VLOW	LOW	VLOW
VLOW	MEDIUM	VLOW
VLOW	HIGH	LOW
LOW	LOW	VLOW
LOW	MEDIUM	LOW
LOW	HIGH	LOW
MEDIUM	LOW	LOW
MEDIUM	MEDIUM	MEDIUM
MEDIUM	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	MEDIUM	HIGH
HIGH	HIGH	VHIGH
VHIGH	LOW	HIGH
VHIGH	MEDIUM	VHIGH
VHIGH	HIGH	VHIGH

The Fuzzy Logic uses the minimum function for inference and the centre of set defuzzification method. With the input sets, the rules base, the output set and the established the entire Fuzzy Logic system can now be used within NNIR. The experiment undertaken with this system will be described in Section 6.3 and the results in Section 6.4.

6.2 NNIR in uncertain environments with interval type-2 fuzzy logic

NNIR is used as described in Chapter 5 with additions to its evaluation function similar to that described in Section 6.1 but now with the use of an interval Type-2 Fuzzy Logic controller (IT2FLC). An interval Type-2 system can be seen as an extension to the Type-1 version presented in Section 6.1. In general the interval Type-2 fuzzy logic is seen to be an improvement over Type-1 Fuzzy Logic at handling uncertainty at the cost of computational complexity [44], [96].

The evaluation function equations are the same as presented in Section 6.1 with a Type-2 Fuzzy Logic controller used to calculate the FS values. (Equations 6.3 and 6.4 as shown below for reader convenience, albeit they are the same as equations 6.1 and 6.2). *RD* represents the route discount, used to ensure there the desired number of routes in a solution. *N* represents the desired number of routes and *SR* represents the number of routes in the given solution. NLC represents the number of edges in the solution and are scaled by 0.25 to reduce its effect on the total cost, NLC exists as tie breaker in case two solutions have the FS and RD.

$$RD = FS^{N-SR} \tag{6.3}$$

$$Total\ cost\ = \frac{1}{RD * FS * (0.25 * NLC)} \tag{6.4}$$

The interval Type-2 Fuzzy Logic system much like its Type-1 predecessor has two inputs, the local input and the global input. The local input is calculated from the edges cost values at each vertex, whereas the global input is based upon the costs of every edge in the graph. Fig 6.3 shows the local input membership functions and Fig 6.4 show the global input membership function each with their footprints of uncertainty. As the points *P1, P2* and *P3* are all taken from the edge costs of the evaluated vertex corresponding to the minimum, mean and maximum values, as are the footprints of uncertainty. A new value *D* is calculated for each point, which is obtained by taking the 10% of the distance between two points. For example, if *P1* is 10 and *P2* is 20 then the corresponding *D* value is 1. The *D* value is then used to calculate the values that describe the footprint of uncertainty with a membership function. With the left and right points of the footprint of uncertainty equal to the *P* value plus or minus *D*. Once again in the global input membership function the values of *P1, P2, P3, P4* and *P5* are calculated from a box plot as described in Section 6.1.

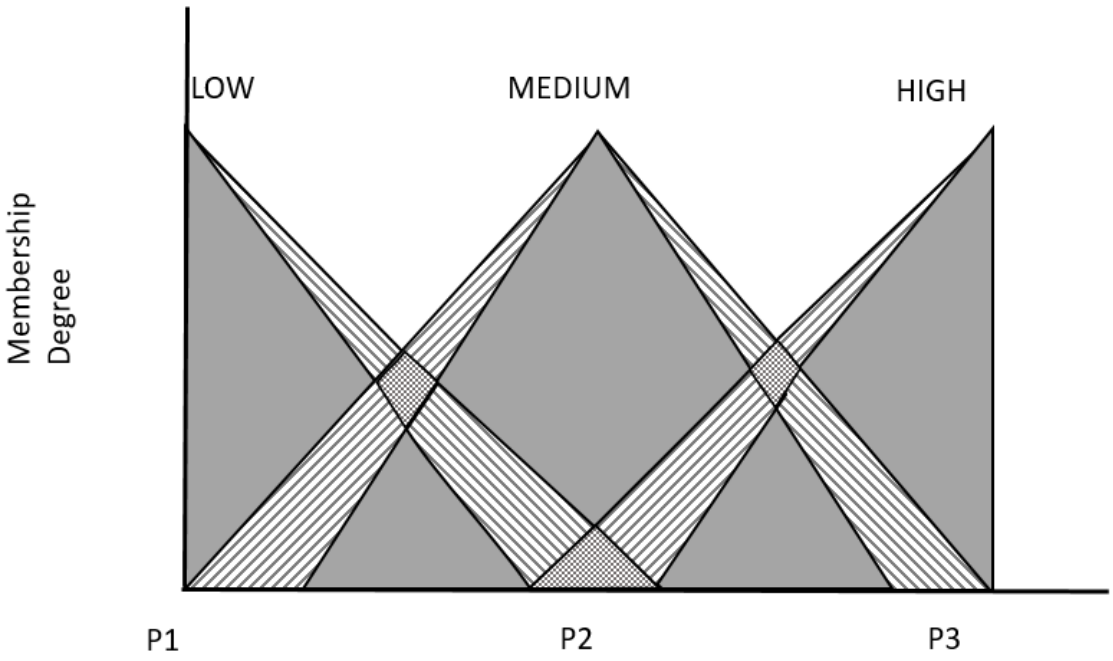


Fig 6.3 Local input of a Type-2 input Membership function with its footprints of uncertainty, and with three inputs LOW, MEDIUM, HIGH.

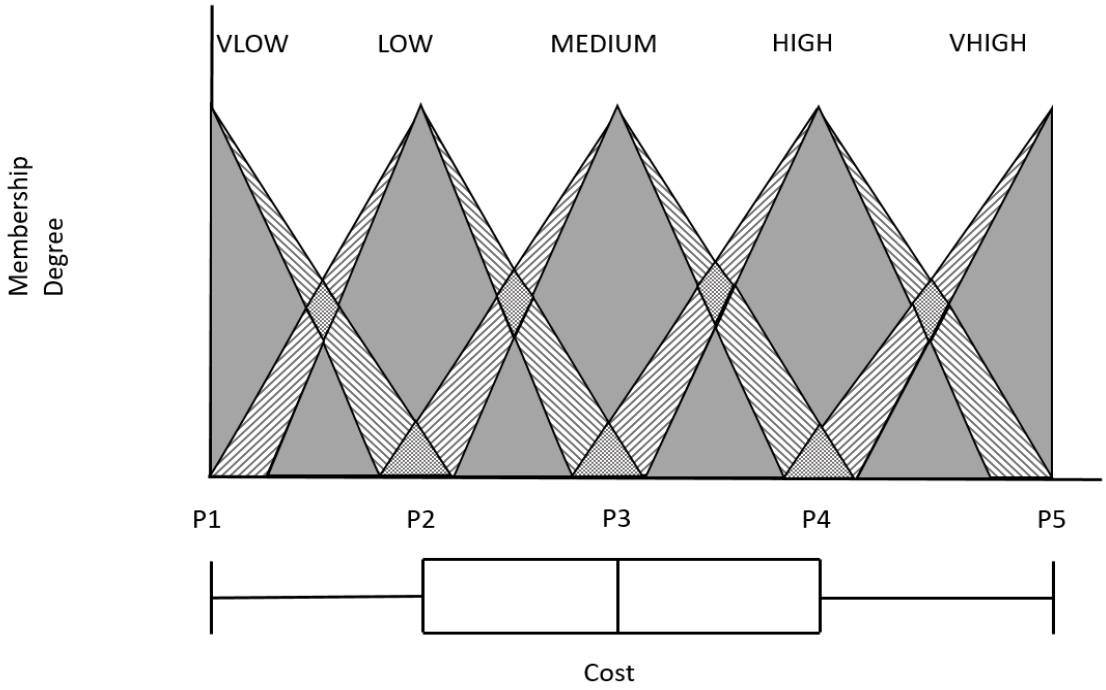


Fig 6.4 Global input of a Type-2 input membership function with its footprints of uncertainty, and with five inputs VLOW, LOW, MEDIUM, HIGH, VHIGH and a boxplot showing how the membership functions are generated.

The output membership functions used have the same values as the global membership function shown in Fig 6.4 with the footprint of uncertainty calculated in the same methodology. A Fuzzy Logic controller is incomplete without a rule base to tie the inputs to the outputs, Table 6.2 shows the rule base for IT2FLC. The minimum function has been used within rule inference and the centre of sets type reduction was used to enable defuzzification.

TABLE 6.2 The rule base used for the interval Type-2 fuzzy logic controller for the evaluation within NNIR

Global input	*Local input*	*Output*
VLOW	LOW	VLOW
VLOW	MEDIUM	VLOW
VLOW	HIGH	LOW
LOW	LOW	VLOW
LOW	MEDIUM	LOW
LOW	HIGH	LOW
MEDIUM	LOW	LOW
MEDIUM	MEDIUM	MEDIUM
MEDIUM	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	MEDIUM	HIGH
HIGH	HIGH	VHIGH
VHIGH	LOW	HIGH
VHIGH	MEDIUM	VHIGH
VHIGH	HIGH	VHIGH

6.3 Experiments & Results

The same experiment has been undertaken for both the Type-1 and Type-2 Fuzzy Logic systems allowing for a comparison of their performance. The goal of the system is to be able to perform resilient routing in uncertain environments characterised within a real-world telecommunications network. The experiments have been set out in such a way to look for

consistency in routes, given differing edge values within the graph. To this end 10 versions of the same graph exist with different values for each of the edges. Each data set is corresponding to one graph, but they are all within a 10% variation of a source graph, from which the other graphs have been generated. The graph its self represents British Telecoms' European network, it is vast and complex allowing for many different routes between geographic locations across the continent.

Each graph is run against three versions of the NNIR algorithm. First the original crisp version of NNIR without a Fuzzy Logic evaluation system. Second the Type-1 version of NNIR with the Fuzzy Logic evaluation system as described in Section 6.1. Finally the interval Type-2 versions of NNIR with the Fuzzy Logic evaluation system as described in Section 6.2. Ten scenarios are selected randomly, with a scenario described as trying to establish a primary and resilient route between two locations. Each scenario is run with each dataset on each system, therefore each scenario is run 30 times.

There are four metrics used to determine the quality of the results from each system. First metric is the number of *Unique Routes* (how many times does a route occur once). Second metric is the number of *Routes of Multiple Occurrence* (how many routes occurred more than once). Third metric is *Max Same Route* (a count of how many times the most common route occurred). The final metric is the *Number of Independent Routes* (a count of how many routes there are that have occurred once or more). The best performing system will minimise the value of *Unique Routes* and *Number of Independent Routes* whilst maximising the *Max Same Route*. The most important metric is *Unique Routes* because if this value is zero is means no route has only occurred once.

TABLE 6.3 Average routing results across the 10 routing scenario

System	*Average Unique Routes*	*Average Routes of Multiple Occurrence*	*Average Max Same Route*	*Average Number of Independent Routes*
Crisp	3.6	2	3.6	5.6
Type-1	3.4	1.8	4.4	5.4
Type-2	2.3	1.6	6.2	4

Table 6.3 shows the average results over the 10 routes scenarios for the three versions of the system. The crisp version has the highest average number of *Unique Routes* at 3.6. The Type-1 version of the system has a slight reduction in the number of *Unique Routes* at 3.4 whereas the Type-2 system has a much larger reduction at 2.3. The crisp system has the highest average number of *Routes of Multiple Occurrence* with a reduction seen by the Type-1 system to 1.8 and a further reduction by the Type-2 system to 1.6. When it comes to the *Max Same Route* metric the crisp system has the worst result of 3.6, with a progressive improvement from Type-1 to Type-2 with 4.4 and 6.2 respectively. Finally the *Number of Independent Routes* shows a decrease from crisp at 5.6 to Type-1 as 5.4 to Type-2 at 4. Not one of these metrics can show in full the performance of the system. The crisp system has relatively low average number of *Max Same Routes* with a slight improvement being shown by the Type-1 system, and a much larger improvement by the Type-2 system. This metric indicates that the Type-1 system will find the same route for a given problem in 44% of cases, whilst the Type-2 system will find the same route in 62% of cases the. The crisp system on the other hand will only find the same route in 36% of cases, showing that by this metric the Type-1 is an improvement of 8% over the crisp system whilst the Type-2 system has an improvement of 26% over the crisp system and an 18% improvement over the Type-1 system.

Table 6.4 shows the best and worst performing routing scenarios for all three of the systems. The best scenario for the crisp system presents 7 identical routes and then 3 identical routes, with no unique routes. The worst case crisp logic has the worst possible outcome with 10 different routes as they are all unique. The best case for Type-1 is only slightly better than that provided by the crisp system, with 8 scenarios having the same route and the other 2 sharing a route. The best case for Type-1 system is a definite improvement over the crisp system, with 8 routes occurring once and 1 route occurring twice. Finally, the best case for the Type-2 systems is the best possible outcome with all 10 scenarios using the same route. The worst case for the Type-2 system is better than its crisp and Type-1 counter parts with 6 unique routes and 2 lots of 2 routes being the same.

TABLE 6.4 The best and worst performing routing scenarios for each of the three versions of NNIR

System	***Unique Routes***	***Routes of Multiple Occurrence***	***Max Same Route***	***Number of Independent Routes***
Best Crisp	0	2	7	2
Worst Crisp	10	0	1	10
Best Type-1	0	2	8	2
Worst Type-1	8	1	2	8
Best Type-2	0	1	10	1
Worst Type-2	6	2	2	8

By looking at all of the metrics it is shown that there is a progressive improvement in the system from crisp to Type-1 and again to Type-2. This can be seen with the reduction in the number

of *Unique Routes*, *Routes of Multiple Occurrence* and the *Number of Independent Routes* whilst the number of times the *Max Same Route* occurred increases.

6.4 Discussion

In this chapter the idea of resilient routing in uncertain environments along with two potential solutions to this problem were introduced. These solutions took the form of the NNIR algorithm with a Type-1 or Type-2 Fuzzy Logic controller built into the evaluation operation. This is a real-world problem faced by British Telecom, and the interval Type-2 version of the system is now in use. This problem presented an interesting predicament as the optimal solution in the experiments was not always the shortest path, as it is known from work in Chapter 5 that the crisp system will always find the shortest path. Unfortunately when the data can't be trusted creating an uncertain environment it is more important to find a route that is short and consistent if the costs of the edges in the graph change. This is due the fact that changing one of these routes can be costly and time consuming.

The work undertaken as shown in the chapter resulted in two conference papers:

- A Fuzzy Genetic System for Resilient Routing in Uncertain Dynamic Telecommunication Networks [93]
- A Type-2 Fuzzy Genetic Approach to Uncertain & Dynamic Resilient Routing within Telecommunications Networks [94]

These papers and the contents of this chapter show that there is an improvement in resilient routing in uncertain environments through the use of Fuzzy Logic as part of the evaluation process. It is also shown that there is a progressive improvement from Type-1 to Type-2 Fuzzy Logic.

Once uncertainties have been dealt with that brings a conclusion to the NNIR algorithm for now but it has an application as part of the wider problem solution. The next chapter will

introduce the Heated Stack Algorithm (HS) and its initial applications digital capacity management.

Chapter 7. A Type-2 Fuzzy Multi-Objective Multi-Chromosomal Optimisation for Capacity Planning within Telecommunication Networks

Single objective optimisation is a well-developed field with many different algorithms providing optimal solutions across many domains. The natural extension to single objective optimisation is multi-objective optimisation. A popular and successful optimisation algorithm is Non-dominant Sorting Genetic Algorithm two (NSGA-II). The operation of NSGA-II is described in Section 4.2.3 and is known to produce high quality solutions in many scenarios [71], [79], [86].

In this chapter the Heated Stack algorithm (HS) a multi-objective evolutionary algorithm with a Type-2 Fuzzy Logic controller will be introduced. The set of experiments presented in this chapter are based on the digital capacity planning with its SVLans and CVLans as introduced in Section 2.5.1. The content of this chapter was first presented at FUZZ-IEEE 2021 with a conference paper [38].

7.1 The Heated Stack Algorithm

The Heated Stack algorithm is an evolutionary algorithm with a temperature system, taking inspiration from the popular and successful optimisation algorithms NSGA-II [77] and Simulated Annealing [56]. HS is a population based evolutionary algorithm employing a Fuzzy Logic controlled temperature system used for sorting and crossover manipulation. It uses the ideas of population, generations, selection, crossover and mutation from a Genetic Algorithm (GA). Fig 7.1 shows a flow chart of the algorithmic process, notice its similarity to GA but with the addition of the temperature control mechanism.

