OPTIMISATION HEURISTICS FOR SOLVING TECHNICIAN AND TASK SCHEDULING PROBLEMS

AMY KHALFAY

Ph.D. 2018

OPTIMISATION HEURISTICS FOR SOLVING TECHNICIAN AND TASK SCHEDULING PROBLEMS

AMY KHALFAY

SCHOOL OF COMPUTING, MATHEMATICS AND DIGITAL TECHNOLOGY
MANCHESTER METROPOLITAN UNIVERSITY AND SERVICE POWER

A thesis submitted in partial fullfilment of the requirements of the Manchester Metropolitan University for the degree of Doctor of Philosophy

February 2018

Dedication

I would like to dedicate this work to my parents and my partner, Drewe, who have always encouraged me, given me the confidence to undertake this research, and supported me wholeheartedly throughout my academic studies.

I would like to thank my supervisors, Dr Alan Crispin and Dr Keeley Crockett, who have provided me with immeasurable support and guidance throughout the three years of this research. Their determination and perseverance has ensured that I have achieved the aims and objectives of this research. I would also like to thank my mentor, Mr Alex Syrichas, and unofficial supervisor, Dr Kris Welsh, who have helped me develop my technical skills and have been a key part of helping shape this research.

Lastly, my appreciation to the sponsors of this research Service Power PLC, a leading field service scheduling provider, without whom this research would not have been possible. Throughout undertaking this research Mr Alan Smith has continued to provide insight into the field and the key challenges faced.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

AMY KHALFAY

February 2018

Acknowledgements

This research is sponsored by Service Power Technologies PLC, a worldwide leader at providing innovative mobile workforce management solutions, in cooperation with MMU and KTP.

Abstract

Motivated by an underlying industrial demand, solving intractable technician and task scheduling problems through the use of heuristic and metaheuristic approaches have long been an active research area within the academic community. Many solution methodologies, proposed in the literature, have either been developed to solve a particular variant of the technician and task scheduling problem or are only appropriate for a specific scale of the problem. The motivation of this research is to find general-purpose heuristic approaches that can solve variants of technician and task scheduling problems, at scale, balancing time efficiency and solution quality. The unique challenges include finding heuristics that are robust, easily adapted to deal with extra constraints, and scalable, to solve problems that are indicative of the real world.

The research presented in this thesis describes three heuristic methodologies that have been designed and implemented: (1) the intelligent decision heuristic (which considers multiple team configuration scenarios and job allocations simultaneously), (2) the look ahead heuristic (characterised by its ability to consider the impact of allocation decisions on subsequent stages of the scheduling process), and (3) the greedy randomized heuristic (which has a flexible allocation approach and is computationally efficient).

Datasets used to test the three heuristic methodologies include real world problem instances, instances from the literature, problem instances extended from the literature to include extra constraints, and, finally, instances created using a data generator. The datasets used include a broad array of real world constraints (skill requirements, teaming, priority, precedence, unavailable days, outsourcing, time windows, and location) on

a range of problem sizes (5-2500 jobs) to thoroughly investigate the scalability and robustness of the heuristics.

The key findings presented are that the constraints a problem features and the size of the problem heavily influence the design and behaviour of the solution approach used. The contributions of this research are; benchmark datasets indicative of the real world in terms of both constraints included and problem size, the data generators developed which enable the creation of data to investigate certain problem aspects, mathematical formulation of the multi period technician routing and scheduling problem, and, finally, the heuristics developed which have proved to be robust and scalable solution methodologies.

Table of contents

List of figures

List of tables

Chapter 1

Introduction

The right person, in the right place, at the right time. This simply-articulated goal underpins a variety of real-world and academic optimisation problems. These problems, commonly referred to as "technician and task scheduling problems", are intractable, no polynomial time algorithm is known to find the globally optimal solution. Efficient solution methodologies are needed for the discovery of high quality solutions in short computational times. Optimised scheduling has the potential to; reduce incurred labour costs, increase customer satisfaction, ensure repeat business, provide a fair workload for employees, and reduce the environmental impacts of vehicles used.

A technician and task scheduling problem in the simplest form requires a workforce to be allocated to complete a set of jobs that each require skills. An example of a technician and task scheduling problem solved in this thesis is the ROADEF 2007 challenge problem. This problem requires a set of jobs to be completed, where each job has skill requirements. The technician or team who services the job must possess the necessary levels of skills to complete it. The process of creating a team to service a job is referred to as "teaming" throughout this research. In addition, each job has a duration, the time it takes the technician or team to complete it. Jobs also have priority levels which is a numerical value that represents how important it is to serve this job early. A complicating factor of this problem is that there are some days when technicians are not available. There are also relationships present between jobs. These relationships are called precedence relationships where one job may not begin until another has been completed. Lastly, this problem also includes outsourcing. Outsourcing means giving a job to a third party to complete which incurs a financial cost. In some datasets of the ROADEF 2007 challenge an outsourcing budget is given which must be utilised. Choosing which jobs to outsource is itself an NP hard optimisation problem.

The sponsor of this research, Service Power PLC, is a leading field service scheduling provider. The aim of this research is to determine the complexities associated with technician and task scheduling problems and find efficient ways to handle them. Currently, research in the field is focusing on solving small scale problems for which exact solution methodologies may be used. In addition, many of these solution methodologies proposed have been tested on a single problem framework with no investigation into how well the heuristic could be adapted to solve closely related problems or the same at scale, such as industrially sized problems.

However, the research undertaken in this thesis is purely focused on heuristic solution methodologies, which can be used to solve real world problems of a realistic size and varying nature, in order to implement the findings into commercial software. This research aims to investigate the most common and complex aspects of technician and task scheduling problems, designing robust and scalable heuristic methodologies to find high quality solutions, whilst balancing time efficiency.

It is important to note that all experiments performed have used a HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ and 12GB of RAM. As hardware becomes more powerful, the number of iterations performed will increase, this can make it difficult to compare algorithms on a like for like basis. Where possible the issue of hardware and processing power has been addressed, by using an appropriate run times, and running heuristics on the same machinery.

1.1 Complexity Theory

The design and development of solution methodologies to solve optimisation/decision problems are inherently related to the complexity of the problem under consideration. Alan Turing proved that some problems are so hard they are “undecidable” (Turing, 1936). The example was "given a computer program and an input, could a polynomial time algorithm decide whether this code would terminate?". It concluded that there was no such algorithm known and therefore this problem is classified as "intractable".

It is important to note, that the complexity classifications are based on decision problems, not optimisation problems, which have a natural representation, where the answer is a "yes" or "no" outcome (Lenstra et al., 1977). However, each optimisation problem can be reduced to a decision problem which can then be classified using the complexity classes. There are four complexity classifications based on the assumption that P $\neq$ NP, as shown in Figure 1.1.

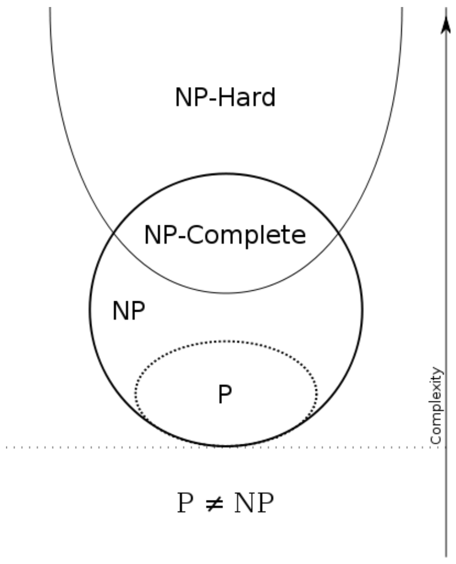


Figure. 1.1 Diagram showing the complexity classes for decision and optimisation problems

P

The complexity class, P, denotes the set of decision problems that can be solved within polynomial time on a deterministic Turing machine (Cook, 2006). An example of three problems which fall into this category, P, are: 1) finding the greatest common denominator between two or more values, 2) deciding whether a number N is prime or not and, finally, 3) finding the shortest path between two nodes. No matter the size of the problem under consideration, it can be solved to optimality when phrased as a decision problem.

The shortest path problem requires the shortest path between two nodes, A and B to be found, where the objective function is the sum of all traversed nodes. There are polynomial time algorithms to solve this problem such as Dijkstra's algorithm (Dijkstra, 1959) and improvements on the efficiency by Ahuja et al. (1990), Abraham et al. (2016), and Jain et al. (2016). The time it takes to solve these problems is a polynomial function of the problem size N. Both versions of the shortest path problem are presented, as an optimisation/search problem and as a decision problem.
Optimisation Problem : "Find the shortest path between nodes A and B"
Decision Problem : "Does there exist a feasible path between A and B of length X ?"

NP

The class NP denotes the set of decision problems that are solvable on a nondeterministic Turing Machine (Szelepcsényi, 1988) within some polynomial function of the problem size N. A problem that is NP is also verifiable on a yes instance, a feasible solution, by a deterministic polynomial time algorithm (Weisstein, 2017).

The implications of the complexity classes P and NP being equivalent is far reaching. If it could be shown that any problem belonging to NP can be solved to optimality using a deterministic polynomial time algorithm, then this would prove that any problem belonging to NP can also be solved, and indeed any problem belonging to the class NP complete could be solved also. This would mean that all problems would have efficient solution methodologies that could be implemented allowing for faster computation of solutions as the scale of problem increases. As expected, there has been much effort to both prove and disprove this theory with neither yet achieved.

NP hard

A problem x is only NP hard if there exists a decision problem y belonging to NP that is reducible to x in polynomial time. Therefore, any problem belonging to the class

NP hard is at least as hard as any problem belonging to NP. Optimisation problems which are NP hard have decision problems that are NP complete. It has been shown that the travelling salesman problem is an NP hard problem (Bonomi and Lutton, 1984). This problem requires a salesman to visit a set of N cities, each at different geographical locations in the shortest distance possible (Hoffman et al., 2013). The vehicle routing problem can be seen as a generalisation of the travelling salesman problem (Eksioglu et al., 2009), where a set of vehicles (salesmen) travel to visit a set of customers in the shortest cumulative distance (Laporte, 1992). Therefore, the vehicle routing problem is also an NP hard problem (Lenstra and Kan, 1981). The vehicle routing problem was first studied by Clarke and Wright (1964) and has been studied extensively throughout the past decades (Beasley (1983), Fisher (1994), Montané and Galvao (2006), and Lalla-Ruiz et al. (2016)). Additionally, the technician and task scheduling problem is a generalisation of the vehicle routing problem where the set of technicians is heterogeneous in terms of skill and is therefore also NP hard. An example of both the optimisation problem and the decision problem for the travelling salesman problem is given.
Optimisation Problem : "Find the shortest tour for the salesman visiting all cities"
Decision Problem: "Does there exist a feasible tour of length X ?"

NP complete

The set of NP complete problems is the set of problems, which belong to NP, for which it is possible to reduce any other problem belonging to NP to this problem. The set of problems belonging to NP complete is both NP and NP hard i.e. the intersection. Problems which have been shown to be NP complete are the graph colouring problem (Jensen and Toft (2011) and Titiloye and Crispin (2011*a*)), 3 SAT (Gu et al., 1999), and the Hamiltonian problem (Garey et al., 1976). The graph colouring problem aims to find the smallest number of colours to colour a graph such that no two adjacent vertices are the same (Lewis, 2016). Again, both the decision and optimisation versions of the graph colouring problem are stated.

Optimisation Problem : "Find the smallest K colouring"

Decision Problem : "Does there exist a feasible colouring where k=3?"

The complexity classifications are based on the assumption that P $\neq$ NP. If P $=$ NP, this means that NP problems can be solved to optimality within polynomial time on a deterministic Turning Machine. This concept was first introduced by Stephen Cooke in 1971, who won the Alan Turing award in 1982 for significant contribution to the field of complexity theory (Cook, 1983) and subsequently a prize of $1,000,000 dollars as part of the Millennium problems (Carlson et al., 2006), has been offered for a correct proof on this matter (Cook, 2003).

In fact, most scheduling problems are NP-hard combinatorial optimisation problems, such as nurse rostering (Qu and He (2009) and Asta et al. (2016)), job shop scheduling (Hoogeveen et al. (1996) and Panwalkar et al. (2016)), and unmanned aerial vehicle scheduling (Drucker et al., 2010). This means that there are no known polynomial time algorithms that can solve these problems to optimality within a discrete time period (Krishnamoorthy et al., 2012). In the literature, exact techniques which examine every possible configuration, are used to solve technician and task scheduling problems that contain scheduling up to 29 jobs. For this reason, exact techniques are prohibitive for many real world and large scale scheduling problem instances and approximate techniques must be used (Blum and Roli, 2003). Approximate techniques have no guarantee of finding the optimal solution, but generally are able to find quality solutions in short computational times (Rabadi, 2016).

1.2 Research Plan

Research Aim

What are the complexities associated with technician and task scheduling problems and how can they be dealt with effectively?

Objectives

The research question will be answered by completing the following objectives.

Objective 1: Undertake a review of approaches for solving technician and task scheduling problems with an emphasis on problem definition, available datasets, heuristic, and metaheuristic methodologies.

Objective 2: Design and develop heuristic methodologies to solve the technician and task scheduling problem and evaluate the performance of the heuristics against other approaches in the literature.

Objective 3: Investigate the constraints and complexities commonly associated with technician and task scheduling problems and strategies to overcome them.

Objective 4: Implement a range of trajectory metaheuristics to solve variants of the technician and task scheduling problem and devise approaches for parameter tuning.

Objective 5: Develop multi-period technician routing and scheduling problem instances and implement the heuristic methodologies developed to test their robustness handling extra constraints. Empirically evaluate all of the heuristic implementations.

Objective 6: Develop a set of large scale technician and task scheduling problem instances to test the scalability of the heuristics developed.

1.3 Publications

The following papers have been published/submitted in connection with this research, full papers can be found in Appendix A-D.

1. Khalfay A., Crispin A., Crockett K. (2016) Solving Technician and Task Scheduling Problems with an Intelligent Decision Heuristic. In: Czarnowski I., Caballero A., Howlett R., Jain L. (eds) Intelligent Decision Technologies 2016. Smart

Innovation, Systems and Technologies, vol 56. Springer, Cham, *Awarded Best Student Paper*, https://doi.org/10.1007/978-3-319-39630-9-6.

2. "A Review of Technician and Task Scheduling Problems, Datasets and Solution Approaches" Amy Khalfay, Alan Crispin and Keeley Crockett, In: Intelligent Systems Conference, IntelliSys September 7th-8th 2017, London. *In Press*

3. Khalfay A., Crispin A., Crockett K. (2018) Applying the Intelligent Decision Heuristic to Solve Large Scale Technician and Task Scheduling Problems. In: Czarnowski I., Howlett R., Jain L. (eds) Intelligent Decision Technologies 2017. IDT 2017. Smart Innovation, Systems and Technologies, vol 72. Springer, Cham, https://doi.org/10.1007/978-3-319-59421-7-7

4. "Solving the Service Technician Routing and Scheduling Problem with Time Windows" Amy Khalfay, Alan Crispin and Keeley Crockett, Submitted To: Operations Research, Springer, April 2017. *Under Review*

1.4 Contributions to Knowledge

1. **A review of the constraints, datasets and solution approaches used in the field of technician and task scheduling problems (Chapter 2).** The review highlights the need to design and develop approximate solution approaches that are both robust and scalable to solve these real world problems that occur in a range of industrial settings. The review demonstrates that previous solution approaches have not been applied to multiple problems, of varying nature and size, in order to prove their efficiency/ability to be applied in commercial settings.

2. **The intelligent decision heuristic characterised by its ability to consider multiple seed jobs and possible team configurations simultaneously (Chapters 3 to 6).** The heuristic considers the utilisation of each possible team by considering the potential further allocations, before making an allocation decision. This heuristic has been used to solve a diverse range of problems in this research

such as the technician and task scheduling problem, the large scale technician and task scheduling problem, the precedence constrained technician and task scheduling problem and, lastly, the multi-period technician routing and scheduling problem. The heuristic has produced competitive results on each set of data tested and proved its validity as a solution approach.

3. **A look ahead heuristic that has a preprocessing phase to calculate the underlying indirect precedence relationships present between jobs (Chapters 3 to 6).** This heuristic considers the subsequent impact of an allocation decision to the scheduling process in regards to the idle teams which are left and the allocations they may be given. This heuristic has also been applied to a range of problems: the technician and task scheduling problem, the large scale technician and task scheduling problem, the precedence constrained technician and task scheduling problem and, lastly, the multi-period technician routing and scheduling problem. The look ahead heuristic has proved to be both a robust (handling extra constraints such as location and travel time) and scalable (problem size) solution approach and has generally outperformed the intelligent decision heuristic.

4. **A data generator which has been designed and developed in order to generate technician and task scheduling problems (Chapters 4 to 6).** In this research, novel datasets have been generated in order to explore certain aspects of the problems, which occur in the real world but are not featured within datasets available in the literature. In this thesis, 12 new large scale technician and task scheduling problems, 25 technician and task scheduling datasets containing varying levels of precedence constraints and 30 multi-period technician routing and scheduling problem instances have been created. These datasets have addressed the problems highlighted in Chapter 2, the need for more datasets featuring a range of constraints and problem sizes available publicly to researchers, for further investigation into the field.

5. **New mathematical formulation of the multi-period technician routing and scheduling problem (Chapter 6).** Due to the extension of the ROADEF 2007 challenge problem which now contains location and travel time constraints, the mathematical formulation had to be updated to account for the extra complexities added. In this thesis, a mathematical formulation for the multi-period technician routing and scheduling problem is described. This is a novel problem in the field as it most importantly includes scheduling over multiple days, constructing teams subject to technician unavailability, and scheduling over a geographical area. This problem also includes other constraints such as priority levels, skill requirements, and outsourcing.

6. **The greedy randomized heuristic which is a flexible scheduling approach with multiple allocation criteria (Chapter 7).** This heuristic is the first sequential heuristic to be tested on the service technician routing and scheduling problem with time windows Kovacs et al. (2012), a single day problem extended from vehicle routing instances. This heuristic was able to find new best known results in 18% of the 72 data instances tested. This is an impressive result due to the complexity of the problems under consideration which include routing, skills, time windows and outsourcing, and the datasets are subject to very short computational run times of less than 90 seconds.

1.5 Thesis Overview

Chapter 2

In Chapter 2, a literature review is presented based on the field of technician and task scheduling problems. This chapter focuses on three key areas; the problem definitions that have been proposed and constraints which were included, the datasets that have been studied and the solution approaches applied. The current gaps in the field are also identified, which form the foundation of this thesis, which includes solving large scale problems, multi day problems, precedence constrained problems, and routing with medium scale datasets. Lastly, the effectiveness of proposed heuristics solving a single type of technician and task scheduling problem has been undertaken numerous times, however, there has been no research conducted that investigates how well the heuristic approach can be adapted in order to solve closely related problems or the same problem at scale. Therefore, the main aim of this research is to explore the common constraints associated with technician and task scheduling problems but also to analyse how well heuristic approaches can be adapted to solve problems with varying frameworks, or sizes.

Chapter 3

Chapter 3 introduces the ROADEF 2007 challenge problem, which is a technician and task scheduling problem, from real world data. The datasets come from a telecommunications company, named France Telecom, who wished to limit the growth of their workforce, whilst providing a high quality service, and reducing their operational costs. Two approaches were developed and applied to solve the 30 datasets, the intelligent decision heuristic, and the look ahead heuristic. The first approach, the intelligent decision heuristic, characterised by its ability to evaluate multiple scenarios, performed well on the majority of datasets but did struggle on datasets with a high level of precedence relationships and skill sparsity amongst technicians. A second approach was then designed, the look ahead heuristic, which focused on the impact/consequences of decisions made during the scheduling process. The look ahead heuristic performed well on all instances, matching the performance of other heuristic approaches in the literature and outperforming the intelligent decision heuristic overall.

Chapter 4

Chapter 4 focuses on solving a set of large scale technician and task scheduling problems. Large organisations, such as the sponsor of this research, Service Power PLC, face scheduling problems where there are hundreds of employees to schedule and thousands of jobs to allocate. For this reason, scalable solution approaches are needed that can tackle large problems efficiently in time constrained scenarios. A set of large scale instances has been created, using the data generator (which was designed), as there is a lack of large scale data available in the literature to test the performance of heuristics. These datasets have been generated under the framework of the ROADEF 2007 challenge, and contain scheduling up to 2500 jobs whilst adhering to many constraints. The intelligent decision and look ahead heuristic, featured in the previous chapter, Chapter 3, were then implemented to solve these large scale problems in order to ascertain whether the heuristics developed are scalable approaches when compared to a simple greedy heuristic (created for comparative purposes).

Chapter 5

Using the data generator, developed in Chapter 4, another set of technician and task scheduling problems were created, the precedence constrained technician and task scheduling problems. Precedence constraints arise in many sectors where tasks must be performed in a certain sequence, such as the housing development trade, utility repair services, and telecommunications. However, the occurrence of precedence constraints within technician and task scheduling problems has remained largely unstudied. The aim of this chapter is to investigate the effect that precedence constraints have on the quality of solution that can be obtained using different heuristic approaches and use the results for forecasting and predictions. The datasets developed contain varying percentages of precedence levels within the same set of jobs. To do this, a precedence algorithm was developed that made multi-level precedence relationships between jobs. Again, the approaches discussed in previous chapters are implemented, the intelligent decision, the look ahead and the greedy heuristic to test the robustness of the heuristics.

Chapter 6

A constraint that has not yet been featured in the problems studied so far was location and travel time. These are important considerations since most organisations, including the sponsor of this research, are required to send out their employees to the client to complete the job. To generate problems that include location, this constraint was added on to the ROADEF 2007 challenge problem as it is a multi-day problem with up to 800 jobs to allocate with many other constraints to satisfy. Chapter 6 introduces the multi-period technician routing and scheduling problem and provides the mathematical formulation of this problem. Each job was randomly given a location and a depot was created that technicians depart from and return to. To solve these datasets, a modified look ahead heuristic, intelligent decision heuristic, and greedy heuristic have been implemented, providing a performance analysis and an assessment of the robustness of the heuristics.

Chapter 7

The last stage of this research solved a technician and task scheduling problem from the literature, the service technician routing and scheduling problem with time windows in Chapter 7. This problem was adapted from vehicle routing instances by Kovacs et al. (2012) and is concerned with scheduling 100 jobs over a single day. This problem included not only the complexity of location and travel time but also time windows in which customers must be visited, an emerging constraint in the field of technician and task scheduling problems. Due to the short computational times, a greedy randomized heuristic, with a flexible allocation criterion, was developed which was coupled with a simulated annealing with restart metaheuristic. The heuristic performed well on these datasets finding new best known results in 18% of data instances.

Chapter 8

Chapter 8 provides a discussion of each of the problems presented, solution approaches implemented and tuning experiments performed in Chapters 3 to 7. This research has focused on five problems; the technician and task scheduling problem, the large scale technician and task scheduling problem, the precedence constrained technician and task scheduling problem, the multi-period technician routing and scheduling problem and, lastly, the service technician routing and scheduling problem with time windows.

Some of the problems have been taken from the literature, others adapted to increase the problem complexity and others developed with a data generator due to a lack of available datasets in the literature. A wide range of constraints has been included in the problems studied in this research, which commonly arises in real world settings. For each problem featured, a comparative analysis of the heuristic performance on the datasets is presented, as well as insight into the robustness and scalability of the heuristic approach.

Chapter 9

Lastly, Chapter 9 provides an overall conclusion about the research undertaken in this thesis. The original aim of this research was to investigate the complexities which arise in the field of technician and task scheduling problems. This research has shown there is an infinite list of complexities that arise in this area due to the real world nature of the problems under consideration. This research has studied some of the most common constraints and complexities featured by studying a range of technician and task scheduling problems and indicates directions for further research. In addition, three new sets of data have been generated, available publicly to researchers, which address some of the key challenges faced. The main challenges that have been identified and explored in this thesis are; solving large scale problems, problems including precedence constraints, problems including location and travel time, problems which involve teaming, and scheduling over multiple days subject to unavailability. Further areas for investigation are also identified such as the inclusion of mandatory breaks and tools and spare parts.

Chapter 2

Literature Review and Associated Technical Background

2.1 Introduction

This chapter presents a review of the current literature based in the field of technician and task scheduling problems. These problems are types of personnel scheduling problems (Castillo-Salazar et al., 2016), characterised by a heterogeneous workforce, and tasks with varying requirements. These problems approximate a number of real-world personnel scheduling problems faced by large organisations (Van den Bergh et al., 2013), with previous research effort studying personnel scheduling problems faced by Sears (Weigel and Cao, 1999), France Telecom (Fırat and Hurkens, 2012), and British Telecom (Lesaint et al. (2000) and Reid et al. (2016)). In addition, there has been some investigation of problems inspired by industry sectors rather than specific companies: Rasmussen et al. (2012) and Rest and Hirsch (2016) reported on work targeted at the home health care sector and Cortés et al. (2014) studied a problem based on maintenance technician dispatching.

Technician and task scheduling problems, as mentioned above, differ from classical personnel scheduling problems in both the available workforce and the tasks to be scheduled. Individual employees in the workforce possess quantified levels of skill across a set of domains (areas of expertise). Tasks impose specific requirements in the number of employees needed with specific skill levels in the same domains. Thus, problem instances may exhibit significant skill shortages or overabundances. Technician and task scheduling problems have been identified in: utility services (Chen et al., 2017), security patrols (Mısır et al. (2015) and Leigh et al. (2016)), gas and electrical services, and general maintenance (Castillo-Salazar et al., 2012).

These problems are NP hard combinatorial optimisation problems. Finding even a feasible solution is a complex task, let alone controlling navigation through the search space to find optimal solutions. In fact, these problems are actually composed of several optimisation subproblems, such as task assignment and tour scheduling (Alsheddy and Tsang, 2011), with overall optimal solutions requiring trade-offs between the subproblems. Typically, the problems' objective is to schedule all tasks in the most time-efficient and least costly manner.

Efficient handling of technician and task scheduling problems has many benefits to the organisations that face them, such as reduced operational costs and improved client satisfaction. Research in the field has focused on many different problem formulations (Fırat and Hurkens (2012), Pillac et al. (2012), and Mathlouthi et al. (2016)) and both exact and heuristic approaches have been used to solve them (Hashimoto et al. (2011) and Zamorano and Stolletz (2017)). However, to date, there has been little research effort in identifying common constraints in these problems, quantifying the complexity added by these constraints, or exploring efficient techniques for handling them.

Research in the field is somewhat fragmented, with most work reporting on the suitability of a single approach to solving a single problem (Castillo-Salazar et al., 2012). Typically, the data used in demonstrating the suitability of the approaches are small scale (Mathlouthi et al. (2016)), and it remains unclear whether the results of experiments using the reported approaches are similar on closely-related problems, or even larger datasets on the same problem. As a result, industries or companies faced with technician and task scheduling problems remain dubious as to the applicability of approaches from the literature to solve their wide ranging real-world problems and more generalised solution methodologies are needed (Ernst et al., 2004).

This remainder of this chapter is structured as follows; section 2.2 describes the constraints that are usually associated with technician and task scheduling problems in order highlight which constraints are used most frequently, and which constraints need further investigation. Section 2.3 presents the technician and task scheduling problem datasets available in the field to identify if there are any additional constraints to be considered or constraints that have not been studied simultaneously. Section 2.4 discusses the solution approaches used to solve technician and task scheduling problems to highlight promising solution approaches that have been applied to solve technician and task scheduling problems and discusses their scalability and robustness. Next, section 2.5 describes the metaheuristics used and section 2.6 identifies the current gaps and limitations within the literature. Section 2.7 discusses related scheduling problems in the literature, and lastly, section 2.8 concludes on key areas for research in the field of technician and task scheduling problems to benefit not only the research community but industry also.

2.2 Constraints

Many constraints have been featured in technician and task scheduling problems. The following subsections will explain each of the constraints included throughout the literature in the variations of the technician and task scheduling problem that have been studied.

Skill Requirements

A significant feature of technician and task scheduling problems are the occurrence of skill requirements. Usually, tasks will impose specific demands of skill that must be met in order to be completed. In most scheduling problems in the literature, there are domains, which are areas of expertise, and levels within the domains which classify proficiency. Skills are usually hierarchical, if a technician is skilled to level 3 in a particular domain, then the technician is also skilled at levels 1 and 2 as well. The set of technicians is heterogeneous, includes members who have varying skill levels, i.e. highly skilled and lower skilled workers.

Most technician and task scheduling problems studied in the literature include skill requirements such as Society (2007), Kovacs et al. (2012), Pillac et al. (2012). Pillac, Gueret and Medaglia (2013), Cortés et al. (2014), Mathlouthi et al. (2016), Chen et al. (2016), and Zamorano and Stolletz (2017). One problem that does not include skill requirements as hard constraints is a problem studied by Xu and Chiu (2001).

Routing

The complexity of routing is an important aspect of many technician and task scheduling problems. In most settings, the set of customers will each have a location, and so skilled personnel must travel to these locations to service customers. The occurrence of routing also usually implies that there is a central depot, from which skilled personnel depart from and return to at the beginning and end of each day. The travel times are accounted for between these locations and the depot, and are usually calculated as Euclidean distance. However, in many practical situations, travel time can be variable dependent on the day of the week or the time of the day that the journey between two customers is made. In addition, the use of a SAT NAV can also estimate the journey time which takes into account traffic incidents, roadworks, and congestion.

Problems featuring the complexity of routing in the literature include Pillac, Gueret and Medaglia (2013), Mathlouthi et al. (2016), and Chen et al. (2016) who studied technician routing and scheduling problems, Kovacs et al. (2012) who studied a service technician routing and scheduling problem, Tricoire et al. (2013) who studied the multi-period field service routing problem, and Zamorano and Stolletz (2017) who studied the multi-period technician routing and scheduling problem.

Pillac, Gueret and Medaglia (2013) studied instances extended from Solomon (1987) that included scheduling up to 100 jobs, which was validated using vehicle routing instances. The heuristic performed well achieving an average gap from optimal results of 0.23% on the vehicle routing instances and provided a set of benchmark results on the proposed instances. Smaller instances were studied in Tricoire et al. (2013), Mathlouthi et al. (2016), Chen et al. (2016), and Zamorano and Stolletz (2017) emphasizing the inability of exact methods to solve realistic sized problems which occur in the real world. In addition, Kovacs et al. (2012) also extended vehicle routing instances by Solomon (1987), containing 100 jobs, meaning the problem only consisted of scheduling a crew over a single day, but did include the complexity of teaming.

Teaming

Building teams is also an important constraint, particularly in the utility and service maintenance sector. Some jobs require skills which cannot be fulfilled by a single skilled worker. In addition, in some industries such as the building trade, it is prohibited that workers work alone on a site because of the Management of Health and Safety at Work Regulations 1999. Therefore, a team must be made where the cumulative skills of the team satisfy the job's skill requirements. In most problems featuring the complexity of teaming in the literature, the team is configured at the beginning of the working day and will stay together for the whole day.

For this reason, team formation decisions require much consideration in order to make sure the team's skills are utilised. In the real world, a manual planner may also take into account other information such as knowing which technicians work well together. There are few problems in the existing literature featuring the complexity of teaming, but some works include the ROADEF 2007 challenge (Society, 2007) which included real world problem instances and a service technician routing and scheduling problem studied by Kovacs et al. (2012) adapted from vehicle routing problem data.