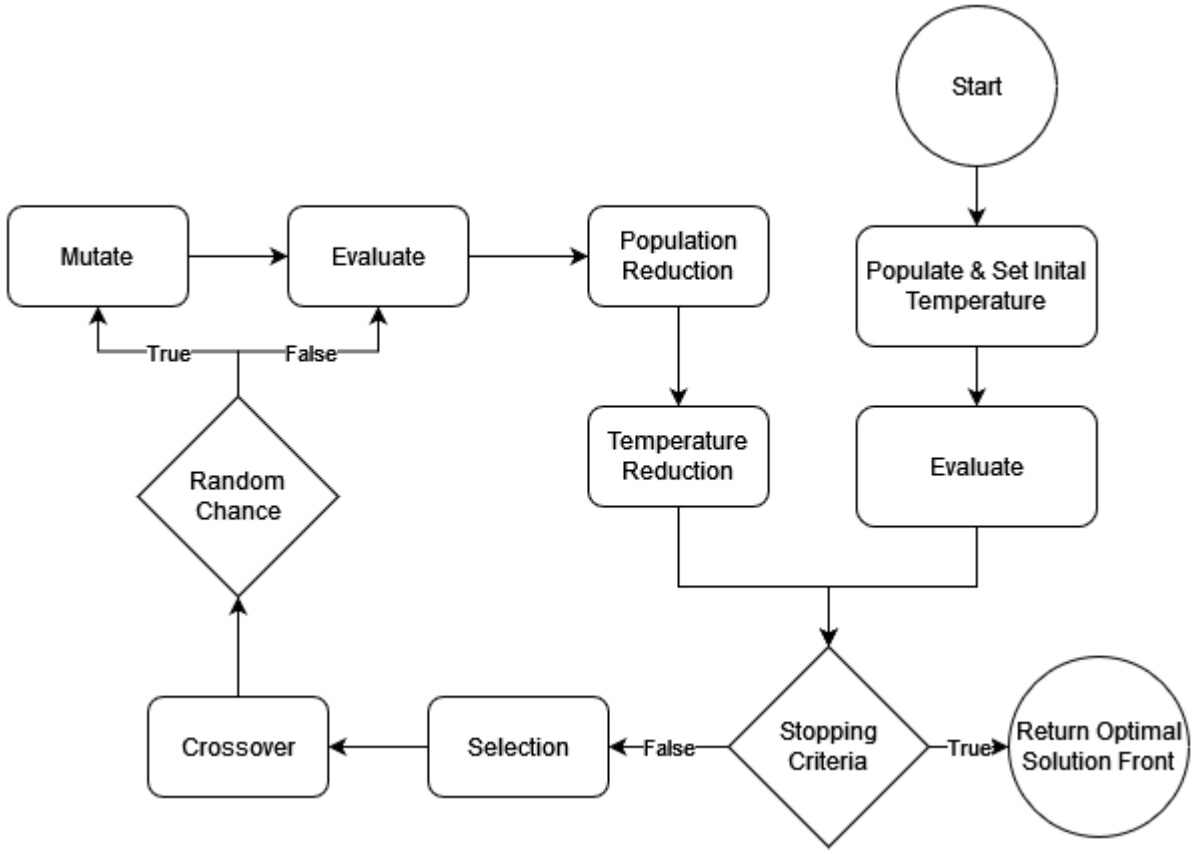


Fig 7.1. An overview of the Heated Stack Algorithm with its temperature initialisation and reduction

As demonstrated by Fig 7.1 HS begins with the creation of the initial population, it then evaluates the initial population for the first generation. The system then cycles through a predetermined set of generations whilst in each generation using selection to sort the population for crossover. Every new member of the population created by crossover has a chance to mutate, the new members of the population are then evaluated for population reduction and the temperature of every remaining member of the population is then reduced. The stopping criteria of the number of generations passed is checked and either it stops returning the optimal solution front or the next generation beings.

7.1.1 Solution Representation and Population Initialisation

In the HS a solution is represented in a three layered structure namely “Stack”, “Chromosome” and “Gene” as shown in Fig 7.2.

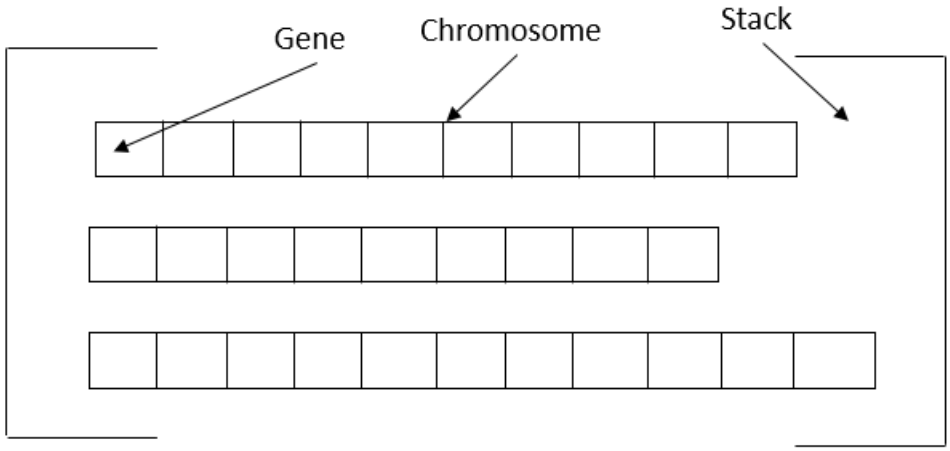


Fig 7.2. The solution representation with its three layers of genes chromosomes and the stack

As seen in Fig 7.2 a stack can have many chromosomes housed within it and each chromosome can have a number of genes, with each stack representing one potential solution to a problem. The number of chromosomes in a stack and the number of genes in a chromosome are variable and problem dependant. Each chromosome has its own temperature value that is initialised to a specific problem dependant value. Initial temperature is a tuneable value but a value of around the value of 1000 tends to be a good starting point. Due to the temperature being based upon each chromosome, the temperature of a stack is the sum of its chromosomal temperatures. The genes of a chromosome can be encoded in much the same as a GA, with real values, binary, or some more abstract representations such vertices in a graph or CVLans in a virtual capacity management problem.

The initial population is generated much the same way as in a GA with the problem and gene’s encoding heavily influencing the initialisation process. The capability of having a multi-chromosomal representation allows for easier representation of problems that have a low

degree of separation in the objective space, but a high degree of separation in its decision variables. Multi chromosomal representation is at its most useful when a problem has a set of decision variables that rarely interact but their objectives are deeply dependant on all of the decision variables. Some examples of this include: digital capacity planning, physical capacity planning and resilient routing. During population initialisation each chromosome has a temperature variable set to the initial temperature value which will be manipulated every generation.

7.1.2 Evaluation and Population Sorting for Selection & Population Reduction

Every solution has its own assigned set of scores according to the optimisation objective, these scores much like in a GA are used to determine the quality of each potential solution. These scores are used to sort the population into fronts based upon the domination rules introduced in Section 4.2.1. Once on a front the crowding distance of every solution is determined exactly the same as if it were NSGA-II using the Euclidian distance between solutions on the objective as described in Section 4.2.3. These fronts are used as the first stage of the sorting process used for selection and population reduction.

Selection with HS uses the principle of survival of the fittest, where the best members of the population are split randomly into two groups and ranked, with the best ranked being picked for crossover. After the population has been sorted into fronts, each front needs to be sorted into a priority order, which is used for both the selection ranking and population reduction ranking. The ranking within a front is sorted by an Interval Type-2 Fuzzy Logic Controller (IT2FLC) using temperature and crowding distance as the inputs. Fig 7.3 shows an example of this ranking for selection visualised. In this example, the population has been randomly split into two sets, with each set sorted into its fronts and each front sorted based upon the sorting fuzzy logic system.

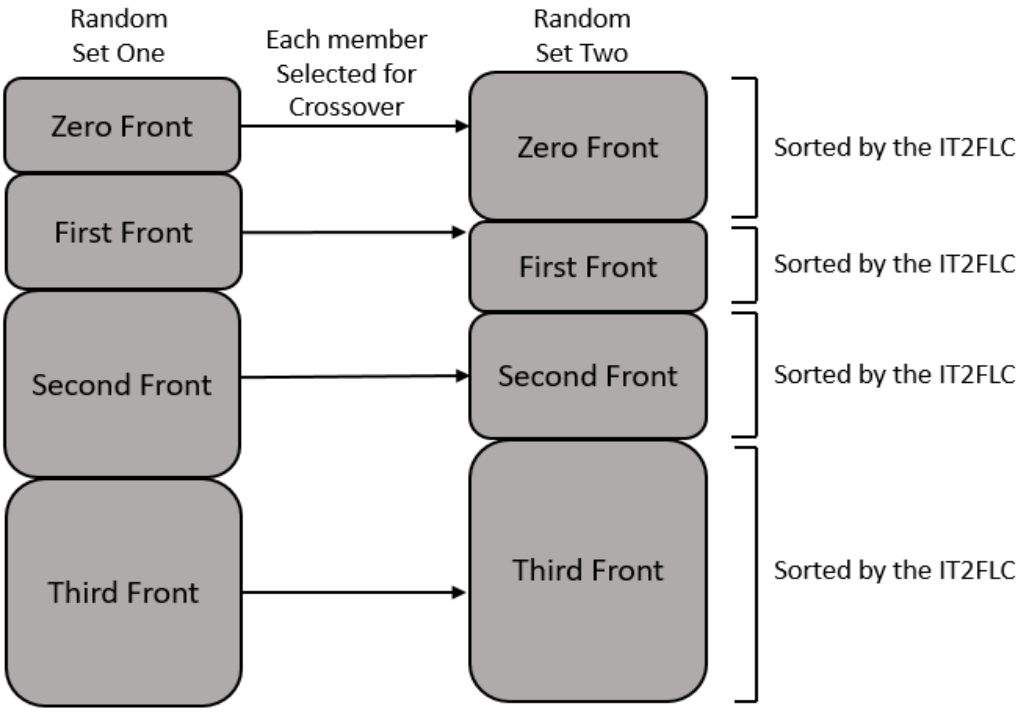


Fig 7.3. Visualisation of sorting as part of selection

The IT2FLC for sorting is used for both selection and population reduction, with its inputs being temperature and crowding distance. Temperature has three input membership functions labelled: “LOW”, “MEDIUM” and “HIGH”. Crowding distance has five input membership functions labelled: “VLOW”, “LOW”, “MEDIUM”, “HIGH” and “VHIGH”. Being that an IT2FLC has been used, each of the inputs has a footprint of uncertainty (FoU). Fig 7.4 shows the input temperature membership functions, the values of temperature are scaled between 0-1 therefore the inputs to the FLC are normalised. This normalisation is shown in equation 7.1 with a minimum value 0 and the maximum set to initial temperature value of a stack and with *x* being the specific temperature of the stack being sorted. Fig 7.5 shows the crowding distance input membership functions. Once again the input membership function values are scaled between 0-1, so the input to the membership function must also be normalised between 0-1. In order to normalise the crowding distance equation 7.1 is used with the minimum value set to

0, the maximum value set to the largest crowding distance in the current population and *x* set to the crowding distance of a given stack.

$$Normalise\ (x)\ =\ \frac{Maximum - x}{Maximum - Minimum} \tag{7.1}$$

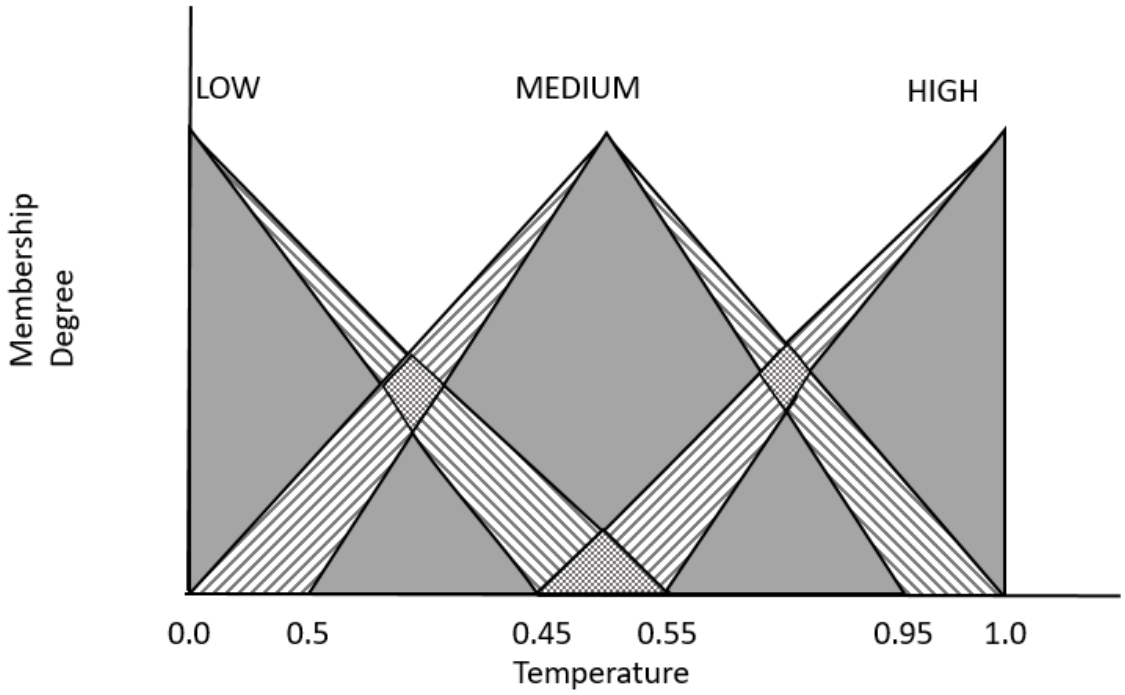


Fig 7.4 Temperature input membership function for sorting with its footprint of uncertainty

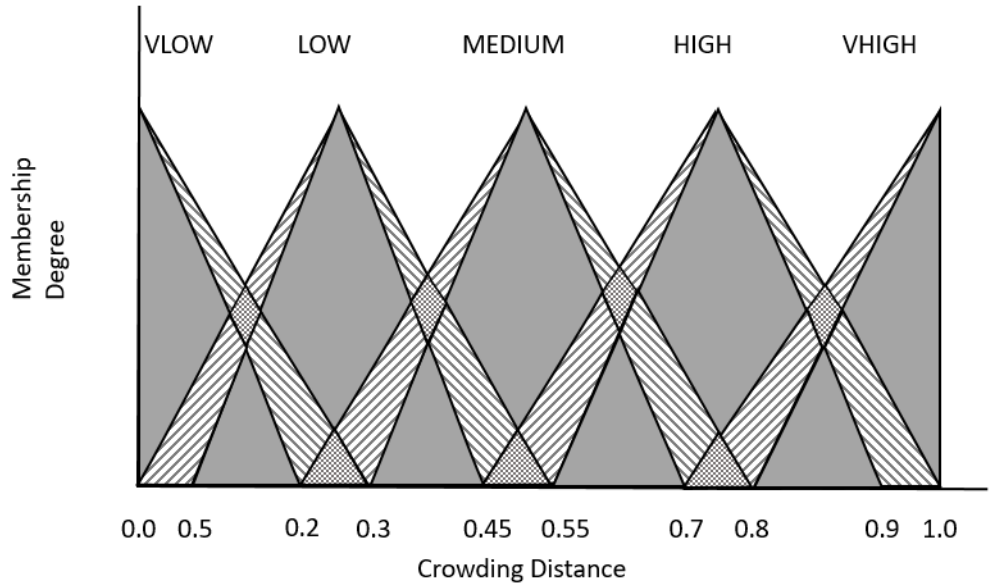


Fig 7.5 Crowding distance input membership function for sorting with its footprint of uncertainty.

TABLE 7.1. The rules for the sorting IT2FLC

Temperature	Crowding Distance	Output
Low	VLow	High
Low	Low	Med
Low	Medium	Low
Low	High	VLow
Low	VHigh	VLow
Medium	VLow	VHigh
Medium	Low	High
Medium	Medium	Medium
Medium	High	Low
Medium	VHigh	VLow
High	VLow	VHigh
High	Low	High
High	Medium	High
High	High	Med
High	VHigh	Low

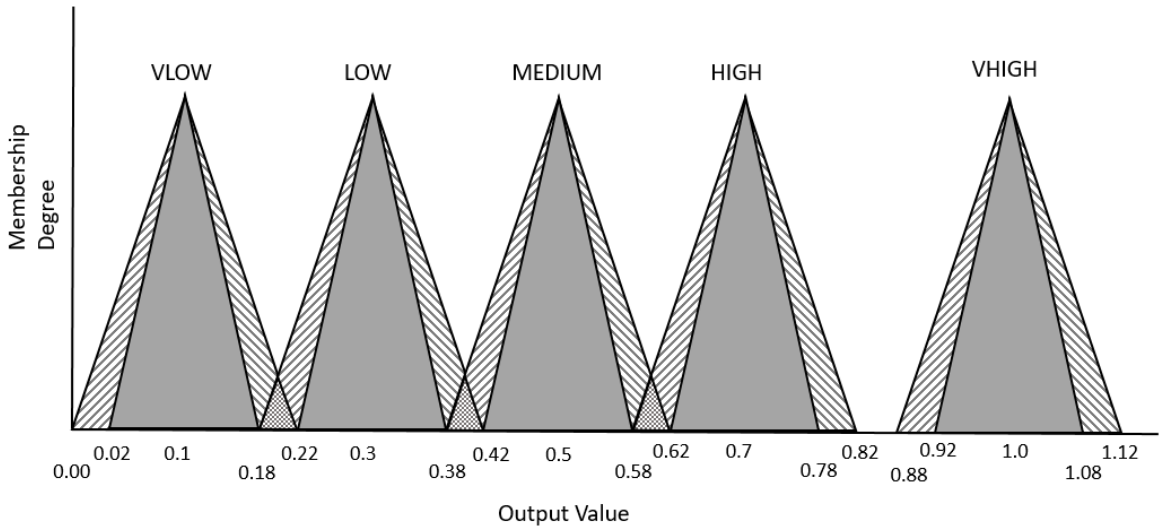


Fig 7.6 Output membership functions for the sorting IT2FLC

The IT2FLC uses the rule base shown in Table 7.1 with two inputs and one output using the minimum inference method and the centre of sets type reduction method. The output set used is shown in Fig 7.6, once every member of the population has a value from the FLC they are sorted into ascending order within their fronts. This means that a lower output from the FLC indicates a more preferable solution for selection.