Single Period/ Multi-Period

There are also differences in the size of the scheduling horizon, the number of working days available used in technician and task scheduling problems. Some problems in the literature have been adapted from vehicle routing problems and so, have only one scheduling day, because in the vehicle routing problem the objective is to minimise the total distance travelled on a single day whilst serving all customers. However, there are some multi-period problems in the field of technician and task scheduling, where typically the schedule can cover over a month of working days (Society, 2007). Consequently, consideration must then be given to the acceptable amount of time allowed to find feasible and high quality solutions. Typically, a schedule that will cover a month should be given more computational time than a single day problem as the impact of a bad solution will be more costly to the organisation.

Single day problems have been studied by Pillac, Gueret and Medaglia (2013) and Kovacs et al. (2012) as the instances were adapted from vehicle routing problems. The ROADEF 2007 challenge problem was a multi-period problem (Dutot et al., 2006) including scheduling up to 800 jobs. Multi-period problems were also studied by Mathlouthi et al. (2016) and Zamorano and Stolletz (2017) although on small scale instances containing no more than 30 jobs to schedule due to the exact solution approaches used.

Time Windows

Time windows are an emerging constraint within the field of technician and task scheduling problems. Service providers are seeking to differentiate themselves from each other in a competitive market to maintain market share. Allowing a customer to choose their own preferred time slot may result in a better customer experience, higher rates of customer satisfaction, and in turn produce repeat business.

Time windows have been included in technician and task scheduling problems studied by Pillac, Gueret and Medaglia (2013), Kovacs et al. (2012), Tricoire et al. (2013), Mathlouthi et al. (2016), and Zamorano and Stolletz (2017). The size of the time window can vary depending on the industry. Time windows may span half a day, a few hours or, in some cases, a single hour. In addition, there may be different levels of time windows present within a set of jobs, in Kovacs et al. (2012) the datasets contained either 50% or 100% time windows.

Precedence Relationships

Precedence and successor relationships are also a common occurrence in many industries. A precedence relationship between jobs i and i' implies that job i' may not begin until job i has been completed. A job i' is a successor of i, and i precedes i'. Precedence and successor constraints occur in sectors such as utilities, electrical maintenance, and housing projects. For example, in housing projects, the decorations may not begin until all the plastering is completed, which cannot be done until the brickwork is complete, which is dependent on the ground work being ready. In addition, in utility services fixing a fault may not begin until the road has been dug up and pipes can be accessed, which cannot be done until the correct equipment is on the site. Furthermore, chains of precedence relationships can develop between a set of jobs which adds to the complexity of the scheduling problem.

Precedence constraints have been used in technician and task scheduling problems such as the ROADEF 2007 challenge, which was tackled by Korteweg (2007) and Fırat and Hurkens (2012), and in Zamorano and Stolletz (2017). There is has been little investigation into the effect that precedence constraints have on the quality of solution that can be obtained. Precedence constraints have also featured in related scheduling problems, such as unmanned aerial vehicle scheduling (Park et al., 2016).

Priority Levels

Priority levels are also included in some technician and task scheduling problems. A priority level quantifies how important it is to service a job as early as possible in comparison to other jobs. These scenarios can arise where there are customers who place a lot of orders to a business and so they will be seen as more important than customers with smaller orders. Priority levels can also be used to classify the seriousness of a fault in the maintenance sector, for example, the water being off in a town or city compared to a minor leak in a single customer's house.

Priority levels have been included in the ROADEF 2007 challenge (Society, 2007) and research by Mathlouthi et al. (2016). Priority levels usually infer that the objective function will be a weighted sum of priority group end times. In addition, priority levels occur in other industries such as the emergency services: ambulances, police, and fire (Defraeye and Van Nieuwenhuyse, 2016).

Tools and Spare Parts

Tools and spare parts are an important complexity in the maintenance and repair sector. Generally, there will be a finite number of tools that must be shared between the technicians. A complicating factor is that the tools are usually located at a central depot, and so may need to be collected and returned. Furthermore, spare parts are non-replenishable, and therefore consideration must be given to stock levels at the central depot and perhaps even nearby DIY shops. A problem where technicians began their routes with a set of tools and spare parts was studied by Pillac, Gueret and Medaglia (2013). Furthermore, an aviation problem also included the collection and drop off of equipment needed to complete tasks (Erkoc and Ertogral, 2016).

Technician Unavailability

In some problems, there are also days or times when some technicians may not be available for work. This is also an emerging constraint in the wider field of personnel scheduling problems, as in law proper consideration must now be given to people with special circumstances in relation to the working patterns and hours, under the Employment Rights Act 1996 and Flexible Working Regulations 2014.

Unavailability of resources particularly affects the process of building a team as all members must be available and there may be a trade off between sending someone over qualified against waiting for a technician to become available. Most work does not consider the unavailability of resources (Van den Bergh et al., 2013), however, the ROADEF 2007 challenge does (Estellon et al., 2009).

Dynamic Arrivals

Dynamic arrivals is another situation that occurs in some industries. A dynamic job is a job that will need to be fitted into the schedule and arrives in real time. It is common in the water, telecommunications, electricity, and gas services, where there are faults that must be fixed quickly as there may be hundreds, perhaps thousands, of customers affected. In addition, the emergency services is also a sector where jobs continually arrive and must be allocated. When emergency jobs arrive in real time, any jobs that have begun are fixed in position. Only jobs that have not yet begun can be moved in order to fit in the emergency job. A dynamic scheduling problem was studied by (Lesaint et al., 2000) and by Pillac et al. (2012) on a telecommunications problem.

Pillac et al. (2012) demonstrated that when new requests appear it is better to construct a new schedule rather than try to reoptimise and fit in the dynamic job, 56 problem instances were tested, again adapted from Solomon (1987). In addition, research by Lesaint et al. (2000) which studied the BT scheduling problem, included scheduling over a large geographical area and has saved significant amounts of operational costs since implementation.

Summary

Section 2.2 has described the individual constraints that are usually associated with technician and task scheduling problems. As demonstrated, there are many constraints some of which are interdependent. For example, the constraint of technician unavailability occurs only in multi day problems. Time windows and tools usually occur alongside routing constraints as the technician may have to travel to collect the tool.

It seems sensible to focus research on the key constraints that arise simultaneously in the real world and to focus on solving problems with constraints or properties that have not been studied before. It appears there have been no problems investigated in the literature studied which include routing, unavailability and teaming in multi day scheduling problems. In addition, there has been no investigation into the effect of precedence constraints on solution quality.

2.3 Datasets

In this section, the datasets available in the literature that are based on variations of the technician and task scheduling problem are presented. Some of the datasets for technician and task scheduling have actually been adapted from vehicle routing problem datasets proposed by Solomon (1987). Each problem contains a different set of constraints. For example, some problems include the complexity of teaming, routing, job relationships, priority levels, and tools and spare parts. Tables 2.1 and 2.2 show the datasets that have been studied in this field, along with the constraints each problem featured, the size of the problems, and the number of problem instances studied.

Technician and Task Scheduling Problem

A technician and task scheduling problem was the basis of the ROADEF 2007 challenge. The ROADEF challenge is a biennial competition proposed by the French Operational Research Society (Dutot et al., 2006), that invites researchers to compete to find efficient ways of solving combinatorial optimisation problems. In 2007, the problem was a technician and task scheduling problem and used real world datasets provided by France Telecom, containing 30 data instances ranging from 5 to 800 jobs to schedule and 5 to 150 technicians. The aim of the problem is to allocate a set of jobs over a scheduling horizon to a set of teams. Teams are made up of technicians, each with intrinsic skill domain levels and days within the scheduling horizon when they are not available. Each job has a priority level indicating how important it is to serve the job as early as possible. In some problem instances, there is an available outsourcing budget that can be used, which itself is an NP hard problem. Jobs can also have relationships with other jobs, which can be precedence or successor relationships, where a job may not begin until another has been completed. Jobs have skill requirements which must be satisfied by the team which serves the job.

Table 2.1 Table showing the technician and task scheduling problems and variations that have been studied in the literature part 1

Author	Problem	Constraints	Jobs	Instances
Society (2007)	Technician and task scheduling	Teaming Technician unavailability Priority levels Outsourcing Job relationships	800	30
Pillac, Gueret and Medaglia (2013)	Technician routing and scheduling	Routing Time windows Tools and spare parts	100	56
Kovacs et al. (2012)	Service technician routing and scheduling	Teaming Routing Time windows	100	144
Xu and Chiu (2001)	Field technician scheduling	Routing Time windows	1000	20
Zamorano and Stolletz (2017)	multi-period technician routing and scheduling	Routing Time windows Teaming	25/27	102

Table 2.2 Table showing the technician and task scheduling problems and variations that have been studied in the literature part 2

Author	Problem	Constraints	Jobs	Instances
Tricoire et al. (2013)	Multi-period field service routing	Routing Time windows Technician unavailability Breaks	100	10
Mathlouthi et al. (2016)	Multi-attribute technician routing and scheduling	Routing Time windows Tools and spare parts Priority	25	20
Pillac et al. (2012)	Dynamic technician routing and scheduling	Routing Time windows Tools and spare parts Dynamic	100	56
Cortés et al. (2014)	Real life technician dispatching	Routing Time windows	100	56

Technician Routing and Scheduling Problem

Technician routing and scheduling problems in the literature include studies by Kovacs et al. (2012) and Pillac, Gueret and Medaglia (2013). The work by Kovacs et al. (2012) extended the vehicle routing problem instances proposed by Solomon (1987) into a service technician routing and scheduling problem with time windows. The problem concatenated skill requirements for each job based on the skill requirements present in some of the ROADEF 2007 challenge datasets. In this problem, teams leave the depot and travel to service customers. This version of the problem negated some constraints that are present in the ROADEF 2007 challenge problem, such as precedence and successor relationships, priority levels, multiple days, technician unavailability, and outsourcing budgets, but did contain the complexity of routing and time windows.

The work by Pillac, Gueret and Medaglia (2013) extended the instances in Solomon (1987) by generating skills, tools and spare parts information randomly. This problem, the technician routing and scheduling problem, did not include the complexity of teaming or precedence relationships. However, this is the first work the author is aware of that included the complexity of tools and spare parts, an important aspect of service maintenance problems.

Field Technician Scheduling Problem

A field technician scheduling problem was proposed by Xu and Chiu (2001), where jobs had to be serviced at different locations within a time window. This research did not treat skill compatibility as a hard constraint (Paraskevopoulos et al., 2017) which is the under pinning constraint that makes a scheduling problem a technician and task scheduling problem. The objective was to maximise the number of served jobs with a predefined time period whilst minimising the cost of the workforce. This research tested problem instances with up to 1000 jobs, which is representative of the scale of real world problems that occur in the industry.

The industrial sponsor of the research presented in this thesis Service Power PLC faces the problem of solving large scale scheduling problems that include skill requirements and many other complexities. Finding efficient ways to solve these problems within reasonable computational times is of great importance.

Multi-period Technician Routing and Scheduling Problem

Mathlouthi et al. (2016) explored the multi-period technician routing and scheduling problem by generating datasets. The datasets contained up to 25 jobs and CPLEX, an industry standard software package, was used to solve the mixed integer programming model. The problem included complexities such as skill requirements, priority levels, time windows, breaks, and overtime. This work found that CPLEX could only solve all instances with 10 jobs to allocate within reasonable computational times. This paper demonstrated how the computational time rapidly increases with problem size and complexity, and the need for approximate scalable solution approaches for use in commercial settings.

Both artificial and real world datasets were used by Zamorano and Stolletz (2017). The artificial datasets contained up to 25 jobs, and the real data instances contained up to 27 jobs. This research again emphasized the difficulties faced with scalability and robustness of exact solution approaches and the need for hybridized approaches.

Multi-Period Field Service Routing Problem

The multi-period field service routing problem studied by Tricoire et al. (2013) used both exact and hybrid solution approaches. The datasets used in this research were artificial and contained two sets, C4 small instances up to 40 jobs and C1 up to 100 jobs, and contained the complexities of routing, time windows, unavailability and breaks. This paper highlighted that even after 7 days of computational time, the branch and price technique was not able to find the optimal solution for 2 out of 5 instances. However, when using some heuristic techniques within branch and bound, a feasible solution can be found for all instances within 24 hours of computational time, further highlighting the need for heuristic solution methodologies.

Dynamic Technician Routing and Scheduling Problem

A dynamic technician routing and scheduling problem was studied by Pillac et al. (2012). In this problem, new job requests appear as the schedule is implemented in real time. This is another aspect of a real world situation faced by the industry. These datasets were created extending vehicle routing problem instances from Solomon (1987), and so contain scheduling 100 jobs over a single day. This paper concluded by suggesting it is more beneficial to create a new schedule rather than to try and reoptimise the existing schedule once a new job arrives.

2.4 Solution Approaches

Many approaches have been applied to solve the technician and task scheduling problem and its variations. Both exact and approximate approaches have been used. Table 2.3 shows some of the most cited solution approaches that have been applied to technician and task scheduling problems and their variants. The following subsections will describe the most effective solution approaches that have been used in the field to solve technician and task scheduling problems.

Approximate Mixed Integer Programming

Approximate mixed integer programming techniques have been used by Hurkens (2009) and Fırat and Hurkens (2012) to solve the technician and task scheduling problem proposed by the ROADEF 2007 challenge (Society, 2007). In these approaches, a construction algorithm was designed that broke down the overall scheduling problem into smaller sub problems. These sub problems then used the CPLEX library to solve these smaller mixed integer programming problems. These approaches did not contain an improvement phase, but the construction algorithm produced high quality solutions and solved problems with up to 800 jobs in time bounded conditions. Fırat and Hurkens (2012) outperforms other approaches on the set X problem instances of the ROADEF 2007 challenge.

Table 2.3 Table illustrating the solution approaches applied for solving technician and task scheduling problems

Solution Approach	Author
Approximate Mixed Integer Programming	Hurkens (2009) Fırat and Hurkens (2012)
Adaptive Large Neighbourhood Search	Cordeau et al. (2010) Kovacs et al. (2012)
Greedy Randomized Adaptive Search Procedure	Hashimoto et al. (2011)
Local Search	Estellon et al. (2009) Tsang and Voudouris (1997)
Parallel Matheuristic	Pillac, Gueret and Medaglia (2013)
Branch and Price	Tricoire et al. (2013) Zamorano and Stolletz (2017) Cortés et al. (2014)
Mixed Integer Programming	Mathlouthi et al. (2016)

Adaptive Large Neighbourhood Search

Adaptive large neighbourhood search heuristics have been used by both Cordeau et al. (2010) and Kovacs et al. (2012) to solve a technician and task scheduling problem and a service technician routing and scheduling problem with time windows. The adaptive large neighbourhood search heuristic is based on the seminal work by Shaw (1998) with the large neighbourhood search heuristic. The adaptive large neighbourhood search uses several destroy and repair operators. A destroy operator removes a portion of the current solution, and the repair operator reinserts the removed tasks back into the solution. In adaptive large neighbourhood search, the effectiveness of each destroy and repair operator is tracked such that operators that have performed well so far are more likely to be selected, which will aid the search for quality solutions. Cordeau et al. (2010) ranked 2nd place in the ROADEF 2007 challenge, whilst Kovacs et al. (2012) provided a set of benchmark results on proposed problem instances.

Greedy Randomized Adaptive Search Procedure

A greedy randomized adaptive search procedure was used by Hashimoto et al. (2011) to solve the ROADEF 2007 challenge (Society, 2007). This approach proved to be a successful one, as Hashimoto et al. (2011) won 1st place in the student category of the ROADEF 2007 challenge. The greedy randomized adaptive search procedure comprises of two components, a greedy heuristic and then a local search phase. A solution is made using a greedy algorithm, iteratively selecting the best decision at each stage of the scheduling process. Local search is then used to try to improve the solution for a short amount of computational time. This process is run multiple times and the best solution found is recorded.

Local Search

Local search has been used by Estellon et al. (2009) to solve the ROADEF 2007 challenge. This heuristic came joint 2nd place with the adaptive large neighbourhood search used by Cordeau et al. (2010). In this heuristic, an initial solution is constructed using a greedy heuristic. Next, in the improvement phase, operators are iteratively applied in order to reduce the objective function, which is priority weighted. This approach was coupled with a hill climbing metaheuristic.

Mathheuristic

The mathheuirtsic is a novel heuristic proposed by Pillac, Gueret and Medaglia (2013). The heuristic was created in order to solve a technician routing and scheduling problem that included many constraints. The mathheuirtsic is composed of three phases, a construction algorithm, a parallel adaptive large neighbourhood search and a mathematical programming post optimisation phase. The parallel adaptive large neighbourhood search takes advantage of the parallel architectures which result in a significant computational speed up.

Branch and Price

Branch and price is an exact solution technique. This method has proved popular as many authors have used this technique: Tricoire et al. (2013), Cortés et al. (2014), and Zamorano and Stolletz (2017). Branch and price combines branch and bound with column generation techniques in order to solve large integer programming problems. However, exact solution approaches are only suitable for small sized problems, medium and large scale problems must be solved using approximate approaches due to the complexity and the need to solve these problems within reasonable computational time.

MIP Programming

Mixed integer programming has been used by Mathlouthi et al. (2016) to solve a multi skill technician routing and scheduling problem. Again, this solution approach is only suitable on small sized problems and so would not be suitable on an industrial scale. In mixed integer programming, variables are restricted to take integer values, and can be solved using commercial software such as CPLEX.

Summary

This section has discussed some of the solution approaches that have been applied to solve technician and task scheduling problems. It is obvious that exact techniques such as MIP programming or branch and price are only suitable approaches for solving small scale problems, which is problematic, as the situation that arises in the real world will be of a realistic size, and so these approaches would not be suitable. It is difficult to compare approaches against each other as not all solution approaches have been applied to solve the same problems. For example, research by Kovacs et al. (2012), Pillac et al. (2012), and Pillac, Gueret and Medaglia (2013) proposed new problem instances and so their approaches provided benchmark results. As the ROADEF 2007 challenge problem instances have been solved by multiple researchers, the results here can be compared. The approximate MIP approaches by Fırat and Hurkens (2012) and Hurkens (2009) performed overall best on the instances especially in the set X instances which are the most complex. Competitive results have also been achieved on these instances with Estellon et al. (2009), Cordeau et al. (2010), and Hashimoto et al. (2011). The research in this thesis will focus on designing and developing heuristic solution approaches, that are robust and are able to solve realistically sized problems whilst balancing time efficiency.

2.5 Metaheuristics

Most approximate solution approaches to solving technician and task scheduling problems are two phase. In the first phase, the construction phase, an initial solution is generated. In the second phase, the improvement phase, local operators are applied which perturb the current solution generating a neighbouring solution. This neighbouring solution has to be evaluated, using a cost function. According to Blum and Roli (2003), *"metaheuristics are high level strategies for exploring search spaces by using different methods"*. It is important to note that metaheuristics are not problem specific like heuristics are, they are tools. The following subsections will describe four popular trajectory metaheuristics: hill climbing, iterative local search, simulated annealing, and tabu search. The use of trajectory metaheuristics is popular in time constrained environments in such complex solution space. Each of these metaheuristics has been used successfully to solve a range of combinatorial optimisation problems.

Hill Climbing

Hill climbing is the most simple and easy to implement metaheuristic as there are no input parameters to tune. Figure 2.1 illustrates the hill climbing metaheuristic as descirbed in Minsky (1961). First, the initial solution generated using the construction heuristic is recorded as the best solution, S_{Best}, on line 1. On each iteration, a local operator o is selected randomly and applied to solution S which generates a neighbouring solution S' on line 4.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators

```
1: $S_{Best} \leftarrow S$
2: while termination criteria not met do
3:     $randomly \quad choose \quad o \in O$
4:     $S' \leftarrow o(S)$
5:     if $S' \leq S$ then
6:         $S \leftarrow S'$
7:         if $S \leq S_{Best}$ then
8:             $S_{Best} \leftarrow S$
9:         end if
10:    end if
11: end while
12: return $S_{Best}$
```

Figure. 2.1 Pseudocode showing the implementation of a hill climbing metaheuristic

If solution S' is of better quality than solution S, has a lower objective value, then it is accepted and becomes the new current solution S on line 6. If solution S is now of better quality than the best solution S_{Best}, then it replaces it on line 8. Once the termination criterion has been met, the best solution is output. Hill climbing has been proven to be successful in solving a range of combinatorial optimisation problems and was used by Estellon et al. (2009). One of the drawbacks of the hill climbing metaheuristic is that it does not incorporate an escape mechanism, it never accepts a worse solution. For this reason, it can get stuck in local optima.

Iterative Local Search

Iterative local search can be considered as an extension to the hill climbing metaheuristic as it does include an escape mechanism. Iterative local search is classed as a multi start technique, first proposed by (Lourenço et al., 2003) and contains a diversification quality (Martí et al., 2010). There are two parameters associated with this metaheuristic, the step size N and the $kickoperator$.

The implementation of iterative local search is shown in Figure 2.2. Again, the initial solution is saved as S_{Best} on line 1 and the variable *count* is assigned the value 0 on line 2. On each iteration, a local operator o is randomly selected and applied to solution S which generates neighbouring solution S' on line 5. If solution S' is of better quality than solution S then it is accepted and becomes the current solution S on line 7.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, N: maximum steps before restarting from best solution, *count*: counter of iterations

```
1:  $S_{Best} \leftarrow S$
2:  $count \leftarrow 0$
3:  while termination criteria not met do
4:      $randomly \quad choose \quad o \in O$
5:      $S' \leftarrow o(S)$
6:      if $S' \leq S$ then
7:          $S \leftarrow S'$
8:          if $S \leq S_{Best}$ then
9:              $S_{Best} \leftarrow S$
10:             $count \leftarrow 0$
11:         end if
12:     else
13:         $count \leftarrow count + 1$
14:     end if
15:     if $count = N$ then
16:         $S' \leftarrow kickop(S)$
17:         $S \leftarrow S'$
18:         $count \leftarrow 0$
19:     end if
20: end while
21: return $S_{Best}$
```

Figure. 2.2 Pseudocode illustrating an implemenation of an iterative local search meta-heuristic

If solution S is now of better quality than the best solution, then it replaces S_{Best} on line 9, and the *count* variable is reset to 0 as an improving move has been found. If the neighbouring solution is not of better quality then the variable *count* is incremented by one. On line 15 a check is performed, if the number of non improving moves, i.e. *count* is equal to the maximum step size N, then a *kickoperator* is applied to S generating S' on line 16. The solution S' then replaces S on line 17 to become the current solution (no matter the quality of the new solution) and the search will continue from this point in the solution space. Once the termination criterion has bet met the best solution is output on line 21. This feature of the iterative local search metaheuristic allows the algorithm to escape local minima when it becomes stuck in a non-improving loop. However, much care must be taken to control the strength of the $kickoperator$ and the number of non improving moves permitted. Iterative local search can help ensure that previously unvisited regions of the solution space are searched, and that computational time is not wasted progressing towards local optima.

Simulated Annealing

Simulated annealing is one of the most widely used metaheuristic techniques. This metaheuristic was first proposed by Kirkpatrick et al. (1983) and has shown to be able to produce quality results not only in the field of technician and task scheduling problems (Cordeau et al., 2010) but in other optimisation problems.

The process of annealing comes from the field of physics and involves the heating and controlled cooling of a metal in order to reduce its imperfections and form a smooth lattice structure. This concept was brought into the field of combinatorial optimisation to help guide heuristics through the search space to find global minima.

This metaheuristic has two parameters, the temperature T and the decrement δT. Simulated annealing has the ability to escape local optima, through the ability to accept a worse quality solution. The chance of accepting a worse quality solution is controlled by the temperature parameter. The simulated annealing metaheuristic is shown in Figure 2.3. On each iteration, a local operator is randomly selected and applied to solution S which generates S'. If S' is of better quality then it replaces S on line 6. If solution S is better than the best-found solution so far then it replaces S_{Best} on line 8. However, if solution S' is of worse quality than solution S then a probability p is calculated on line 12. The probability p is the negative exponential of the difference in solution quality divided by the temperature parameter. If the probability p is greater than a random number generated on the interval $\{0,1\}$ then S' is accepted and replaces S on line 14.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, T: multi-period technician routing andmperature, δT: the cooling rate

```
1: $S_{Best} \leftarrow S$
2: **while** termination criteria not met **do**
3:     $randomly \quad choose \quad o \in O$
4:     $S' \leftarrow o(S)$
5:     **if** $S' \leq S$ **then**
6:         $S \leftarrow S'$
7:         **if** $S \leq S_{Best}$ **then**
8:             $S_{Best} \leftarrow S$
9:         **end if**
10:    **else**
11:        $r \leftarrow random(0,1)$
12:        $p \leftarrow exp^(S' - S)/T$
13:        **if** $p \geq r$ **then**
14:            $S \leftarrow S'$
15:        **end if**
16:    **end if**
17:    $T \leftarrow T \cdot \delta T$
18: **end while**
19: return $S_{Best}$
```

Figure. 2.3 Pseudocode describing the implementation of the simulated annealing metaheuristic

After each iteration the temperature parameter is decremented using δT, reducing the likelihood of accepting a worse quality solution. Once the termination criterion has been met the best solution is output.

Tabu Search

Tabu search is another powerful trajectory metaheuristic and has been applied to many combinatorial optimisation problems. The main feature of tabu search is adaptive memory (Glover and Taillard, 1993). This heuristic approach was first used by Glover (1986) but since has included many adaptations and improvements to solve problems such as the vehicle routing problem (Gendreau et al., 1994), multi depot vehicle routing problem (Cordeau et al., 1997) and the travelling salesman problem (Gendreau et al., 1998).

The aim of this heuristic is to force the search into previously unvisited areas of the search space. A list of restricted solutions, usually the past N visited solutions, is kept which restricts the allowable moves to unvisited areas.

Variables: S: current solution, S': chosen neighbouring solution, S_{Best}: the best solution, $Restricted_{L}ist$: the list of previously visited moves, $sNeighbourhood$: list of possible moves, $CandidateList$ list of allowable moves

```
1: $S_{Best} \leftarrow S$
2: $RestrictedList$= null
3: while termination criteria not met do
4:   $CandidateList$=null
5:   for $sCandidate \in sNeighbourhood$ do
6:     if $sCandiate! \in RestrictedList$ then
7:       $CandidateList \leftarrow sCandiate$
8:     end if
9:   end for
10:  $S' \leftarrow bestsol(CandidateList)$
11:  if $S' \leq S_{Best}$ then
12:    $S_{Best} \leftarrow S'$
13:  end if
14:  Update $RestrictedList$
15: end while
16: return $S_{Best}$
```

Figure. 2.4 Pseudocode illustrating the tabu search metaheuristic

Figure 2.4 demonstrates a tabu search metaheuristic. On line 1, the solution generated by the construction heuristic is stored as S_{Best}, and the $RestrictedList$ is set to null. Whilst there is computational time remaining a $CandidateList$ is constructed, for each candidate solution in the neighbourhood of possible solutions, a check is performed on line 6. If the solution is not a member of the $RestrictedList$ then it is permitted join the $CandidateList$. On line 10, the best solution amongst the candidate solutions is selected as S'. If S' is better than S_{Best} then it replaces it on line 12. On line 14, the restricted list is updated, and once the computation time has been reached the best solution is output on line 16.

Comparison of Metaheuristics

The aim of a metaheuristic is to guide the lower level heuristic and so it seems natural that an analysis is made on the relative freedom and convergence properties of the techniques. One of the important qualities of a metaheuristic is the ability to escape local optima. In such complex problems, the solution space is full of peaks and troughs. It can be easy to end up in a trough and so, there is a need to be able to navigate through the solution space by accepting a worse solution in the hope of being able to find a better quality one. However, as is clear in the literature the amount of freedom given to exploring worse quality solutions must be weighed up against the amount of computational time remaining, the formulation of the objective function and the complexity of the problem under investigation.

For example, simulated annealing contains a probability at all times of accepting a worse quality solution that decreases over time. Iterative local search however at times transports the search space to another area when a period of non-improvement has occurred. Tabu search contains a memory aspect that forces the search away from previously visited solutions and towards unvisited solution space.

Some trajectory metaheuristic approaches have been discussed, however, there has been no mention of population based approaches such as genetic algorithms or quantum annealing. There are a few reasons for this, one of the characteristics of genetic algorithms is the cross over function, where two parent solutions join to form a child solution. The child solution will have some information from each parent. In the context of two scheduling solutions over a multi day horizon, with different team formations and job distributions, it would be extremely difficult to perform this (Aickelin and Dowsland, 2004). Often problem specific knowledge is needed to successfully implement a genetic algorithm on a complex optimisation problem, such as the use of a repair function (Aickelin and Dowsland, 2000). It has been attempted in the nurse rostering problem, where a repair function was used to ensure the child solution was valid, however, this research concluded that the computational expense outweighed its benefits (Kundu et al., 2008). Furthermore, using a population based approach is computationally expensive, especially on large scale problems (Tanomaru, 1995).

Quantum annealing has been used successfully in related optimisation problems such as the travelling salesman problem (Martoňák et al., 2004), graph colouring problem (Titiloye and Crispin, 2011*b*), and the vehicle routing problem (Crispin and Syrichas, 2013). In order to implement quantum annealing on the technician and task scheduling problem, a binary solution representation would have to be designed in order to perform the interactions, stating the team formations, job allocations and job order, across each day.

2.6 Discussion

As demonstrated, the field of technician and task scheduling problems is a vast research field that needs continuing investigation. There are a wealth of industrial applications which means that breakthroughs in this field have the potential to make both environmental and financial impacts in this and other closely related fields. Furthermore, as demonstrated in Burke et al. (2010) there is a need to develop solution approaches that *"work well, not only across different instances of the same problem but also across different problem domains"*.

The complexity of teaming has been studied on relatively small problem sizes, for example in Pillac, Gueret and Medaglia (2013) and Kovacs et al. (2012) the problem instances contain at most 100 jobs that were adapted from vehicle routing problems. Additionally, work by Gérard et al. (2016) concluded on the need to solve realistic problems that include teaming constraints. The ROADEF 2007 challenge does deal with teaming with larger instances, up to 800 jobs, but routing is not part of the problem definition. The complexity of routing has been studied in many problems, with usually up to 100 jobs. Routing was considered in Xu and Chiu (2001) with 1000 jobs, but skill compatibility was not treated as a hard constraint, a fundamental property of technician and task scheduling problems.

Precedence constraints are featured in the ROADEF 2007 challenge but are absent from many other problems that have been studied. Precedence constraints can occur in many other application areas such as home health care. In the field of technician and task scheduling problems, precedence constraints are featured in service maintenance, housing developments, utility, and electrical services problems. Precedence constraints add significant complexity to the problem and may influence the design of a solution approach. It is vital that research is undertaken that considers precedence constraints because they can occur in many sectors and pose significant challenges when designing a solution approach.

Furthermore, the largest problems studied have included up to 1000 jobs in Xu and Chiu (2001) although skill was not treated as a hard constraint. A problem with up to 800 jobs was studied in Society (2007). However, in most problems up to 100 jobs are considered due to the datasets which have been developed from single day vehicle routing problems. In many industrial scenarios, there will be many jobs to schedule over a larger geographical area and realistically sized problem instances are needed in order to validate solution approaches and assess their suitability for commercial purposes.

The lack of multi-period problems has also meant that the complexity of the unavailability of resources needs further research and investigation. Typically many technician and task scheduling problems and their variants are either adapted from vehicle routing problem instances or are artificial. The use of exact solution approaches has also prohibited the use of multi-period problems as the problem size has had to be kept relatively small for computational time. The unavailability of resources is also directly linked to the complexity of teaming, as all team members must be available to join the team. In large organisations, it is necessary to schedule a workforce over multiple days, accounting for technician unavailability and, in some organisations, create teams to complete jobs.

It appears that research in the field would benefit from being focused on multi-period problems, large scale problems, technician unavailability, teaming and precedence constraints.

2.7 Related Personnel Scheduling Problems

The field of personnel scheduling is an interrelated one. There are many types of problems, which can be classified into groups such as technician and task, travelling salesman (Barketau and Pesch, 2016), vehicle routing (Bektaş et al., 2016), nurse rostering (Santos et al., 2016) and home healthcare (Castillo-Salazar et al., 2016). Within each of these groups, there are many variations of each problem, including different collections of constraints/complexities. There are many commonalities and differences between the groups of scheduling problems, which are discussed in further detail in the following subsections.

Travelling Salesman Problem

The travelling salesman problem is one of the most widely studied combinatorial optimisation problems since first being studied by Dantzig et al. (1954). It requires a tour to be designed for a salesman such that all cities are visited in the shortest distance possible (Held and Karp, 1970). Many solution approaches have been applied to this problem: guided local search (Voudouris and Tsang, 1999), branch and cut (Fischetti et al., 1997), ant colony optimisation (Dorigo and Gambardella, 1997), tabu search (Li and Alidaee, 2016), genetic approaches (Kang et al., 2016), and hybrid heuristic (Hernández-Pérez et al., 2016). The travelling salesman problem in comparison to the other groups of problems is less complex, as it includes scheduling one salesman, over a single day, with the objective function being calculated as the total distance travelled (usually Euclidean).

Vehicle Routing Problem

The most notable problem related to the technician and task scheduling problem is the vehicle routing problem, which evolved from the truck dispatching problem studied by Dantzig and Ramser (1959) in the 1950's. This problem has been studied extensively throughout the decades (Lenstra and Kan (1976), Golden and Yee (1979), Christofides et al. (1981), Paessens (1988), Osman (1993), Taillard et al. (1997), Golden et al. (2008), Szeto et al. (2011), and Bouzid et al. (2017)). The vehicle routing problem requires scheduling a fleet of vehicles to serve a set of customers (Crispin and Syrichas, 2013). Each vehicle leaves the depot and visits a subset of customers before returning to the depot. The constraints of the problem are that each customer is visited exactly once. The objective of the vehicle routing problem is to minimise the sum of the distance travelled by the vehicles (Baker and Ayechew, 2003). Usually, the Euclidean distance between customers is used. The vehicle routing problem is a generalisation of the travelling salesman problem, where instead of single salesman there are multiple vehicles and instead of serving all customers, each vehicle will serve a subset.

The vehicle routing problem has also been studied with various side constraints. Constraints such as the capacity of the vehicles (Fukasawa et al., 2006), multiple depots (Cordeau et al., 1997), and time windows in which customers must be visited (Solomon, 1987). The technician and task scheduling problem can be thought of as a generalisation of the vehicle routing problem. The main differences between these scheduling problems are the homogeneousness of the vehicles (capacity) in the vehicle routing problem compared to heterogeneousness of the workers (skill set) in the technician and task scheduling problem. The vehicle routing problem is usually concerned with making a schedule for a single day, whereas some technician and task scheduling problems require scheduling for multiple days. Many solution approaches have been applied to solve the vehicle routing problem such as tabu search (Badeau et al., 1997), bee colony optimisation (Szeto et al., 2011), variable neighbourhood search (Todosijević et al., 2017), and column generation (Desrochers et al. (1992) and Mahvash et al. (2017)). Comprehensive reviews based on vehicle routing problems and dynamic vehicle routing problems were undertaken by Lenstra and Kan (1981) and Pillac, Gendreau, Guéret and Medaglia (2013) respectively.

Nurse Rostering

The nurse rostering problem also belongs to the set of personnel scheduling problems. Unlike the other problems discussed, this problem typically includes various hard and soft constraints (Asensio-Cuesta et al., 2012). A hard constraint is one that must be satisfied in order for the solution to be feasible (Shi and Landa-Silva, 2016). A soft constraint is one which if possible is desirable to satisfy but if not a penalty cost may be incurred. One feature of the nurse rostering problem which makes it different from other scheduling problems is its cyclicness (Baker and Magazine, 1977), where solutions can be reapplied since the demand of a hospital is stable and known. Comprehensive surveys in this field have been undertaken by Sitompul and Randhawa (1989), Cheang et al. (2003), Burke et al. (2004), Wright et al. (2017), and Gartner and Padman (2017).

Home Healthcare Problem

Another related scheduling problem is the home healthcare problem. The home healthcare problem requires the scheduling of skilled personnel to travel to patients based in different locations to administer medication or provide care. This problem can be thought of as a generalisation of the vehicle routing problem with time windows, multiple depots, and compatibility constraints (Cheng and Rich, 1998). This problem is becoming more prevalent in society as the population ages and private companies begin to work in this area (Bertels and Fahle (2006) and Fikar and Hirsch (2017)). In fact, some local authorities outsource as much as 100% of home care (Akjiratikarl et al., 2007) and it is a rapidly growing industry (Rest and Hirsch, 2016). The main purpose of this problem is to allow elderly patients to be treated in their own homes for as long as possible (Rasmussen et al., 2012). The care given can be anything between cleaning and making food, to changing bandages and dressings, and administering medication. There are also many side constraints to be considered such as patient preference on the nurses who visit (Braekers et al., 2016) and travel time between visits dependent on the mode of transportation (walking, cycling, bus, tram, or car) (Hiermann et al., 2015). Recent approaches such as variable neighbourhood search (Pinheiro et al., 2016) have been implemented which provided a set of benchmark results on generated instances, genetic algorithms (Shi et al., 2017) which were tested on literature instances performing efficiently, and branch and price (Rasmussen et al., 2012) which used both real world and generated problem instances.

Summary

Figure 2.5 shows the commonalities between three of the personnel scheduling problems discussed: technician and task scheduling problems, vehicle routing problems and the home healthcare problem. It is evident that there are many shared constraints/complexities between these problems and also some notable differences. One of the notable differences between the vehicle routing problem and the other problems is that the vehicle routing problem is concerned with a single day rather than multiple days and contains a homogeneous workforce.

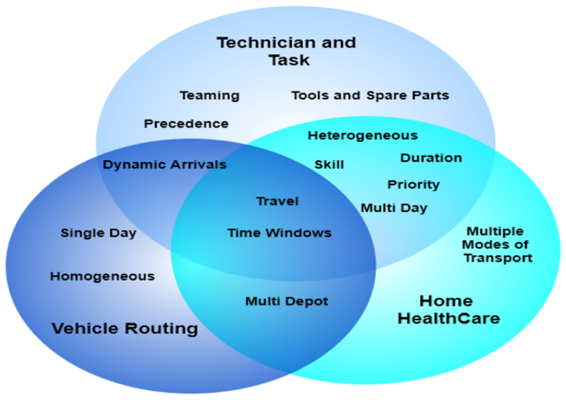


Figure. 2.5 Venn diagram showing the commonalities between three personnel scheduling problems, the technician and task scheduling problem, the vehicle routing problem and the home healthcare problem

As demonstrated in section 2.7 the technician and task scheduling problem is related to many other personnel scheduling problems. There are shared constraints between problems such as skill compatibility (assigning a worker/nurse who can perform/serve the job/patient), travel time (between locations of jobs/patients) and time windows (in which to serve a job or administer medication/care). It is, therefore, reasonable to assume that approaches to solving a technician and task scheduling problem could be adapted in order to solve the vehicle routing problem or the home healthcare problem since problems share common constraints and so research in the field of technician and task scheduling problems have the potential to impact other industries.

2.8 Conclusion

This chapter has given a comprehensive study of the field of technician and task scheduling problems and related personnel scheduling problems. A review of the constraints that have been associated with the technician and task scheduling problems and variants has been presented, and an analysis of which constraints have been studied simultaneously and which constraints need further investigation.

This review has also discussed the datasets used in the field which has highlighted some the limitations in the datasets available: such as the size of the scheduling problem (many problems have studied up to 100 jobs), the number of scheduling days (some datasets consider a single scheduling day only) and the source of the data (some problems have been adapted from vehicle routing problem instances).

In addition, the literature has illustrated that exact techniques are only appropriate for small scale problem instances. There is a need to develop heuristic approaches to solve industry sized problems that arise in the real world. The use of metaheuristics has also been discussed, it seems that most research in the field relies on trajectory methods for computational efficiency in time constrained environments.

In the next chapter the first scheduling problem addressed in this research is introduced, the technician and task scheduling problem, a real world problem, proposed by the ROADEF 2007 challenge.

Chapter 3

Technician and Task Scheduling Problems

3.1 Introduction

In this chapter, a technician and task scheduling problem is solved using two novel heuristic procedures designed and developed, the intelligent decision heuristic and the look ahead heuristic. A technician and task scheduling problem in the simplest form requires a set of technicians, each with different skills, to be assigned to complete jobs, which each require skills. As illustrated in chapter 2, there are many different variations of technician and task scheduling problems that have been studied. In this chapter, the technician and task scheduling problem chosen for investigation is the ROADEF 2007 challenge problem, as the datasets are from the real world and other researchers have already studied this problem.

The ROADEF 2007 challenge, organised by the French Operational Research Society, encouraged researchers to compete to find efficient ways of solving France Telecom's optimisation problem (Society, 2007). The aim of the ROADEF 2007 challenge problem is to construct a set of teams to service a set of jobs over a scheduling horizon $K = [1...k]$. A scheduling horizon is a collection of days that make up a solution. One of the interesting features of this problem is that it requires the workforce to be scheduled over multiple days. France Telecom wished to protect their market share and maintain a high level of customer service whilst limiting the growth of their workforce (Dutot et al., 2006). Characteristics featured in the ROADEF 2007 challenge problem are still applicable to the problems faced today in many organisations (Montoya et al., 2015).

The ROADEF 2007 challenge problem can be summarised as follows: each job has domain skill requirements that need a team to be built in order to satisfy the demand. A domain may be a particular area of expertise and a skill level will represent the proficiency within that domain. Teams are made up of technicians who have intrinsic skill domain levels and days where they are unavailable. In addition, there are also dependency relationships between jobs, prohibiting some jobs being started until others have been completed. Jobs also have a priority level, representing how important it is to serve the job as early as possible. Jobs have a completion time and must be started and finished on the same day. Furthermore, in some instances, there is an outsourcing budget available (outsourced jobs do not contribute to the objective function).

Technician and task scheduling problems are classified as NP-hard problems, there are no polynomial time algorithms known that can solve the problems optimally within reasonable computational time. Therefore, the ROADEF 2007 challenge problem is also NP hard, in fact even the sub problem of outsourcing jobs has been shown to be NP hard (Estellon et al., 2009). For this reason, the approaches proposed in the literature to solve the ROADEF 2007 challenge are all approximate approaches. The ROADEF 2007 challenge attracted a lot of research attention and a number of solution approaches such as; adaptive large neighbourhood search (Cordeau et al., 2010), approximate mixed integer programming (Fırat and Hurkens, 2012; Hurkens, 2009), local search heuristics (Dongala, 2006; Estellon et al., 2009), greedy algorithms (Jaskowski and Wasik, 2007; Pokutta and Stauffer, 2009), and greedy randomized adaptive search algorithms (Hashimoto et al., 2011) have been proposed.

Two new heuristic approaches have been designed in this research to solve the ROADEF 2007 challenge problem. The first approach, the intelligent decision heuristic can be categorised by its ability to assess multiple team configurations and job allocations at once before making a decision. The look ahead heuristic is the second approach that has been applied, that contains a preprocessing phase and considers the impact of an allocation decision on the idle teams and subsequent allocations that can be made.

The remainder of this chapter is organised as follows, section 3.2 presents the formulation of the ROADEF 2007 challenge problem and section 3.3 describes the real world datasets. Section 3.4 introduces the intelligent decision heuristic and computational experiments undertaken. Section 3.5 presents the look ahead heuristic and computational experiments. This chapter sets the foundation for the exploration of constraints associated with technician and task scheduling problems and shapes the design and development of heuristic approaches.

3.2 Problem Formulation

The mathematical formulation of the ROADEF 2007 challenge problem is given, this has already been stated in the literature but is given here just for completeness. Equations 3.1 to 3.20 describe the ROADEF 2007 challenge problem.

The aim of the ROADEF 2007 challenge problem is to construct a set of teams to service a set of jobs over a scheduling horizon $K = [1...k]$, where k is a day belonging to the scheduling horizon. Each day is of length 120 time units. Each job i belonging to set N has certain properties, a priority level p where $p \in [1...4]$, an execution time d_i, a domain skill requirement matrix $s^i_{\delta\alpha}$ (where δ is the domain and α is the skill level), an outsourcing cost c_i and a set of successor jobs σ_i. The set of teams is denoted by $M = [1...m]$, which are made up of technicians $T = [1...t]$.
The objective function set in the challenge is shown in Equation 3.1. The objective function is a weighted sum of the latest ending times, e_p, of each priority group where $w_p = [28, 14, 4, 1]$ for $p = [1, 2, 3, 4]$.

$$Minimize \sum_{p=1}^{4} w_p * e_p \tag{3.1}$$

The start times of jobs are denoted as b_i. Equation 3.2 ensures that the latest ending time for each priority group, $p \in [1...3]$, must be greater than, or equal to, the start time of every job plus the duration of the job.

$$e_p \geq b_i + d_i \qquad \forall p \in 1,2,3, i \in N_p \tag{3.2}$$

In addition, Equation 3.3 ensures the latest ending time overall e_4, is greater than, or equal to, the start time of every job plus the duration of every job belonging to the entire set of jobs.

$$e_4 \geq b_i + d_i \quad \forall i \in N \tag{3.3}$$

Let $x_{t,k,m} = 1$ if technician t belongs to team m on day k. Equation 3.4 guarantees that if a technician is available to work, belongs to the set T_k, then the technician may only be a member of one team that day.

$$\sum_{m \in M} x_{t,k,m} \leq 1 \quad \forall k \in K, t \in T_k \tag{3.4}$$

Conversely, Equation 3.5 confirms if a technician may not work, does not belong to the set T_k, then the technician is not a member of any team on that day.

$$\sum_{m \in M} x_{t,k,m} = 0 \quad \forall k \in K, t \notin T_k \tag{3.5}$$

Let $y_{i,k,m} = 1$ if job i is assigned to team m on day k. Equation 3.6 states that every job belonging to the set of jobs N, must be either outsourced, $z_i = 1$, or scheduled during the scheduling horizon.

$$z_i + \sum_{k \in K} \sum_{m \in M} y_{i,k,m} = 1 \quad \forall i \in N \tag{3.6}$$

Equation 3.7 ensures that if a team is assigned a job, $y_{i,k,m} = 1$, then the collective skill levels of the team are greater than or equal to the skill requirements needed to complete the job.

$$y_{i,k,m} * s^{i}_{\delta\alpha} \leq \sum_{t \in T_k} v^{t}_{\delta\alpha} * x_{t,k,m} \quad \forall i \in N, k \in K, m \in M, \alpha \in A, \delta \in D \tag{3.7}$$

Equation 3.8 deals with the precedence relationships between jobs, so that if job i' is a successor of job i, belongs to the set σ_i, i' may not begin until i has been completed.

$$b_i + d_i \leq b'_i \quad \forall i \in N, i' \in \sigma_i \tag{3.8}$$

Equations 3.9 and 3.10 deal with the working hours of the day. Equation (9) ensures that if a job is scheduled to begin on day k, then the start time of the job is greater than or equal to the beginning of that day. Equation (10) states that if a job is scheduled to be completed on day k then the job must be completed before the working day ends.

$$120(k-1) * \sum_{m \in M} y_{i,k,m} \leq b_i \quad \forall i \in N, k \in K \tag{3.9}$$

$$120(k) * \sum_{m \in M} y_{i,k,m} \geq b_i + d_i \quad \forall i \in N, k \in K \tag{3.10}$$

Let $u_{i,i'} = 1$ if jobs i and i' are assigned to the same team on the same day and i' begins after i is completed. Equation 3.11 ensures time continuity, if two jobs happen sequentially then the end time of job i is less than or equal to the start time of the job i'. Here, G is a large number to satisfy the constraint when jobs do not happen sequentially.

$$b_i + d_i - G(1 - u_{i,i'}) \leq b'_i \quad \forall i, i' \in N, i \neq i' \tag{3.11}$$

Equation 3.12 ensures the correct ordering of jobs. If two jobs happen sequentially, then they must both be allocated to the same team and one must be scheduled before the other.

$$y_{i,k,m} + y_{i',k,m} - u_{i,i'} - u_{i',i} \leq 1 \quad \forall i, i' \in Ni \neq i', k \in K, m \in M \tag{3.12}$$

In some problem instances of the ROADEF 2007 challenge problem there is an outsourcing budget available, C. Jobs that are outsourced do not contribute to the objective function, therefore utilisation of this budget is important. Let $z_i = 1$ if job i is outsourced. Equation 3.13 ensures that the outsourcing budget is not exceeded.

$$\sum z_i * c_i \leq C \quad \forall i \in N \tag{3.13}$$

The set of jobs that are outsourced must adhere to precedence constraints, so if a job is outsourced then so are all successor tasks, Equation 3.14.

$$|\sigma_i| * z_i \leq \sum_{i \in \sigma_i} z'_i \quad \forall i \in N^{\sigma} \tag{3.14}$$

Equations 3.15-3.18 show that variables; $x_{t,k,m}$, $y_{i,k,m}$, $u_{i,i'}$ and z_i are binary.

$$x_{t,k,m} = [0,1] \quad \forall k \in K, m \in M, t \in T \tag{3.15}$$

$$y_{i,k,m} = [0,1] \quad \forall k \in K, m \in M, i \in N \tag{3.16}$$

$$u_{i,i'} = [0,1] \quad \forall i, i' \in N, i \neq i' \tag{3.17}$$

$$z_i = [0,1] \quad \forall i \in N \tag{3.18}$$

Lastly, Equations 3.19 and 3.20 show that the start and end times of jobs are non-negative.

$$e_p \geq 0 \quad \forall i \in N_p \tag{3.19}$$

$$b_i \geq 0 \quad \forall i \in N \tag{3.20}$$

3.3 Datasets

The ROADEF challenge is a biennial competition set by the French Operational Research Society. Each competition proposes a combinatorial optimisation problem and researchers compete to find the most efficient solution techniques within time constrained environments. In 2007, the problem was a technician and task scheduling problem. This problem was based on France Telecom's optimisation problem and 30 real world datasets were used. France Telecom wished to maintain their market share whilst optimising their workforce.

The ROADEF 2007 challenge comprised of three sets of data; A, B and X, each increasing in complexity, see Appendix A. Each set of data contained 10 data instances. There are no outsourcing budget in the Set A instances and the size of the instances range from 5 to 100 jobs. Additionally, there are at most 20 domain skill levels (number of domains $\times$ number of levels), and up to 20 technicians available in these datasets. The Set A instances were the first to be released in the competition.

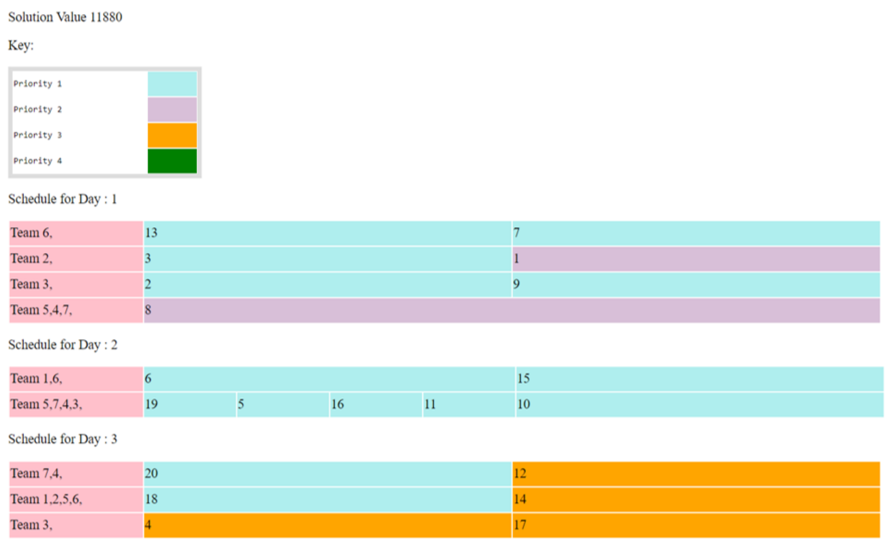
The Set B instances contain outsourcing budgets for each dataset. When there is an outsourcing budget, the outsourcing budget is utilised before scheduling begins. Jobs that are outsourced may not enter the schedules, and scheduled jobs may not enter the outsourcing list. Furthermore, if a job is outsourced then so are all successor jobs. Instances in Set B range from 120 to 800 jobs. There are between 15 and 120 domain skill levels in these datasets and between 20 to 150 technicians available.

The Set X instances are the most complex instances in the ROADEF 2007 challenge and were the last to be released. These instances range from 100 to 800 jobs, contain outsourcing budgets and many domain skill levels, 36 to 105.

Interestingly, during analysis performed in this research of the datasets it was noted that there are similarities between certain datasets. For example, datasets B9 and B10 contain the same set of jobs, however the set of available technicians for dataset B9 is larger and dataset B10 has a larger outsourcing budget. This pattern was also observed for datasets X9 and X10.

3.3.1 Solution Visualisation

A visual representation of a solution is important, an employer who needs to know who is working, where they are, which jobs they are doing and who are they with. In industrial software, these charts are often interactive. This allows a manager to make


Solution Value 11880

Key:

Priority 1	🟦
Priority 2	🟪
Priority 3	🟧
Priority 4	🟩

Schedule for Day : 1

Team 6,	13	7			
Team 2,	3	1			
Team 3,	2	9			
Team 5,4,7,	8				

Schedule for Day : 2

Team 1,6,	6	15			
Team 5,7,4,3,	19	5	16	11	10

Schedule for Day : 3

Team 7,4,	20	12
Team 1,2,5,6,	18	14
Team 3,	4	17

Figure. 3.1 HTML output of a solution to dataset A3 of the ROADEF 2007 challenge detailing the team configurations and job assignments over multiple days

changes to a solution according to local knowledge that is not present in the datasets. For example, removing or adding a team member to improve the working dynamics of the team.

Figure 3.1 shows a HMTL output of a solution to dataset A3, an instance of 20 jobs in the ROADEF 2007 challenge problem. The solution shown is the best known, with an objective value of 11880 (Cordeau et al., 2010). Figure 3.1 shows that this solution is made up of three days. On the first day, there are four teams (*team6*, *team2*, *team3* and *team4,5,7*), and each team has job assignments. The colouring of the jobs corresponds to the priority set that the job belongs to.

As the length of a working day is 120 time units, it can be deduced that on day one *team6* is allocated two jobs, jobs *13* and *7*, both priority one jobs, and each is 60 time units. Due to the way the objective function is calculated, priority one jobs are the most costly, followed by priority two jobs, and so on.

.

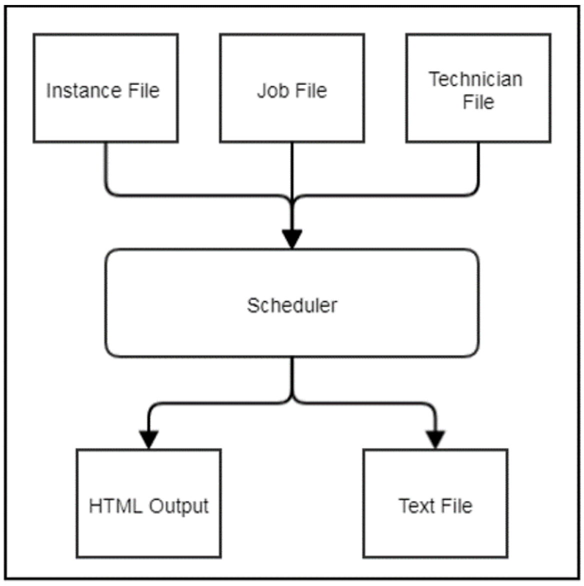


Figure. 3.2 Flowchart describing a solution process for solving the ROADEF 2007 challenge

3.3.2 Solution Process

Solving technician and task scheduling problems is usually a multi stage process. The first stage is the input of the data which describes the problem instance that is being solved. The second stage is the scheduling stage where a solution is created and explored. Lastly, the solution will be output, perhaps to a text file, a visual representation of the solution or both.

For the ROADEF 2007 challenge problem, each instance comprises of three input files, an instance file, a job file and a technician file as shown in 3.2. The instance file contains the dataset name, the number of technicians, number of jobs, and available outsourcing budget. The technician file contains all the information about the technicians, their skills and days on which they are not available. Lastly, the job file contains all the information about the jobs, their skill requirements, priority levels, outsourcing costs, and precedence relationships.

Next, this information is passed to the scheduler. The scheduler creates an initial solution using the heuristic approach and tries to improve it with the metaheuristic. Once the computational time has been reached, a HTML file is output as well as a text file. The HTML file is a pictorial representation of the solution that can be used by a manager to disseminate the schedule to technicians, as shown in Figure 3.1. The text file is used for verification, where the solution is put through an additional checker to ensure feasibility and quality.

3.4 Intelligent Decision Heuristic

In this section the first of two approaches that have been designed to solve the ROADEF 2007 challenge problem is introduced.

3.4.1 Introduction

The intelligent decision heuristic is unlike other approaches that have previously been applied to solve the ROADEF 2007 challenge problem instances. Other approaches construct a team based on which job a scoring mechanism perceives to be the most important job to allocate. A team is then created to service the job and if the team can be allocated more jobs, then more jobs are allocated. The previous approaches in the literature do not consider the utilisation of a team before it is created, for example, a team could be created to service a job but then could be allocated additional low skilled jobs and their collective skills would be wasted. The intelligent decision heuristic is a new approach, which at each stage of the scheduling process evaluates many team and job configuration scenarios.

The intelligent decision heuristic begins by selecting multiple seed jobs. For each seed job a dummy team is then created which is able to service the job. Next, for each dummy team, a check is then performed which finds which other jobs could also be allocated to the dummy team if it was formed. A utility score is then calculated for each dummy team, which represents the quality of the possible job assignments to the team. The highest scoring scenario is then selected, the team is configured and the job assignments are made. This continues until all jobs are allocated.

In other research based on the ROADEF 2007 challenge, team configurations have been rigid and difficult to alter. New operators presented in this research were developed to provide flexibility in team configurations using the intelligent decision heuristic; *remove a team*, *remove N teams*, *remove N jobs*, *decompose and rebuild*, and *decompose and rebuild N*. These operators allow distant solutions to be evaluated as they have the ability to change not only team configurations but also the allocation position of a job or the distribution of a set of jobs.

In addition, experiments into strategies for outsourcing have previously not been carried out in the evaluation of other heuristics to determine whether the quality of the solution produced is dependent on the choice of outsourcing strategy. It was also of interest to measure the effectiveness of each local operator over each set of data to see if there are any patterns. For example, if the chance of finding an improvement is equal or unequal, or if the performance is the same across each set of data.

The intelligent decision heuristic is coupled with an iterative local search meta-heuristic. Iterative local search has not been applied previously to the ROADEF 2007 challenge problem. During the improvement phase, when N non-improving moves have been performed the search is moved to another area of the solution space, through the use of local operators, in the hope of discovering new solutions.

3.4.2 Construction Phase

Figure 3.3 shows the pseudocode for the intelligent decision construction heuristic. The variables associated with the intelligent decision heuristic are; the scheduling horizon K, which holds all schedules and is an entire solution and a schedule k that represents a single day within the scheduling horizon K. There is also a set of technicians T_k who are available for schedule k, an array called $Hypoteams$ which contains a list of dummy teams that could be made to service jobs and a team $T1$ which is the team chosen. There is an array $AllJobs$ containing all jobs that need to be scheduled, an array $Pjobs$ containing jobs of priority p that are under consideration for allocation, an array called

OtherJobs which contains further allocations that could be made to each *Hypoteams* and, lastly, a *PrecedenceArray*, which contains jobs that may not yet be allocated.

On line 1, a scheduling horizon K is initialised, which holds individual schedules and makes up an entire solution to the technician and task scheduling problem instance. The intelligent decision heuristic iterates through all jobs until they have been allocated to teams (i.e. the *AllJobs* array is empty).

Variables: K: scheduling horizon, k: a schedule, *AllJobs*: array of jobs, T_k: set of technicians available on day k, *Pjobs*: array of jobs of priority p, *PrecedenceArray*: array containing jobs that cannot be scheduled

```
 1: Initialise scheduling horizon $K$
 2: while $AllJobs > 0$ do
 3:    Create schedule $k$
 4:    Initialise technicians $T_k$
 5:    while $p \leq 4$ do
 6:       $Pjobs \leftarrow AllJobs(p)$
 7:       $Hypoteams \leftarrow MakeTeams(Pjobs)$
 8:       if $Hypoteams! = null$ then
 9:          $Otherjobs \leftarrow findjobs(Hypoteams)$
10:          $T1 \leftarrow HighestUtility(Hypoteams)$
11:          $MakeTeam(T1)$
12:          $AddJobs(Pjobs)$
13:          $UpdatePrecedenceArray$
14:       else
15:          $p \leftarrow p+1$
16:       end if
17:    end while
18: end while
19: Return $K$
```

Figure. 3.3 Pseudocode describing the implementation of the intelligent decision heuristic

A new day schedule k is created on line 3, and all available technicians are initialised as single technician teams on line 4. The inner while loop is entered on line 5 and the set of jobs with priority p is found and stored in an array *Pjobs* on line 6. For each job belonging to *Pjobs*, a hypothetical team is made if possible and added to an array *Hypoteams*, which, if constructed, has the time and skills to complete the job. In this heuristic, teams are made in a greedy fashion, at each step, the team member who covers the most skill and wastes the least skill is added as the next member of the team until the job requirements are fulfilled or no members can be added to the team.

A check is performed on line 8, if no teams can be created for any job belonging to the $Pjobs$, then p is incremented and the loop is iterated through again. However, if at least one team can be created for any job belonging to the set $Pjobs$ of jobs with priority p, then the heuristic proceeds to line 9. The heuristic then checks which other jobs from $Pjobs$ could also be added onto each job list belonging to each hypothetical team.

On line 10, a utility score is calculated for each possible team belonging to $Hypoteams$, as seen in Equation 3.21. The utility function is made up of two components, the average overskill of the team to the jobs they would be allocated, Equation 3.22, and the slack time that would be left after job allocations, Equation 3.23. The highest scoring utility function is selected; the best hypothetical team is recorded as $T1$ on line 10. Team $T1$ is constructed on line 11 and added to the schedule k.

$$Utility = \frac{Skill}{Slacktime} \tag{3.21}$$

$$Skill = \frac{1}{size} * \sum_{i}^{size} \frac{1}{Max(Overskill(hypoteam, i), 1)} \tag{3.22}$$

$$Slacktime = 120 - \sum_{i}^{size} d_i \tag{3.23}$$

All available job assignments from $Pjobs$ are made to the team on line 12 and the heuristic then checks whether any jobs are now eligible for allocation due to satisfied precedence constraints on line 13. Once no more jobs can be allocated to the current schedule, the heuristic checks whether $AllJobs$ have now been allocated, if not another schedule k is created and the heuristic continues. The construction heuristic terminates once all jobs have been allocated, and an initial solution has been created. The intelligent decision heuristic is used for the initial solution construction and it is also used in the improvement phase. The intelligent decision heuristic is used by the local operators which are discussed further in subsection 3.4.3.