This process of sorting into fronts and then sorting fronts with the IT2FLC is used for both selection and population reduction. In regards to sorting for population reduction, only the final front that has population members needs to be sorted as the prior fronts will already be in the new population in their entirety. Fig 7.7 shows a visualisation of the portion of the front that is sorted within the sorting FLC for population reduction.

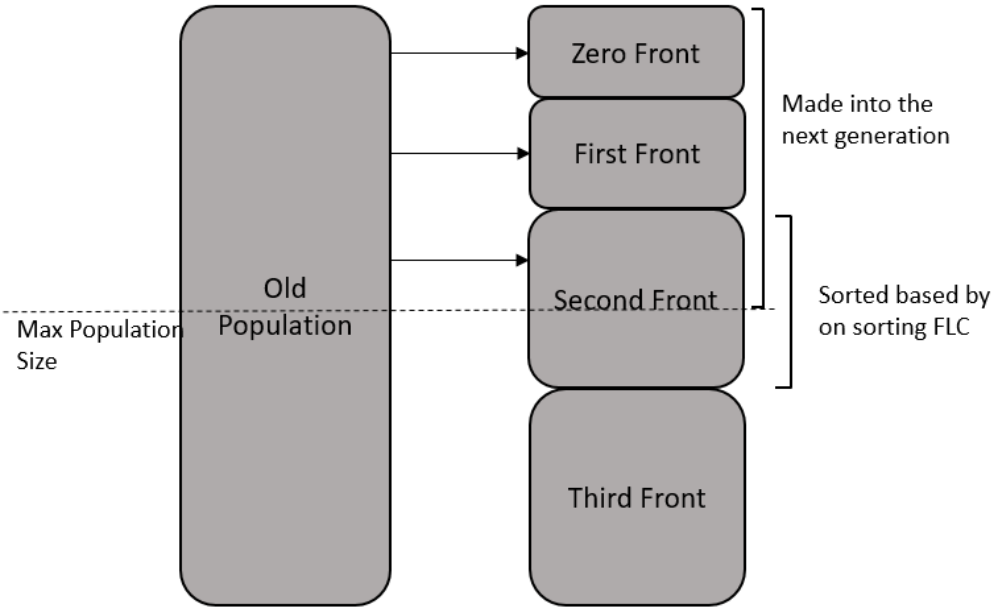


Fig 7.7. Visualisation of the parts of the population that need to be sorted for population reduction

7.1.3 Offspring Creation

In HS offspring are created through the use of the crossover and mutation operators. Members of the population are selected for crossover with the selection operator taking into account their temperature and crowding distance. Crossover takes place on the chromosomal level, in the digital capacity planning problem occurring with a corresponding chromosome from the other parent as seen in Fig 7.8. It is a problem dependant decision to decide which chromosomes to crossover, much like how the crossover strategy in a GA is a problem dependent decision.

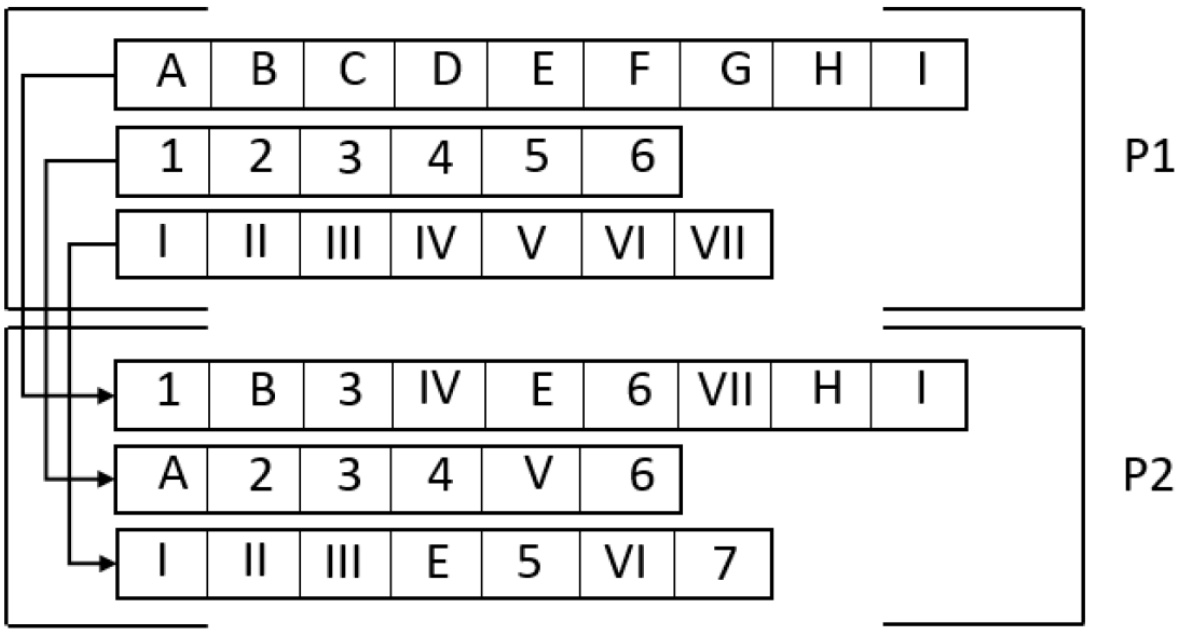


Fig 7.8 Example of two parents (P1, P2) demonstrating which chromosomes crossover

Crossover in the HS is affected by the temperature of the two parent solutions, with a temperature comparison determining how much information to take from each parent solution. This decision is determined through the use an IT2FLC with two inputs and one output. The inputs are the temperature of the parent solutions, with the output determining crossover quantity. Both inputs use the same Type-2 input membership functions. As seen in Fig 7.9, these inputs have three labels: “LOW”, “MEDIUM” and “HIGH”. The inputs to this fuzzy system are between 0-1 therefore temperature of each chromosome must be normalised between 0-1. The rule base for the IT2FLC is show in Table 7.2, it uses centre of set type reduction and the minimum inference method and the output sets are shown in Fig 7.10.

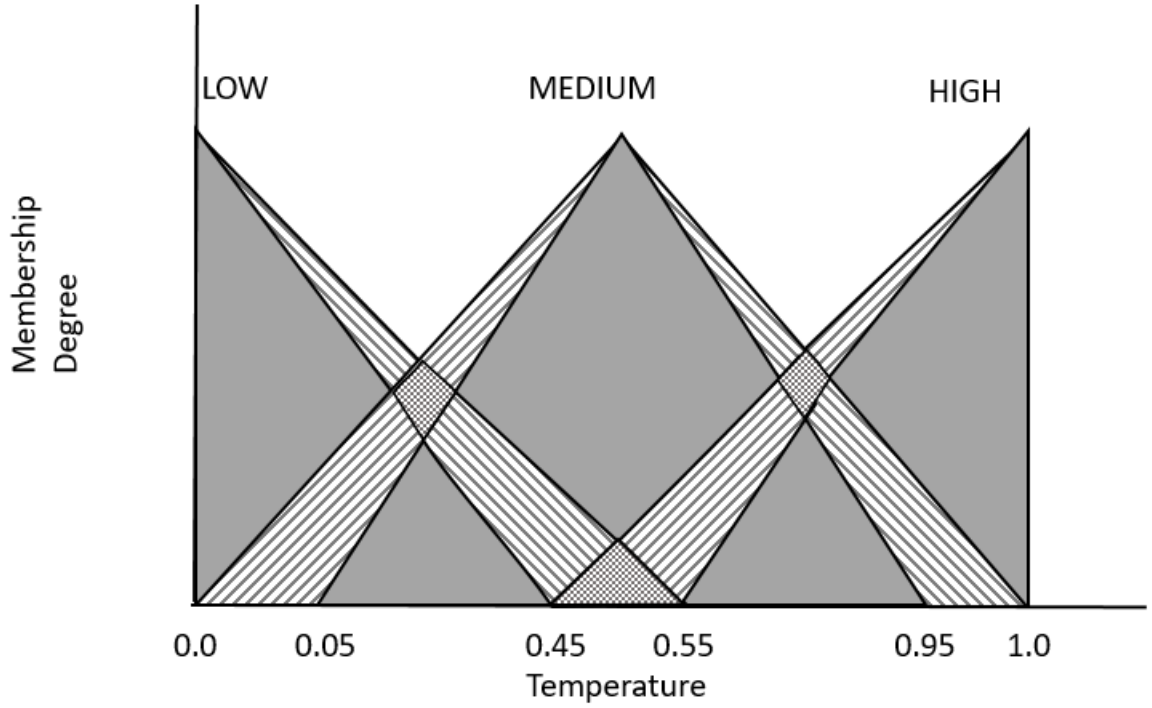


Fig 7.9 Input membership function for temperature as part of the crossover quantity FLC

TABLE 7.2 Rules for the crossover quantity FLC

Chromosome One Temperature	Chromosome Two Temperature	Crossover Quantity
Low	Low	VLow
Low	Medium	Low
Low	High	Medium
Medium	Low	Low
Medium	Medium	Medium
Medium	High	High
High	Low	Med
High	Medium	High
High	High	VHigh

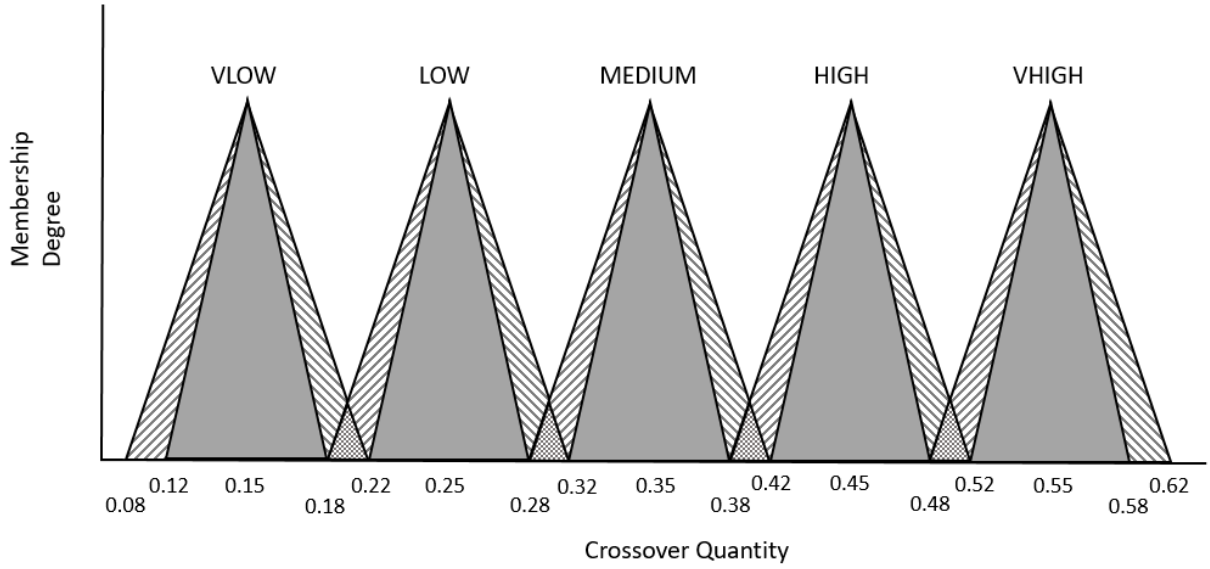


Fig 7.10 Output membership functions for crossover quality

As crossover is at the chromosomal level as is this IT2FLC, the crossover quality is determined for each child chromosome individually. For example if there are 3 chromosomes in each parent solution and each crossover operation is due to create two offspring then the FLC will be run 6 times. Once for each crossover that occurs with one parent being the first input for the first child chromosome and the second parent being the first input for the second child. Fig 7.11 demonstrates and example of which chromosomes crossover within the two parent stacks.

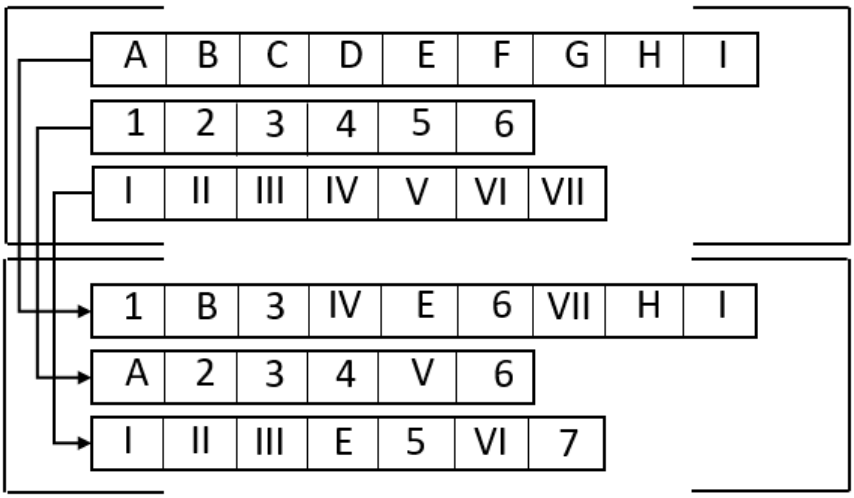


Fig 7.11 Demonstration of which chromosomes crossover over in the Stack

Crossover in the digital capacity planning problem works by taking a copy of parent *P1* randomly replacing genes with the appropriate counter parts from the other parent *P2*. The number of genes replaced is represented by the crossover quality. For example if *P1* is the first input and *P2* is the second input then the first offspring is a copy of *P1* with a number of genes randomly replaced by the *P2* counterpart, then *P1* and *P2* swap as inputs. Fig 7.12 shows an example of this crossover taking place with 2 parents and 2 children.

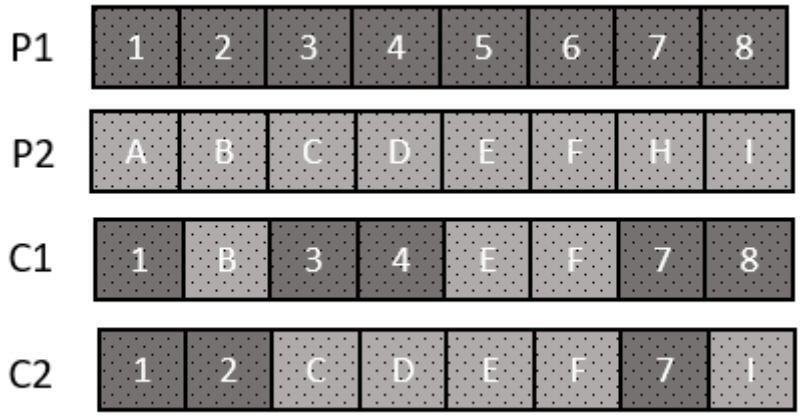


Fig 7.12 Example of crossover between chromosomes

In the final stage of crossover each chromosome inherits its temperature from its parent with the highest temperature. Once inherited the temperature is increased by a small amount. In the case of the digital capacity planning problem this is an increase of 10%. The quantity that the temperature increases by is a tuneable value, much like mutation chance in a GA.

Once crossover has occurred throughout the entire stack, there is random chance that mutation can happen. A predetermined amount of the genes are randomly swapped throughout a stack. Each gene can only be swapped once, ensuing that mutation doesn't make a change and revert it in the same operation.

7.1.4 Temperature

At the end of every generation the population is reduced based upon the sorting described in Section 7.1.2 which incorporates the temperature of solutions into its decision making process. During crossover a comparison of parent temperatures is used to determine how much information should be passed onto the next generation from each parent as described in Section 7.1.3. During crossover new members of the population are created with a higher temperature than that of their parents, introducing a temperature gain into the algorithm. But in order to maintain a balance over temperature in the population the temperature of every population member must be periodically reduced every generation. Given that every solution has its own temperature at a specific generation T_G and that the temperature reduced Tr has been set temperature of the next generation T_{G+1} can be calculated as shown in equation 7.2. Temperature for the next generation is the temperature of the current generation reduced by some small amount, this is expressed as a multiplication in equation 7.2 thus the reduction is one minus the specified amount.

$$T_{G+1} = T_G * (1 - Tr) \tag{7.2}$$

In the case of the digital capacity planning problem the temperature is reduced by 10% every generation. With these temperature manipulation mechanisms in place the effects of temperature on population reduction and crossover become less prominent overtime allowing more traditional ranking to effect the decision making. Due to temperature in some cases potentially more promising members of the population are not prioritised as part of selection and sometimes even removed from the population. This is a purposeful design choice of the algorithm, allowing worse performing member of the population to contribute genetic information to the overall population prior to being removed. This effect happens as temperate has a strong influence on the population sorting.

7.2 Experiments and Results

In the case of the digital capacity planning problem the goal of the system was to be able to reorganise the customers represented on VLans in the form of SVLans and CVLans. In order to establish the quality of the results a comparison between HS and NSGA-II was undertaken. This comparison is a simple yet effect one with the dominance of a solution front determining which algorithm performs better at this problem. A dominance comparison is used due to the problem being a multi-objective problem. In order to keep the comparison as fair as possible the experiment is run 15 times with three different configurations of population size and number of generations. NSGA-II and HS have the same mutation change and crossover strategy with NSGA-II taking 50% of each of it parents' genes as opposed to the variable amount in HS.

Prior to the comparison between NSGA-II and HS first an internal comparison of HS is performed, comparing a Type-1 FLC and an IT2FLC to determine if the increased computational complexity of Type-2 fuzzy logic yields better results. In this comparison the experiment is run 25 times with a configuration of 250 generations with a max population of 250. The results to this comparison are shown in table 7.3

TABLE 7.3 Type-1 and Tpye-2 comparison within the Heated Stack algorithm

	Type-1 Dominance	**Type-2 Dominance**
The Proposed Heated Stack Algorithm	7/25	18/25

As seen in Table 7.3 18 out of the 25 times the experiment was run Type-2 Fuzzy Logic provided a dominate solution front whereas in 7 out of the 25 times Type-1 Fuzzy Logic provides a dominant solution front. This shows that Type-1 is capable of providing a dominant

solution front, but Type-2 provides is more consistent at 72 % of the time. From this point onwards the HS will be using an IT2FLC as it tends to yield better results.

Table 7.4 shows the results from the comparison between the HS and NSGA-II, in these experiments three configurations are used to see how NSGA-II and HS performs. The configurations are as follows: 100 generations with a max population of 100, 250 generations with max population of 250 and finally 500 generations with a max population of 500.

TABLE 7.4. Comparison between HS and NSGA-II

Generations/ Populations	NSGA-II Dominance	Heated Stack Dominance
100	7/15	8/15
250	3/15	12/15
500	1/15	14/15

As shown in Table 7.4, the first case with a configuration of 100 population and generations the difference between NSGA-II and HS is very small with them providing almost the same result, not indicating that either is a better algorithm at the digital capacity planning problem. In the second case with the configuration of 250 population and generations HS is producing a dominant result in 12 of the 15 cases showing that it is slightly better than NSGA-II at this problem. Finally in the last configuration with 500 population and generations HS is producing a dominant result in 14 out of the 15 cases, clearly showing that it is producing a better result for this problem when given an increased number of generations and population members allowing temperature the time to work as intended.

7.3 Discussion

In this chapter the Heated Stack Algorithm was introduced for the digital capacity planning problem. HS is an NSGA-II inspired multi-objective evolutionary algorithm with a temperature

control system. It has been shown that HS outperforms NSGA-II at the digital capacity planning problem and that a Type-2 version of HS will outperform its Type-1 counterpart.

The Heated Stack has been designed with capacity planning problems in mind hence the solution representation having the three layers of gene, chromosome and stack, which is no coincidence that it is very similar to the digital capacity planning problem representation of VLan, SVLan and CVLan.

The key aspect of the HS is its temperature used to guide the population through the search space by manipulating crossover, selection and population reduction. As seen in the experiments, the temperature mechanism requires a larger number of generations to be able to perform. Temperature is used to balance exploration and exploitation, allowing worse members of the population to mate with more promising ones. During crossover temperature is increased to allow new solutions more time in the population, and if they are promising now they will be able to pass the temperature increase onto the next generation, propagating the effects of temperature over the course of multiple generations. Temperature reduces every generation meaning the oldest member of the population will have the lowest temperatures. Fig 7.13 shows the median temperature of the population over the course of an experiment with 250 population members for 250 generations.

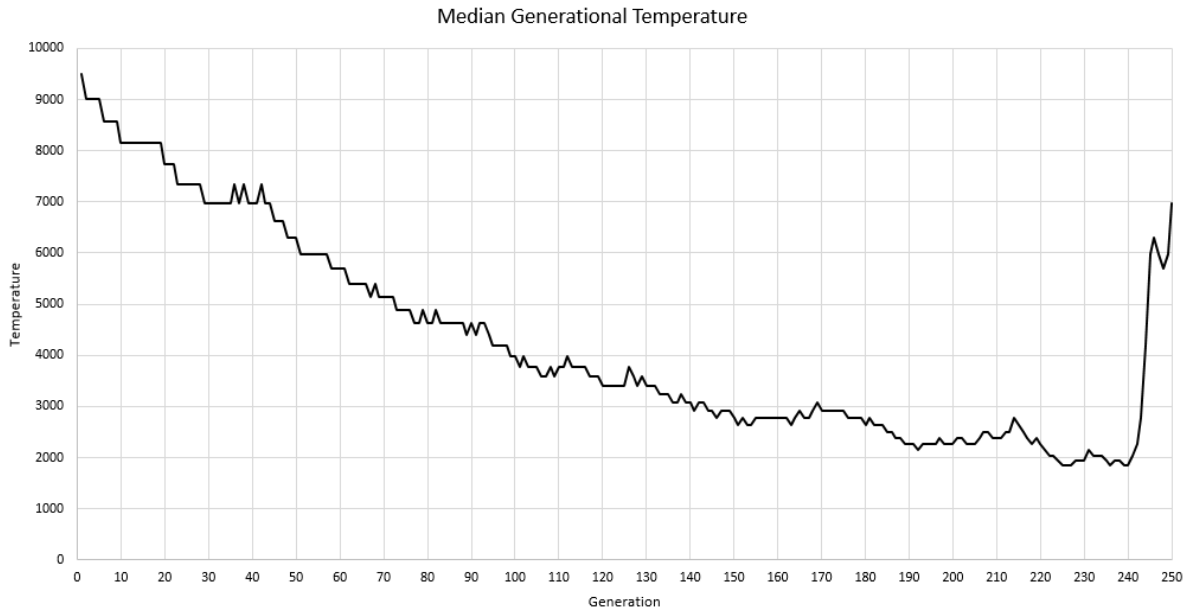


Fig 7.13 Median temperature of the population of the course of 250 generations

In Fig 7.13 it can be seen that the median temperature in the population decreases over the course of a number of generations. In several occasions the median temperature increases by small amounts, such as around generation 40 and 170. In one case the median temperature rises rapidly (at around generation 240) implying lots of new members of the population were created from one another in quick succession. Due to the temperature system effects on HS being controlled through the FLC, the FLCs have been designed to change if HS is in an explorative state or exploitative state, with a high temperature indicating exploration and a low temperature indicating exploitation.

The content of this chapter was first presented at FUZZ-IEEE 2021 in a conference paper named "A Type-2 Fuzzy Multi-Objective Multi-Chromosomal Optimisation for Capacity Planning within Telecommunication Networks" [38].

In the next chapter the progression of the HS as a more general purpose optimisation algorithm will be presented whilst also being used to tackle the much more complex and constrained problem of physical capacity planning.

Chapter 8. A Heated Stack based Type-2 Fuzzy Multi-Objective Optimisation System for Telecommunications Capacity Planning

As the complexity of multi-objective problems has increased the introduction of constraints was inevitable. In order to deal with this, evolutionary algorithms have been adapted to deal with constraints as seen in Section 4.3. NSGA-II and NSGA-III (Section 4.2.3 & Section 4.2.4) are both known to be capable optimisation algorithms that can take constraints into account.

In the previous chapter the Heated Stack Algorithm was introduced as a multi objective evolutionary algorithm for the digital capacity planning problem (Section 2.5.1). It was shown to be more effective than NSGA-II at solving this problem whilst also showing that there was an increase in the quality of results by using a Type-2 system over its Type-1 counterpart. The effectiveness of HS is good when compared to NSGA-II but more effective methods exist in the literature, so the algorithm has been improved in order to be capable of comparison with one of the best evolutionary algorithms in the literature, namely NSGA-III.

In this chapter the changes to the HS will be outlined and a description of how the HS is used to solve the physical capacity planning problem (as seen in Section 2.5.2) shall be given. The physical capacity planning problem is a constrained multi-objective problem, so an explanation of the constraint handling technique used in HS will also be given. Additionally, to show the capabilities of the HS a set of open-source problems will be optimised, allowing for reproducibility of results.

8.1 The Heated Stack Improvements

The HS is the same as described in Chapter 7 with its generational operations of crossover, selection, mutation, population reduction and temperature reduction. It still has a focus on using temperature to manipulate whether the system is in an explorative or exploitative state.

The HS algorithm does however have one minor change to its sorting system. In the previous version the sorting Fuzzy Logic system took its inspiration from NSGA-II using temperature and crowding distance as the inputs. In the improved version HS uses the distance metric from the Niche-Preservation operation as seen in Section 4.2.4 instead of the crowding distance. In NSGA-III Niche-Preservation is used in conjunction with the domination rules to rank the population for selection and population reduction. In HS the population is ranked by the domination rules and then placed upon a reference plane much the same as NSGA-III, but this is where the similarities end.

In NSGA-III, only the members of the population with the lowest distance are allowed into the next generation in the case of population reduction and in the case of selection the population members with the lowest distance are the best ranked. In the HS members of the population placed upon the reference plane allowing their distances to be calculated. The distance of a given population member is used as an input to the FLC alongside its temperature. In order to calculate the niche distance there must be a normalised reference plane. An equidistance reference plane with a number of reference points equalling that to the maximum population size is used, as seen in Fig 8.1.

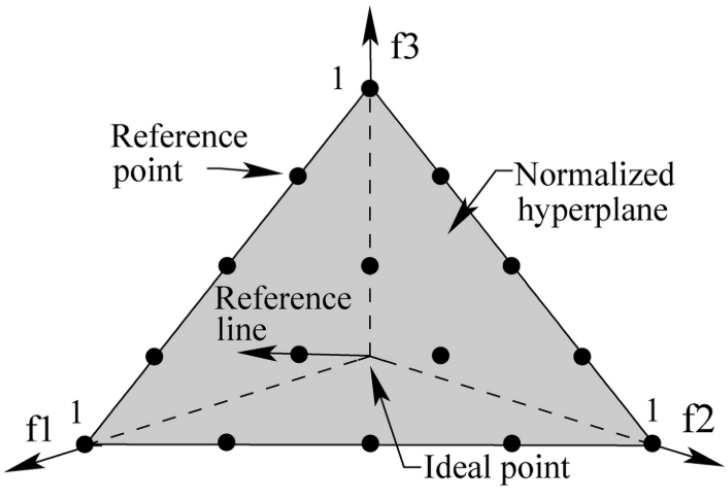


Fig 8.1. Example of an equidistance refinance plane in a three dimensional space [72]

In order to use the reference plane the objective values of each population member must been normalised between 0-1, using the normalisation equation as seen in equation 8.1. Given that the objective functions are minimisation objectives, the minimum value for normalisation is the lowest possible value an objective can be. The maximum value is the largest value seen so far across all generations and finally x is the objective value of a given solution.

$$Normalise\ (x)\ =\ \frac{Maximum - x}{Maximum - Minimum} \tag{8.1}$$

As seen in a GA and in the HS sorting is used for two processes: selection and population reduction. In both cases the fuzzy logic system is used to sort the members of the fronts after they have been sorted by the domination rules. In the case of population reduction, the members of the final front that is capable of contributing to the next generation need to be sorted, this is visualised in Fig 8.2.

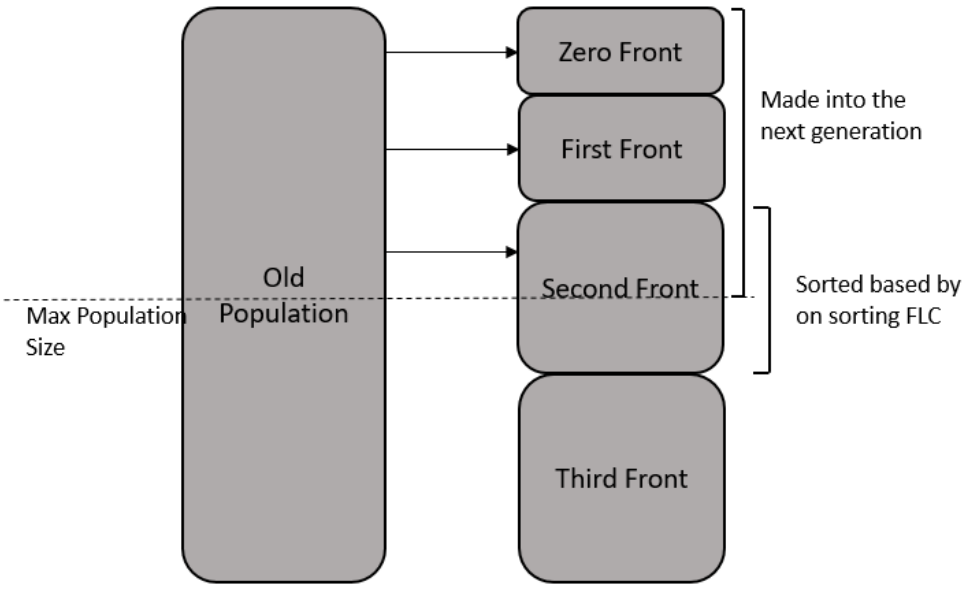


Fig 8.2 Visualisation of population that needs to be sorted for population reduction

In the case of selection, the population is broken down into two sets randomly then each of these sets must be sorted. This sorting happens in two stages, first into their fronts and then each of the fronts is sorted by the FLC. Fig 8.3 shows the input fuzzy set for temperature, Fig 8.4 shows the input set for niche distance, Table 8.1 shows the rules and Fig 8.5 shows the output set of the sorting FLC. The new sorting FLC is identical to the previous version, but this version uses the niche distance as opposed to crowding distance. Otherwise HS is unchanged in its operations to as described in Chapter 7.

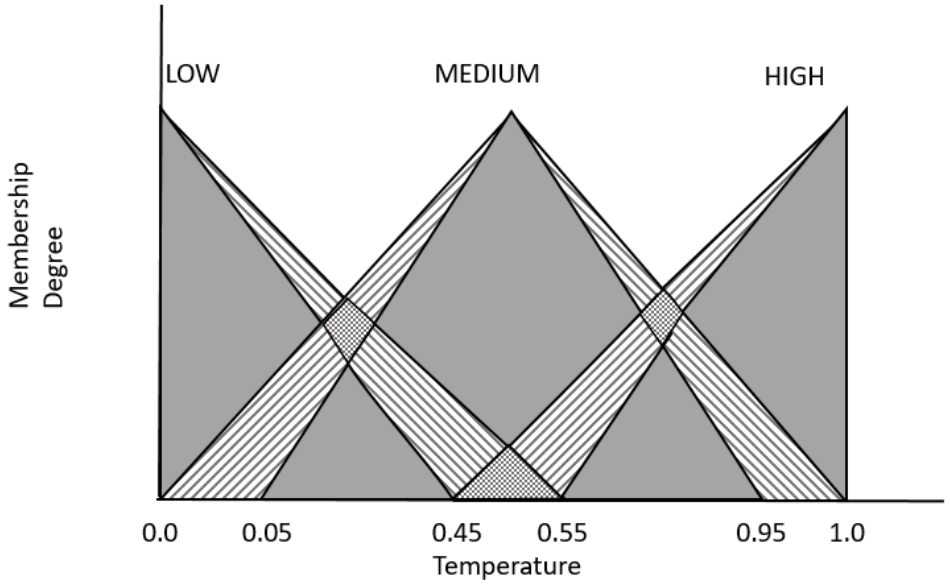


Fig 8.3. Input membership function for temperature, as part of the sorting FLC

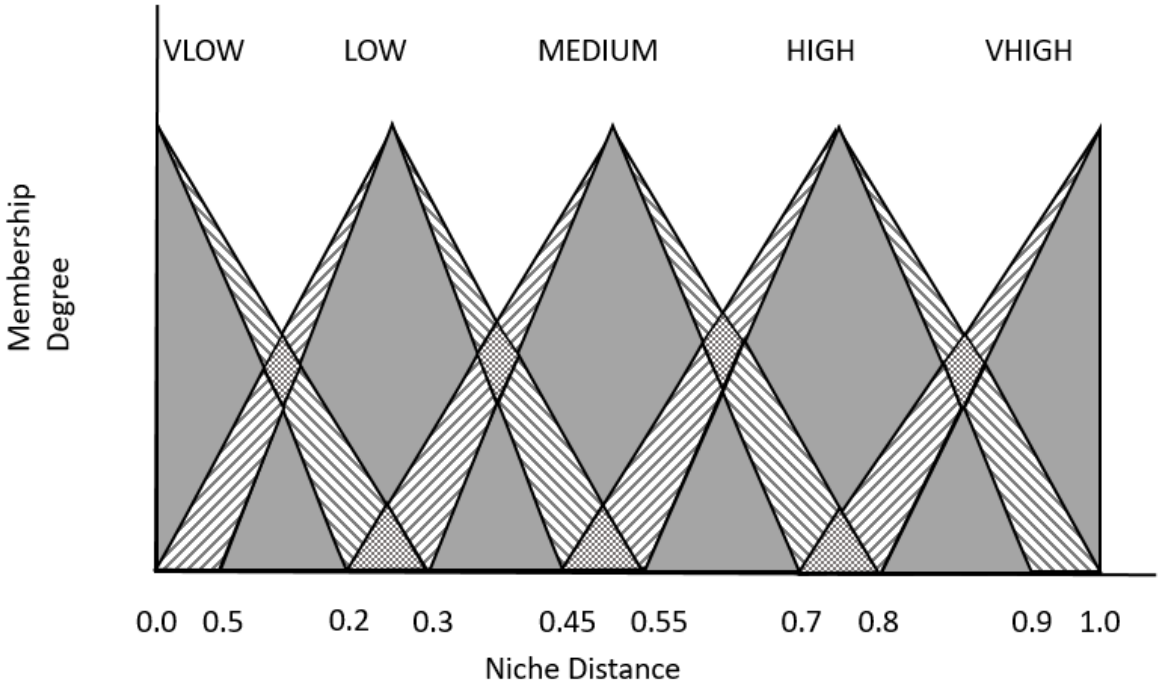


Fig 8.4 Input membership function for niche distance, as part of the sorting FLC

TABLE 8.1 Rules for the sorting FLC

Temperature	Niche Distance	Output
Low	VLow	High
Low	Low	Med
Low	Medium	Low
Low	High	VLow
Low	VHigh	VLow
Medium	VLow	VHigh
Medium	Low	High
Medium	Medium	Medium
Medium	High	Low
Medium	VHigh	VLow
High	VLow	VHigh
High	Low	High
High	Medium	High
High	High	Med
High	VHigh	Low