3.4.3 Improvement Phase

In this section, the improvement phase of the intelligent decision heuristic is described. An initial solution has been constructed and now the search to improve it begins. To do this, local operators are used, which perturb the current solution to generate a neighbouring solution, which can then be evaluated.

A variety of local operators were used in this research, some from other combinatorial optimisation work (Estellon et al. (2009) and Cordeau et al. (2010)) such as; *move a job*, *swap two jobs*, *shuffle*, and *swap 1 with N*. The following operators were designed and developed for this research; *remove a team*, *remove N teams*, *remove N jobs*, *decompose and rebuild*, and *decompose and rebuild N*. These operators were designed in order to provide flexibility not only in the distribution of the jobs but also the configurations of the teams.

- *Move a job:* a single job belonging to a team on a day within the scheduling horizon is chosen. This job is then removed from its current position, and reallocated to a different team on a day within the scheduling horizon.

- *Swap two jobs:* two teams within the scheduling horizon are selected and a single job belonging to each team is removed. The jobs are then reallocated to the opposite team, if skill and time constraints allow.

- *Shuffle:* this operator randomly selects a team within the scheduling horizon, and reorders the order of the jobs that have been allocated to the team.

- *Swap 1 with N jobs:* a job is randomly chosen belonging to a team, and the heuristic tries to swap the job with multiple jobs belonging to another team.

- *Decompose and rebuild:* a single day within the scheduling horizon is selected, and all jobs that have been allocated to this day are removed, and all team configurations are also removed. The set of jobs is then reallocated, using the construction heuristic allowing for new team configurations to form.

- *Decompose and rebuild N:* multiple days within the scheduling horizon are chosen, removing all jobs and team configurations. The construction heuristic is then used to reallocate the set of jobs allowing for new team configurations to form.

- *Remove N jobs:* a value *N* is randomly chosen and the heuristic removes this number of jobs belonging to the scheduling horizon. The construction heuristic is then used to reallocate the set of removed jobs.

- *Remove a team:* this operator randomly selects a team and removes all jobs belonging to the team. The removed jobs are then reallocated using the construction heuristic.

- *Remove N teams:* a value *N* is chosen and this number of teams is removed from the scheduling horizon. All jobs are then removed from the teams and reallocated using the construction heuristic.

Iterative Local Search

Once a neighbouring solution has been created using a local operator, the solution must be evaluated using a metaheuristic. Blum and Roli (2003) provide a comprehensive review of metaheuristics used in the field of optimisation, concluding that there are many research opportunities into the application of metaheuristics due to the *"importance of combinatorial optimisation problems for the scientific as well as the industrial world"*.

Most metaheuristic techniques incorporate a strategy to escape local minima and perform a robust search of the solution space (Martí et al., 2010). In this research, iterative local search is used which is an extension of hill climbing that incorporates an escape mechanism. Iterative local search can be classified as a multi start technique and is conceptually simple (Lourenço et al., 2003).

On each iteration, an operator is randomly selected and applied to the current solution s, generating neighbour s'. If the s' solution has a lower objective value (better quality) then s' is accepted as the current solution s. If the neighbouring solution s' has a lower objective than the best solution, the best solution is updated to s'. However, if the neighbouring solution s' is worse, then the variable count is incremented by one. If the count is equal to the step size N, a kick is performed. The kick selects an operator and applies it to the current solution s, generating s'. The solution s' is always accepted and the iterations continue. When implementing iterative local search many decisions are left to the researcher within regards to the step size N used and the type of kick to be performed (Lourenço et al., 2003).

3.4.4 Computational Experiments

In this section, the experiments undertaken on the ROADEF 2007 challenge problem instances in order to utilise the performance of the intelligent decision heuristic are described. Four sets of experiments were performed: (1) outsourcing experiments, (2) local operator experiments, (3) iterative local search tuning, and (4) priority permutation experiments.

1. Outsourcing: experiments into the best outsourcing strategies for the ROADEF 2007 challenge problem datasets have not previously been performed. It is expected that a single outsourcing strategy will not be best suited to all datasets due to the variations within the data. Jobs that are outsourced do not contribute to the objective function, and so utilisation of the outsourcing budget is important. A series of experiments are performed examining the average objective values produced and standard deviations across a subset of problem instances, with different characteristics, to ascertain the best strategies for outsourcing.

2. Local Operators: the probability of selecting a local operator to perturb a solution is equal, but the success rate (finding a better quality solution) of each local operator may not be equal. For example, an operator may be particularly suited to a certain type of dataset, or equally, unsuited. A hypothesis test will be carried

out to ascertain whether the success rates across each local operator is equal on each set of data A, B and X.

3. Iterative Local Search: due to the variations within the datasets in terms of problem size (the size of the problems range from 5 to 800 jobs), the number of iterations performed is not equal across the different datasets. One of the parameters associated with the iterative local search metaheuristic is the step size N, the number of non improving moves before applying the kick operator. It is reasonable to assume that on the larger scale problems the step size N will be smaller than the step size for smaller problem instances as fewer iterations are performed overall.

4. Priority Permutations: the objective function for the ROADEF 2007 challenge problem is a weighted sum of the latest ending job time for each priority class. However, scheduling with the priority permutation *1234* may not always lead to the best quality results. At times, it may be more beneficial to schedule lower priority groups first, for example, if there is a particular job which requires in demand skills. A range of priority permutations have been tested on each dataset and the mean results recorded for each dataset to find the best priority permutations for each dataset.

Outsourcing Experiments

Experiments into the choice of outsourcing strategies have previously not been performed in the literature to the author's knowledge, and therefore the strategies featured in this research are "best guess" strategies, which use factors identified that the author believes will most strongly affect the quality of solution that can be obtained. The strategies contain features such as the duration of a job (strive to reduce the time taken to perform all jobs), the skill requirements (technicians who make up the teams have finite skills), and outsourcing cost (use this budget as efficiently as possible). Multiple outsourcing strategies were designed as shown in Table 3.1.

Table 3.1 Table showing the seven different outsourcing strategies used in the outsourcing experiments

Strategy Number	Outsourcing Strategy
1	Duration
2	Skill Requirements
3	Outsourcing Cost
4	Duration, Skill Requirements
5	Duration, Skill Requirements, Outsourcing Cost
6	Duration, Outsourcing Cost
7	Skill Requirements, Outsourcing Cost

Multiple datasets from the Set B and Set X datasets (as the Set A datasets do not contain the complexity of outsourcing) were chosen for the outsourcing experiments, as one unifying outsourcing strategy may not be suitable for all datasets. The datasets chosen are shown in Table 3.2, the datasets were chosen because they contain a varying number of jobs, available technicians, and skill domain levels.

Table 3.2 Table showing the datasets used for the outsourcing experiments

Dataset	Jobs	Technicians
B4	400	30
B8	800	150
X2	800	100
X7	300	50
X10	500	40

For each dataset, each outsourcing strategy was tested (1-7), using ten runs with each run lasting 20 minutes. The final solution obtained for each run was recorded. The average objective value obtained and the standard deviations have been calculated and are shown in Figures 3.4 to 3.8.

Figure 3.4 displays the mean objective values and standard deviations obtained by the intelligent decision heuristic for dataset B4. It appears that dataset B4 is heavily influenced by the choice of outsourcing strategy used, as there is a large variation in the objective values produced. It appears that the best strategies are 2, 4 and 5 which have lower means and smaller standard deviations than the other strategies. These results

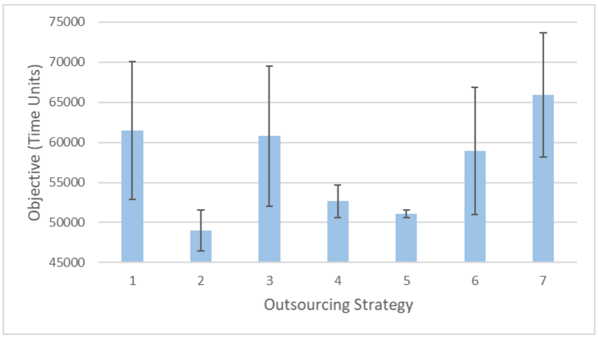


Figure. 3.4 Bar chart showing the mean objective results found including one standard deviation on dataset B4 using different outsourcing strategies

imply that the most important factors when choosing jobs to outsource are the skill requirements followed by the duration of a job.

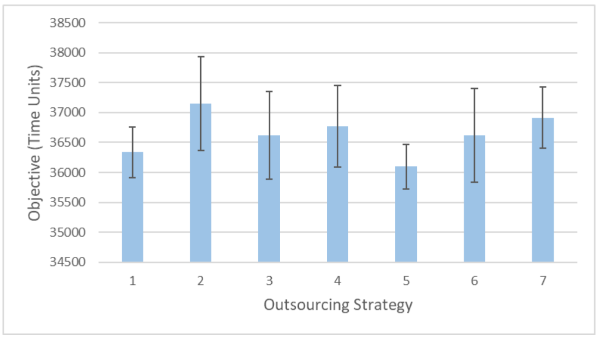


Figure. 3.5 Bar chart showing the mean objective results found including one standard deviation on dataset B8 using different outsourcing strategies

Conversely, Figure 3.5 illustrates that dataset B8 is not as sensitive to changes in outsourcing strategy. The results indicate that regardless of outsourcing strategy, the performance seems to be consistent in terms of the mean objective value produced and standard deviations. This implies that the quality of the solution produced using the intelligent decision heuristic is independent of the outsourcing strategy chosen.

The results for outsourcing strategy testing for dataset X2 are displayed in Figure 3.6, which demonstrates that this dataset is not heavily influenced by the choice of outsourcing strategy. Most strategies appear to have a large standard deviation and the most promising strategy appears to be strategy 1, which is job duration. On closer inspection of the dataset it is observed that this dataset has many jobs of short durations.

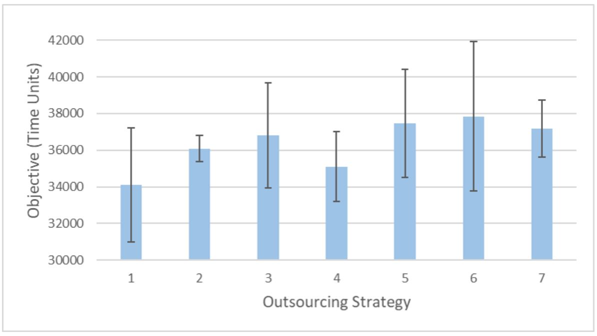


Figure. 3.6 Bar chart showing the mean objective results found including one standard deviation on dataset X2 using different outsourcing strategies

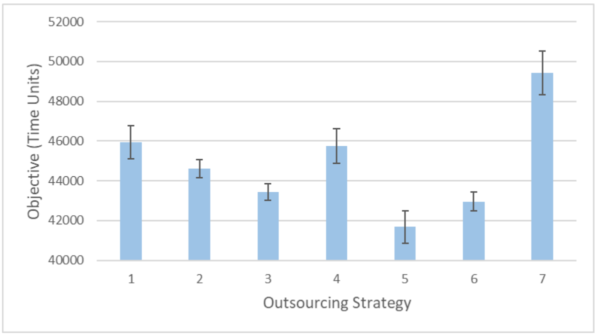


Figure. 3.7 Bar chart showing the mean objective results found including one standard deviation on dataset X7 using different outsourcing strategies

Dataset X7 illustrates that like dataset B4, it is also heavily influenced by that choice of outsourcing strategy. Figure 3.7 shows the mean objective values found appear to be of better quality when using strategies 5 and 6, which both include duration time and the outsourcing cost. Interestingly, each strategy tested appears to have a small standard deviation.

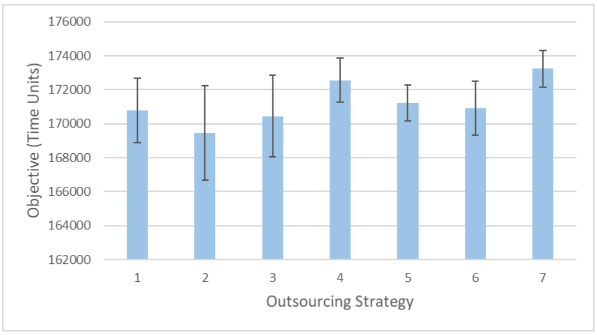


Figure. 3.8 Bar chart showing the mean objective results found including one standard deviation on dataset X10 using different outsourcing strategies

Lastly, Figure 3.8 illustrates that the results on dataset X10, like B8 and X2, are not heavily influenced by the choice of outsourcing strategy used. However, the lowest mean objective value is produced by strategy 2, suggesting skill requirements are the most critical factor.

The outsourcing experiments performed have demonstrated that there is variability amongst the datasets which influence the quality of solutions that can be obtained when using different outsourcing strategies. Some datasets, B4 and X7, are heavily influenced by the choice of outsourcing strategy used, and others are less influenced such as B8, X2 and X10. These results infer that a single outsourcing strategy is not suitable for all datasets as expected.

Local Operator Performance Experiments

In this section, experiments are conducted to ascertain the successfulness of the local operators used. The test considers whether each local operator finds the same number of improving moves, or whether there are differences. A hypothesis test has been carried out for each set of data in the ROADEF 2007 challenge problem, sets A, B, and X. The hypotheses are that either the chance of success is equal, shown in Equation 3.24, or that the chance of success is unequal across the local operators, shown in Equation 3.25.

$$H_0 : \pi_1 = \pi_2 ... = \pi_n \tag{3.24}$$

$$H_1 : \pi_1 \neq \pi_2 ... \neq \pi_n \tag{3.25}$$

For each set of data, 100 runs were performed across ten datasets, with each run lasting 20 minutes, and the number of improving moves per local operator was recorded. Figures 3.9, 3.10, and 3.11 show the percentage improvement achieved by each local operator across each set of data, A, B and X respectively.

Figures 3.9, 3.10, and 3.11 appear to illustrate that hypothesis H_1 is correct, that the improvement proportion is not the same across all operators. In order to prove this a goodness of fit test must be carried out. The test calculates whether the difference in

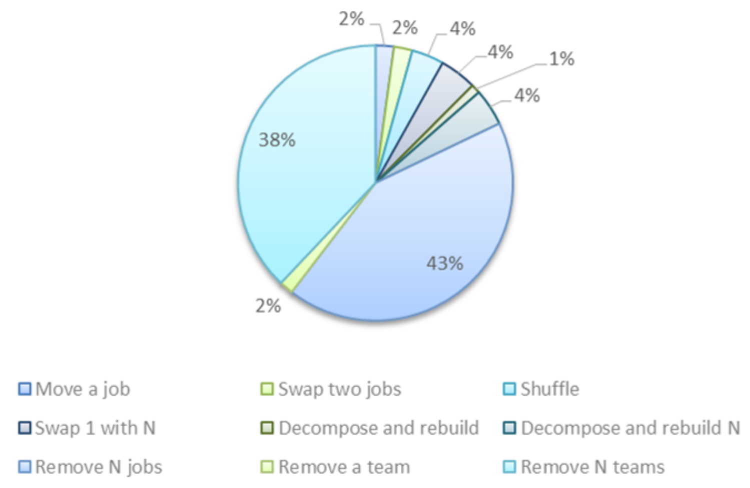


Figure. 3.9 Pie chart showing the percentage improvement of the local operators on the set A problem instances

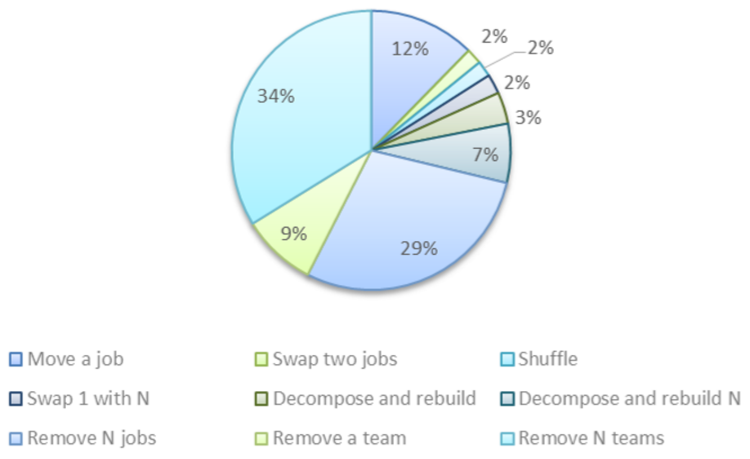


Figure. 3.10 Pie chart showing the percentage improvement of the local operators on the set B problem instances

expected values and observed values can be attributed down to randomness or whether there are significant differences. A chi-square statistic is the sum of the differences squared divided by the expected values.

$$X^2 = \sum_{i=1}^{n} \frac{(ob_i - ex_i)^2}{ex_i} \tag{3.26}$$

The chi-square statistic calculated for each set of data, A, B and X, as shown in Table 3.3.

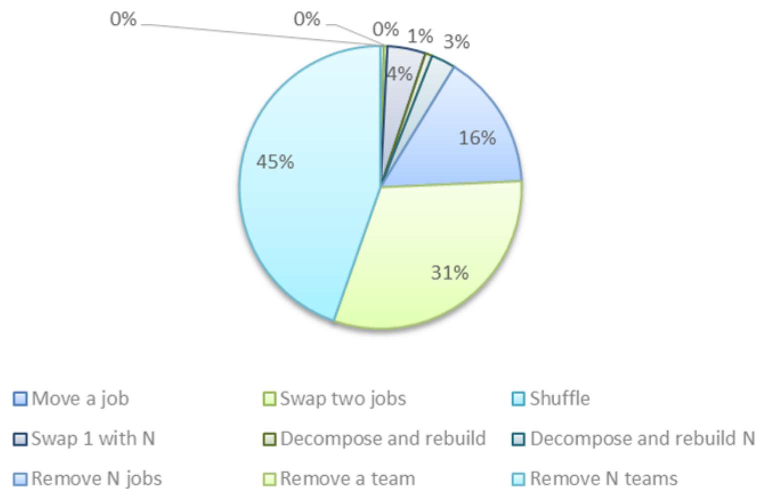


Figure. 3.11 Pie chart showing the percentage improvement of the local operators on the set X problem instances

Table 3.3 Table showing the chi-square statistics calculated for set A, B and X datasets

Set	X^2
A	7137.14
B	459.97
X	379.04

The chi-square statistic compared with the critical value from the chi-square tables on (n-1) (11-1) 10 degrees of freedom on a 95% confidence interval. The critical value at 95% confidence on 10 degrees of freedom is 18.307. As all of the chi-square statistics are significantly greater than 18.307, there are strong grounds to reject H_0 and accept the alternate hypothesis H_1, the local operators have varying levels of success.

Iterative Local Search Experiments

In this section, the implementation of the iterative local search metaheuristic is examined. This metaheuristic has two parameters the step size N and the kick operator k. The step size N controls how many non-improving moves will be performed before applying the kick operator k, as discussed in subsection 3.4.3. The kick operator k is applied to the solution modifying it, and the search continues. This allows worse solutions to be accepted after a period of non-improvement in the hope of finding better quality solutions by exploring another area of the search space.

Two levels for each parameter have been tested as shown in Table 3.4. The step size *N* is either 2500 or 5000 non-improving moves. The KickType has two levels, the first level consisting of a pool of local operators that make small changes to the current solution i.e. *move a job*. The second level consists of a pool containing all operators from the lower level and operators which make large changes such as *decompose and rebuild*.

Table 3.4 Table showing the parameter settings for the iterative local search experiments

Experiment	Step Size *N*	KickType
1	2500	1
2	2500	2
3	5000	1
4	5000	2

Two datasets have been chosen to demonstrate that the variations amongst the datasets may require different step sizes to be used in order to utilise the performance of the metaheuristic. The datasets chosen are A7 and X6. Dataset A7 has 100 jobs to schedule amongst 20 technicians with 20 domain skill levels, and dataset X6 has 200 jobs to schedule amongst 20 technicians with 36 domain skill levels.

Each dataset was tested 10 times using a 20 minute run time for each of the tuning experiments. The mean objective value obtained was then calculated and the main effects of each of the parameters were examined. The results are shown in Figures 3.12 and 3.13. Parameter A corresponds to the step size *N* and parameter B corresponds to the kick type.

Figure 3.12 illustrates that dataset A7 produces better quality results when using a larger step size and a kick that is able to alter the current solution significantly. This is in line with the original assumptions that smaller datasets will require a larger step size as there are more iterations performed, and so the operators provide additional searching power.

Figure 3.13 conversely demonstrates that it is more beneficial to use a smaller step size. This can be explained by the fact larger datasets perform fewer iterations and therefore using a smaller step size prevents the metaheuristic from behaving like a hill

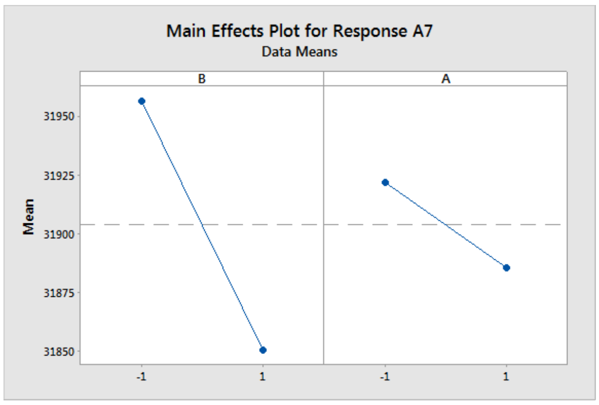


Figure. 3.12 Chart describing the effects on objective function using different parameter settings for iterative local search experiments on dataset A7

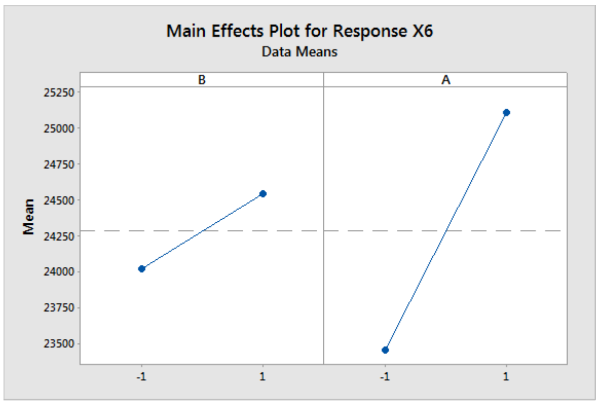


Figure. 3.13 Chart describing the effects on objective function using different parameter settings for iterative local search experiments on dataset X6

climber. Furthermore, in the larger datasets, there are more combinations and a larger search space, frequently moving the search may assist in finding high quality solutions. Additionally, using a small pool of local operators that are computationally inexpensive also aids the search for high quality solutions.

Priority Permutations

The objective function of a solution to any dataset from the ROADEF 2007 challenge problem is calculated as a weighted sum of the latest ending time of each job belonging to each priority group. The weights for priority groups [1, 2, 3, 4] is [28, 14, 4, 1]. However, the ordering of "1234" may not always lead to the best quality solutions.

There may be times when it is more beneficial to schedule lower priority groups first, perhaps due to the skill requirements, duration, or predecessor relationships.

A series of experiments were performed which evaluated different priority permutations to ascertain the best one that leads to the lowest objective values. Table 3.5 shows the priority permutations and outsourcing strategies chosen for each dataset of the ROADEF 2007 challenge. Column one shows the dataset, column two displays the priority permutation used, and column three shows the outsourcing strategy chosen (shown to be the best in section 3.4.4). "1234" indicates that jobs are allocated in the order of priority 1 then 2 and then 3 and finally 4. The outsourcing strategies are either "1" uses a combination of skill requirements, duration and outsourcing cost, or "2" just the skill requirements of the job.

3.4.5 Experimental Results

Tables 3.6 and 3.7 display the results obtained by applying the intelligent decision heuristic to the ROADEF 2007 challenge problem instances compared with the heuristics previously applied in the literature. Column 1 represents the dataset name and column 2 shows the best result (BKS) obtained in the literature. Columns 3 to 5 show the results obtained by Hurkens (2009)(Hu), Cordeau et al. (2010) (C) and Estellon et al. (2009) (E). Columns 6 to 10 display the results found by other researchers, Fırat and Hurkens (2012) (F), Dongala (2006) (D), Jaskowski and Wasik (2007) (J), Pokutta and Stauffer (2009)(P), Korteweg (2007) (K), and Hashimoto et al. (2011) (Ha). Lastly, Column 11 displays the result found by the intelligent decision heuristic. The heuristic was tested using the competition rules, which limits the run time to 20 minutes, with 5 runs and the best found score recorded. The intelligent decision heuristic was programmed in Java and tested on an HPZ230 tower workstation with an i7 processor and 16 GiB RAM.

3.4.6 Discussion

This section 3.4 has presented the intelligent decision heuristic which can be categorised as an ’intelligent’ approach because it considers multiple scenarios before making an

Table 3.5 Table showing the priority permutations used for the ROADEF 2007 challenge datasets

Dataset	Permutation	Outsourcing
A1	1234	-
A2	1234	-
A3	2134	-
A4	1234	-
A5	2134	-
A6	2134	-
A7	1234	-
A8	2134	-
A9	2134	-
A10	1234	-
B1	1234	1
B2	1234	1
B3	1234	1
B4	2134	2
B5	1234	2
B6	1234	1
B7	1234	1
B8	1234	1
B9	2134	1
B10	2134	1
X1	1234	2
X2	1234	1/2
X3	1234	1/2
X4	1234	1
X5	1234	2
X6	2134	1/2
X7	1234	1
X8	1234	1
X9	1234	2
X10	1234	2

allocation or team configuration decision. The heuristic considers the utilisation of a possible team before deciding to create the team. Utilisation is measured in terms of skill and time.

This intelligent heuristic was tested on 30 technician and task scheduling problems that were taken from the ROADEF 2007 challenge and was compared against other heuristic approaches featured in the literature. The heuristic behaved well on the Set A instances finding 5 out of 10 best known results as shown in Table 3.6. However,

Table 3.6 Table showing the computational results on the ROADEF 2007 challenge datasets for the intelligent decision heuristic part 1

Dataset	BKS	Hu	C	E	F	D	J	P	K	H	ID
A1	2340	2340	2340	2340	2340	2340	2490	2340	2340	2340	2340
A2	4755	5580	4755	4755	4755	4755	4755	4755	4755	4755	4755
A3	11880	12600	11880	11880	11880	13068	12600	11880	11880	11880	11880
A4	13452	13620	13452	14040	13452	13620	14040	14760	13452	13452	13452
A5	28845	30150	29335	29700	29335	31236	32400	33480	29335	28845	29040
A6	18795	20280	18795	18795	20005	21576	21120	22380	19935	18870	18795
A7	30540	32520	30540	30540	30960	40116	32520	33360	31050	30840	30660
A8	16920	18960	17700	20100	17335	23115	19380	21180	17587	17335	20100
A9	27348	29328	27692	28020	28280	34056	28280	30000	28028	27692	28020
A10	38296	40650	38636	38296	39300	52348	41580	42740	40350	40020	39000
B1	33900	34710	37200	34395	34575	58968	46995	44025	43620	43860	34410
B2	15870	17970	17070	15870	16775	28989	19890	21240	20010	20665	18600
B3	16005	18060	18015	16020	16275	34368	20340	20280	19575	20565	18210
B4	23775	26115	23775	25305	23925	56382	29460	31815	35385	26025	45855
B5	88680	94200	117540	89700	88920	N/A	100080	122760	119160	120840	119820

Table 3.7 Table showing the computational results on the ROADEF 2007 challenge datasets for the intelligent decision heuristic part 2

Dataset	BKS	Hu	C	E	F	D	J	P	K	H	ID
B6	26955	30450	27390	27615	28785	N/A	24330	37965	32760	34215	37775
B7	31620	33300	33900	38200	31620	N/A	36060	38820	41220	35460	37140
B8	33030	35490	33240	37440	35520	N/A	35550	34440	39240	33030	36000
B9	28080	28200	29760	32700	28080	N/A	29460	33360	30000	29550	33360
B10	34680	34680	35640	41280	35040	N/A	36960	44640	38040	34920	40680
X1	146220	151140	159300	188595	146220	N/A	N/A	N/A	N/A	181575	178560
X2	7260	9090	8280	8370	7740	N/A	N/A	N/A	N/A	7260	32925
X3	48720	50400	50400	50100	48720	N/A	N/A	N/A	N/A	52680	52920
X4	64600	65400	66780	68120	64600	N/A	N/A	N/A	N/A	72860	74880
X5	144750	147000	157800	183700	144750	N/A	N/A	N/A	N/A	172500	182820
X6	9480	10320	9900	10440	9690	N/A	N/A	N/A	N/A	9480	13020
X7	32040	33240	47760	37200	32040	N/A	N/A	N/A	N/A	46680	40320
X8	23220	23460	24060	25480	23220	N/A	N/A	N/A	N/A	29070	27420
X9	122800	134760	152400	159660	122800	N/A	N/A	N/A	N/A	168240	159600
X10	120330	137040	140520	152040	120330	N/A	N/A	N/A	N/A	178560	160860

as the datasets became more complex the heuristic could not match the performance of other approximate approaches such as Cordeau et al. (2010); Estellon et al. (2009); Hashimoto et al. (2011). The intelligent decision heuristic did find competitive solutions in 23 out of 30 datasets.

The intelligent decision heuristic particularly struggled with data instance X2 due to the high number of jobs and small average job durations. This is because the heuristic, when deciding to make a team, looks at which jobs could be allocated to the team, and because of the short job durations, there are many combinations to consider.

The intelligent decision heuristic was coupled with an iterative local search meta-heuristic to solve the technician and task scheduling problem instances. Other meta-heuristics previously applied to this problem include hill climbing and simulated annealing.

Overall, it can be concluded that the intelligent decision heuristic is a reasonable approach to use on the technician and task scheduling problem instances but the author does suggest that results could be improved by using the precedence and successor relationship information to drive the algorithm.

Summary

The intelligent decision heuristic has matched some of the best known solutions to the ROADEF 2007 challenge problem. In 23 out of 30 datasets, the intelligent decision heuristic has produced a competitive solution with regard to solutions found by other researchers.

This research has highlighted that there are many complexities in the ROADEF 2007 datasets which arise due to the real world nature of the technician and task scheduling problem, especially relating to the constraints and their relationships.

The next section of this chapter will investigate the precedence relationships within the datasets to ascertain if results can be improved in instances which have a high number of precedence and successor relationships, by designing a new heuristic approach.

3.5 Look Ahead Heuristic

3.5.1 Introduction

In the previous section, 3.4, an intelligent decision heuristic was proposed to tackle the ROADEF 2007 challenge instances. It was concluded that the precedence and successor relationships between jobs were not fully understood or exploited in the solution approach. The precedence and successor relationships between jobs cause tree like structures which subsequently create indirect precedence and successor relationships between jobs. The additional complexity that precedence constraints introduce to a combinatorial optimisation problem is discussed in Lenstra and Rinnooy Kan (1978). Recent papers deal with precedence constraints in application areas such as machine production line scheduling (Davari et al. (2013) and Carrasco et al. (2013)) and warehouse scheduling (Park et al., 2016).

For this reason, a new heuristic approach, called the look ahead heuristic, has been developed to solve the ROADEF 2007 challenge problem instances, which will account for these multi level precedence and successor relationships present within the datasets.

Relationships within the ROADEF 2007 challenge datasets

Figure 3.14 highlights the complexity of the precedence and successor relationships between a set of jobs in dataset B6 of the ROADEF 2007 challenge. In this example, jobs *346* and *347* are very important jobs to allocate because there are many jobs (*348*, *349*, *350*, *351*, *352*, *353*, *354*, *355*, *356* and *357*) that cannot begin until they have been completed, due to direct and indirect precedence and successor relationships. This characteristic in the datasets can be problematic, especially in low priority groups that are typically not scheduled until later, as it will elongate the makespan of the scheduling horizon, resulting in a higher objective value. The look ahead heuristic proposed in this section has a preprocessing phase, where the indirect relationships between jobs are calculated, which aids the heuristic during the construction and improvement phase.

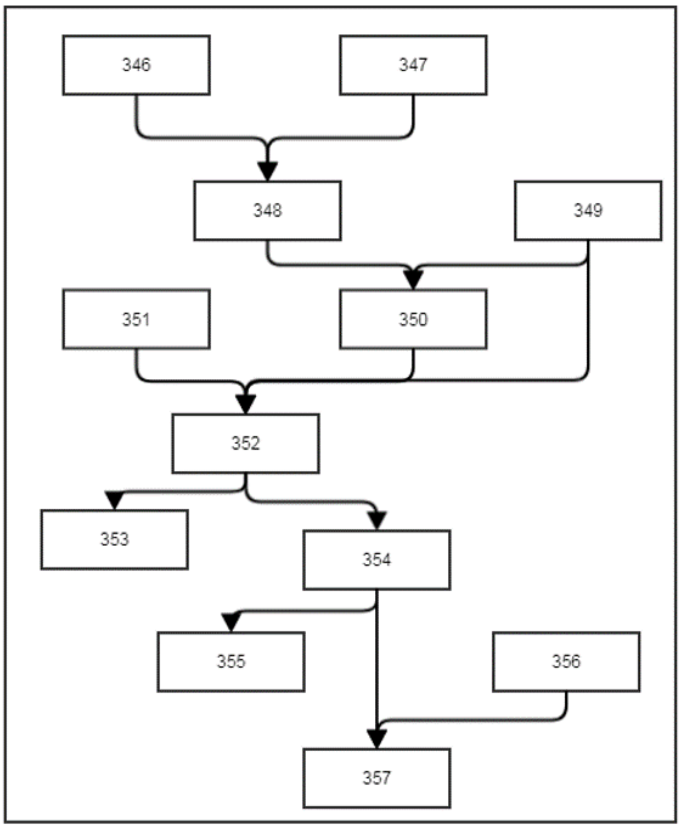


Figure. 3.14 Flowchart illustrating the complex precedence relationships present between jobs in dataset B6

3.5.2 Construction Phase

The look ahead heuristic is a novel approach to solving the ROADEF 2007 challenge problem. The main feature of the look ahead is to consider the impact a job allocation decision will have on subsequent stages of the scheduling process, thereby making smarter allocation decisions. This is the first look ahead heuristic that has been applied to a technician and task scheduling problem. A two phase approach is proposed to solve the ROADEF 2007 challenge problem; construction and improvement.

Look ahead heuristics have been used by Atkinson (1994) and Ioannou et al. (2001) to solve related scheduling problems such as the vehicle routing problem with time windows. In this work, the impact of an allocation decision was measured by metrics relating to distance and time measures. In this implementation, proposed in this thesis, to solve the technician and task scheduling problem, the impact of an allocation decision is measured by the skill wastage of the idle teams that would be left without job assignments.

The construction phase of the look ahead heuristic can be split into four main components as shown in Figure 3.5.2. The components of the construction heuristic are; job selection, team construction, team utilisation and team extension.

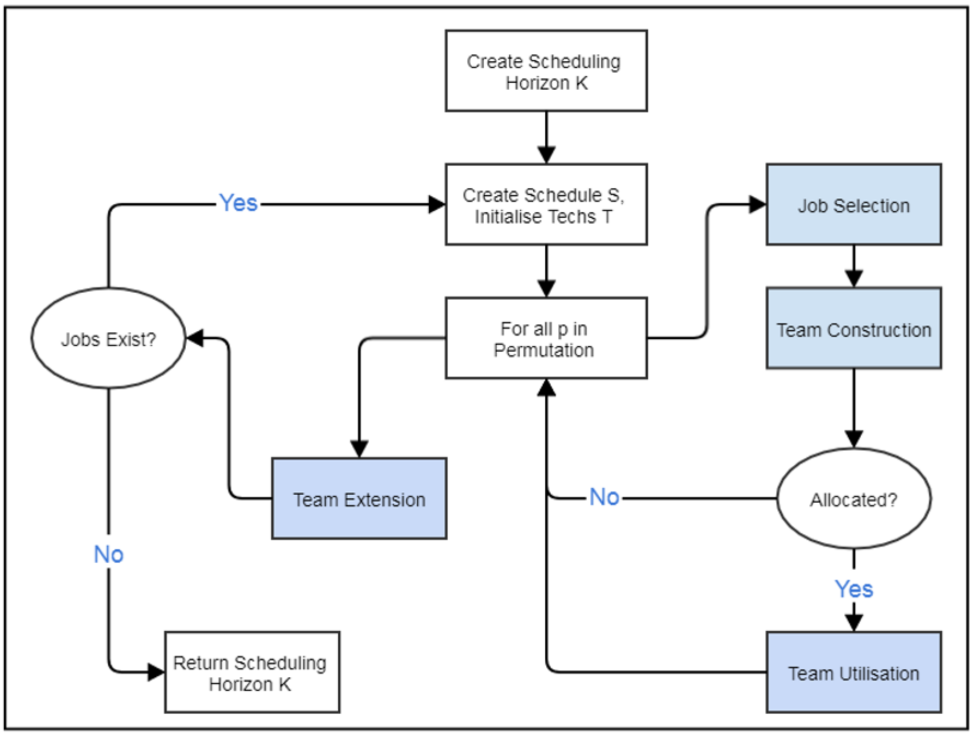


Figure. 3.15 Flow chart shwing the solution path of the look ahead construction heuristic

The construction heuristic begins by initialising a scheduling horizon, which is a collection of schedules that make up a solution. A schedule is then initialised and all available technicians on this day are added to the schedule as single technician teams. When allocating jobs to a schedule, the heuristic iterates through the priority classes of the jobs. While allocations can still be made to the schedule, the heuristic selects a job, constructs a team, makes further job allocations to the team and uses team extension. Once all jobs have been allocated an initial feasible solution has been created.

Job selection

At each stage of the scheduling process, the current priority set under consideration is found. A validity check on this priority set is then performed, where jobs not eligible for allocation are removed. Jobs are not eligible for allocation if a job's predecessors have not been completed or if the combined skill levels of the idle teams are less than the skill levels required by the job. The look ahead is then performed on the remaining set of jobs that are eligible for allocation. The look ahead heuristic refines the set of remaining jobs, removing jobs that will negatively impact subsequent stages of the scheduling process.

The look ahead feature examines the combined skill set of the current idle teams in relation to each job's skill requirements. An estimate of the remaining combined skill set of the idle teams left after allocation of this job is calculated, and the resulting set of feasible moves is found. Decisions that will leave idle teams without assignments are discarded as this would result in high amounts of skill waste. The look ahead step produces a list of jobs that are intelligent allocation decisions in terms of the way they will affect/shape subsequent stages of the scheduling process.

As shown in subsection 3.5.1, there are complex relationships between jobs in some of the ROADEF 2007 challenge datasets. In these scenarios, it is important to recognise how these relationships can constrain the quality of solution found. Jobs that appear in complex relationship trees, like jobs *346* and *347* from Figure 3.14, are important to allocate early in the scheduling horizon. Scheduling these jobs will allow their successors to be eligible for allocation, reducing the scheduling horizon makespan. The look ahead heuristic uses direct and indirect precedence and successor relationship information between jobs to help calculate which job from the set of intelligent allocation decisions should be allocated next, shown in Equation 3.27. The importance of a job can be calculated as; the job's duration multiplied by its priority weight plus the sum of all indirect successors i' belonging to the set $indirect_i$ durations multiplied by their corresponding priority weight wp.

$$Imp_i = d_i * wp_i + \sum_{i' \in indirect_i} d'_i * wp_{i'} \tag{3.27}$$

In addition, the difficulty of a job is also an important consideration especially in the more complex data instances of the ROADEF 2007 challenge. There are some datasets that contain jobs where there are fewer technicians who possess certain domain skill levels (sparse skill distribution). This means that it is more difficult to find a team configuration because there are fewer technicians to choose from who cover the required skills of the job. This is also further complicated by the fact that technicians are not available every day. The difficulty of a job can be calculated as the sum of the domain skill requirements, $s^i_{\delta,\alpha}$, divided by the combined total skill levels of the technicians $T_{\delta,\alpha}$, as described in Fırat and Hurkens (2012) and shown in Equation 3.28.

$$Diff_i = \sum_{\delta \in D} \sum_{\alpha \in A} \frac{s^i_{\delta,\alpha}}{T_{\delta,\alpha}} \tag{3.28}$$

Team construction

Once a job has been selected for allocation, a team must be built to service the job. The team must collectively possess all the skills necessary to service the job and the time available. The skill domain requirements of the job are stored in a matrix, which is reduced accordingly as members are added to the team. During team construction, all teams who have no job assignments are considered. In this approach, teams are built iteratively adding a member at a time. On each iteration, the heuristic calculates the skill cover and skill waste of each eligible team to the remaining uncovered skill requirements of the job. The team with the highest skill cover and lowest skill waste is selected as the next team member. When a member is added to the team, the matrix containing the uncovered skill requirements of the job is reduced. When all elements in the requirements matrix are zero, a team τ has been built to service the job.

Job list utilisation

Once a job has been selected for allocation and a team has been constructed to service the job, the rest of the team's available time and skill can be utilised. In the ROADEF 2007 challenge problem, the length of a working day is limited to 120 time units, and this cannot be exceeded. Jobs in the ROADEF 2007 challenge problem generally have durations that are multiples of 15 time units i.e [15, 30, 45, 60, 75, 90, 105 and 120]. In the look ahead heuristic, a sequential add on approach is used so that the heuristic can recognise when jobs become available for allocation due to fulfilled precedence constraints. The look ahead heuristic calculates which jobs from the current priority set are eligible to be scheduled to the team (in terms of skill requirements, precedence relationships and the slack time of the team) at this time t. From this list of feasible job allocations F_i, the heuristic calculates the over skill of the team τ to each job i and the lowest scoring is selected as U_i, the job i which most utilises the team's skill shown in Equation 3.29.

$$U_i = \min_{\forall i \in F_i} Overskill(\tau, i) \tag{3.29}$$

Team extension

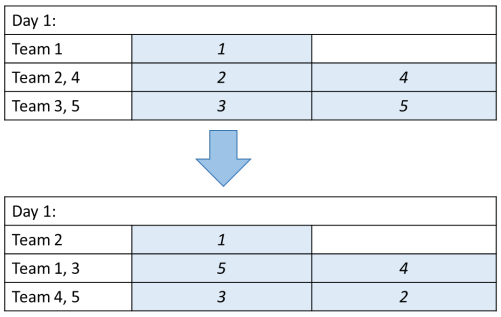
The last component of the look ahead construction heuristic is team extension. This happens when no jobs from the current priority set, N_p, can be scheduled by creating a new team to service the job. This situation may happen when there is a lack of skill amongst the available teams or unfulfilled precedence constraints between jobs. First, a check is performed to see whether any of the teams who already have job assignments have the slack time available to service any of the jobs from the current priority set. If any team has the time available, a validity check on precedence constraints must be performed. The heuristic checks whether precedence constraints would be violated if the job was added to the end of the team's job list. If precedence constraints are not violated, the last step is to check whether this team can be extended (add a team member) in order to satisfy the job’s skill requirements. If an idle technician can be added, the member is added to the team and the job is allocated.

3.5.3 Improvement Phase

Once an initial solution has been created, the improvement phase begins. The heuristic tries to iteratively improve this solution by randomly applying an operator from the set of local operators. Each local operator is a transformation, which modifies the current solution to a neighbouring solution that can be evaluated. The aim of the improvement phase is to find the least costly feasible solution. The local operators featured are the same as the previous section, 3.4.2. In addition, a more sensitive objective function will be used as in Cordeau et al. (2010) in order to guide the metaheuristic. The look ahead heuristic has been coupled with simulated annealing metaheuristic, a widely used metaheuristic in the field of combinatorial optimisation and in particular personnel scheduling problems.

Improved operators

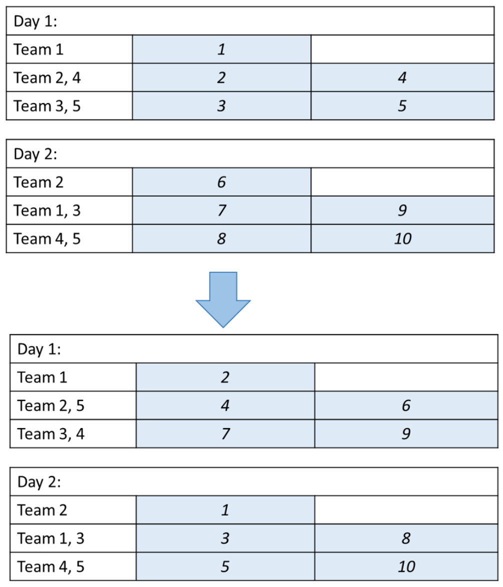
Two operators have been enhanced from the previous section. These operators are; *decompose and rebuild* shown in Figure 3.16 and *decompose and rebuild N* shown in Figure 3.17. Previously, these operators selected a schedule or *N* schedules, removed all jobs and all teaming configurations and used the construction heuristic to reallocate the set of jobs to the existing empty schedule/schedules.



Day 1:		
Team 1	*1*	
Team 2, 4	*2*	*4*
Team 3, 5	*3*	*5*

Day 1:		
Team 2	*1*	
Team 1, 3	*5*	*4*
Team 4, 5	*3*	*2*

Figure. 3.16 Figure showing the local operator *decompose and rebuild* change a day within a solution



Day 1:		
Team 1	*1*	
Team 2, 4	*2*	*4*
Team 3, 5	*3*	*5*

Day 2:		
Team 2	*6*	
Team 1, 3	*7*	*9*
Team 4, 5	*8*	*10*

Day 1:		
Team 1	*2*	
Team 2, 5	*4*	*6*
Team 3, 4	*7*	*9*

Day 2:		
Team 2	*1*	
Team 1, 3	*3*	*8*
Team 4, 5	*5*	*10*

Figure. 3.17 Figure showing the local operator *decompose and rebuild N* change multiple days within a solution

In this work, these operators have been modified to behave more intelligently. If a job is selected to be removed then so are all indirect successor jobs, using the indirect precedence and successor relationship information generated during the preprocessing phase, which should make the reallocation of jobs easier. For operator *decompose and rebuild N*, sequential schedules are removed in order to make more local changes to the solution and maximise the chance of finding a feasible solution. Reallocation of these jobs to the schedule/schedules is performed using the look ahead construction heuristic.

Objective function

The objective function provided in the ROADEF 2007 challenge problem description was not very responsive to changes in the solution landscape. This is because only four jobs' end times are included within the objective function calculation. This is the case whether there are 50 or 800 jobs to schedule, which means that many solutions will have the same objective function value but may be very different in terms of composition.

For this reason, the look ahead heuristic uses a modified objective function, similar to Cordeau et al. (2010). In Cordeau et al. (2010), the last ε job's end times contributed to the objective function calculation . In this implementation, shown in Equation (3.30), the additional term added on to the original objective function is a weighted priority sum of all job's end times. This term aims to help identify promising solution configurations. The additional term ensures that every job's end time makes a contribution to the objective function. Each job's end time, e_i, is multiplied by an ordered priority weight ow_i that corresponds to the position of the job within the scheduling horizon, thereby helping differentiate between different solution states.

$$Objective = \sum_{p=1}^{4} w_p * e_p + \sum_{i=1}^{N} e_i * ow_i \tag{3.30}$$

Simulated annealing

Simulated annealing was first introduced by Kirkpatrick et al. (1983) to solve combinatorial optimisation problems. The simulated annealing metaheuristic has two parameters, the temperature T and the cooling rate δT. If a better quality solution is found then it is always accepted, however, if a worse solution is found the Metropolis criterion is used. The Metropolis criterion, shown in Equation (3.31), calculates a probability, which is the negative exponential of the difference in the current and candidate solution, δE, divided by the current temperature, T.

$$p = e^{\frac{-\delta E}{T}} \tag{3.31}$$

This probability, p, is compared to a random number generated on the interval $\{0,1\}$, and, if larger, the candidate solution is accepted. As the algorithm iterates, the temperature parameter is decremented thus decreasing the likelihood of accepting worse quality solutions.

3.5.4 Computational Experiments

Simulated annealing tuning

In order to implement simulated annealing effectively, the optimal starting temperature for the metaheuristic must be found. Using a temperature that is too low, will make the metaheuristic behave more like a hill climber, and a high temperature may cause the search to diverge towards low quality solutions. Finding the balance between a search that is too constrained and a search that is too free can be a difficult task. Other considerations are the variations amongst the dataset in terms of the size of the ROADEF 2007 challenge problem instances. Therefore, each dataset or set of data may need a different starting temperature value. This is to ensure the metaheuristic reaches its ground state (convergence) before the computational time limit has been reached. In this implementation of simulated annealing, the decrement factor has been fixed in order to minimise the number of computational experiments that need to be performed.

The aim of these tuning experiments is to find an optimal starting temperature for the metaheuristic which achieves convergence within the computational run time. Due to the variations amongst the datasets, it is not expected that a single temperature will suit all datasets, as the number of iterations performed is not uniform. For this reason, two datasets were used to demonstrate the need for different starting temperatures within the implementation of the look ahead heuristic. Datasets B1 and X8, have been trialled with a range of starting temperatures, as shown in Table 3.8, which were chosen to ensure the metaheuristic reaches ground state before the run is over.

Table 3.8 Initial temperatures for the simulated annealing tuning experiments on the ROADEF 2007 challenge datasets

Temperature
10
20
40
80
100

Figures 3.18 and 3.19 show the average objective function value obtained over 10 runs for each dataset for the range of temperatures tested. Figure 3.18 illustrates that for dataset B1 the best starting temperature appears to be 100, suggesting that this dataset is better suited to a higher starting temperature. It appears that using a lower temperature results in a larger objective value, and a curve that is less steep (finds less

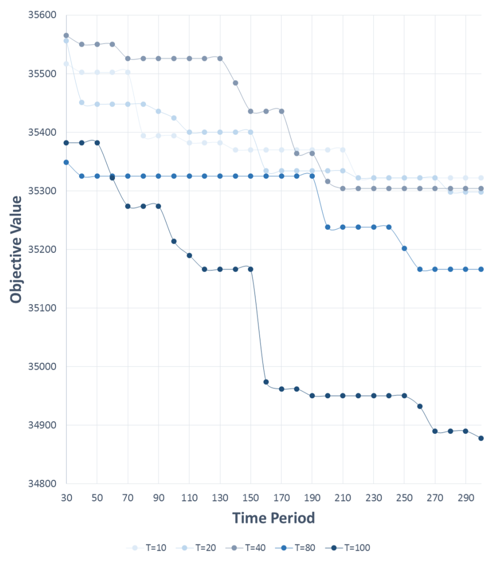


Figure. 3.18 Graph showing the solution value over time using a range of initial simulated annealing temperatures for dataset B1

improvement). The graph also suggests that the algorithm converges much quicker using a higher starting temperature.

Conversely, Figure 3.19 illustrates that the best starting temperature for dataset X8 is 20 or 40. Unlike the previous graph, higher temperatures appear to provide to free a search in the early iterations and find less improvement. Using a lower temperature results in a much faster convergence speed in the early iterations of the algorithm. These results are in line with the assumptions that due to the varying nature of the problem datasets a range of simulated annealing temperatures are optimal.

3.5.5 Experimental Results

The ROADEF 2007 challenge allowed 20 minutes computational time for each dataset, however, the look ahead heuristic approach was allowed a 5 minute run time to prove its effectiveness as a solution approach. The look ahead heuristic was programmed in Java and tested on an HPZ230 tower workstation with an i7 processor and 16 GiB RAM. Each dataset was tested 5 times, as in the ROADEF 2007 challenge, and the best results obtained are shown in Tables 3.9 and 3.10. The look ahead heuristic is compared to the mixed integer programming approaches by Fırat and Hurkens (2012)

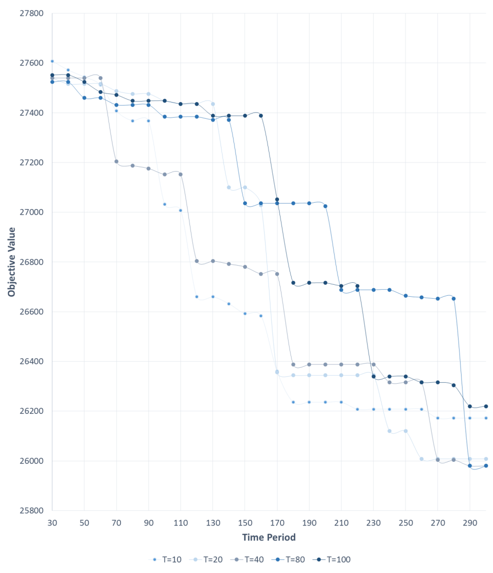


Figure. 3.19 Graph showing the solution value over time using a range of initial simulated annealing temperatures for dataset X8

(MIP-1) and Hurkens (2009) (MIP-2). The look ahead heuristic is also compared to heuristics that include an improvement phase; Cordeau et al. (2010) with an adaptive large neighbourhood search (ALNS), Estellon et al. (2009) with a local search (LS) algorithm and Hashimoto et al. (2011) with a greedy randomized adaptive search algorithm (GRASP). The results for the look ahead heuristic (LA) are displayed in the last column of each of the tables.

3.5.6 Discussion

Section 3.5 has demonstrated that the ROADEF 2007 challenge problem is a complex combinatorial optimisation problem. Section 3.5.1 showed that there are many indirect precedence and successor relationships between jobs which cause chain like structures, increasing the importance of allocating these jobs early in the scheduling horizon. These types of relationships occur in many sectors such as housing projects and production line assembly.

The look ahead heuristic has been coupled with a simulated annealing metaheuristic. Due to the run time used in these experiments, 5 minutes, considerations about the number of iterations had to be made. For example, in the more complex and larger

Table 3.9 Computational results found on the ROADEF 2007 challenge datasets using the the look ahead heuristic part 1

Dataset	BKS	MIP-1	MIP-2	ALNS	LS	GRASP	LA
A1	2340	2340	2340	234 0	2340	2340	2340
A2	4755	4755	4755	4755	4755	4755	4755
A3	11880	11880	11880	11880	11880	11880	11880
A4	13452	13452	13620	13452	14040	13452	13452
A5	28845	29335	29335	29335	29700	28845	30660
A6	18795	20055	20280	18795	18795	18870	18795
A7	30540	30960	32520	30540	30540	30840	30960
A8	16920	17355	18960	17700	20100	17335	17400
A9	27348	28280	28320	27692	28020	27692	27796
A10	38296	39300	40650	38636	38296	40020	39000
B1	33900	34575	35460	37200	34395	40020	33900
B2	15870	16755	18300	17070	15870	20655	17700
B3	16005	16275	16975	18015	16020	20565	17820
B4	23775	23295	27015	23775	25395	26025	27360
B5	88680	88920	94200	117540	89700	120840	108120

Table 3.10 Computational results found on the ROADEF 2007 challenge datasets using the the look ahead heuristic part 2

Dataset	BKS	MIP-1	MIP-2	ALNS	LS	GRASP	LA
B6	26955	28785	30510	27390	27615	34215	32500
B7	31620	31620	33060	33240	37440	35460	32940
B8	32160	35520	32160	33240	37440	33030	33480
B9	28080	28080	28080	29760	32700	29550	29760
B10	34680	35040	35040	35460	41280	34920	36840
X1	146220	146220	151980	159300	188595	181575	158520
X2	7260	7740	9090	8280	8370	7260	9540
X3	48720	48720	50400	50400	50100	52680	52200
X4	64600	64600	65640	66780	68120	72860	68560
X5	144750	144750	147000	157800	183700	172500	158280
X6	9480	9690	10440	9900	10440	9480	11140
X7	32040	32040	33120	47760	37200	46680	39360
X8	23220	23220	23580	24060	25480	29070	25140
X9	122700	122700	136030	152400	159660	168240	144360
X10	120330	120330	131700	140520	152040	178560	141720

problem instances, fewer iterations are performed due to the computationally expensive operators such as *decompose and rebuild* and *decompose and rebuild N*. For this reason, it was necessary to find the optimal starting temperatures to ensure convergence within the run time.

Tables 3.9 and 3.10 show the results obtained using the look ahead heuristic. On the Set A instances the look ahead heuristic performs well with regards to the solutions found by other researchers. In 5 out of 10 datasets the look ahead heuristic finds the best known score. Additionally, the look ahead performs third best on these instances with an average gap of 1.4% to best known solutions. These instances are small compared to the datasets in the Set B and X instances and they do not contain the complexity of outsourcing.

In the Set B instances the look ahead heuristic has an average gap of 10% to the best known score and performs fifth best. In data instance B1 the look ahead finds the best known score, 33900. The look ahead performs particularly well on data instance B7 which contains 50 domain skill levels and 500 jobs. The look ahead approach struggled with data instance B6, which may be attributed to a large number of indirect precedence and successor relationships in low priority groups coupled with jobs that have short durations.

In the Set X instances the look ahead heuristic performs fourth best out of all the approaches. The heuristic performs particularly well on the larger instances in terms of the scheduling horizon size X1, X5, X9 and X10. The look ahead heuristic performs least well on the smaller scheduling horizon instances X2 and X6 which contain jobs which have very short durations and therefore contain many combinations of solutions. Section 3.5 has shown that competitive results can be achieved on the ROADEF 2007 instances using the look ahead heuristic. The look ahead heuristic also significantly outperformed the intelligent decision heuristic on these problem instances.

Summary

This section has presented a look ahead heuristic to solve the technician and task scheduling problem presented by the ROADEF 2007 challenge. The look ahead heuristic is characterised by its pre-processing phase where indirect precedence and successor information is collected and mapped into the problem, the look ahead element within the heuristic where the "state" of the scheduling process is considered, the flexibility in team configurations through improved operators and team extension tools.

The results presented in this section illustrate that the look ahead heuristic is an appropriate solution approach and has highlighted the complexity of these types of problem. The look ahead heuristic has shown that competitive results can be produced in complex ROADEF 2007 challenge data instances (B1, B7, X9 and X10). The look ahead heuristic can outperform the intelligent decision heuristic on these technician and task scheduling problem instances, except on problem instances A5 and A7.

3.6 Comparison of the Intelligent Decision and Look Ahead Heuristic

In this chapter two heuristic procedures have been designed and developed to solve 30 instances of technician and task scheduling problems of varying complexity and problem size. The results demonstrate that the look ahead heuristic is a superior approach to the intelligent decision heuristic. Although the intelligent decision heuristic in most cases can produce a competitive result, it struggled on the set X problem instances, the most complex instances. In particular, the intelligent decision heuristic did not perform well on X2 or X6, these datasets include jobs that have short durations, this may be attributed to the multi allocation decision making property of the heuristic.

The look ahead heuristic performed well on all instances, matching the performance of the adaptive large neighbourhood search and local search heuristic who ranked 2nd in the competition and outperformed the greedy randomized adaptive search procedure. The look ahead could not beat the performance of the mixed integer programming approaches, who consistently found superior solutions in the set X problem instances, through the efficient exploitation of problem structure solving small MIP problems.

The contributions to the field in this chapter are; (1) intelligent decision heuristic which considers the impact of a team configuration decision in regards to the utilisation of the team, considering slack time and further allocations, (2) look ahead heuristic characterised by its preprocessing phase and by considering the subsequent impacts of allocation decisions to the scheduling process and (3) novel operators designed and developed to provide flexibility in team configurations and job distributions.

These problem instances studied have contained up to 800 jobs to allocate, however, in the real world often there are many more jobs to schedule to a large workforce. For this reason, the next chapter, chapter 4, will focus on designing some large scale problems that are indicative of the scale of problem faced in industry and apply the heuristics developed in this chapter (the intelligent decision heuristic and the look ahead heuristic) to test their scalability.

Chapter 4

Large Scale Technician and Task Scheduling Problems

4.1 Introduction

In this chapter, a set of large scale technician and task scheduling problems are created using a data generator and solved using the previous heuristic approaches, described in chapter 3. As illustrated in chapter 2, there is a lack of large scale data available to researchers and for this reason heuristic approaches used to solve smaller problems have not had their scalability tested on larger problem instances. It is necessary to test heuristic procedures not only to assess scalability but robustness too, in order to apply them to real world scenarios, therefore new datasets have been generated to address this issue.

The ROADEF 2007 challenge problem includes the most relevant features of the problems studied in the literature. However, the problems featured range from 5 to 800 jobs. There is no current literature that includes solving large scale (1000+ jobs) scheduling problems that have precedence relationships, skill requirements, and teaming. There is a great need to solve these large scale problems that occur in industrial settings, a fundamental problem faced by the sponsor of this research Service Power PLC.

In this chapter, large scale technician and task scheduling problems (created under the ROADEF 2007 challenge problem definition) have been generated using a novel data generator. The datasets contain all aspects of the ROADEF 2007 datasets such as priority, precedence, outsourcing, and skill requirements. 12 new large scale datasets have been generated ranging from 1000-2500 jobs to schedule. The scalability of heuristic procedures; the intelligent decision heuristic, look ahead heuristic and the greedy heuristic on the datasets have been examined to see which heuristic is the most suitable and could be used to solve real world industrial problems.

The lower and upper bound on the number of jobs has been carefully chosen for several reasons. Firstly, to test the scalability of the heuristics developed the problems must be larger than the ROADEF 2007 instances, which contain up to 800 jobs. Furthermore, work by Xu and Chiu (2001) contained up to 1000 jobs (with skill not treated as a hard constraint), and so this was chosen as the lower bound. In addition, research in the literature on a shift scheduling problem by Krishnamoorthy et al. (2012) containing up to 2105 jobs, reported that in some problem instances, even a heuristic algorithm could not find a feasible solution in 30 minutes of computational time. For this reason, constrained by the ROADEF 2007 competition rules of using 20 minutes computational time, it was decided that 2500 would be the upper limit on the number of jobs to schedule. In addition, the size of the scheduling horizon had to be given due consideration. Although there is no number of scheduling days given in the problem description, the objective is to complete all jobs as quickly as possible. In the ROADEF 2007 challenge datasets, the scheduling horizon spans up to 58 days in the more complex instances. For this reason, in the large scale datasets, it was expected that the scheduling horizon would span from 20-250 days depending on the problem size and the number of available technicians. It may seem like this is a rather large scheduling period that may not be practical in some situations, such as repair work, but in the field of yearly maintenance checks of appliances, this is acceptable. Furthermore, the purpose of this chapter is to push the heuristics developed to solve difficult problem instances in short computational times.

The rest of this chapter is structured as follows; section 4.2 describes the large scale technician and task scheduling problem datasets that have been generated and section 4.3 describes the three heuristics, an intelligent decision heuristic, the look ahead heuristic and a greedy heuristic that have been used to test the large scale data instances. Section 4.4 outlines the improvement phase and Section 4.5 presents the experimental results. Lastly, section 4.6 discusses the results presented and section 4.7 concludes on the research undertaken in this chapter.