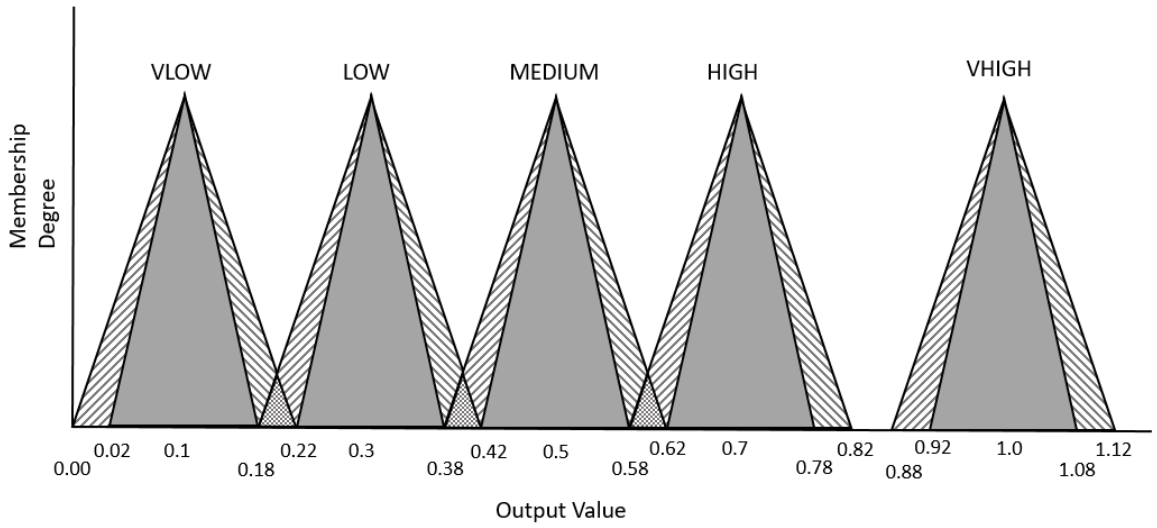


Fig 8.5 Output membership functions as part of the sorting FLC

8.2 Experiments and Results

There are two data sets used for the experiment in this section, firstly the Telecoms data set presented by British Telecom, the second are a set of open source problems from the IEEE Congress on Evolutionary Computation (CEC). Due to the sensitive nature of the Telecoms data set I am unable to go into extended detail, whereas the CEC data sets are openly available online.

In the physical capacity planning problem as described in Section 2.5.2 the goal of the system is to reorganise which ports are occupied by cables. This reorganisation allows for older bandwidth equipment to be replaced with new more effective equipment. The experiments are using a real data set representing part of a networking exchange. The data is highly constrained with lots of specific constraints but only 4 general constraints, made up of 1000s of decisions variables but with only two objectives.

Every year the conference GECCO (the Genetic and Evolutionary Computation Conference) and CEC (Congress on Evolutionary Computation) run a wide array of competitions. These competitions provide open source data sets for many different problems. From the selection of competitions and data sets the best fit for HS are the Real-World Multi-Objective Constrained Optimisation otherwise known as the Constrained Multi-Objective Problem (CMOP)[97]. A subset of these problems have been selected for the experimentation. The problems vary between 2-3 objectives and between 2-10 constraints, Table 8.2 outlines these problems.

8.2.1 Open Source Problems

TABLE 8.2 List of open source Constrained Multi Objective Problems used in the Heated Stack Experiments

Name	Topic	Number of Objectives	Number of Constraints
RCM01	Pressure Vessel Design	2	2
RMC02	Vibrating Platform Design	2	5
RMC03	Two Bar Truss Design	2	3
RCM04	Welded Beam Design	2	4
RCM05	Disc Brake Design	2	4
RCM08	Car Side Impact Design	3	10
RCM17	Bulk Carrier Design	3	6
RCM18	Front Rail Design	2	3
RCM22	Haverly's Pooling Problem	2	6
RCM23	Reactor Network Design	2	5
RCM24	Heat Exchanger Network Design	3	8
RCM25	Process Synthesis Problem	2	2
RCM26	Process Synthesis and Design Problem	2	2

Each problem shown in Table 8.2 is used in a comparison between HS, NSGA-II and NSGA-III. NSGA-II has been selected for comparison as it is a popular multi-objective optimisation algorithm and as it was used in the last set of HS experiments. NSGA-III has been selected for comparison as it is a direct influence on HS, additionally it is listed as one of the joint top performing algorithms in the competition.

The experiment undertaken uses the Hyper Volume Indicator (HVI) as the performance metric as described in Section 4.4. The HVI is used due to its acceptance in both GECCO and CEC

as an appropriate metric for optimisation algorithm comparison. The algorithm that produces the higher HVI creates a better solution front and thus is the better algorithm in this instance. In order to validate the results found by using the HVI, the solution fronts are checked for domination to ensure that the HVI is accurately portraying the solution picture. The comparison between HS, NSGA-II and NSGA-III uses three different configurations across the experiments, as shown in Table 8.3.

TABLE 8.3 List of the open source experiments and their configurations

Experiment Name	Population	Generation
NSGA-II vs HS 100	100	100
NSGA-II vs HS 250	250	250
NSGA-II vs HS 500	500	500
NSGA-III vs HS 100	100	100
NSGA-III vs HS 250	250	250
NSGA-III vs HS 500	500	500

The comparisons are conducted at different configurations to try and generate a clearer picture of the overall performance of each algorithm. For each of the 13 experiments the open source problems are run 25 times with each of the 6 configurations, therefore a total of 1950 experiments are conducted. Fig 8.6, Fig 8.7 and Fig 8.8 show the comparisons of NSGA-II and HS at each of the configurations, each graph shows a count of which algorithm produced the higher HVI for each problem.

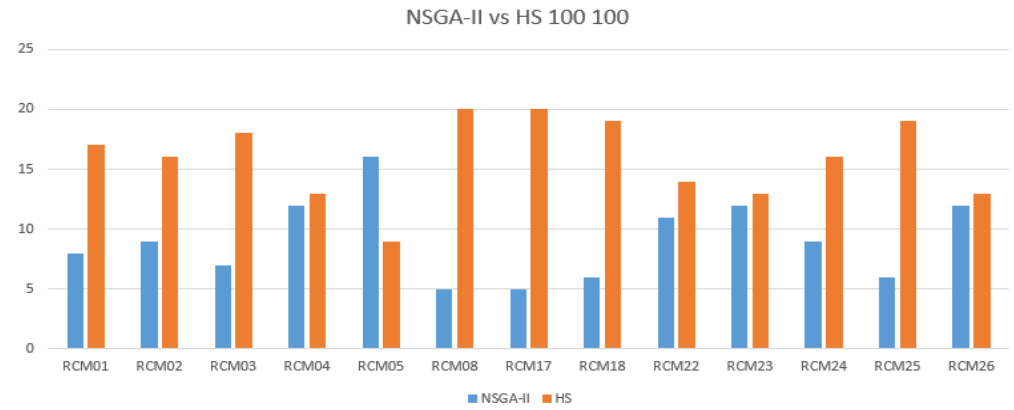


Fig 8.6. A comparison of NSGA-II and HS with a configuration of 100 population members for 100 generations.

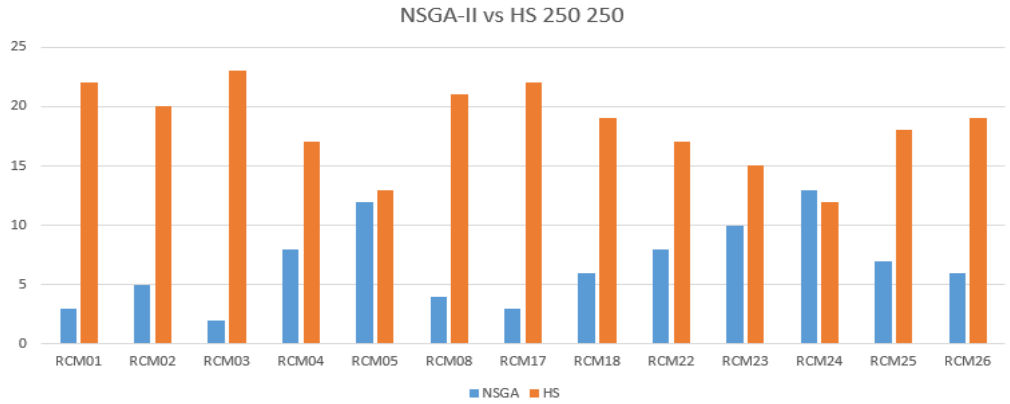


Fig 8.7. A comparison of NSGA-II and HS with a configuration of 250 population members for 250 generations

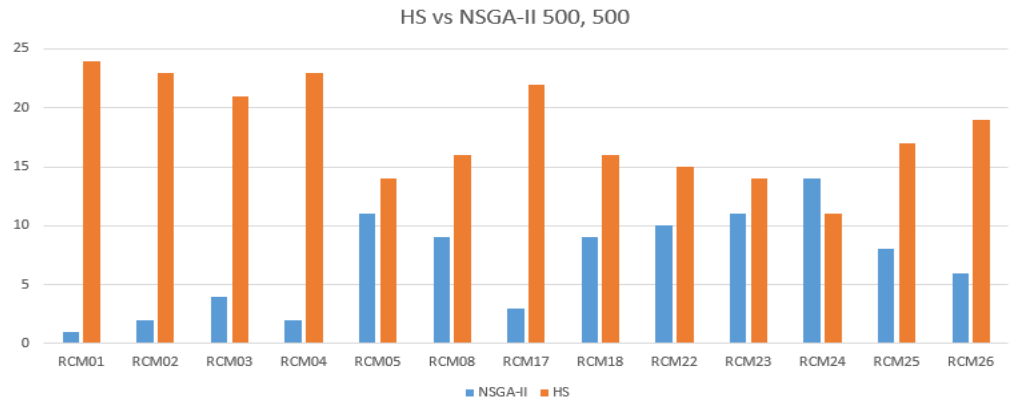


Fig 8.8. A comparison of NSGA-II and HS with a configuration of 500 population members for 500 generations

Fig 8.6 shows that in HS has a larger HVI more often in 11 out of the 13 open source problems. Fig 8.7 an improvement in the performance of HS with 12 out of the 13 open source problems having a larger HVI more often. Fig 8.8 shows the same result a Fig 8.7 with HS having a higher HVI move often in 12 of the 13 problems. This comparison reinforces the result from Chapter 7 and is to be expected especially after the changes made to the HS.

Fig 8.9, Fig 8.10 and Fig 8.11 show the comparison between NSGA-III and HS at each of the configurations, again each graph shows a count of which algorithm produced the higher HVI for each of the problems.

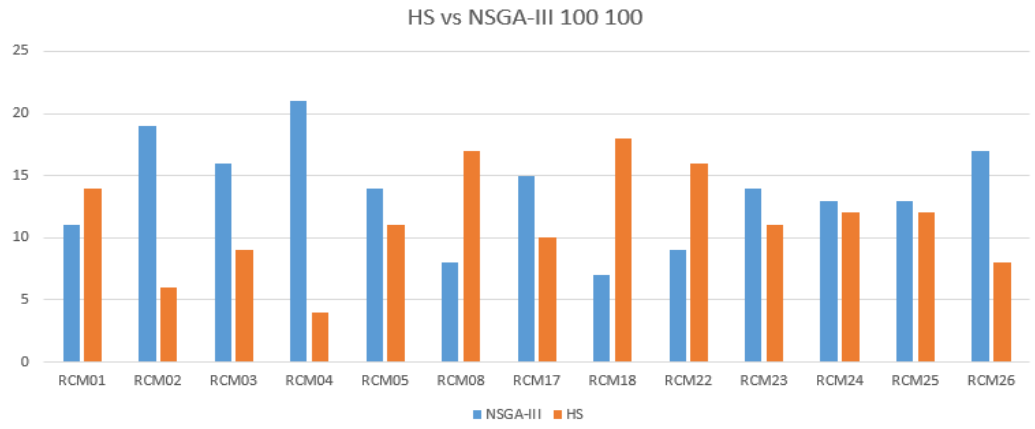


Fig 8.9. A comparison of NSGA-III and HS with a configuration of 100 population members for 100 generations

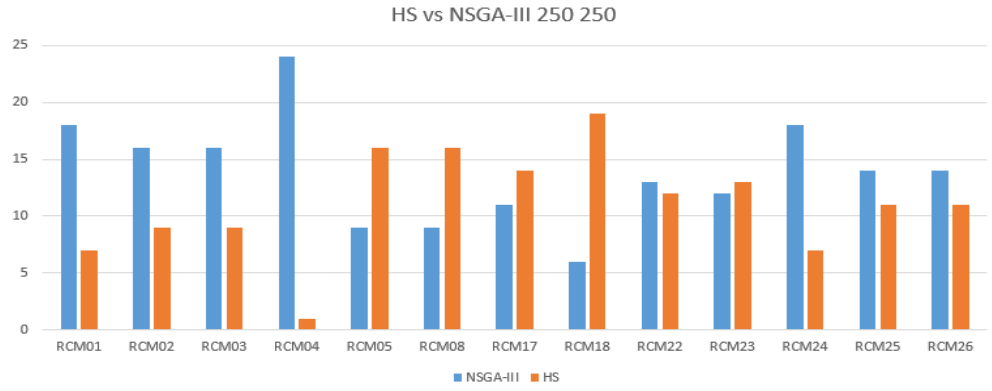


Fig 8.10. A comparison of NSGA-III and HS with a configuration of 250 population members for 250 generations

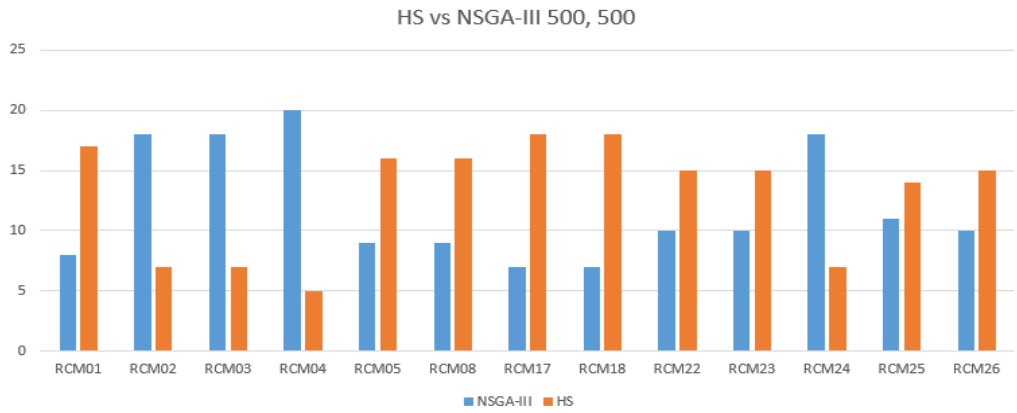


Fig 8.11. A comparison of NSGA-III and HS with a configuration of 500 population members for 500 generations

Fig 8.9 shows the results of the comparison between NSGA-III and HS, it indicates that HS only returns a larger HVI more often in 4 out of the 13 problems. Fig 8.10 shows little improvement over the results in Fig 8.9 with HS having a larger HVI more often than NSGA-III in only 5 out of the 13 cases, showing that NSGA-III is still better performing overall. Fig 8.11 shows a notable improvement over the results in Fig 8.9 and Fig 8.10. In Fig 8.11 HS has a better HVI more often in 9 out of 13 cases, this is a drastic improvement over the previous configurations. But it is to be expected that HS will perform better with an extend number of generations as it gives temperature time to move the algorithm between explorative and exploitive states.

Table 8.4 shows a comparison of median HVI values of NSGA-II, NSGA-III and HS for the 500 population and 500 for generations experiment. This comparison is presented in this table with the median values as the comparisons in Fig 8.6 – 8.11 do not paint an accurate picture of the results.

TABLE 8.4 Median comparison of hyper volume indicator values between NSGA-II, NAGA-III and HS with the configuration of 500 maximum population and 500 generations for the open source problems.

Problem	HS Median	NSGA-II Median	NSGA-III Median	HS vs NSGA-II	HS vs NSGA-III
RCM01	2.15e+11	4.25e+8	1.32e+11	HS	HS
RCM02	2.0e+5	7.9e+3	1.41e+6	HS	NSGA-III
RCM03	7.06e+5	7.9e+3	4.32e+4	HS	HS
RCM04	6.84e+4	2.6e+3	2.53e+5	HS	NSGA-III
RCM05	4.98e+4	5.81e+4	3.7e+4	NSGA-II	HS
RCM08	4.35e+11	5.88e10	5.44e+11	HS	NSGA-III
RCM17	1.82e+13	5.16e+11	3.13e+12	HS	HS
RCM18	8.46e+3	8.39+e3	8.35e+3	HS	HS
RCM22	2.23e+5	8.47e+4	1.70e+5	HS	HS
RCM23	2.98e+4	8.41e+4	2.97e+4	HS	HS
RCM24	3.57e+7	2.02e+4	5.66e+8	HS	NSGA-III
RCM25	1.50e+6	9.70e+5	1.13e+6	HS	HS
RCM26	1.41e+6	8.96+e5	1.28e+6	HS	HS
TOTALS	**1.45e+12**	**4.45e+10**	**2.93e+11**	**12 (92.3%)**	**9 (69.2%)**

Table 8.4 shows the results in a different light to how they are represented in Fig 8.6- Fig 8.11 but it shows the same story. The final row of Table 8.4 named Totals, collected the averages of the median values for comparison, and count the number of times HS outperforms NSGA-II and NSGA-III. NSGA-II is by far the worst performing out of the three algorithms with HS producing a better result on average in 92.3% of cases. NSGA-III outperforms NGSA-II in 11 of the 13 cases only being outperformed in problem RCM05 and RCM18. HS outperforms NSGA-III in 69.2% of the cases. NSGA-II performs exceptionally well at RCM05 out performing both NSGA-III and HS, whereas RCM18 NSGA-II only outperforms NSGA-III by a small margin but is itself outperformed by HS. Looking at the final row of Table 8.4 it summarises that HS will on average outperform NSGA-II and NSGA-III.

8.2.2 Capacity Planning Problem

The main purpose of HS is to optimise the capacity planning problems faced by BT. This problem is comprised of two objectives with four constraints using 1200 decision variables represented in the multi-layered structure of the Heated Stack. Fig 8.12 shows a comparison between NSGA-II and HS for the three configurations of the number of generations and the maximum number of the population members. Fig 8.13 shows the same comparison but between NSGA-III and HS. In all six cases each experiment was performed 25 times, totalling 150 experiments.

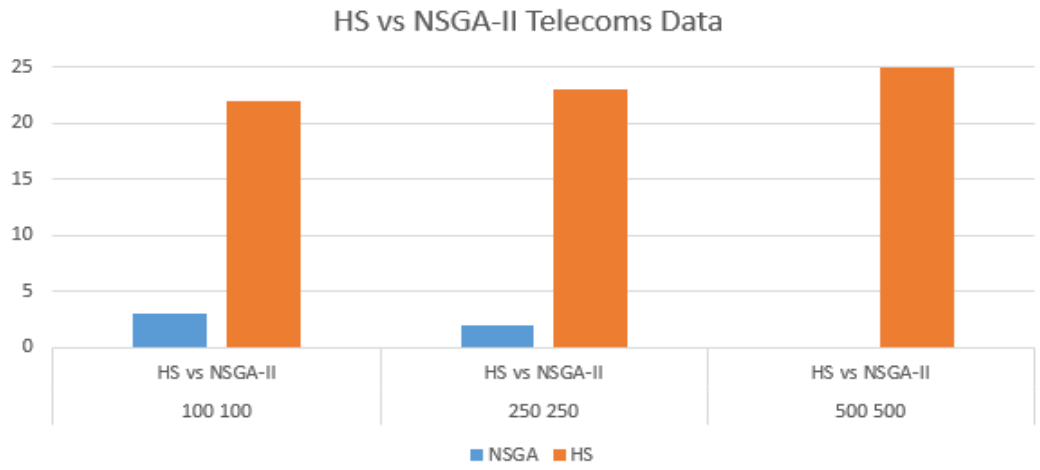
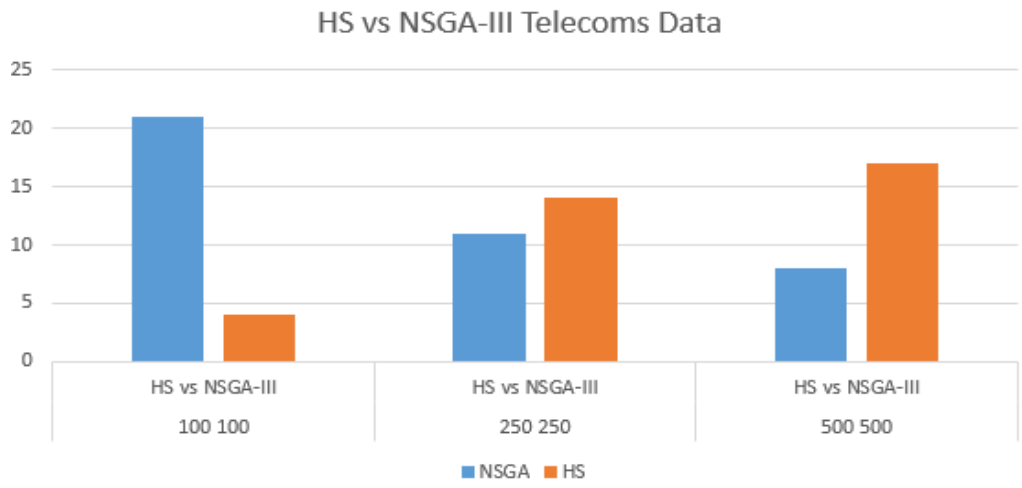


Fig 8.12. Comparison of NSGA-II and HS in the three different algorithm configurations over the course of the 25 experiments

Fig 8.12 shows that HS drastically outperforms NSGA-II in all three configurations with an outperformance in every single experiment in the 500, 500 configuration. Fig 8.13 shows the comparison between HS and NSGA-III, NSGA-III performs much better than NSGA-II when compared to HS. In the first scenario (100, 100) NSGA-III outperforms HS in 21 of the 25 experiments, in the second scenario (250, 250) HS outperforms NSGA-III in 14 of the 25 experiments, in the final scenario (500, 500) HS outperforms NSGA-III in 17 of the 25 cases.



8.13. Comparison of NSGA-III and HS in the three different algorithm configurations over the course of the 25 experiments.

Once again looking at the information presented by Fig 8.12 and Fig 8.13 alone is not enough to get an accurate picture of results. Table 8.5 shows the median HVI values of the experiments in each of the configurations for each of the algorithms.

TABLE 8.5 Median Hyper Volume Indicator values for the NSGA-II, NSGA-III and HS in the three difference algorithmic configurations.

Configuration	HS Median	NSGA-II Median	NSGA-III Median
100, 100	3.43e+12	2.99e+12	3.78e+12
250, 250	6.25e+12	3.11e+12	3.84e+12
500, 500	6.43e+12	3.31e+12	3.89e+12

Looking at Table 8.5 it shows that HS produces a better median HVI value than NSGA-II in all three configurations. The table also shows the NSGA-III produces a better median value than NSGA-II in all configurations. Finally it shows that HS produces a better median value than NSGA-II in 2 of the 3 configurations. HS is only outperformed by NSGA-III in the first scenario, but it has a major increase in its median value in the second and third scenarios.

Capacity planning is not a simple problem to solve but the ability to move cables around in an exchange is a one of the most difficult and important steps to get correct. The results from both sets of experiments (capacity planning and CMOP) show that HS is a capable optimisation algorithm given that it has enough iterations for temperature to have an effect on the algorithm's search.

8.3 Discussion

This chapter has presented the changes to the Heated Stack algorithm namely the way it uses distance within its sorting IT2FLC. It also presented a far more comprehensive set of open source experiments that proves the ability of HS as a general optimisation algorithm.

The results from both sets of experiments show that HS can outperform NSGA-II without any real issues and given that it has enough iterations for temperature to have an effect it can also outperform NSGA-III. These iterations are required to allow the IT2FLC time to use all the rules throughout its rules base and push the algorithm between exploration and exploitation. The system is in a high state of exploration when temperature is high and a high state of exploitation when temperature is low. The work presented in this Chapter is currently under review by the Elsevier journal named "Knowledge-Based Systems".

Given that the Heated Stack can be used to effectively optimise ports as part of the physical capacity planning problem the next step is to construct a plan that can be followed by an engineer in order to make the changes to the exchange. There are some difficulties that come with this due to the fact that an exchange is a live part of the networking infrastructure and any changes made must not disconnect consumers from the network for extended period of time. The next chapter will present a solution to this problem in the form Monte Carlo Tree Search (MCTS) using backward induction.

Chapter 9. Backwards Induction using Monte Carlo Tree Search for Network Configuration Planning

There are many constraints upon a networking exchange that deal with compatibility, functionality or are set in place as part of a wider business goal. One of the most important objectives of BT as an organisation is to minimise down time, in other words to ensure that people have a connection to the network for the maximum amount of time possible. As part of this goal any changes to the bandwidth infrastructure within the exchange must happen overnight, specifically between the hours of 0200 and 0600. All changes must also be reversible within that time frame as well. Therefore, half of the time can be used to make changes and half to reverse the changes if something went wrong meaning the 4-hour time slot to make changes is actually a 2-hour timeslot.

The work presented in Chapter 8 on the optimisation of physical capacity planning provides a set of solutions to the optimal configuration of the cables indicating which port should have which cable in it. In an ideal world to reconfigure the networking exchange every cable would be unplugged and moved to their new locations. This is unfortunately not possible as it would take days if not weeks in most locations to make changes like this. As previously stated, there is only a 2-hour window every night to make changes. Therefore, a step-by-step plan of how the changes presented by the Heated Stack should be made is required.

This chapter begins with an explanation of Monte-Carlo Tree Search (MCTS) and how through the use of backwards induction it can be applied to capacity planning solution representation. Then a simple worked example of the solution representation in the format of a step-by-step guide.

9.1 MCTS

Monte-Carlo Tree Search (MCTS) [98] is a heuristic tree search algorithm that has generated a lot of attention in recent years with success in board games such as Go and Chess [99]. MCTS is a powerful technique in searching through possible states when actions can be taken to manipulate the state. Prime examples of this are board games and video games.

Given that a tree can represent a problem and rewards can be applied to each move represented, MCTS can be used to search the problem space. MCTS operates over the course of 4 stages: Selection, Expansion, Simulation and Backpropagation. These four stages can be seen in Fig 9.1.

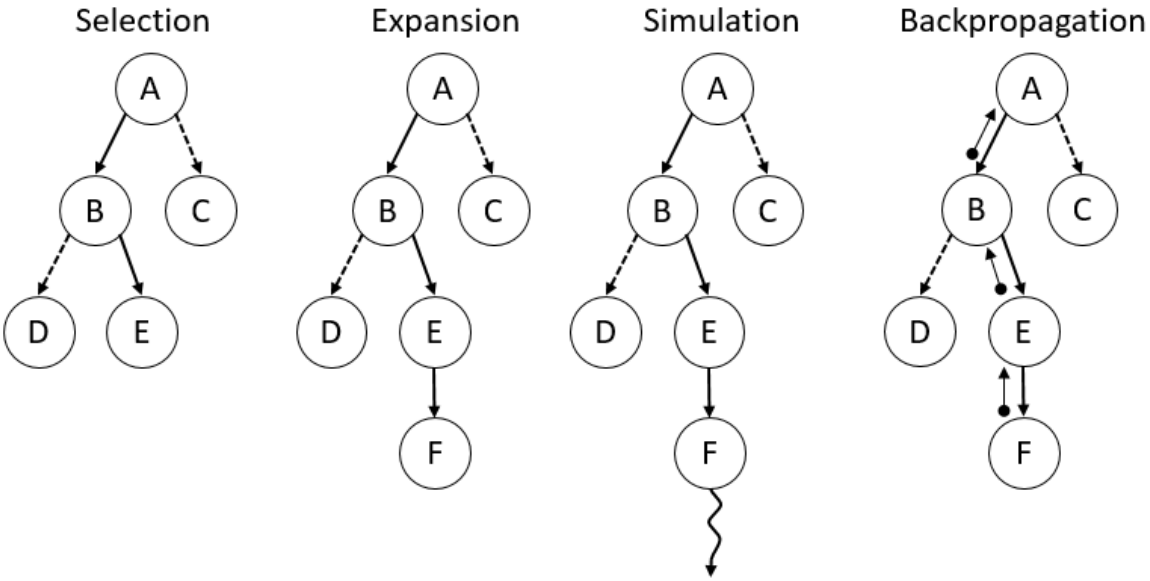


Fig 9.1. A Visualisation of the four stages of MCTS: Selection, Expansion, Simulation and Backpropagation

In the first stage the node for expansion and simulation is selected, in Fig 9.1 this is node *E*. Selection is based upon a calculation known as the UCB1 or UCT algorithm [100] [101] which tries to balance exploration and exploitation. The equation for UCB1 is shown in equation 9.1

where x_j is the average reward obtained from the node j, n_j is the number of times a node has been visited and n is the total number of nodes visited in the tree.

$$\overline{x_j} + \sqrt{\frac{2 \ln n}{n_j}} \tag{9.1}$$

The next stage of MCTS is expansion, where new nodes are added to the selected node, in the case of Fig 9.1 only the one node *F* is added. The third stage is the simulation phase, sometimes known as rollout shown as the downwards line under node *F* in Fig 9.1. In simulation a random set of moves are taken until a terminal state or a stopping criteria have been reached [102]. The simulation is then evaluated and the score is stored in the node. The final stage of MCTS is the backpropagation where information is passed up the tree to each of the parent nodes. The score of a child is added to its parent score and the number of visits is incremented by one. For example in Fig 9.1 the score from the simulation is passed from node *F* to node *E* and the number of visits to node *E* is incremented by one. The new score at node *E* is passed up to its parent node *B* and node *B* has its number of visits incremented.

The four stages of MCTS continue until the tree search is terminated by reaching its goal state or by an outside influence, for example a user or it has been running for a designated amount of time. One of the powerful aspects of MCTS is that it can be stopped at any time returning the best solution that it has found thus far, hence its success in games in single player [103] and multi-player games [104].

9.2 Achieving Backwards Induction with MCTS for Network Configuration Planning

In Chapter 7 and Chapter 8 the idea of using HS to optimise physical and digital capacity planning was introduced. The solutions presented by HS are optimal solutions, showing how the hardware and software should be configured, which is great albeit useless without a plan

of how to implement those configurations. So BT require an implementable solution that gives a step by step guide from the current configuration to the optimal configuration. This is where MCTS can be used to perform backwards induction.

Backwards induction is the idea of traversing a problem space from the goal state back to an origin state [105]. Usually only applied to games or anything that has game-like quality such as economics, backwards indication is seen as an effective way to find an optimal strategy. A prime example of this is completing a maze. A maze can be harder to complete from its start state to the goal state than it is to complete backwards (from the goal to the start).

For MCTS to function optimally it requires a state can be manipulated through actions plus an evaluation function and a goal to search towards. By using the principles of backwards induction the traversal of the problem space can be achieved with MCTS. By starting at the goal state (the output of HS) and making changes that abide by the constraints of the problem it is possible to determine a set of steps that can be used to reorder the cables in an exchange.

This technique presents two interesting qualities due to the fact that HS produces a set of solutions each representing an equally valid optimal solution. Firstly it can be used to determine optimal configurations that are not possible to implement without breaking any of the problem constraints. As the optimal solution presented by HS does not break any constraints but making the changes in a step-by-step may break the constraints, MCTS can be used to find the solutions that does break the constraints in this step by step manner. Secondly it can be used as another evaluation stage to determine which of the optimal solutions are the easiest, fastest and cheapest to implement.

9.3 A Simple Example Solution Representation

Due to the sensitive nature of the contents of a networking exchange a simple example solution has been created in order to demonstrate the results of backwards induction through the use of

MCTS. A given problem has origin and goal states as seen in Fig 9.2 then a set of moves can be set out to rearrange the state. In the example, when an element is moved the other elements shift to the left or right to fill the vacant space. Fig 9.3 shows the set of moves required to manipulate the origin state to the goal state.

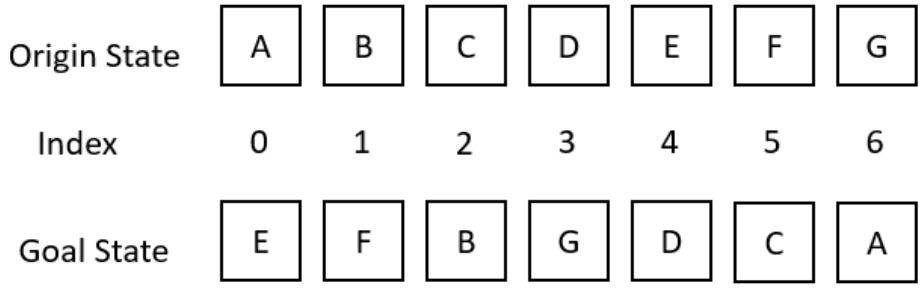


Fig 9.2. Example problem's origin state and goal states with their indexes annotated

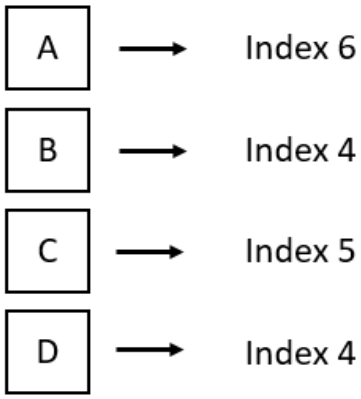


Fig 9.3. The list of moves required in order to get from the origin state to the goal state.