4.2 Generating Large Scale Technician and Task Scheduling Problem Instances

The datasets created in this research are novel, they involve solving a large scale multi-period scheduling problem, with an outsourcing budget, respecting unavailability of resources and teaming. It is believed that only one piece of research exists that has generated technician and task scheduling problem datasets independently, which was by Krishnamoorthy et al. (2012). This work included scheduling up to 2105 jobs, but jobs had to be performed on a particular day, and the objective is to minimise the number of staff used. Two other works extended existing vehicle routing datasets (100 jobs), Kovacs et al. (2012) and Pillac et al. (2012), by concatenating skill requirements from other scheduling problems or generating them randomly. There is a need to be able to test heuristic procedures on large scale and complex problems to ascertain whether solution approaches that work well on small scale problems could be implemented in the real world.

4.2.1 Generating Large Scale Instances

Each data instance was made up of three files, an instance file (containing the dataset name, number of jobs, number of technicians, number of domains, number of levels, and the available outsourcing budget), a technician file (containing the name of each technician, their domain skill levels, and days on which they are unavailable), and a job file (containing each job's name, outsourcing cost, priority level, duration, skill requirements, and predecessors).

Figure 4.1 shows how the large scale datasets have been generated. Firstly, the instance file was created which specifies the parameters of the problem. Next, the set of technicians can be created. Each technician is randomly given a level of expertise in each of the domains and assigned days off within the scheduling horizon. Lastly, the job file is created. In the ROADEF 2007 challenge problem, the length of a working day is limited to 120 time units. In the original problem instances, the job durations ranged from 15 time units to 120 time units, in 15 time unit intervals. Therefore, the job durations in the new datasets have been randomly assigned to be of a length that is a multiple of 15 time units and not greater than 120 time units [15, 30, 45, 60, 75, 90, 105, 120]. An outsourcing cost is randomly assigned and a priority level between 1 and 4. However, the jobs also have two other important attributes, domain skill requirements and precedence and successor relationships.

Generating Domain Skill Requirements

The total number of technicians skilled in each domain skill level is recorded. When it comes to generating the skill domain requirements of a job, for the first level in each domain a random number is selected from 0 to 5 or 0 to the maximum number of technicians who possess this area of expertise, shown in Equation 4.1.

$$domainskill(i) = \begin{cases} random[0,5], & \text{if } numtech(i) >= 5 \\ random[0, numtech(i)], & \text{otherwise} \end{cases} \tag{4.1}$$

For each subsequent level of expertise in a domain, shown in Equation 4.2, a random number is selected between 0 and the previously required level of expertise. This is to ensure that skill levels are hierarchical i.e if four technicians are required to be skilled in domain 2 to level 3, then the next level, i.e domain 2 level 4 must require four or fewer technicians.

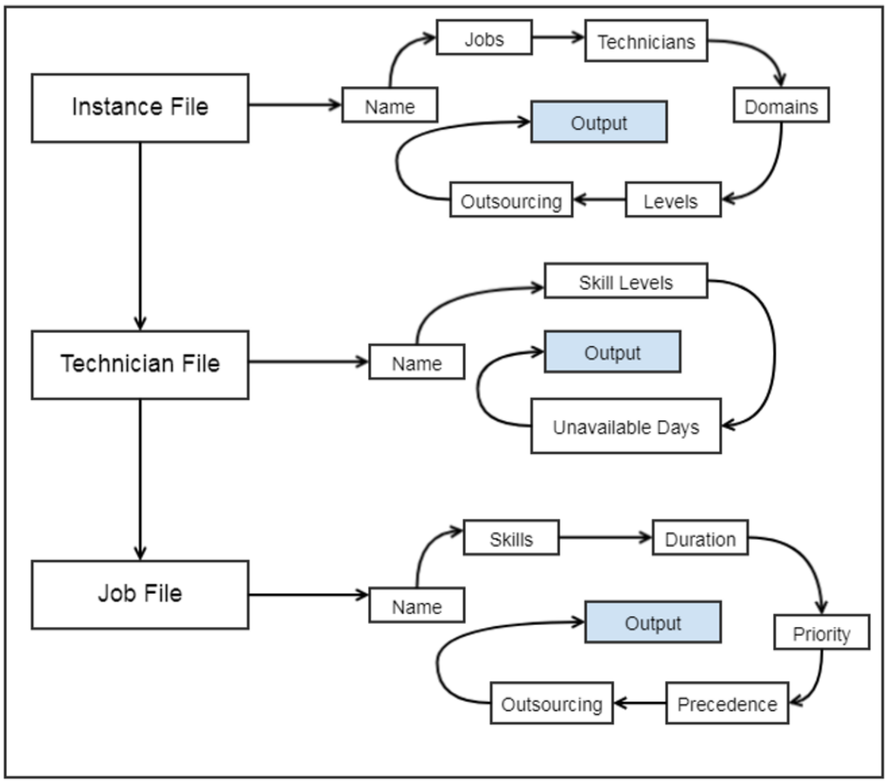


Figure. 4.1 Flowchart showing the creation of the large scale technician and task scheudling problem datasets

$$domainskill(i) = random[0, domainskill(i-1)] \tag{4.2}$$

Generating Precedence and Successor Relationships

Generating precedence and successor relationships between jobs was a complex task. In the ROADEF 2007 challenge problem, there were multi layered precedence and successor relationships that contained many layers and many jobs. In order to generate these types of constraints, an algorithm had to be designed shown in Figure 4.2.

This algorithm on line 1 randomly selects a set of jobs S from the set of all jobs N. On line 2, the set of jobs is ordered in terms of their priority levels in descending order of importance [1...4]. This is to ensure that jobs of priority group p are dependent on jobs that are priority p or higher, as this is the way the objective function is calculated, a weighted sum of priority end times.

Variables: N: set of all jobs, S: subset of jobs from N, L: set of levels

```
1: $S \subseteq N$
2: Priority Order ($S$)
3: while $S$ >0 do
4:     create a level $l$
5:     add jobs from $S$ to level $l$
6: end while
7: for $l \in L$ do
8:     for $job \in l$ do
9:         Build Connections
10:    end for
11: end for
```

Figure. 4.2 Pseudo code illustrating the creation of multi-level precedence relationships between a subset of jobs

The algorithm then enters a while loop on line 3, and while there are jobs remaining in S a level l is created. This level l is given a random subset of jobs from S. This continues until all jobs belonging to S have been assigned a level l. Next on line 7, for each level belonging to l, and for each job belonging to l the algorithm ensures each job has at least one connection to another layer; either upwards (successor) or downwards (precedence).

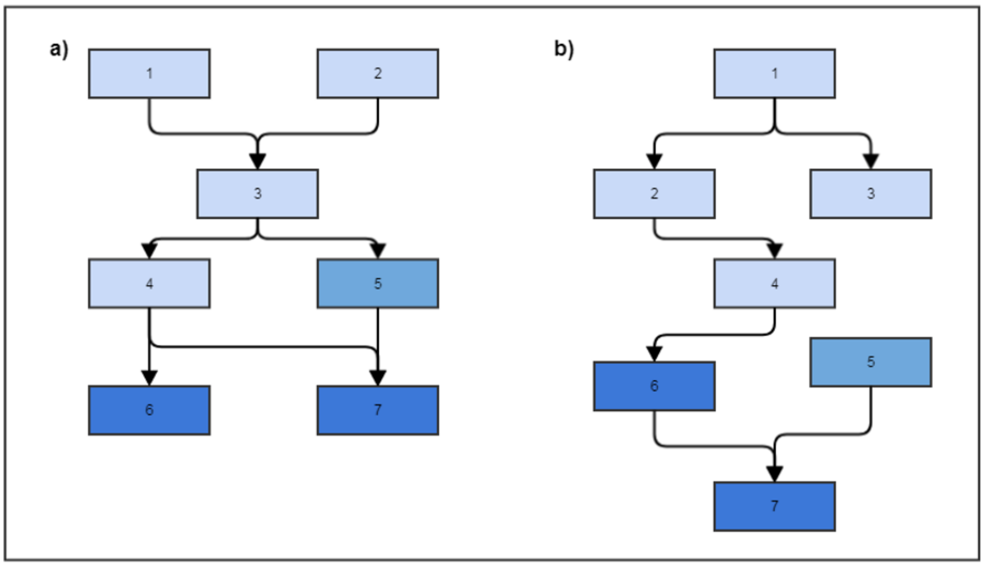


Figure. 4.3 Example of multi-level precedence and successor relationships between a set of jobs using the precedence generator algorithm

As this algorithm has a random nature the following relationship trees, as shown in Figure 4.3, have been created using the same set of jobs. In Figure 4.3a, the relationship tree has four layers. On the first layer are jobs *1* and *2*, on the second, job *3* (which is a successor of jobs *1* and *2*), the third layer contains jobs *4* and *5* (which are successors of job *3*), and lastly, on the fourth layer, jobs *6* and *7* (both dependent on job *4*, and one dependent on job *5*).

In Figure 4.3b, the relationship tree has 5 layers, with one initial job node and one end job node. This figure depicts that jobs can be a member of the relationship tree without having to have both a successor and predecessor (jobs *3* and *5*), reiterating the complexity of job relationships within the ROADEF 2007 challenge problem framework.

4.2.2 Large Scale Instances

Table 4.1 shows the large scale technician and task scheduling problems that have been designed for this research. Column one shows the name of each dataset created, column two (Jobs) shows the number of jobs to be scheduled, column three (Techs) displays the number of available technicians and column four (Budget) displays the outsourcing budget available. Lastly, columns five and six (Domains and Levels) show the number of domains and levels.

There are twelve data instances that have been designed which range from scheduling 1000 to 2500 jobs. The size of the scheduling problems was chosen with help from the industrial sponsor with relation to the size of the scheduling problems which they face daily. These new datasets can be split into four groups; L1-L3, L4-L6, L7-L9 and L10-L12. Each group of data contains the same set of jobs (skill requirements, outsourcing cost, priority, precedence and duration) to be scheduled, but contains a varying number of available technicians, thus altering the size of the scheduling horizon.

Table 4.1 Table showing the large scale technician and task scheduling problem instances that were generated using the data generator

Dataset	Jobs	Techs	Budget	Domains	Levels
L1	1000	25	500	3	3
L2	1000	50	500	3	3
L3	1000	100	500	3	3
L4	1500	25	1000	4	4
L5	1500	50	1000	4	4
L6	1500	100	1000	4	4
L7	2000	25	1500	3	3
L8	2000	50	1500	3	3
L9	2000	100	1500	3	3
L10	2500	25	2000	4	4
L11	2500	50	2000	4	4
L12	2500	100	2000	4	4

4.2.3 Dataset Analysis

The datasets created, L1-L12, have been studied and the mean job length, as well as the priority proportions, are shown in Table 4.2. The mean job length and priority proportions are also shown for datasets A7, B1 and X3 from the ROADEF 2007 challenge problem. It is important to note that there are many variations within the ROADEF 2007 challenge datasets in terms of both mean job length and priority proportions.

Table 4.2 Table showing the analysis of a subset of the large scale technician and task scheduling problem instances against some the ROADEF 2007 challenge datasets

Dataset	Job Length	Priority 1	Priority 2	Priority 3	Priority 4
1-3	67.35	0.0214	0.244	0.288	0.254
4-6	65.91	0.246	0.246	0.261	0.247
7-9	66.23	0.2445	0.245	0.242	0.267
10-12	68.47	0.243	0.255	0.238	0.263
ROADEF- A7	61.71	0.69	0.19	0.12	0
ROADEF- B1	72.52	0.25	0.235	0.265	0.25
ROADEF- X3	108.4	0.193	0.29	0.24	0.276

4.3 Heuristic Approaches

The heuristics featured in chapter 3, the intelligent decision heuristic and the look ahead heuristic are applied to solve the large scale technician and task scheduling problems. On the ROADEF 2007 challenge instances both heuristics found competitive results but overall the look ahead heuristic had better performance particularly on the Set X instances. It is, therefore, reasonable to assume that on the large scale instances the look ahead will again outperform the intelligent decision heuristic. To compare these approaches a simple heuristic approach, a greedy heuristic, has also been designed to provide a more in depth comparison.

4.3.1 Greedy Heuristic

In order to benchmark the intelligent decision and look ahead heuristic, a greedy heuristic is also implemented to ascertain the performance of an unsophisticated and computationally efficient solution approach. There was no need to implement the greedy heuristic on the ROADEF 2007 challenge datasets as there are already benchmark results from the literature using a range of solution approaches.

On line 1 the scheduling horizon K is created, and while jobs are left to allocate a schedule k is created on line 3 and all available technicians are initialised on line 4. On line 5 each priority class is iterated through, the current priority set of jobs is obtained on line 6 and stored as P_{jobs}. On line 7 a job j is selected from P_{jobs} which is the job chosen to allocate. The heuristic then tries to make a team $T1$ on line 8. If team $T1$ is not null, i.e. has the expertise to serve job j, then j is allocated to the team. Next, on line 11 the heuristic continues to add jobs to $T1$. On line 12, the $PrecedenceArray$ is updated. This process continues until each job has been allocated and an initial solution has been found and output on line 18. The greedy heuristic is far less computationally expensive than the other heuristics.

Variables: K: scheduling horizon, k: a schedule, *AllJobs*: array of jobs, T_k: set of technicians available on day k, *Pjobs*: array of jobs of priority p, *PrecedenceArray*: array containing jobs that cannot be scheduled

```
1: Create scheduling horizon $K$
2: while $AllJobs > 0$ do
3:    Create schedule $k$
4:    Initialise technicians $T_k$
5:    while $p \leq 4$ do
6:       $Pjobs \leftarrow AllJobs(p)$
7:       $j \leftarrow SelectJob(Pjobs)$
8:       $MakeTeam(T1)$
9:       if $T1 != null$ then
10:         $AddJob(T1, j)$
11:         $AddOtherJobs(T1)$
12:         Update $PrecedenceArray$
13:      else
14:         $p \leftarrow p + 1$
15:      end if
16:   end while
17: end while
18: Return $K$
```

Figure. 4.4 Pseudo code detailing the implementation of the greedy heuristic for the large scale technician and task scheduling problem datasets

4.4 Improvement Phase

Once an initial solution has been generated by the construction heuristic, the improvement phase begins. During the improvement phase, a local operator is applied generating a neighbour that can be evaluated using a metaheuristic. As the problems are large scale, a subset of the local operators featured in sections 3.4 and 3.5 were chosen: *Decompose and rebuild*, *Decompose and rebuild N*, *Remove N jobs*, *Remove a team*, and *Remove N teams*. These operators were chosen, as the local operator experimentation in section 3.4.4 indicated that in complex instances, set X datasets, these operators accounted for 94.8% of all improvements found.

In addition, these operators are able to change multiple parts of a solution rather than a single part. For example, these operators can change the distribution of the jobs and the configuration of the teams. Due to the flat objective function, and the size of the problems, operators that can provide significant change are needed. For the large scale technician and task scheduling problem instances a hill climbing metaheuristic is used. It can be thought of as the simplest metaheuristic procedure, as there are no parameters and therefore no tuning experiments to be performed.

4.5 Experimental Results

Under the competition rules of the ROADEF 2007 challenge, each run of the heuristic is allowed a 20 minute computational time limit and so in this research, a 20 minute run time is used. The heuristics were programmed in Java and tested on an HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ with 12GB of RAM. Table 4.3 presents the best result obtained over five runs for each heuristic. In columns two, three and four the best results found are displayed for the intelligent decision heuristic, the greedy heuristic and the look ahead heuristic respectively. The experimental aim is to find which heuristic procedure is the most effective when solving industrial sized scheduling problems.

4.6 Discussion

This chapter has presented a methodology for creating large scale technician and task scheduling problems. The data generator has been used to develop 12 large scale novel problem instances that will allow researchers to test the scalability of their solution approaches. These datasets are available from https://akhalfay.wordpress.com/large-scale-ttsps.

In section 4.5, the results obtained for each heuristic approach on the large scale technician and task scheduling problem instances is presented. It is clear that overall the intelligent decision heuristic outperforms the greedy heuristic in all problem instances, and the look ahead heuristic outperforms both heuristics in all but one of the problem instances. As the number of jobs to allocate increases the gaps between the heuristics increase with respect to the number of available technicians. For example

Table 4.3 Computational results for the intelligent decision, greedy and look ahead heuristic on the large scale technician and task scheduling problem instances

Dataset	Intelligent Decision	Greedy	Look Ahead
L1	192810	203850	**184440**
L2	97725	103440	**93690**
L3	48330	50700	**46380**
L4	296940	315210	**296295**
L5	**147480**	156960	147930
L6	76110	80880	**75660**
L7	420660	445335	**396015**
L8	198900	207405	**187170**
L9	97080	102870	**92160**
L10	574465	607890	**553560**
L11	280260	290745	**271020**
L12	140970	144840	**133800**
Average	214311	225844	**206510**

when evaluating the heuristics scheduling 1000, 1500, 2000 and 2500 jobs amongst 100 technicians (L3, L6, L9 and L12) the gap between the worst performing and best performing heuristic increases (4320, 5220, 10710 and 11040) as the number of jobs to allocate increases.

As the number of technicians available increases within the same set of jobs to allocate, the gap in solution quality generally decreases. This can be expected because as the number of available technicians on each day increases, there are fewer shortages of skills, and therefore less consideration needs to be made for team configurations.

Overall, the intelligent decision heuristic finds a solution that is on average 5% better than the quality of solution found by the greedy heuristic. A saving of 5% in personnel costs has the potential to save significant amounts of money in large businesses. Furthermore, the look ahead outperforms the intelligent decision heuristic finding a solution on average that is 3.7% better quality in terms of cost.

This research has demonstrated that although the intelligent decision heuristic is far more computationally expensive than the greedy heuristic, it can produce better quality results in the same computational time limit. In addition, it has shown that in these instances, the look ahead is a better solution approach than the intelligent decision heuristic by consistently producing superior results. The chapter has shown that both the intelligent decision and look ahead heuristic are both scalable approaches to solving large scale technician and task scheduling problems within strict computational time limits.

4.7 Summary

This chapter has illustrated the need to solve large scale technician and task scheduling problems that arise in real world industrial settings. As there are no large scale datasets available in the literature, which include the scale of complexities featured within the ROADEF 2007 challenge, the data has been generated in order to allow researchers to assess the scalability of solution approaches.

The benefits of finding efficient ways to solve large scale technician and task scheduling problems in time constrained conditions using the large scale data instances have been shown. In these large scale problems, finding a better quality solution of even 1% can result in large financial savings. The contributions to the field in this chapter are; (1) a methodology for creating large scale technician and task scheduling problem instances (which can be adapted in order to create other scheduling problems), (2) twelve novel large scale technician and task scheduling problems that can be used to test the scalability and suitability of heuristic approaches on realistically sized scheduling problems and (3) a comparative analysis of the greedy heuristic, intelligent decision heuristic, and the look ahead heuristic, which has provided a set of benchmark results on the large scale technician and task scheduling problem instances.

In this chapter, a precedence generating algorithm has been designed in order to add the complexity of precedence and successor relationships into the datasets. These constraints affect the quality of solution that can be obtained and arise in many practical situations such as housing development. However, no research has addressed the effect of precedence constraints on solution quality and the ability of forecasting. Therefore, in the next chapter, the complexity of precedence relationships across different scales of data instances is discussed.

Chapter 5

Precedence Constrained Technician and Task Scheduling Problems

5.1 Introduction

The purpose of work described in this chapter is to assess the complexity of precedence constraints within the field of technician and task scheduling problems. The problems studied so far in this thesis have included the complexity of precedence constraints, which arise in many application areas. To the author's knowledge, there has been no research undertaken in the field that specifically focuses on the complexity of precedence or its effect on the quality of solution that can be obtained. In this chapter, a set of precedence constrained technician and task scheduling problems are presented that have been created with the data generator, described in the previous chapter, which created the large scale problem instances.

The datasets generated have used the definition of the ROADEF 2007 challenge (Society, 2007) problem, where skilled workers must service jobs that require certain skill levels and includes the complexity of teaming, priority levels, precedence and an outsourcing budget. Research in this area aims to aid the task of accurately forecasting and predicting the expected objective value for a given level of precedence constraints between jobs and to understand the algorithmic performance of the heuristics developed so far, as the problems become more constrained as the level of precedence constraints between jobs increases.

Many organisations face the complex problem of scheduling staff to complete a set of jobs in the least costly way (Castillo-Salazar et al., 2012). Precedence constraints occur in industries such as maintenance and repair, for example, the need to collect a tool or spare part before going to complete a job (Pillac, Gueret and Medaglia, 2013). Additionally, in the housing development trade, there must be coordination of many different types of skilled workers, for example, plumbers, plasterers, roofers and scaffolders, and certain tasks must be completed before others may commence. Similarly, in the home health care industry, which is growing due to the number of private health care services and an ageing population. Many people prefer to be cared for in their own home and rely on third parties to administer care, such as medications, that have strict guidelines (Hiermann et al., 2015). More recent work, by Park et al. (2016), required scheduling unmanned aerial vehicles to perform tasks within the manufacturing sector. This problem included precedence constraints between tasks that must be completed. This work also included constraints such as travel time, recharge time of the unmanned aerial vehicle and time windows in which the task must be completed.

As shown in Chapter 2, there are limited problems in the literature that have included the complexity of precedence relationships, even though it is a common complication in many sectors. One of the most notable problems that include precedence constraints is the ROADEF 2007 challenge. Although this work included precedence constraints, there was no comparison made against the complexity that is added or the effect on the quality of solution that can be obtained.

This chapter is focused on the occurrence of precedence constraints within the field of technician and task scheduling. It is believed that this is the first research that specifically focuses on the impact that precedence relationships have on the quality of solution that can be obtained. In this research, 25 precedence constrained technician and task scheduling datasets, split into 5 groups that include scheduling between 100 and 1000 jobs have been created. Within each group, the percentage of precedence relationships between the jobs varies from 0% to 100%, in order to measure the effects of precedence relationships.

The remainder of this chapter is structured as follows, section 5.2 discusses how the varying levels of precedence relationships were created using the data generator and presents the datasets created, and section 5.3 describes the three heuristic approaches used, featured in Chapters 3 and 4. Section 5.4 describes the metaheuristic used to guide the lower level heuristics, a multi start hill climbing metaheuristic, and section 5.5 shows the computational results obtained on the precedence constrained technician and task scheduling problems. Lastly, section 5.6 discusses the results presented and section 5.7 concludes on the research undertaken in this chapter.

5.2 Generating Precedence Constrained Technician and Task Scheduling Problem Instances

5.2.1 Creating Varying Precedence Levels

In order to create datasets which contain varying levels of precedence constraints within the same set of jobs (in terms of skill requirements, priority levels, and outsourcing costs) a methodology for dataset creation had to be designed. The methodology had to ensure that every dataset, containing the same number of jobs, of precedence level p has all the precedence relationships present in the datasets of precedence levels lower than p. For example, dataset P2 has 100 jobs to allocate and contains 25% precedence relationships, therefore dataset P3, contains all the precedence relationships contained in dataset P2, plus another 25% of precedence relationships.

Figure 5.1 illustrates the precedence relationships present within a small scale dataset that has 30 jobs to schedule. This dataset contains 50% precedence relationships, meaning that half of the jobs have at least one connection to another job (precedence or successor). It is clear there are four sets of precedence relationships. In one set of

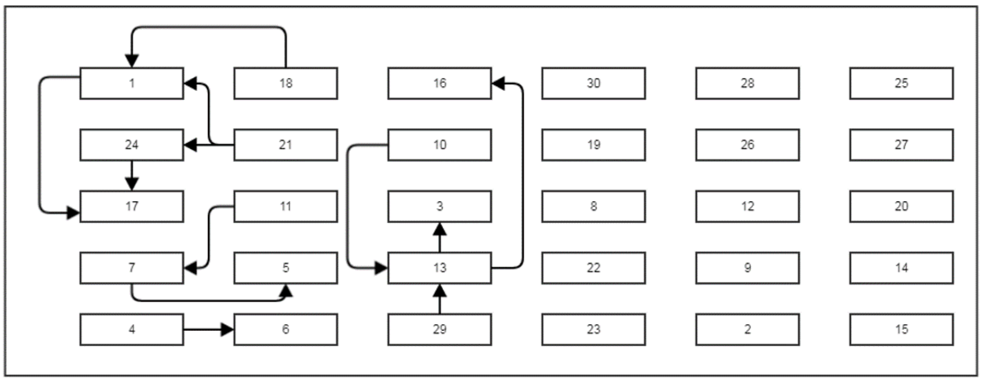


Figure. 5.1 Diagram showing 50 % precedence relationships between a set of jobs

precedence relationships, containing jobs *1*, *17*, *18*, *21* and *24*, there are three levels. On the first level are jobs *18* and *21*. On the second level are jobs *1* and *24*, and finally on the third level is job *17*. In this relationship tree, job *17* is dependent on job *1* and job *24*. Both Job *1* and *24* are dependent on job *21*. Lastly, job *1* is also dependent on job *18*. Scheduling jobs *18* and *21* early in the scheduling horizon is of high importance, as jobs *1* and *24* may not be scheduled until these jobs have been completed.

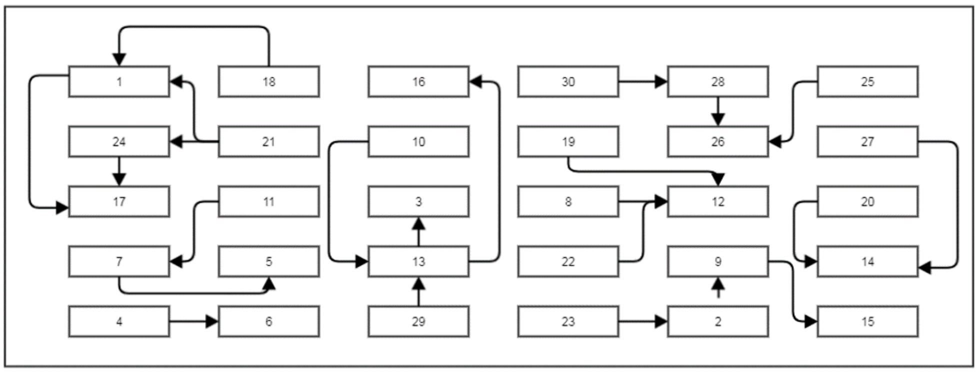


Figure. 5.2 Diagram showing 100 % precedence relationships between a set of jobs

In Figure 5.2, there are 100% precedence relationships between the set of jobs, meaning that all of the jobs have at least one connection to another job (precedence or successor). This dataset contains all of the relationships that were present in Figure 5.1.

5.2.2 Dataset Generator

Figure 5.3 illustrates how the precedence constrained datasets were generated under the framework of the ROADEF 2007 challenge problem. Firstly, the instance file is generated containing information about the problem. Next, the set of technicians is created and each is assigned skills and days off. The jobs are then created and assigned skill requirements, durations, priority levels, and outsourcing costs. The precedence algorithm then begins which creates multi layered relationships between jobs. Precedence levels to 25% are created and an output file is written, then the precedence relationships are built to 50% and again an output file is written. This continues until the precedence levels have reached 100%.

This data generator differs from the data generator featured in the previous chapter in Figure 4.1 because for each run 5 job files will be created. The first will contain no precedence constraints i.e 0%, then 25%, 50%, 75% and finally 100%. This will enable the complexity of precedence relationships to be understood and help ascertain which heuristic procedures work well on datasets of varying size and complexity.

5.2.3 Precedence Constrained Instances

The data can be separated into 5 groups, P1-P5, P6-P10, P11-P15, P16-P20 and P21-25 with 100, 200, 400, 800 and 1000 jobs to schedule respectively. Within each group, the percentage of precedence constraints vary, 0%, 25%, 50%, 75% and 100%. Within each group of data, the remaining characteristics of the datasets are the same, number of technicians, domains and levels, and outsourcing budgets, in order to evaluate the effect of precedence constraints on the quality of solution that can be achieved. A table showing each datasets characteristics can be found in Appendix A Table A.2.

5.3 Heuristic Approaches

In order to provide a performance comparison, the three heuristic procedures, the intelligent decision heuristic, the look ahead heuristic, and the greedy heuristic, which

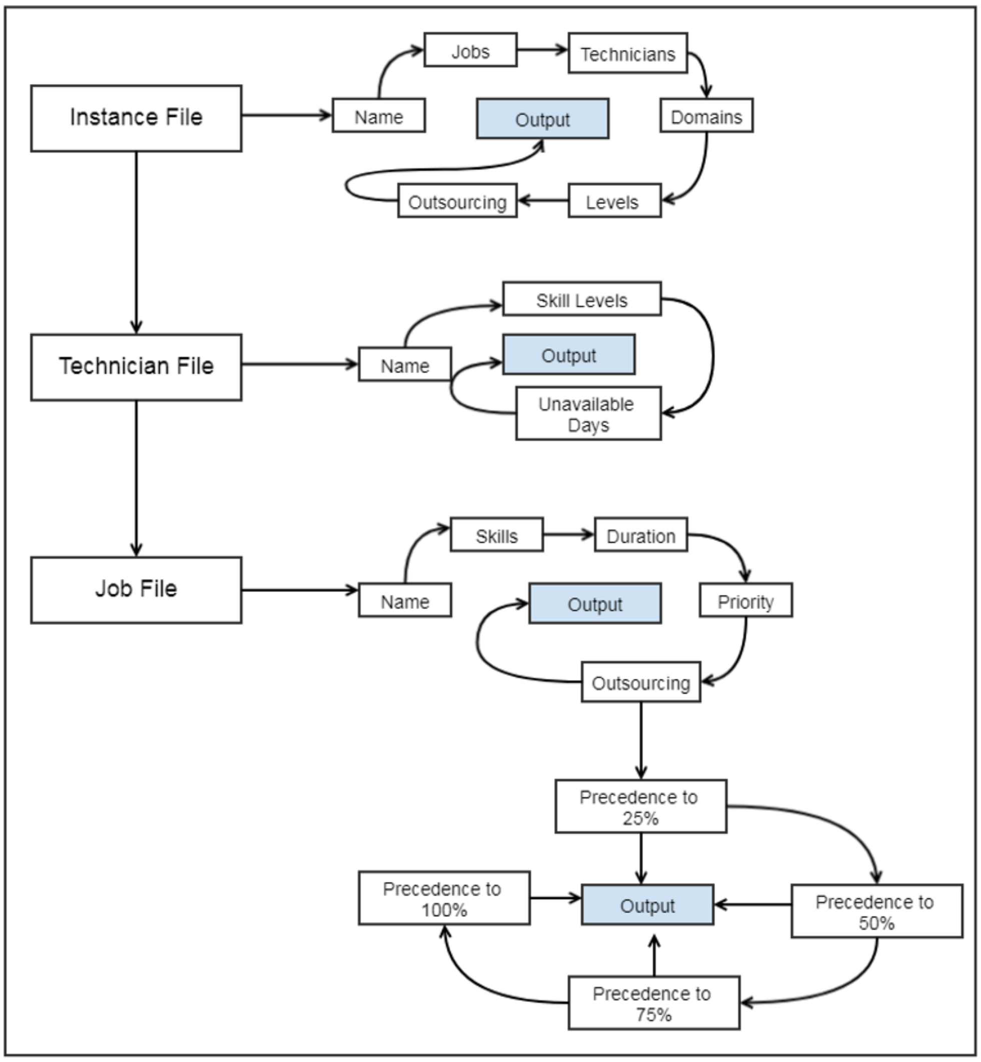


Figure. 5.3 Flowchart describing the creation of the precedence constrained technician and task scheduling problem datasets

were also applied to the large scale problems in Chapter 4, are applied to precedence constrained technician and task scheduling problem instances to test performance and provide a comparative analysis. Once an initial solution has been generated, local operators are used to try and find better quality solutions. The operators featured in Chapter 4 are used on the precedence constrained problem instances. The operators used are *Decompose and rebuild*, *Decompose and rebuild N*, *Remove N jobs*, *Remove a team*, and *Remove N teams*.