The moves shown in Fig 9.3 must be performed in order to move between the origin state and the goal state. Fig 9.4 shows in a step-by-step guide how the elements are moved around to manipulate the state. In the first move element *A* is moved to index 6 moving all the other elements once to the left. In the second move element *B* is moved to index 4, moving elements *F*, *E*, *D* and *C* once to the left. In the third move element *C* is moved to index 5, moving

elements *G, B, F, E,* and *D* once to the left. In the fourth and final move element *D* is moved to index 4, moving elements *G, B, F* and *E* once to the left.

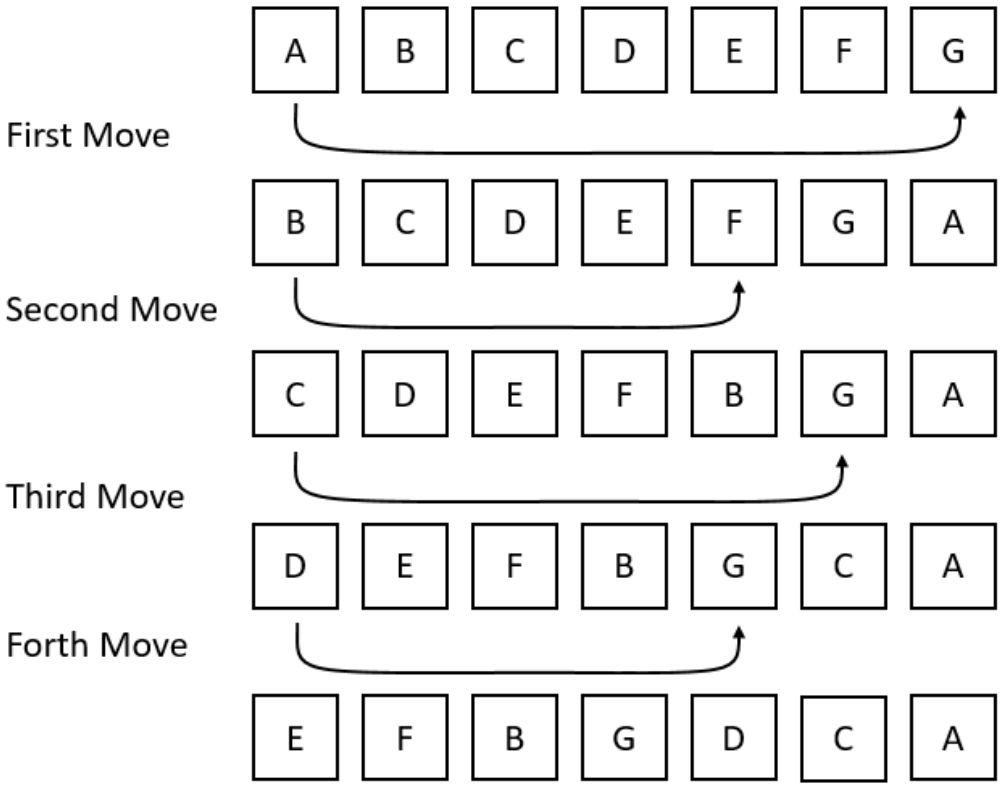


Fig 9.4 Step-by-Step representation of the state changes by performing the moves presented in Fig 9.3

The set of moves used for this example were found through the use of MCTS. The output of the MCTS is the set of moves required to make the changes to the state. In the case of the telecommunications capacity planning problems the set of moves represent the cables that must be moved. In the actual capacity planning application, a cable can only be moved to an unoccupied port on a card and the other ports are not shifted to the left as they are in the example.

9.4 Discussion

In this chapter the idea of using MCTS to perform backwards induction was introduced, using this technique to determine a strategy of implementing the optimal solution from HS without breaking any of the implementation constraints. There a wide variety of very specific constraints set upon the networking equipment and one overarching constraint when it comes to making configuration changes; any changes made to the exchange must be made within a 2-hour time window. This time window exists for two reasons: firstly so any downtime as a result of these changes is during the period of time in which the least bandwidth is required therefore minimising disruption. Secondly so any unsuccessful changes can be reverted with another 2-hour time window and still have minimal disruption.

Currently at BT the task of allocating a set of moves to make changes to the equipment is done by hand without any specific software aid, meaning that even simple changes can takes hours and more complex changes can take days to plan. But by using MCTS even the more complex planning problems can be solved with minimal human effort in a fraction of the time.

There are two additional benefits to using MCTS to produce this plan. In multi-objective optimisation a front or set of solutions are returned as the answer to the objectives, this front can comprise of one or more solutions. MCTS for network configuration planning can be used to rank the members of this front by any number of attributes including: how easy it is to implement the changes, the cost of the changes or the speed at which the changes are possible. The second benefit is that an optimal solution may not break any of the constraints, but there is no series of steps that can get from the current configuration to a configuration presented by HS without breaking some the constraints. MCTS can find these cases and remove them from HS's resulting solution front, therefore reducing the members of the resulting solution front that require a human decision in the final step.

Chapter 10. Conclusion and Future Work

In this thesis, three distinct systems have been discussed: the N-Non-Intersecting-Routing algorithm (NNIR) for resilient routing, the Heated Stack algorithm (HS) for digital and physical capacity planning and backwards induction through the use of Monte Carlo Tree Search (MCTS) for configuration planning. These three systems work together solving different aspects of the wider aim of the thesis.

10.1 Conclusion

The overarching goal of this thesis is to improve the speed, effectiveness and reliability of capacity planning and optimisation within large scale telecommunication networks by decreasing down time and increasing the speed at which upgrades can be performed to an exchange. This goal is broken down into four aims:

I. How to provide assurances steps have been taken to keep a reliable connection between two endpoints in a complex network.
II. How to provide those same assurances in a network when the data used for routing is uncertain or in constant flux.
III. How an ideal version of the current networking infrastructure would be configured to allow planned upgrades to be implemented.
IV. How to make changes to the network infrastructure to without disconnecting consumers for extended periods of time.

These four aims are constructed taking into account British Telecom's business objectives for telecommunications capacity planning and optimisation, which are as follows: first a consumer's connection must be reliable with disconnections being minimised and as infrequent as possible, but if a consumer is disconnected they must be reconnected as quickly as possible.

Second the bandwidth capacity given to a consumer must increase with the demands of a modern interconnected society.

Five interconnected tasks that fulfil the aims of the thesis have been achieved resulting in the main body of work in this thesis and are as follows:

- Perform a comparison into techniques that can perform multi-path resilient routing and determine if one can outperform the currently implemented technique of Dijkstra's algorithm.

This was achieved by performing a comparison between traditional routing methods and single objective optimisation algorithms (as seen in Chapter 5). When the traditional methods are represented by Dijkstra's algorithm and the A-Star algorithm, and the single objective evolutionary algorithms are represented by a Genetic Algorithm (GA) and Simulated Annealing (SA). In the experiments (Section 5.2) it was found that a modified GA that utilised Dijkstra's algorithm named NNIR outperformed the traditional routing methods and SA. Both traditional methods found the same result and found a shortest optimal primary route and a shortest optimal resilient route when evaluated individually, but when evaluated together as single solution these two routes are no longer consider optimal. SA didn't perform much better than the traditional methods only slightly outperforming them. NNIR on the other hand could find a cheaper resilient route in 18% of cases when considering both routes as a single solution seeing, whilst in 20% of cases finding resilient routes where Dijkstra's algorithm couldn't.

- A comparison of type-1 and interval type-2 fuzzy logic for uncertainty handling in resilient routing as part of the selected aforementioned technique resulting in effective and consistent multi path resilient routing in uncertain environments.

This was achieved through an extension to the NNIR algorithm to include a fuzzy logic system (as seen in Chapter 6). A comparison between use of NNIR with a crisp system and NNIR with

type-1 Fuzzy Logic Controller (FLC) and NNIR with an Interval Type-2 Fuzzy Logic Controller (IT2FLC) was performed. In the results (Section 6.3) it was shown that there was a progressive improvement from a crisp to a type-1 FLC to an IT2FLC, at finding a consistent route more often in an uncertain or dynamic environment.

- The creation of a novel evolutionary algorithm to optimise virtual groups within exchange hardware which are used for bandwidth allocation.

This was achieved through the creation of the Heated Stack algorithm (HS) a multi-objective evolutionary algorithm with type-2 controlled fuzzy temperature system (as seen in Chapter 7). This temperature system is used to control the extent that the algorithm is focusing on exploration or exploitation. The system greatly benefited from the use of type-2 fuzzy logic to control the temperature systems as the comparison between type-1 and type-2 showed that the type-2 system outperform its type-1 counterpart in72% of cases. The heated stack proved to be an affective evolutionary algorithm at this specific task outperforming the popular and successful NSGA-II in 93% of cases.

- Improving the aforementioned novel evolutionary algorithm to optimise existing exchange hardware allowing planned upgrades to exchange equipment.

This was achieved through some simple modifications to the population sorting system which greatly improved its performance, whilst also adding the ability to handle constraints within optimisation (as seen in Chapter 8). HS was applied to the task of optimising hardware configurations within an exchange showing to that it performed better than NSGA-II 100% cases and NSGA-III in 68% of cases allowing exchange upgrades to take place (as seen in Section 8.2.2). In addition to the physical capacity planning problem HS was compared to and outperformed NSAG-II in 92 % of cases and NSGA-III in 69% of cases on a set of open-source constrained problems (as seen in Section 8.2.1). The heated stack works well as an optimisation

algorithm given that it has enough generations for temperature to guide the search space back and forth between exploration and exploitation.

- A step-by-step guide of how implement changes to the exchange allowing for the optimised solutions to become reality whilst minimising consumer downtime.

This was achieved with the proposed Monte Carlo Tree Search (MCTS) using the idea of backwards induction (as seen in Chapter 9). The set of solutions presented by HS are useless without a step by step guide of how to implement the changes from the initial configuration to one of the optimal ones presented by HS. MCTS is used to create this guide whilst also serving a secondary function, allowing a final ranking of the set of solutions presented by HS. It can determine the cost and viability of each solution whilst eliminating any solution that would break the constraints of actually implementing the solution, as they are different to the optimisation constraints.

The combination of these five tasks has resulted in the completion of the overarching goal of thesis. Creating system that delivers a faster and better solution than what was in use a BT for these problems.

10.2 Real-World Impact

The speed, effectiveness and reliability of capacity planning and optimisation has been improved within BT through the use of the HS algorithm. As prior to the implementation of this technique all capacity planning optimisation was done by hand without any specific software aid. Now HS is used to aid in the planning process showing what an optimal configuration of the exchange would look like in digital capacity planning, allowing those changes to be made increasing speed and reliability. HS is also used to allocate cables to ports allowing parts of the exchange hardware to be removed without a loss of service to the

consumers. The work undertaken as part of the capacity management with HS has been patented [106] by BT as they believe it is an idea worth protecting.

Backwards induction through MCTS is not currently in use at BT but should be in use in the near future. With the use of MCTS the most efficient set of changes to the exchange can be made decreasing downtime and increasing the speed at which upgrades can be performed.

NNIR is not yet integrated into the wider network allowing for seamless switching of packet routing, but that is being explored. Currently it has taken on another role, being used as part of a pricing tool to outbid competitors across the Europe for routing contracts. The tool and thus the use of the NNIR algorithm has already proved its worth by securing several contracts that would not have otherwise been obtained in the past. Prior to the use of NNIR this pricing tool used Dijkstra's Algorithm to obtain the shortest resilient routes. The work undertaken as part of resilient routing with NNIR has been patented [107] by BT as they believe it is an idea worth protecting.

These outcomes show that AI technologies are having a real-world influence on business profits and the decision making process. These techniques include: Genetic Algorithms, Fuzzy Logic, the Heated Stack and Monte Carlo Tree Search.

10.3 Future Work

There are several avenues of exploration that could result in future work these encompass one of the three area, the NNIR algorithm, the Heated Stack or MCTS.

The heated stack has already proved itself as a high quality optimisation algorithm, but there are always improvements to be made. Firstly the temperature of each chromosome could change independently based on how promising the information within the chromosomes is allowing for the fuzzy crossover system to play a greater role in the balancing for exploration and exploitation. Secondly, the temperature reduction system of the HS should not just be a

single consistent reduction of temperature it should be based upon the information within the population. This would require the IT2FLC for temperature control to be modified based upon the new reduction system otherwise members of the population would not be correctly sorted.

Efforts should be made to improve the execution speed of the Heated Stack allowing it to be used in application that could require it in real time. This would be difficult to make run in under 5 minutes, but for an optimisation application users should be willing to wait given that there is some feedback about the progress of optimisation.

Backwards induction with MCTS could be integrated into HS at the evaluation stage, allowing members of the population to be ranked by the configuration implementation constraints not just the optimisation constraints. Additionally when this next layer of evaluation happens, it should be linked to temperature as not to impact its control over exploration.

MCTS for Backwards induction should be explored in greater detail with an array of experiment across different problems to identify its strengths and weaknesses as an explainable AI tool.

Exploration and exploitation are controlled by the UCB equation within MCTS, at its core this trade-off between exploration and exploitation is one of uncertainty. IT2FLC is well known to be an effective strategy at handling uncertainty so a fuzzy system that decides which node to expand next should be investigated.

NNIR has shown that traditional routing methods are not able to perform adequately at the resilient routing problem. NNIR should be extended to deal with multiple objectives and constraints allowing for a fair comparison of NNIR to HS at the resilient routing problem. NNIR could be streamline to allow it to operate much faster to allow it to be used by application that require real-time routing solutions.

References

[1] A. Bozyiğit, G. Alankuş, and E. Nasiboğlu, "Public transport route planning: Modified dijkstra's algorithm," in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 502–505.

[2] Y. Cuan and X. Chen, "Application of improved Dijkstra algorithm in selection of gas source node in gas network," *Proc. 2012 Int. Conf. Ind. Control Electron. Eng. ICICEE 2012*, pp. 1558–1560, 2012.

[3] J. R. Jiang, H. W. Huang, J. H. Liao, and S. Y. Chen, "Extending Dijkstra's shortest path algorithm for software defined networking," *APNOMS 2014 - 16th Asia-Pacific Netw. Oper. Manag. Symp.*, pp. 1–4, 2014.

[4] V. Bulitko, Y. Björnsson, N. R. Sturtevant, and R. Lawrence, "Real-time heuristic search for pathfinding in video games," in *Artificial Intelligence for Computer Games*, 2011, pp. 1–30.

[5] Z. Zhu, L. Li, W. Wu, and Y. Jiao, "Application of improved Dijkstra algorithm in intelligent ship path planning," in *2021 33rd Chinese Control and Decision Conference (CCDC)*, 2021, pp. 4926–4931.

[6] Cisco, "Cisco: 2020 CISO Benchmark Report," 2020.

[7] K. Kurihara, H. Maruyama, and K. Masuda, "Hierarchical planning method for product supply based on multi objective genetic algorithm," *Proc. 15th IEEE Int. Conf. Emerg. Technol. Fact. Autom. ETFA 2010*, pp. 1–8, 2010.

[8] O. Gerstel *et al.*, "Multi-Layer Capacity Planning for IP-Optical Networks," *IEEE Communications Magazine*, no. January, IEEE, pp. 44–51, 2014.

[9] M. M. and K. C. R. Prasad, C. Dovrolis, "Bandwidth Estimation: Metrics, Measurement Techniques, and Tools," in *in IEEE Network*, 2003, no. December, pp. 27–35.

[10] J. Tu, R. Guo, and Z. Fang, "Capacity planning of ERP based on the state of manufacturing resources," in *2011 International Conference on Consumer Electronics, Communications and*

Networks (CECNet), 2011, pp. 134–137.

[11] Chandan S N, Vinayaka N, and Sandeep G M, "Design, Analysis and Optimization of Race Car Chassis for its Structural Performance," *Int. J. Eng. Res.*, vol. V5, no. 07, 2016.

[12] G. S. Hornby, A. Globus, D. S. Linden, and J. D. Lohn, "Automated antenna design with evolutionary algorithms," *Collect. Tech. Pap. - Sp. 2006 Conf.*, vol. 1, pp. 445–452, 2006.

[13] J. Van Leeuwen, *Handbook of Theoretical Computer Science. Vol. A, Algorithms and complexity*. Elsevier, 1998.

[14] N. Cesa-Bianchi, "Multi-armed Bandit Problem," in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Boston, MA: Springer US, 2014, pp. 1–5.

[15] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire, "Gambling in a rigged casino: the adversarial multi-armed bandit problem," in *Annual Symposium on Foundations of Computer Science - Proceedings*, 1995, pp. 322–331.

[16] C. A. Floudas and P. M. Pardalos, *Encyclopedia of Optimization*. Springer US, 2001.

[17] E. Tomita, "Knapsack," in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Boston, MA: Springer US, 2014, pp. 1–4.

[18] Dake, "Illustration of the knapsack problem, (https://commons.wikimedia.org/wiki/File:Knapsack.svg)." 2006.

[19] S. K. Chaharsooghi and A. H. M. Kermani, "An intelligent multi-colony multi-objective ant colony optimization (ACO) for the 0-1 knapsack problem," in *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, 2008, pp. 1195–1202.

[20] E. L. Lawler, "Fast approximation algorithms for knapsack problems," *Proc. - Annu. IEEE Symp. Found. Comput. Sci. FOCS*, vol. 1977-Octob, no. 4, pp. 206–213, 1977.

[21] "Bin packing problem." [Online]. Available: https://en.wikipedia.org/wiki/Bin_packing_problem.

[22] H. Fujiwara and K. M. Kobayashi, “Bin Packing with Cardinality Constraints,” in *Encyclopedia of Algorithms*, M.-Y. Kao, Ed. Boston, MA: Springer US, 2008, pp. 1–4.

[23] D. S. Johnson, “Near-Optimal Bin Packing Algorithms,” *Thesis*, p. 400, 1973.

[24] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2011.

[25] E. W. Dijkstra, “Solution of a problem in concurrent programming control,” *Commun. ACM*, vol. 8, no. 9, p. 569, 1965.

[26] S. Julius Fusic, P. Ramkumar, and K. Hariharan, “Path planning of robot using modified dijkstra Algorithm,” *2018 Natl. Power Eng. Conf. NPEC 2018*, pp. 1–5, 2018.

[27] Yujin and G. Xiaoxue, “Optimal Route Planning of Parking Lot Based on Dijkstra Algorithm,” *Proc. - 2017 Int. Conf. Robot. Intell. Syst. ICRIS 2017*, pp. 221–224, 2017.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, 1968.

[29] J. Kubacki, L. Koszalka, I. Pozniak-Koszalka, and A. Kasprzak, “Comparison of heuristic algorithms to solving mesh network path finding problem,” in *4th International Conference on Frontier of Computer Science and Technology, FCST 2009*, 2009, pp. 388–392.

[30] K. Khantanapoka and K. Chinnasarn, “Pathfinding of 2D & 3D game real-time strategy with Depth Direction A* algorithm for multi-layer,” in *2009 8th International Symposium on Natural Language Processing, SNLP ’09*, 2009, pp. 184–188.

[31] R. A. Kirkman, “Evaluating Single Point Failures for Safety & Reliability,” *IEEE Trans. Reliab.*, vol. R-28, no. 3, pp. 259–263, 1979.

[32] T. Li and Z. Ge, “A Multiple QoS Anycast Routing Algorithm Based Adaptive Genetic Algorithm,” in *2009 Third International Conference on Genetic and Evolutionary Computing*, 2009, pp. 89–92.

[33] L. Veryard, H. Hagras, A. Starkey, A. Conway, and G. Owusu, “NNIR: N-non-intersecting-routing algorithm for multi-path resilient routing in telecommunications applications,” *Int. J. Comput. Intell. Syst.*, vol. 13, no. 1, pp. 352–365, 2020.

[34] *Collins English Dictionary – Complete and Unabridged*, 12th Editi. 2014.

[35] N. Yusupova, D. Rizvanov, and E. Chernyshev, “Evaluation of the Effectiveness of the Multi-Agent Approach for Capacity Planning,” *Proc. - ICOECS 2020 2020 Int. Conf. Electrotech. Complexes Syst.*, 2020.

[36] Y. K. Lu and W. Y. Chen, “Implementation of required capacity planning for virtual machine development,” *Proc. 2012 8th Int. Conf. Intell. Inf. Hiding Multimed. Signal Process. IIH-MSP 2012*, pp. 331–334, 2012.

[37] S. G. Hamed and M. S. Pishvaee, “A system dynamics approach for capacity planning and price adjustment in a closed-loop supply chain,” *EMS 2009 - UKSim 3rd Eur. Model. Symp. Comput. Model. Simul.*, no. 2, pp. 435–439, 2009.

[38] L. Veryard, H. Hagras, A. Conway, and G. Owusu, “A Type-2 Fuzzy Multi-Objective Multi-Chromosomal Optimisation for Capacity Planning within Telecommunication Networks,” *IEEE Int. Conf. Fuzzy Syst.*, vol. 2021-July, 2021.

[39] R. Blahut, *Fast Algorithms for Signal Processing*. Cambridge: Cambridge University Press, 2014.

[40] L. A. Zadeh, “Fuzzy Sets,” *Infromation Control*, vol. 8, pp. 338–353, 1965.

[41] L. A. Zadeh, “The concept of a linguistic variable and its application to approximate reasoning,” *Inf. Sci. (Ny).*, vol. 8, no. 3, pp. 199–249, 1975.

[42] E. H. Mamdani, “Applications of fuzzy algorithms for simple dynamic plan,” *Proc. IEEE*, vol. 121, pp. 1585–1588, 1974.

[43] H. A. Hagras, “A hierarchical type-2 fuzzy logic control architecture for autonomous mobile

robots," *IEEE Trans. Fuzzy Syst.*, vol. 12, no. 4, pp. 524–539, 2004.

[44] A. Sakalli, T. Kumbasar, E. Yesil, and H. Hagras, "Analysis of the performances of type-1, self-tuning type-1 and interval type-2 fuzzy PID controllers on the Magnetic Levitation system," *IEEE Int. Conf. Fuzzy Syst.*, pp. 1859–1866, 2014.

[45] M. Antonelli, D. Bernardo, H. Hagras, and F. Marcelloni, "Multiobjective Evolutionary Optimization of Type-2 Fuzzy Rule-Based Systems for Financial Data Classification," *IEEE Trans. Fuzzy Syst.*, vol. 25, no. 2, pp. 249–264, 2017.

[46] H. Leon-Garza, H. Hagras, A. Pena-Rios, A. Conway, and G. Owusu, "A big bang-big crunch type-2 fuzzy logic system for explainable semantic segmentation of trees in satellite images using HSV color space," *IEEE Int. Conf. Fuzzy Syst.*, vol. 2020-July, 2020.

[47] H. Hagras, V. Callaghan, M. Colley, and M. Carr-West, "Fuzzy-genetic based embedded-agent approach to learning & control in agricultural autonomous vehicles," *Proc. - IEEE Int. Conf. Robot. Autom.*, vol. 2, no. May, pp. 1005–1010, 1999.

[48] B. Yao, H. Hagras, M. J. Alhaddad, and D. Alghazzawi, "A fuzzy logic-based system for the automation of human behavior recognition using machine vision in intelligent environments," *Soft Comput.*, 2014.

[49] R. Chimatapu, H. Hagras, M. Kern, and G. Owusu, "Hybrid Deep Learning Type-2 Fuzzy Logic Systems For Explainable AI," in *2020 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2020, pp. 1–6.

[50] J. M. Mendel, *Uncertain Rule-Based Fuzzy Systems*. Springer US, 2017.

[51] J. M. Mendel, "Advances in type-2 fuzzy sets and systems," *Inf. Sci. (Ny).*, 2007.

[52] J. M. Mendel and R. I. B. John, "Type-2 fuzzy sets made simple," *IEEE Trans. Fuzzy Syst.*, vol. 10, no. 2, pp. 117–127, 2002.

[53] N. N. Karnik and J. M. Mendel, "Introduction to type-2 TSK fuzzy logic systems," *IEEE Int.*

Conf. Fuzzy Syst., vol. 3, pp. 915–920, 1998.

[54] J. H. Holland, "Genetic algorithms," *Sci. Am.*, vol. 267, no. 1, pp. 66–72, 1992.

[55] O. K. Erol and I. Eksin, "A new optimization method: Big Bang-Big Crunch," *Adv. Eng. Softw.*, vol. 37, no. 2, pp. 106–111, 2006.

[56] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science (80-.).*, vol. 220, no. 4598, pp. 671–680, 1983.

[57] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[58] B. Chopard and M. Tomassini, "Particle swarm optimization," in *Natural Computing Series*, 2018.

[59] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, no. 21, pp. 1087–1092, 1953.

[60] R. A. Rutenbar, "Simulated Annealing Algorithms: An Overview," *IEEE Circuits and Devices Magazine*, no. 1, pp. 19–26, Jan-1989.

[61] K. Sharp and F. Matschinsky, "Translation of Ludwig Boltzmann's paper 'on the relationship between the second fundamental theorem of the mechanical theory of heat and probability calculations regarding the conditions for thermal equilibrium' Sitzungberichte der kaiserlichen akademie d," *Entropy*, vol. 17, no. 4, pp. 1971–2009, 2015.

[62] M. A. S. Barbosa and M. M. Gouvêa, "Access point design with a genetic algorithm," *Proc. - 2012 6th Int. Conf. Genet. Evol. Comput. ICGEC 2012*, pp. 119–123, 2012.

[63] D. Bernardo, H. Hagras, and E. Tsang, "A genetic type-2 fuzzy logic based system for the generation of summarised linguistic predictive models for financial applications," *Soft Comput.*, 2013.

[64] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning.*, Addison-

We. Reading, MA, 1989.

[65] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, *Genetic algorithms in wireless networking: techniques, applications, and issues*, vol. 20, no. 6. Springer Berlin Heidelberg, 2016.

[66] K. Deb and A. Kumar, "Real-coded Genetic Algorithms with Simulated Binary Crossover : Studies on Multimodal and Multiobjective Problems," *Complex Syst.*, vol. 9, pp. 431–454, 1995.

[67] C. W. Ahn and R. S. Ramakrishna, "A genetic algorithm for shortest path routing problem and the sizing of populations," *IEEE Trans. Evol. Comput.*, vol. 6, no. 6, pp. 566–579, 2002.

[68] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An Indicator-Based Multiobjective Evolutionary Algorithm with Reference Point Adaptation for Better Versatility," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 609–622, 2018.

[69] Z. Z. Liu, Y. Wang, and P. Q. Huang, "AnD: A many-objective evolutionary algorithm with angle-based selection and shift-based density estimation," *Inf. Sci. (Ny).*, vol. 509, pp. 400–419, 2020.

[70] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A Coevolutionary Framework for Constrained Multiobjective Optimization Problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 102–116, 2021.

[71] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[72] H. Jain and K. Deb, "An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 577–601, 2014.

[73] V. Chankong and Y. Y. Haimes, *Multiobjective Decision Making Theory and Methodology*. New York, NY,: USA: North-Holland, 1983.

[74] M. Garza-Fabre, G. T. Pulido, and C. A. C. Coello, "Ranking Methods for Many-Objective

Optimization," in *MICAI 2009: Advances in Artificial Intelligence*, 2009, pp. 633–645.

[75] H. Y. Alhammadi and J. A. Romagnoli, "Chapter B4 - Process design and operation: Incorporating environmental, profitability, heat integration and controllability considerations," in *The Integration of Process Design and Control*, vol. 17, P. Seferlis and M. C. Georgiadis, Eds. Elsevier, 2004, pp. 264–305.

[76] S. Suffian, R. Dzombak, and K. Mehta, "3 - Future directions for nonconventional and vernacular material research and applications," in *Nonconventional and Vernacular Construction Materials (Second Edition)*, Second Edi., K. A. Harries and B. Sharma, Eds. Woodhead Publishing, 2020, pp. 63–80.

[77] N. Srinivas and K. Deb, "Muiltiobjective Optimization Using Nondominated Sorting in Genetic Algorithms," *Evol. Comput.*, 1994.

[78] A. J. Starkey, H. Hagras, S. Shakya, and G. Owusu, "A Genetic Algorithm Based Approach for the Simultaneous Optimisation of Workforce Skill Sets and Team Allocation," in *Research and Development in Intelligent Systems XXXIII*, 2016.

[79] L. Beasley, H. Hagras, A. Conway, and G. Owusu, "A Type-2 Fuzzy Based Multi-Objective Optimisation for Strategic Network Planning in the Telecommunication Domain," *IEEE Int. Conf. Fuzzy Syst.*, vol. 2021-July, pp. 1–7, 2021.

[80] D. Xu, Q. Zhang, C. Lu, and S. Liu, "NSGA-II for slab selecting and reheating furnace scheduling in hot rolling production," *Proc. 28th Chinese Control Decis. Conf. CCDC 2016*, no. 2, pp. 6776–6779, 2016.

[81] I. Das and J. Dennis, "Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems," *SIAM J. Optim.*, vol. 8, no. 3, pp. 631–657, 1998.

[82] E. Mezura-Montes and C. A. Coello Coello, "Constraint-handling in nature-inspired numerical optimization: Past, present and future," *Swarm Evol. Comput.*, vol. 1, no. 4, pp. 173–194, 2011.

[83] Z. Ma and Y. Wang, “Evolutionary Constrained Multiobjective Optimization: Test Suite Construction and Performance Comparisons,” *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 972–986, 2019.

[84] H. Jain and K. Deb, “An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach,” *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 602–622, 2014.

[85] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, “Constraint handling in multiobjective evolutionary optimization,” *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 514–525, 2009.

[86] H. Geng, M. Zhang, L. Huang, and X. Wang, “Infeasible Elitists and Stochastic Ranking Selection in Constrained Evolutionary Multi-objective Optimization,” in *Simulated Evolution and Learning*, 2006, pp. 336–344.

[87] L. While, L. Bradstreet, and L. Barone, “A fast way of calculating exact hypervolumes,” *IEEE Trans. Evol. Comput.*, vol. 16, no. 1, pp. 86–95, 2012.

[88] E. Zitzler, D. Brockhoff, and L. Thiele, “The hypervolume indicator revisited: On the design of pareto-compliant indicators via weighted integration,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 4403 LNCS, pp. 862–876, 2007.

[89] H. Lebesgue, “Intégrale, longueur, aire,” Université de Paris, 1902.

[90] K. Dooley, *Designing Large-scale LANs*. O’Reilly, 2002.

[91] W. Zeng and R. L. Church, “Finding shortest paths on real road networks: The case for A*,” *Int. J. Geogr. Inf. Sci.*, vol. 23, no. 4, pp. 531–543, 2009.

[92] “Open Source Road Data, MasterMap highways network roads,” 2019. .

[93] L. Veryard, H. Hagras, A. Starkey, and G. Owusu, “A Fuzzy Genetic System for Resilient

Routing in Uncertain Dynamic Telecommunication Networks," in *IEEE International Conference on Fuzzy Systems*, 2019, vol. 2019-June.

[94] L. Veryard, H. Hagras, A. Conway, and G. Owusu, "A Type-2 Fuzzy Genetic Approach to Uncertain & Dynamic Resilient Routing within Telecommunications Networks," in *IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2020.

[95] D. Asanka and A. S. Perera, "DEFINING FUZZY MEMBERSHIP FUNCTION USING BOX PLOT," *Int. J. Res. Comput. Appl. Robot.*, vol. 5, no. 11, pp. 1–10, 2017.

[96] H. Hagras and C. Wagner, "Introduction to Interval Type-2 Fuzzy Logic Controllers - Towards Better Uncertainty Handling in Real World Applications," *The IEEE Systems, an and Cybernetics eNewsletter*, no. 27, 2009.

[97] A. Kumar *et al.*, "A Benchmark-Suite of real-World constrained multi-objective optimization problems and some baseline results," *Swarm Evol. Comput.*, vol. 67, 2021.

[98] C. B. Browne *et al.*, "A survey of Monte Carlo tree search methods," *IEEE Trans. Comput. Intell. AI Games*, vol. 4, no. 1, pp. 1–43, 2012.

[99] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[100] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," in *Machine Learning*, 2002, vol. 47, no. 2–3, pp. 235–256.

[101] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2006, vol. 4212 LNAI, pp. 282–293.

[102] C. F. Sironi and M. H. M. Winands, "Comparing randomization strategies for search-control parameters in monte-carlo tree search," in *IEEE Conference on Computatonal Intelligence and Games, CIG*, 2019, vol. 2019-Augus, pp. 1–8.

[103] M. P. D. Schadd, M. H. M. Winands, H. J. Van Den Herik, G. M. J. Chaslot, and J. W. H. M. Uiterwijk, “Single-Player Monte-Carlo Tree Search,” in *Proceedings of the Intrnational Conference on Computers and Games*, 2008, pp. 1–12.

[104] H. Finnsson and Y. Björnsson, “CADIA PLAYER : A Simulation-Based General Game Player,” *IEEE Trans. Comput. Intell. AI Games*, vol. 1, no. 1, pp. 4–15, 2009.

[105] E. Ben-Porath, “Rationality, Nash Equilibrium and Backwards Induction in Perfect- Information Games,” *Rev. Econ. Stud.*, vol. 64, no. 1, p. 23, 1997.

[106] L. Veryard, A. Hardwick, and A. Conway, “A35662 Resource Capacity Management,” 2021.

[107] L. Veryard, A. Starkey, H. Hagras, and G. Owusu, “A33790SUSw01 Resilient Network Routing.”

Glossary

Agent: An entity that exist in an environment and can observe and act upon that environment.

Crisp: Using real numbers, or to be used without Fuzzy numbers/sets.

Decision Variables: A set of variables that can be manipulated in-order to effect the evaluated value.

Edge: The connection between two vertices in graph theory.

Exploitation: The act of trying to improve upon a known state, with the intention of always improving the evaluated score.

Exploration: The act of looking for new or unseen states, that don't necessarily improve the evaluated score.

Global Minima: The actual best state that cannot be improved upon.

Local Minima: A state that appears the best that can't be improved upon without making the whole system worse.

Objective: A goal that can be represented as function to be maximised or minimised as a numerical score.

Objective space: A region in which the objective scores can exist, and moved across through the manipulation of the search space.

Optimisation: The act of selecting a set of variables in-order to maximise or minimise some evaluation function.

Search space: A region in which the decision variable can exist, and the entire area that can be traversed during optimisation.

Solution: A set of decision variable that represent a state found by artificial intelligence.

Vertex / **Vertices**: A single point or location in graph theory.