5.4 Metaheuristic

In this chapter, a multi start hill climbing metaheuristic to guide the lower level heuristics has been implemented. Multi start techniques have proved popular strategies in many combinatorial optimisation problems (Blum and Roli, 2003).

5.4.1 Multi Start Hill Climbing

The multi start hill climbing metaheuristic presented in this chapter repeatedly takes an initial solution generated by the heuristic (intelligent decision, greedy, or look ahead), and then uses a pre-defined amount of computational time to try and improve it. The best solution found is updated as better quality results are discovered. This process continues, until the total amount of computational time has been reached.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators

```
 1: Initialise $S_{Best}$
 2: while total time not met do
 3:   $S \leftarrow$ Generate initial solution using heuristic
 4:   while improvement time not met do
 5:     $o \leftarrow O$
 6:     $S' \leftarrow S(o)$
 7:     if $S' \leq S$ then
 8:       $S \leftarrow S'$
 9:       if $S \leq S_{Best}$ then
10:         $S_{Best} \leftarrow S$
11:       end if
12:     end if
13:   end while
14: end while
15: return $S_{Best}$
```

Figure. 5.4 Pseudo code showing the implementation of the multi start hill climbing metaheuristic

Figure 5.4 shows the implementation of the multi start hill climbing metaheuristic. On line 1, the best solution is initialised, which will keep track of the best quality solution found over the whole run. Whilst the algorithm's total run time has not been met, an initial solution is generated on line 3 using the heuristic procedure (either intelligent decision, greedy, or look ahead). Next, while there is improvement time remaining, a local operator is selected and then applied to the current solution, which generates the neighbouring solution S' on line 6. If solution S' is of better quality than S then it replaces S on line 8. If the new solution S is of better quality than the best solution found then S_{Best} is updated. The algorithm continues to iterate, generating initial solutions, and then trying to improve them until the total run time has been used. Once the total run time has been used, the best solution found over the whole run is output.

5.4.2 Multi Start Tuning

In order to use multi start hill climbing, the frequency of restart had to be specified. The experiments use 10 minute runs in total and so, four time intervals for restarts were initially proposed. The restarts ranged from 15 to 60 seconds in 15 second increments. The parameter is powerful as it dictates the amount of search the heuristic does before beginning the search again. This approach differs from iterative local search in the following ways, (1) when a restart is triggered a new initial solution is created and (2) the triggering of a restart is time dependent not search dependent. Each experiment was run 5 times on each of the datasets tested and the mean objective value was recorded. Figures 5.5, 5.6 and 5.7 show the results obtained for problem instances with 100, 400, and 800 jobs.

Figure 5.5 indicates there are differences in the expected solution value dependent on the value of the restart. Each level of precedence has been tested from 0 to 100%. On analysis of the mean results produced across the different restart values, using 30 seconds seems to produce on average higher quality objective values.

Figure 5.6 illustrates that these datasets are suited to a smaller restart time, with the lower objective values found using a 15 second restart. These results may be justified by the size of the problem which means there are many more combinations and therefore frequent searches from different starting points may lead to better quality solutions. .

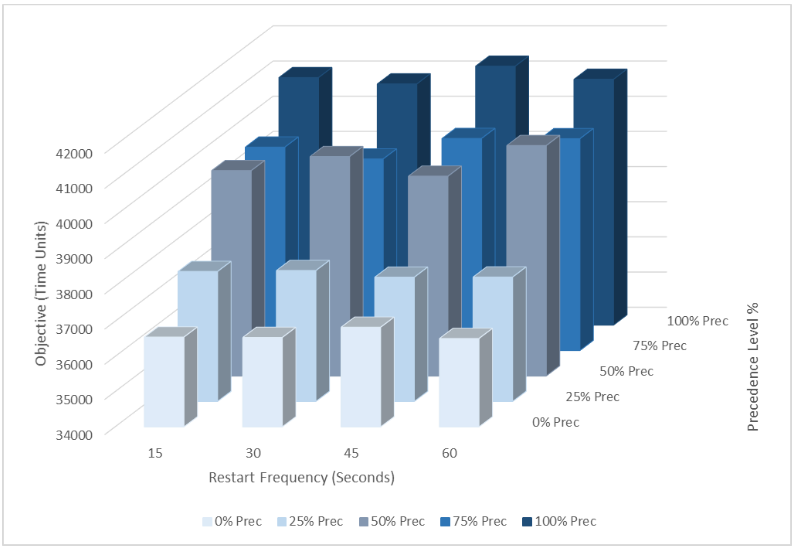


Figure. 5.5 Bar chart showing the average objective values obtained using different values of restart on jobs with between 0-100 % on a 100 job problem instance

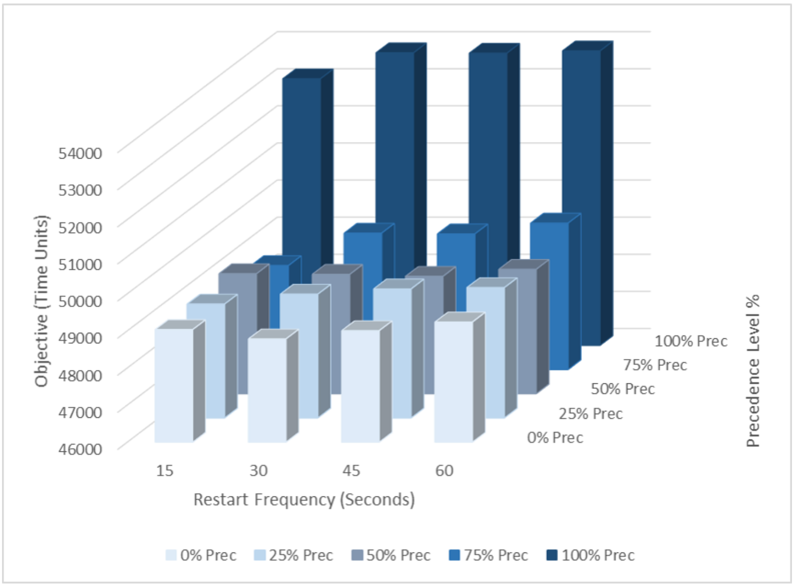


Figure. 5.6 Bar chart showing the average objective values obtained using different values of restart on jobs with between 0-100 % on a 400 job problem instance

Figure 5.7 demonstrates that these datasets generally find lower objective values using a 30 second restart. These results, again, may be explained by the size of the problem being solved. For example, in such a large problem frequent restarts may be needed but coupled with some search time to navigate towards quality solutions.

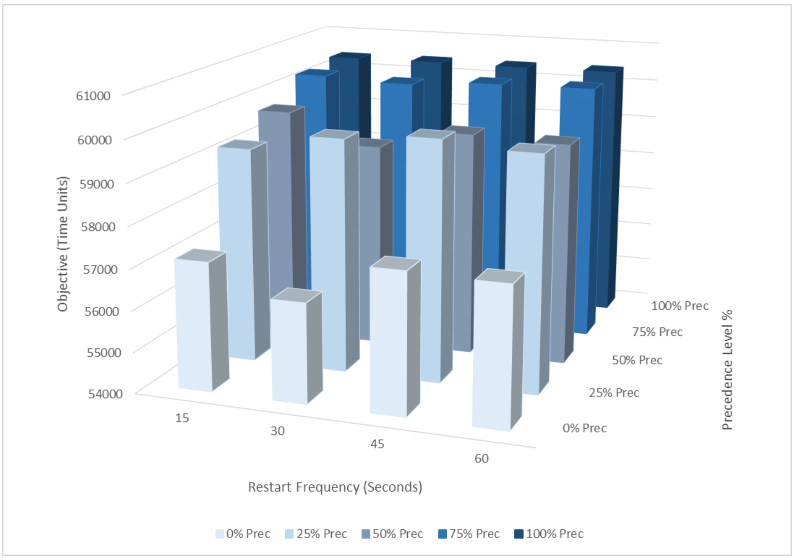


Figure. 5.7 Bar chart showing the average objective values obtained using different values of restart on jobs with between 0-100 % on a 800 job problem instance

5.5 Experimental Results

The precedence constrained datasets have been tested using the following experimental framework. For each run, a 10 minute computational time is allowed, in order to accurately assess heuristic performance in reasonable computational times. Each heuristic was run 5 times and the best solution found was recorded. The algorithms were written in Java, and tested on an HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ with 12GB of RAM, to be able to compare the results achieved for each heuristic tested. The aim of these experiments is to (1) obtain a measure of how solution quality is affected by precedence relationships and (2) provide a comparative analysis of the heuristics developed in this research and their robustness dealing with precedence constraints.

Table 5.1 shows the results obtained using the three heuristic procedures, the intelligent decision, the greedy heuristic, and the look ahead heuristic. It is clear that the greedy heuristic does not perform as well as the intelligent decision or the look ahead heuristic, finding the highest objective value in each of the datasets tested.

Table 5.1 Table showing the average objectives achieved using the intelligent decision, greedy and look ahead heuristic on the precedence constrained problem instances

Name	Intelligent Decision	Greedy	Look Ahead
P1	**31500**	33960	31650
P2	**32340**	36240	32700
P3	**35040**	38220	36180
P4	**35580**	39150	37560
P5	**36240**	40440	37890
P6	52200	54480	**51990**
P7	53730	54630	**52440**
P8	57510	59400	**57240**
P9	62520	63750	**60450**
P10	65220	66330	**63960**
P11	46290	49680	**45690**
P12	**46290**	49770	46515
P13	46410	49770	**46080**
P14	**46670**	49800	47820
P15	48600	54060	**48480**
P16	**53310**	63840	54720
P17	**54270**	63960	54720
P18	55320	64950	**54840**
P19	56220	68235	**54855**
P20	58530	69180	**58020**
P21	**48840**	51720	49095
P22	49830	51720	**49740**
P23	51180	56880	**51120**
P24	51210	56850	**51120**
P25	51450	57480	**51120**
Average	49052	53779.8	**49039.8**

5.5.1 Equality of Means across Precedence levels

This research has been conducted based on the belief that precedence constraints impact the quality of solution that can be obtained. If two datasets are taken, for example, P1 and P2, which contain the same set of 100 jobs, but P1 has no precedence constraints whereas P2 has 25% precedence constraints. It is expected that the mean objective function value on each of these datasets will not be equal.

To test this, a hypothesis test has been carried out. The null hypothesis, shown in Equation 5.1, h_0 claims there is no difference in the mean objective value between the groups. Conversely, the alternate hypothesis described in Equation 5.2, h_1 specifies that there is a difference in the mean objective value obtained, such that the mean value of group 1, x_1 is less than the mean objective of group 2, x_2.

$$h_0 : x_1 - x_2 = 0 \tag{5.1}$$

$$h_1 : x_1 - x_2 < 0 \tag{5.2}$$

A series of experiments were conducted, the look ahead heuristic was run 20 times on each dataset, P1 and P2, with the objective values recorded. The mean value obtained for each was calculated along with the standard deviation. Next, the standard error between both groups was calculated as well as the degrees of freedom. A T statistic was calculated, 14.47, which is compared to a T value, in a one tailed test on 37/38 degrees of freedom. A T value is calculated and compared to the T statistic. In this case, with a significance level of α = 0.01, on a one tail test, the T Value is 2.431 which is less than 14.747. This confirms that there are significant grounds to reject h_0 and accept the alternate hypothesis, that the mean objective of group 2 is higher than the mean objective of group 1.

5.5.2 Effect of Precedence Constraints on Solution Quality

The purpose of this research is to estimate the effect of precedence constraints within the field of technician and task scheduling. Within each set of data, varying levels of precedence constraints were used and the effect that it has on the quality of solution can be examined, across different scales of the problem. The mean score obtained for each scale of the problem across varying levels of precedence constraints has been plotted. The regression line which has been calculated, shown in Figure 5.8, demonstrates how the quality of the solution is affected by the percentage of precedence constraints within the datasets.

Table 5.2 Table showing the calculated values for the one sided T Test on the equality of means

	Group 1	Group 2
Sample Size	20	20
Mean	36493.5	37700.25
Standard Deviation	251.276	266.03
T Statistic	$\frac{(37700.25-36493.5)}{81.827} = 14.747$	
T Value	2.431	

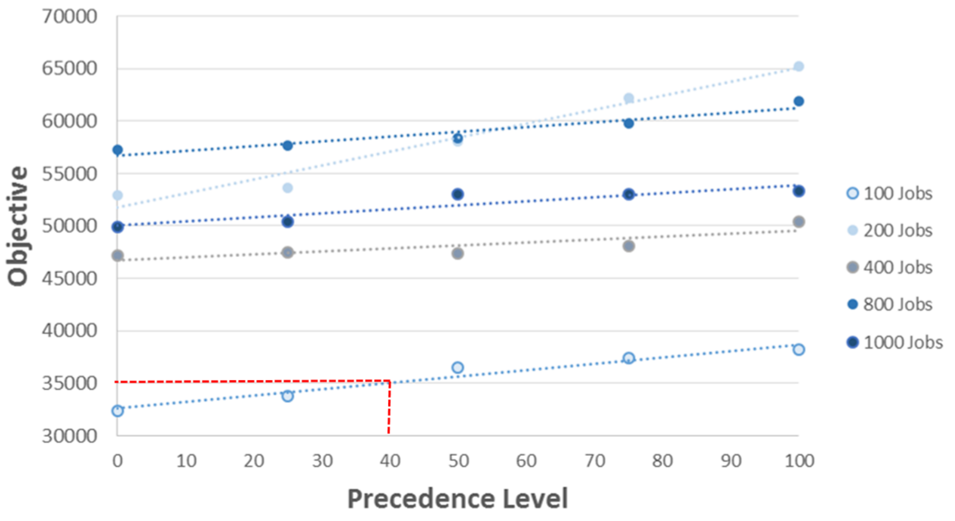


Figure. 5.8 Graph showing the mean value produced for each dataset using each level of precedence exhibiting a positive correlation

Figure 5.8 illustrates that as the level of precedence constraints increases, a larger objective value should be expected. This is due to the extra complexity precedence constraints bring. Using this plot, an estimate of the expected objective value given a precedence level of $x\%$ can be made. For example, using a dataset with 100 jobs, with a precedence level of 40% will result in an expected objective value of 35000.

This power of estimation may aid the industrial world. In the building/housing development industry large projects are often forecasted or estimated both in terms of time and cost without mathematical aid. Naturally, it is common for projects to overrun both in terms of time and cost which can be attributed to the knock on effects a single point delay can have on the rest of the schedule. This work may help reduce the difference between the forecasted and actual costs incurred, thereby saving money.

5.6 Discussion

This chapter has been focused on creating a set of precedence constrained technician and task scheduling problems and solving the datasets with the heuristic procedures featured in the previous chapters. A set of 25 instances was created ranging from 100-1000 jobs to allocate. The instances can be split into 5 sets, in each set the level of precedence constraints ranges from 0% to 100%.

The results presented showed that the greedy heuristic did not perform as well as the look ahead or the intelligent decision heuristic, and consistently found the highest objective values in all datasets. This is because the greedy heuristic makes the locally optimal choice at each stage without considering how this may affect the current team or subsequent teams that may form or future allocations. Furthermore, as the problem size increases, the gap between the performance of the greedy and other heuristic increases indicating that the greedy heuristic is not a scalable approach.

With regards to the performance of the intelligent decision heuristic and the look ahead heuristic, they both achieve a similar average objective value overall. When considering each level of precedence there is a notable difference. On every set of data, except 0% precedence constraints, the look ahead heuristic finds a lower objective value than the intelligent decision heuristic.

5.7 Summary

This chapter has shown that precedence relationships are an important consideration for many personnel scheduling problems that arise in multiple sectors, such as; service maintenance, housing developments, and home health care.

This chapter has: (1) presented 25 precedence constrained technician and task scheduling problem datasets that include varying levels of precedence constraints available to other researchers at https://akhalfay.wordpress.com/precedence-constrained-technician-and-task-scheduling, (2) a precedence constrained data generator and precedence algorithm, (3) provided a set of benchmark results on the datasets for future comparative analysis, and (4) a comparison of the effects of precedence relationships across three heuristic procedures, the intelligent decision heuristic, the greedy heuristic, and the look ahead heuristic.

The next chapter will consider location and travel time constraints, an important consideration in technician and task scheduling problems, that is not featured in the ROADEF 2007 challenge.

Chapter 6

Multi-Period Technician Routing and Scheduling Problem

6.1 Introduction

In this chapter, a set of novel multi-period technician routing and scheduling problems are solved. Multi-period means that the problem covers more than one scheduling day. So far, the problems studied in this thesis have included some of the most important considerations within the field of technician and task scheduling (priority, teaming, precedence, and outsourcing), however routing and travel time have not yet been considered. The aim of this chapter is to add another dimension (location) to the ROADEF 2007 challenge problem. By adding the complexity of location and travel time, a set of novel problem instances have been created, which are available to researchers. These datasets have been used to test the robustness of the heuristics developed so far in this thesis to understand how easily they can be adapted to changing problem definitions and constraints to ascertain their applicability in commercialised scenarios.

The aim of this problem is to construct a set of teams over multiple days to complete a set of jobs. Each job has a set of skill domain requirements that must be satisfied by the team which services the job, a priority level indicating the importance of the job, and an outsourcing cost. On each day, teams (made up of technicians) must depart from and return to a central depot. The travel time between jobs and the depot is accounted for and is calculated as the Euclidean distance. Euclidean distance was chosen as it is the most common way used in the literature of calculating the distance, refer to Pillac, Gueret and Medaglia (2013) and Kovacs et al. (2012).

In some cases, there is also an outsourcing budget which may be used. Outsourced jobs do not contribute to the objective function so utilisation of this budget is of paramount importance. This multi-period technician routing and scheduling problem is a novel problem as it requires scheduling a workforce over multiple days, includes the unavailability of resources, teaming, priority levels, and solves problems ranging from 5 to 800 jobs, with a workforce of 5 to 150 technicians, and comes from real world data.

The remainder of this chapter is structured as follows, section 6.2 presents the mathematical formulation of the multi-period technician routing and scheduling problem and section 6.3 discusses the datasets that have been creating by adapting the ROADEF 2007 challenge instances. Section 6.4 describes each of the heuristics implemented; a greedy heuristic, intelligent decision heuristic, and a look ahead heuristic. Section 6.5 provides a brief description of the local operators used and acceptance criteria and section 6.6 presents the computational experiments performed. Lastly, section 6.7 displays the experimental results and section 6.8 discusses the results presented.

6.2 Multi-Period Technician Routing and Scheduling Problem Formulation

In this chapter, the problem definition of the ROADEF 2007 challenge problem has been extended to include the complexity of location and travel time. However, the complexity of precedence and successor relationships has been removed from the datasets as early computational experiments showed that they conflicted with the optimisation of minimising the routing costs. Th mathematical formulation of this multi-period technician routing and scheduling problem is one of the contributions of this research.

This problem is novel due to the complexity of the problem instances that are used, the planning period is multi day, adapted from real world problems, scheduling a large number of jobs, building teams, and includes the complexity of unavailability of resources. Other works in the literature are generally based on solving single period problems that contain up to 100 jobs, for example, Kovacs et al. (2012) and Pillac, Gueret and Medaglia (2013). In most real world applications, the workers will be scheduled over a planning period, for example, a week or a month etc. with many jobs to allocate, therefore focusing on solving realistically sized problems (with hundreds of jobs) will have the most benefit to the industrial world.

The multi-period technician routing and scheduling problem can be defined as a complete directed graph $G = \{V, A\}$, where V is the set of all vertices i.e the set of jobs, and A a set of arcs between the jobs. Once some jobs have been outsourced the set V is reduced to V'. The solution comprises of a scheduling horizon, a set of days $K = \{1, \ldots, k\}$. There is a set of technicians $T = \{1, \ldots, t\}$, and the set of available technicians on day k is denoted as T_k. On each day k there is also a set of teams τ_k, made up of technicians. The budget for outsourcing is denoted as C. The length of a working day is $Maxday$. The starting depot on each day is denoted as 0_k and the ending depot as N_k.

Technicians have intrinsic skills $s \in S$ and varying levels $l \in L$ within each area of skill. A technician's skill can be represented by an $L \times S$ [0,1] matrix where $p^t_{l,s}$ denotes the level of expertise the technician has in skill area s to level l. Skill levels are hierarchical so if $p^t_{l,s} = 1$ then, $p^t_{l',s} = 1$ for $l' < l$.

Jobs have a service time denoted by d_i, and an outsourcing cost o_i. Each job also has a priority level p_i where $p_i \in \{1, \ldots, 4\}$ describing how important it is to serve the job as early as possible. Each job i has a skill requirement matrix, of size $L \times S$, denoted as q_i.

The problem can now be represented as;

$$\min \sum_{p=1}^{4} w_p \cdot m_p \tag{6.1}$$

The objective function is a weighted sum of the latest ending times, m_p, of each priority group for $p = \{1 \dots 4\}$ where $w = [28, 14, 4, 1]$.

Subject to;

$$m_p \geq E_i^\tau \quad \forall p = [1, ..., 3] \quad i \in V_p' \tag{6.2}$$

Equation 6.2 states that the latest ending time of jobs of priority p where $p = \{1 \dots 3\}$, must be greater than or equal to the ending time of all jobs of that priority level, V_p' .

$$m_4 \geq E_i^\tau \quad \forall \quad i \in V' \tag{6.3}$$

Equation 6.3 states that m_4 must be greater than or equal to the ending time of all jobs belonging to the set V'.

$$\sum_{\tau \in \tau_k} v_{t,k}^\tau \leq 1 \quad \forall \quad t \in T_k, \quad k \in K \tag{6.4}$$

Equation 6.4 ensures that if a technician is available to work on day k then the technician may belong to one team only.

$$\sum_{\tau \in \tau_k} v_{t,k}^\tau = 0 \quad \forall \quad t \notin T_k, \quad k \in K \tag{6.5}$$

Equation 6.5 guarantees that if a technician is not available to work on day k then the technician may not belong to any team.

$$\sum_{k \in K} \sum_{\tau \in \tau_k} y_{ik}^\tau + z_i = 1 \quad \forall \quad i \in V \tag{6.6}$$

Equation 6.6 ensures that each job is either outsourced or it is allocated to a team during the scheduling horizon.

$$\sum_{j \in V'} x_{0_k,j,k}^\tau = 1 \quad \forall \quad k \in K, \quad \tau \in \tau_k \tag{6.7}$$

$$\sum_{i \in V'} x_{i,N_k,k}^\tau = 1 \quad \forall \quad k \in K, \quad \tau \in \tau_k \tag{6.8}$$

Equations 6.7 and 6.8 ensure that on each day the teams depart from and return to the central depot.

$$\sum_{j \in V'} x^{\tau}_{j,i,k} = y^{\tau}_{i,k} \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.9}$$

$$\sum_{j \in V'} x^{\tau}_{j,i,k} - \sum_{j \in V'} x^{\tau}_{i,j,k} = 0 \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.10}$$

Equations 6.9 and 6.10 ensure that if a job is assigned to a team on day *k* then the team enters and leaves the job's location.

$$\sum_{i \in V} z_i \cdot o_i \leq C \tag{6.11}$$

Equation 6.11 states that the total costs incurred by outsourcing jobs do not exceed the maximum budget permitted.

$$B^{\tau}_{j} \geq (E^{\tau}_{i} + c_{i,j}) \cdot x^{\tau}_{i,j,k} \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.12}$$

Equation 6.12 states that if two jobs happen sequentially, the start time of j must be equal to or greater than the ending time of i plus the travel time between i and j.

$$B^{\tau}_{0_k} = (k-1) \cdot Maxday \quad \forall \quad k \in K \quad \tau \in \tau_k \tag{6.13}$$

Equation 6.13 sets the beginning time of each day within the scheduling horizon.

$$B^{\tau}_{N_k} \leq (k) \cdot Maxday \quad \forall \quad k \in K \quad \tau \in \tau_k \tag{6.14}$$

Equation 6.14 ensures that each team arrives back at the depot before the end of the working day.

$$y^{\tau}_{i,k} \cdot q^{i}_{l,s} \leq \sum_{t \in T_k} p^{t}_{l,s} \cdot v^{\tau}_{t,k} \quad \forall \quad i \in V', \quad l \in L, \quad s \in S, \quad k \in K \tag{6.15}$$

Equation 6.15 ensures that if a job is allocated to a team, the team collectively has the required expertise to service the job.

$$E_i^\tau \geq (B_i^\tau + d_i) \cdot y_{i,k}^\tau \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.16}$$

Equation 6.16 sets the ending time of each job belonging to V' as the beginning time of the job, plus the duration of the job.

$$E_i \cdot y_{i,k}^\tau \leq (k-1) \cdot MaxDay \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.17}$$

$$B_i \cdot y_{i,k}^\tau \geq (k-1) \cdot MaxDay \quad \forall \quad i \in V', \quad k \in K, \quad \tau \in \tau_k \tag{6.18}$$

Lastly, equations 6.17 and 6.18 ensure that the start and end times of jobs are within the working day. With variables;

$$B_i^\tau \geq 0 \quad \forall \quad i \in V' \tag{6.19}$$

$$E_i^\tau \geq 0 \quad \forall \quad i \in V' \tag{6.20}$$

$$x_{i,j,k}^\tau = \{0,1\} \quad \forall (i,j) \in A, \tau \in \tau, k \in K \tag{6.21}$$

$$y_{i,k}^\tau = \{0,1\} \quad \forall i \in V', \tau \in \tau, k \in K \tag{6.22}$$

$$v_{t,k}^\tau = \{0,1\} \quad \forall t \in T, \tau \in \tau, k \in K \tag{6.23}$$

$$z_i = \{0,1\} \quad \forall \quad i \in V \tag{6.24}$$

6.3 Generating Multi-Period Technician Routing and Scheduling Problem Instances

In the ROADEF 2007 challenge problem, there were three sets of data; A, B and X, that increased in complexity. Complexity can be measured in terms of the number of jobs to allocate, the number of technicians available, outsourcing budget and the number of domains and skill levels. Within each set of data, there were 10 datasets. Each dataset had a set of jobs, a set of technicians, an outsourcing budget, and skill domain levels.

The aim of the multi-period technician routing and scheduling problem is to utilise the outsourcing budget and serve the remaining jobs by travelling to service them in the least costly manner.

Table 6.1 Table showing the multi-period technician routing and scheduling problem data instances, T1 to T30

Dataset	Jobs	Techs	Budget	Domains	Levels
T1	5	5	0	3	2
T2	5	5	0	3	2
T3	20	7	0	3	2
T4	20	7	0	4	3
T5	50	10	0	3	2
T6	50	10	0	5	4
T7	100	20	0	5	4
T8	20	0	5	4	4
T9	100	20	0	5	4
T10	100	15	0	5	4
T11	20	300	400	4	4
T12	300	30	300	5	3
T13	400	40	500	4	4
T14	400	30	300	40	3
T15	500	50	900	7	4
T16	30	300	800	3	3
T17	500	100	500	10	5
T18	800	150	500	10	4
T19	120	60	100	5	5
T20	120	40	500	5	5
T21	600	60	50	15	4
T22	800	100	500	6	6
T23	300	50	1000	20	3
T24	800	70	50	15	4
T25	600	60	50	15	4
T26	200	20	500	6	6
T27	300	50	1000	20	3
T28	100	30	150	15	7
T29	500	50	50	15	4
T30	500	40	500	15	4

Table 6.1 shows the ROADEF 2007 challenge problem datasets that have been extended to become multi-period technician routing and scheduling problem datasets. Each dataset from the ROADEF 2007 challenge has been extended to include the complexity of location and travel time. In each instance, the customers' locations are randomly generated to be contained within the grid which is of size [100, 100]. The depot, which teams depart from and return to, is centrally located on the grid at coordinates [50, 50]. In order to add the complexity of location and travel time, the length of a working day also had to be modified from the ROADEF 2007 challenge problem formulation. Originally, the maximum day length was 120 time units, it is now 500 time units. The cost of travelling between customers i and j is calculated as the Euclidean distance between the points, as shown in Equation 6.25.

$$c(i,j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \quad \forall i,j \in V' \tag{6.25}$$

6.3.1 Solution Visualisation

Figure 6.1 shows a solution to dataset T5, generated by the adapted look ahead heuristic, which was previously dataset A5 in the ROADEF 2007 challenge. This dataset has 50 jobs to schedule and no available outsourcing budget. A key is used to colour code, priority one, two, three and four jobs, and travel time. On day one, it is clear that there are four teams ($Team9,2$, $Team6$, $Team3,1,7,8,10$, and $Team4$) that each have job allocations.

Figure 6.1 shows a tightly packed schedule. Each team returns to the depot before the end of the working day. There appears to be a minimal amount of slack time left by each team, meaning that the algorithm has utilised the available time. Furthermore, the distances travelled by the teams are relatively short, meaning that the routing costs have been minimised.

The distribution of customer locations is shown in Figure 6.2 for dataset T5. It is evident that the depot is situated in the middle of the region at location [50,50], and that

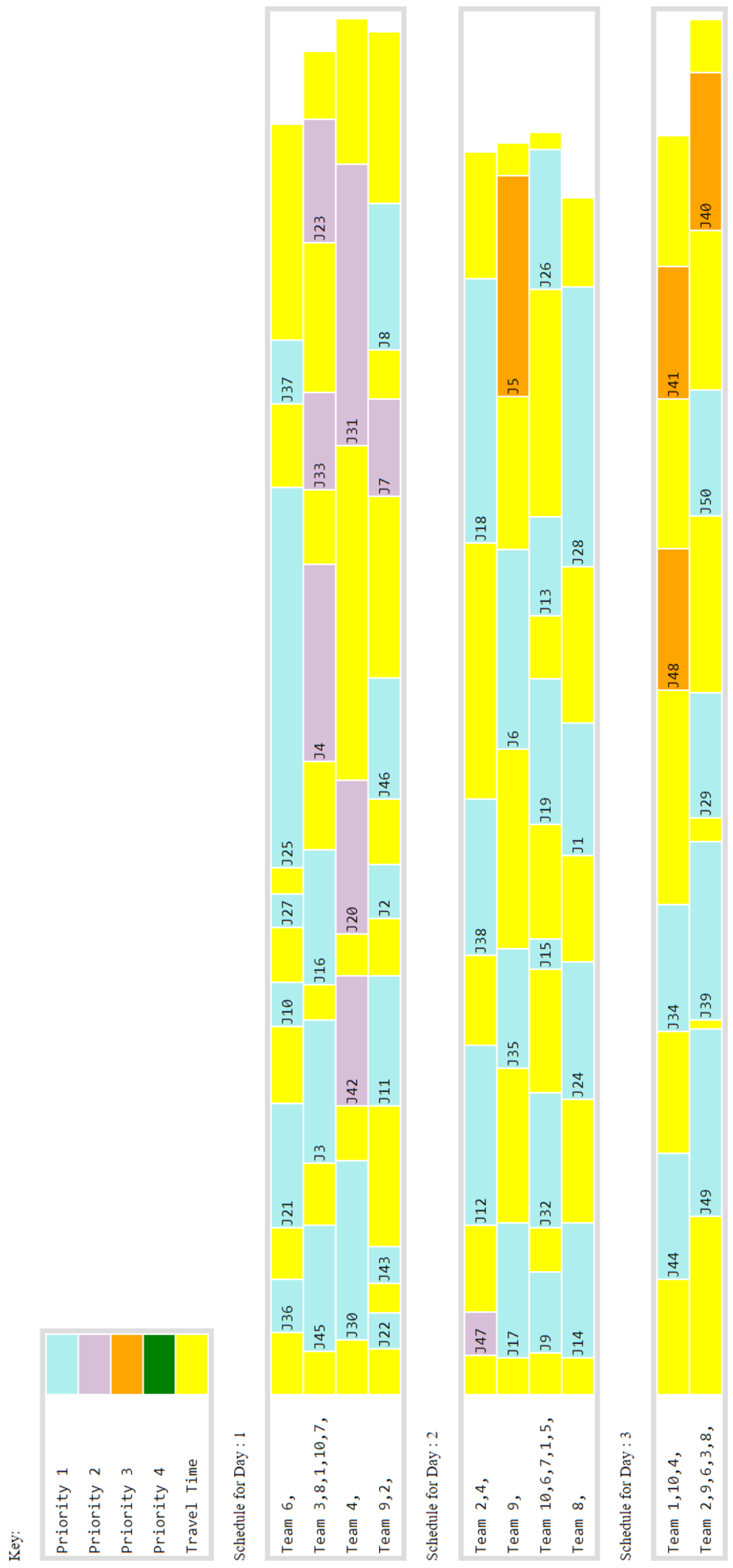


Figure. 6.1 HTML output showing a solution to dataset T5, which has 50 jobs and a pool of 10 technicians. Jobs are colour coded to show their priority levels.

the customers are located randomly within the region. A random location placement was chosen, ensuring that no customers were situated in the same place.

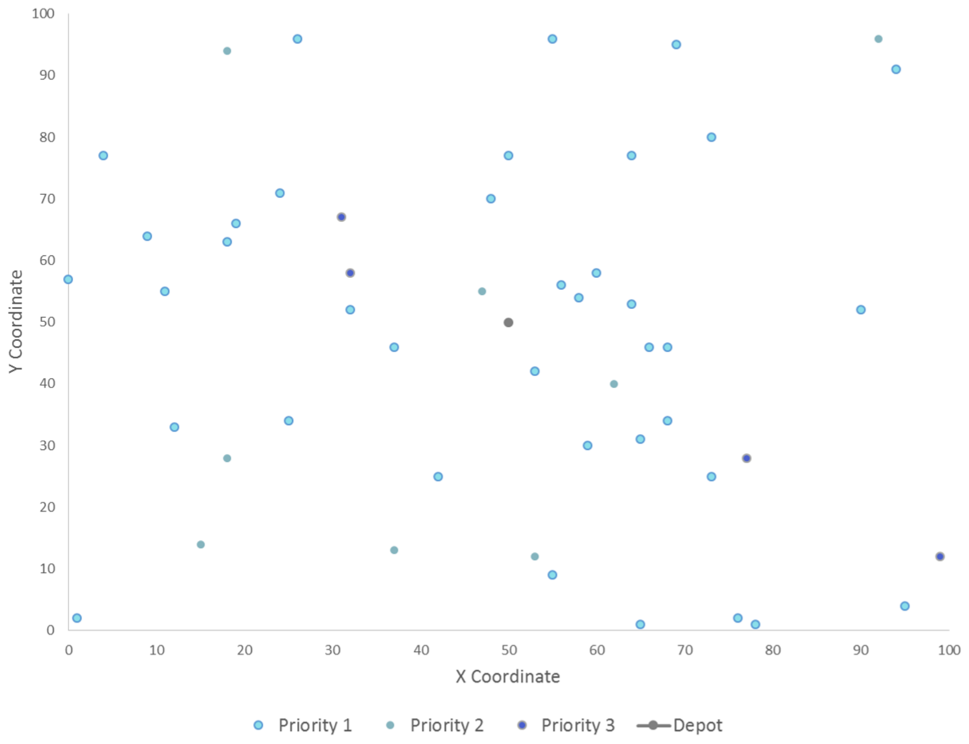


Figure. 6.2 Scatter plot showing the distribution of the customer's locations in dataset T5, each job is randomly located

6.4 Heuristic Approaches

In the previous chapters, the performance of the intelligent decision heuristic and the look ahead heuristic has been compared using the greedy heuristic as a baseline. In Chapters 3, 4, and 5, the look ahead heuristic has outperformed the intelligent decision heuristic, whilst both have outperformed the greedy heuristic.

6.4.1 Adapted Greedy Heuristic

The greedy heuristic presented makes the locally optimal choice at each stage of the scheduling process. At each stage, it selects a job for allocation and builds a team to service the job. If the team has slack time after being allocated an initial job, jobs are iteratively added to the team using a weighted scoring function. The weighted scoring function takes into account the jobs the team can complete (in terms of skill), the distance to the jobs from the team's current location, and the slack time the team would have if allocated the job. The greedy heuristic behaves differently to the look ahead heuristic because it is not prohibitive (does not contain the look ahead element), as no jobs are excluded from being selected for allocation. The pseudo code for the adapted greedy heuristic is shown in Figure 6.3.

Variables: V: set of jobs, V': set of jobs that are allocated to teams, τ: the set of teams, K: the scheduling horizon (collection of days), k: a day in the scheduling horizon, O: outsource list, i: job selected for allocation, τ_i: team selected to serve job i

```
1:  $O \leftarrow selectoutsourced(V)$
2:  while $V'$ is not empty do
3:     Initialise $k$
4:     while $k$ is not full do
5:        $i \leftarrow selectjob(V')$
6:        $\tau_i \leftarrow maketeam(k, \tau)$
7:        $addjob(i, \tau_i)$
8:        while $\tau_i$ $canallocate$ do
9:           $\tau_i \leftarrow addjob(V')$
10:       end while
11:    end while
12: end while
13: return $K$
```

Figure. 6.3 Pseudo code showing the adapted greedy construction heuristic used to solve the multi-period technician routing and scheduling datasets

On line 1 the set of outsourced jobs O is selected from the full set of jobs V leaving V'. The set of jobs V' are all the jobs that must be allocated to the teams. While there are jobs to be scheduled, a day is created k on line 3. While this day can accommodate more job allocations, a job i is selected to be allocated on line 5. On line 6, a team τ_i is created in order to service job i, and the assignment is made on line 7. While team τ_i has idle time left to service more jobs, jobs are added onto τ_i's job list. Once day k cannot be allocated any more jobs, a new day is created. This process is repeated until all jobs belonging to v' have been assigned to a team. The initial solution K is then output on line 13.

6.4.2 Adapted Intelligent Decision Heuristic

To solve the multi-period technician routing and scheduling problem instances, the intelligent decision heuristic had to be modified to consider the routing aspect of the problem. At each stage of the scheduling process, the heuristic considers the distance to eligible jobs and creates a hypothetical team for each. The heuristic then checks which jobs could be allocated to the team adhering to travel time and skill requirements. A utility score is calculated considering the travel time, utilisation of the route and the quality of job assignments to the team.

6.4.3 Adapted Look Ahead Heuristic

In order to tackle the multi-period technician routing and scheduling problem instances, the look ahead has also been adapted to cope with the complexity of travel time and location. The heuristic now considers the distances to jobs and back to the depot. The pseudo code for the adapted look ahead heuristic is shown in Figure 6.4.

Variables: V: set of jobs, V': set of jobs that are allocated to teams, V'': set of smart allocation decisions, τ: the set of teams, K: the scheduling horizon (collection of days), k : a day in the scheduling horizon, O: outsource list, i: job selected for allocation, τ_i: team selected to serve job i

```
1: $O \leftarrow selectoutsourced(V)$
2: while $V'$ is not empty do
3:     Initialise $k$
4:     while $k$ is not full do
5:         $V'' \leftarrow reduceset(V')$
6:         $i \leftarrow selectjob(V'')$
7:         $\tau_i \leftarrow maketeam(k, \tau)$
8:         $addjob(i, \tau_i)$
9:         while $\tau_i$ $canallocate$ do
10:            $\tau_i \leftarrow addjob(V')$
11:        end while
12:    end while
13: end while
14: return $K$
```

Figure. 6.4 Pseudo code describing the adapted look ahead construction heuristic used to solve the multi-period technician routing and scheduling datasets

On line 1 the set of outsourced jobs O is selected from the full set of jobs V leaving V'. The set of jobs V' are all the jobs that must be allocated to the teams. While there are jobs to be scheduled, a day is created k on line 3. While this day can accommodate more job allocations, the set V' is reduced to V'', on line 5, by removing jobs that would impact subsequent stages of the scheduling process in a negative way. A job i is selected to be allocated from the set V'' on line 6. On line 7, a team τ_i is created in order to service job i, and the assignment is made on line 8. While team τ_i has idle time left to service more jobs, jobs are added onto τ_i's job list. Once day k cannot be allocated any more jobs, a new day is created. This process is repeated until all jobs belonging to V' have been assigned to a team. The initial solution K is then output on line 14.

6.5 Improvement Phase

This section outlines the local operators that have been used in order to perturb a solution generated by the construction heuristic. Six local operators are used, all of which are featured in the previous chapters, but they have been modified slightly to cope with the added complexities. The computational expense of each operator varies, operators such as *Move a job*, *Swap two jobs* and *Shuffle a route* are relatively inexpensive compared to *Decompose and rebuild*, *Decompose and rebuild N* and *Remove N jobs*. The latter operators provide more flexibility as they allow new team configurations to be formed. On each iteration the chance of selecting an operator is uniform.

- *Move a job:* a day k belonging to the scheduling horizon is randomly selected. A team τ is then randomly chosen belonging to day k. A job j is then chosen belonging to τ's route. This job is removed from its current position. The construction heuristic then tries to reallocate the job to another team, adhering to skill. time, and routing constraints.

- *Swap two jobs:* two teams belonging to the scheduling horizon are randomly chosen, $\tau 1$ and $\tau 2$. The teams can belong to the same or different days. A job is then chosen from each team's route, jobs *j1* and *j2* respectively. The heuristic then tries to swap the jobs between the teams such that $\tau 1$ is allocated *j2* and $\tau 2$ is allocated job *j1*. Again, routing constraints are checked to ensure the team's route length does not exceed the maximum length of a working day.

- *Shuffle a route:* this operator randomly selects a day k belonging to the scheduling horizon, and then a team τ belonging to k. The operator then gets the route belonging to τ and shuffles the order of the jobs.
- *Decompose and rebuild;* a day k is selected within the scheduling horizon. Day k is then destroyed by removing all jobs that were assigned to it, and removing all team formations. The heuristic then tries to reallocate the set of jobs using the construction heuristic. Team formations are rebuilt as jobs are allocated.
- *Decompose and rebuild N days:* this operator randomly selects a number of days that will be destroyed and rebuilt. Once the number of days is chosen the heuristic selects the days, and consecutive days are removed. All jobs belonging to the selected days are removed and all team formations are removed. Once again, the construction heuristic is used to reallocate the set of jobs, making new team configurations.
- *Remove N jobs:* a number N is randomly chosen, which is the number of jobs to be removed from the scheduling horizon. While u jobs have not been removed, the operator randomly selects a day k, a team belonging to the day, τ, and a job j from the team's route. Job j is removed and added to the list of removed jobs. If at any time, a team is left with no job assignments, the team is dispersed into single technician teams. Once u jobs have been removed, the construction heuristic is used to reallocate the set of removed jobs.

6.6 Computational Experiments

Once a solution has been modified by a local operator, it must be evaluated to see if it is of better quality than the current solution. In this chapter, a simulated annealing metaheuristic has been used due to its success in Chapter 3. A series of computational experiments have been performed in order to utilise the performance of the metaheuristic.

6.6.1 Simulated Annealing Tuning

In this research a range of starting temperatures was used, the decrement and cooling scheme remained the same. Equation 6.26 shows how the temperature parameter is reduced after each iteration.

$$T = T \cdot \delta T \tag{6.26}$$

The tuning experiments have used a range of datasets, T10, T13 and T28, previously datasets A10, B3 and X8 from the ROADEF 2007 challenge. The initial temperatures tested are shown in Table 6.2 and the results of the experiments are shown in Figures 6.5, 6.6, and 6.7.

Table 6.2 Table showing the set of initial temperatures for the simulated annealing metaheuristic experiments

Initial Temperature
10
20
40
80
100

For each dataset, five initial temperatures were tested as shown in Table 6.2. These experiments were performed in order to strike the right balance of freedom whilst still ensuring convergence during the timed experimental runs. Each run of the algorithm takes 300 seconds.

As an example, Figure 6.5 shows the results obtained for tuning on dataset T10. This dataset has 100 jobs to allocate, no outsourcing budget, and 15 available technicians. The graph shows that using a higher initial temperature leads to lower objective values. This may be the case because there are a small number of jobs, and a relatively small

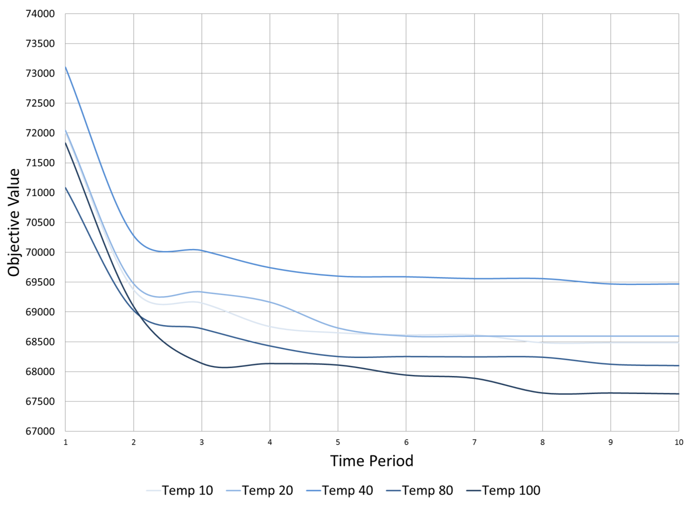


Figure. 6.5 Graph showing the average objective value over time using different initial starting temperatures for dataset T10

pool of technicians, so fewer combinations. The freedom of a higher temperature allows the heuristic to move through these combinations, finding high-quality solutions.

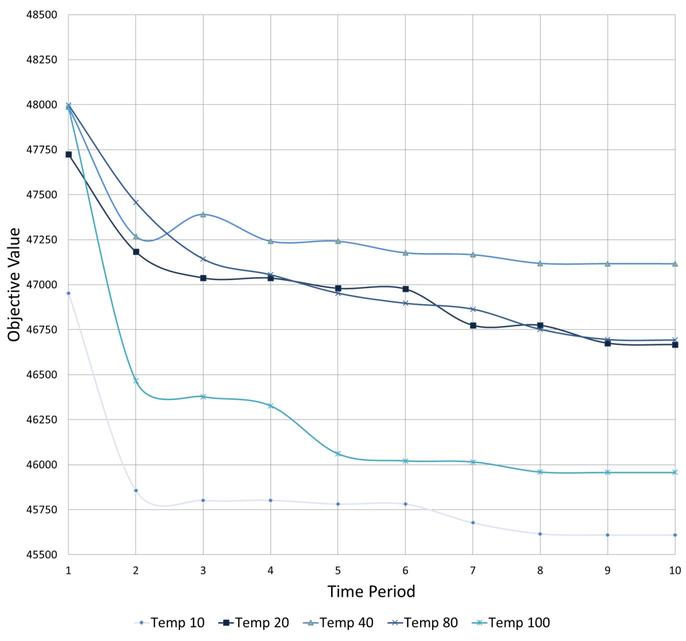


Figure. 6.6 Graph showing the average objective value over time using different initial starting temperatures for dataset T13

Figure 6.6 shows the results obtained for tuning on dataset T13. This dataset has 400 jobs to allocate, an outsourcing budget of 500, and 40 available technicians. The graph shows that using the lowest initial starting temperature leads to the best quality solutions. This may be due to the number of jobs, which leads to more combinations and therefore, a more focused search is necessary so that the heuristic does not diverge into poor quality solution space.

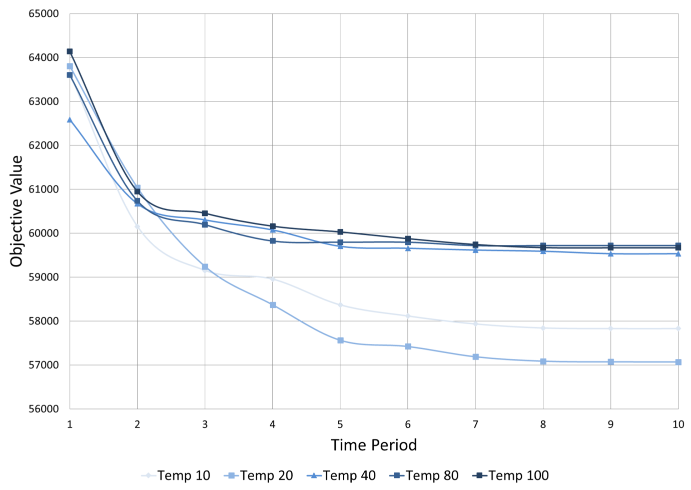


Figure. 6.7 Graph showing the average objective value over time using different initial starting temperatures for dataset T28

Figure 6.7 shows the results obtained for tuning on dataset T28. This dataset has 100 jobs to allocate, an outsourcing budget of 150, and 30 available technicians. The graph illustrates that using a lower initial starting temperature leads to lower objective values. This may be due to the large number of technicians available which means there are more combinations than dataset T10, which has 15 technicians.

The optimal starting temperature of the simulated annealing metaheuristic can be dependent on many factors as these experiments have shown, and there is not a single optimal starting temperature for all datasets. For example, both T10 and T28 have the same number of jobs, however, T10 is a much less complex dataset than dataset T28. Dataset T10 has 20 domain skill levels whereas T28 has 105. Dataset T10 has 15 available technicians and T28 has 30. Furthermore, T28 also includes an outsourcing budget of 150.

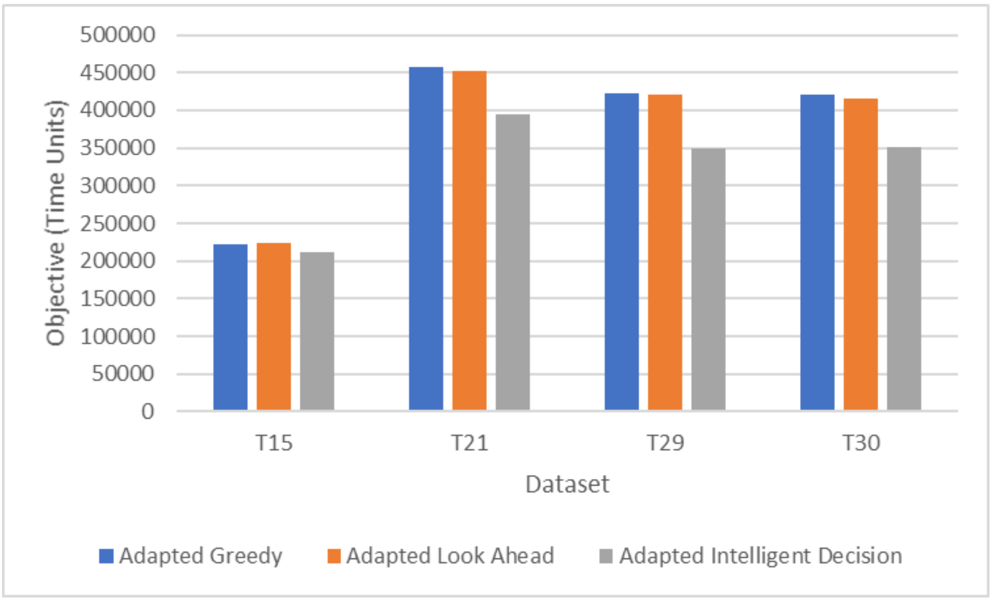


Figure. 6.8 Barchart showing the average objective values found for the adapted greedy, adapted look ahead and the adapted intelligent decision heuristic part 1

6.7 Experimental Results

The multi-period technician routing and scheduling problem instances were tested under the following experimental framework. Each heuristic was allowed a total run time of 300 seconds (5 minutes). Figures 6.8 and 6.9 shows the best results obtained over the five runs for each of the heuristics on the multi-period technician routing and scheduling problem datasets plotted as a bar chart. The heuristics were programmed in Java and tested on an i7 HP Z230 Workstation with 16 GiB.

6.8 Discussion

This chapter has presented the multi-period technician routing and scheduling problem. These datasets were created by extending the technician and task scheduling problem instances featured in Chapter 3 by adding the complexity of location and travel time. The results shown in Figures 6.8 and 6.9 indicate a difference in performance between the adapted greedy, the adapted look ahead, and the intelligent decision heuristic.

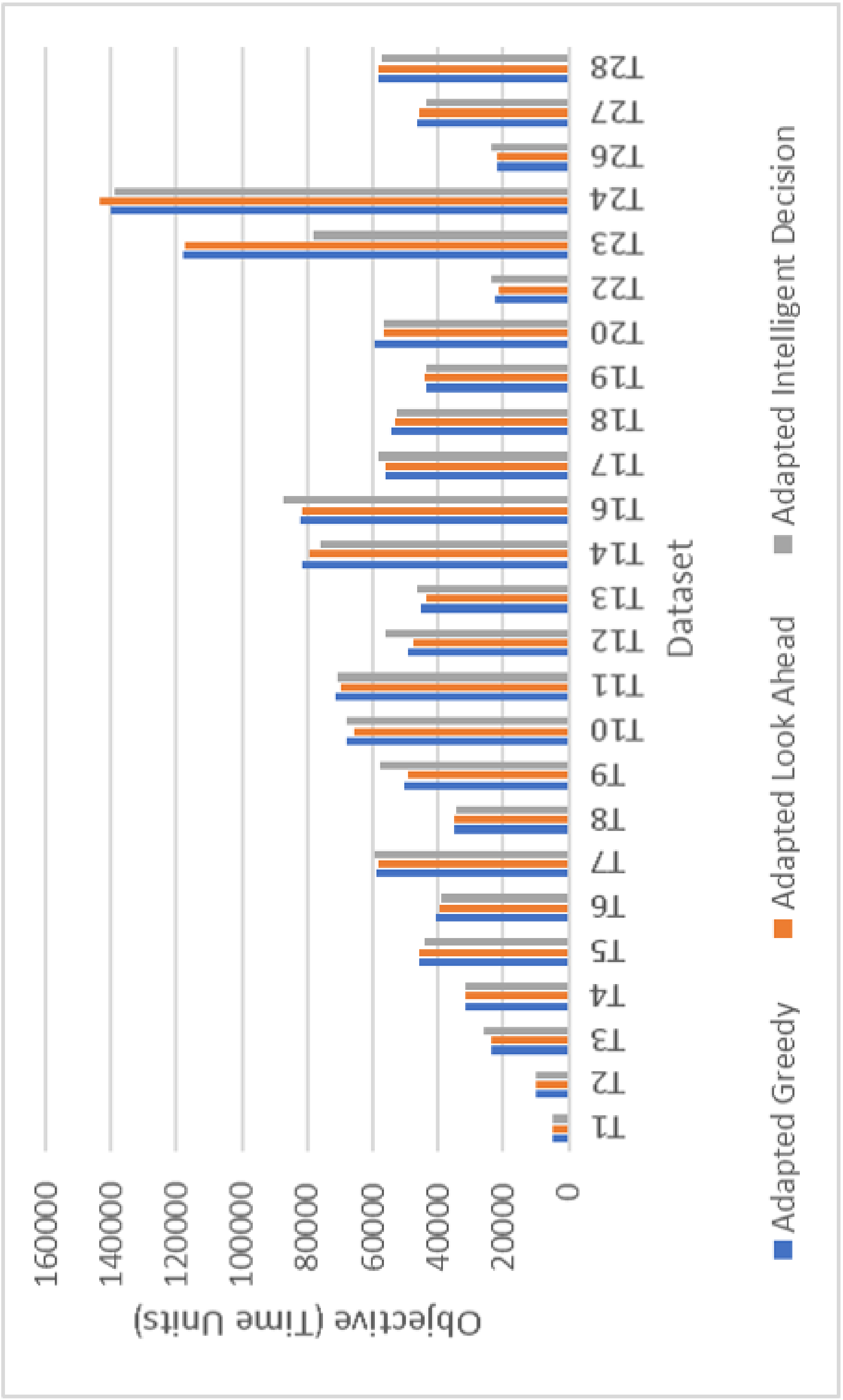


Figure. 6.9 Barchart showing the average objective values found for the adapted greedy, adapted look ahead and the adapted intelligent decision heuristic part 2

In the multi-period technician routing and scheduling problem instances T1-10, the adapted look ahead heuristic finds on average the smallest gap from best known results. Each of the heuristics finds the same solution on instances T1 and T2, which contain scheduling just 5 jobs and so it can be assumed this is the optimal result. Each of the heuristics also finds the same objective value on instance T5, which has 20 jobs to allocate. For the rest of the data instances, the best result is found by either the look ahead or intelligent decision heuristic. On instances T1-10 the look ahead significantly outperforms both other heuristics, with the intelligent decision heuristic performing least favourably.

Similarly, in the multi-period technician routing and scheduling problem instances T11-T20, the adapted look ahead heuristic again outperforms the adapted greedy heuristic and intelligent decision heuristic based on the average gap from best known results. In these instances, the adapted look ahead heuristic finds the lowest objective value in 4 out of 10 instances, whilst the adapted greedy heuristic finds the lowest objective value in just 1 instance, and the intelligent decision heuristic finds 5 of the best results. On average, the adapted greedy heuristic achieves a gap of 2.4%, the adapted look ahead achieves a gap of 1.4% and, lastly, the intelligent decision heuristic finds a gap of 3.6% from best known results.

Lastly, in the multi-period technician routing and scheduling problem instances T21-T30 the adapted intelligent decision heuristic surprisingly finds better quality results on average than the other heuristics. In these datasets, the adapted intelligent decision heuristic finds the lowest objective values in 8 out of 10 instances, whereas the adapted greedy heuristic and adapted look ahead heuristic find the lowest objective values in 1 out of 10 instances each. On average the adapted look ahead has a gap from best known results of 12% whilst the greedy heuristic has a gap of 13% and the intelligent decision heuristic has a gap of just 1.6%.

It appears that overall the adapted intelligent decision heuristic finds the best quality solutions, in terms of the objective value, when taking the average gap from best known results across all datasets. This is helped largely by its superior performance on the

T21-30 problem instances, particularly datasets T21, T25, T29 and T30. The results illustrate that using computationally expensive heuristic approaches has benefits in terms of solution quality obtained when compared against a simpler heuristic approach, such as the greedy heuristic.

6.9 Summary

This chapter has presented a set of multi-period technician routing and scheduling problem datasets. The problem has included scheduling a large set of jobs over multiple days, by constructing teams who travel to service the jobs, subject to unavailability. In the computational experiments, the intelligent decision heuristic outperformed the greedy heuristic and look ahead heuristic on the multi-period technician routing and scheduling problem instances. This suggests that using a more computationally expensive method can in fact out perform a faster heuristic in time constrained experiments.

The contributions of this chapter are: (1) a comparative analysis of the adapted greedy heuristic, adapted intelligent decision heuristic and the adapted look ahead heuristic, (2) a mathematical formulation of the multi-period technician routing and scheduling problem and (3) created 30 instances of multi-period technician routing and scheduling problems which are publicly available at https://akhalfay.wordpress.com/technician-routing-and-scheduling.

In the next chapter, instances from the literature are solved, the service technician routing and scheduling problem with time windows, which is an important aspect of service maintenance scheduling problems.

Chapter 7

Service Technician Routing and Scheduling Problem with Time Windows

7.1 Introduction

This chapter is based on solving the service technician routing and scheduling problem with time windows. The occurrence of time windows is becoming increasingly popular with service maintenance providers as evidenced in recent literature (Mathlouthi et al. (2016) and Zamorano and Stolletz (2017)), and directly affects the scheduling and routing of employees. From a customer's perspective, knowing that a technician/skilled worker will be arriving between time a_i and b_i can improve his/her customer experience. It may allow the customer waiting for a service to take less time off work, and even choose his/her preferred time slot, providing not only convenience but satisfaction.

In this chapter, the first sequential heuristic, the greedy randomized heuristic, to solve the service technician routing and scheduling problem with time windows data instances is tested. These problem instances were created by Kovacs et al. (2012) who enhanced vehicle routing instances proposed by Solomon (1987), by combining the datasets with skill domain information taken from the ROADEF 2007 challenge

(Society, 2007). The research presented in this paper studies 72 service technician routing and scheduling problems with time windows datasets, each with 100 customers, a varying crew size, and different proportions of jobs with time windows. The service technician routing and scheduling problem with time windows datasets have only been solved by Kovacs et al. (2012) who used a parallel adaptive large neighbourhood search algorithm providing a set of benchmark results.

This problem requires tours to be designed for teams, such that all jobs are served or outsourced. The objective of the problem is to minimise the sum of the routing and outsourcing costs over a single day. Each team is made up of one or more technicians, who each have intrinsic skills and levels of competency within each skill area. Each team leaves a central depot at the beginning of the working day, travels to service customers, and returns to the depot before the end of the working day. Each job requires a set of skills that must be satisfied by the team who serves the job. Each job also has a time window $[a_i, b_i]$, and the beginning of service B_i must lie within the time window such that $a_i \leq B_i \leq b_i$. This problem is a single period problem as the solution consists of a single working day. Lastly, there is also the option to outsource some jobs (if they are unable to be allocated), which incurs a penalty cost.

The greedy randomized heuristic, proposed in this chapter has some similarities with the well known greedy randomized adaptive search procedure metaheuristic. The greedy randomized heuristic, like greedy randomized adaptive search procedure, uses multiple scoring criteria to decide which job should be allocated next and includes a degree of randomness in order to avoid deterministic results. In contrast, the greedy randomized heuristic does not generate multiple initial solutions. Instead, the greedy randomized heuristic generates a single initial solution and the rest of the computational time is spent trying to iteratively improve it. During the improvement phase, the chance of selecting a local operator is uniform. A greedy randomized heuristic was chosen as opposed to a greedy randomized adaptive search procedure heuristic due to the short run times that were used in the initial experiments by Kovacs et al. (2012) as it is believed a multi start heuristic procedure is better suited to longer run times.

The remainder of the chapter is organised as follows: section 7.2 presents the mathematical formulation of the service technician routing and scheduling problem with time windows and section 7.3 describes the greedy randomized heuristic. Section 7.4 outlines the improvement phase and section 7.5 presents the metaheuristic used. Section 7.6 shows the results of the computational experiments performed and section 7.7 presents the computational results. Lastly section 7.8 discusses the performance of the greedy randomized heuristic and section 7.9 concludes on the research undertaken.

7.2 Problem Description

The service technician routing and scheduling problem can be described as follows. On a single day, a set of technicians must be assigned to complete a set of jobs. Each job has skill requirements that must be satisfied by the technician who serves the job. The jobs also have durations specifying how long the job takes to complete. Each technician departs from and returns to the centralised depot before the end of the working day. All travel time is accounted for and calculated as Euclidean distance. Each job has a time window in which it must be served. If it is not possible to serve the job within the time window then the job is outsourced which incurs a financial penalty. The objective function is a sum of the distance travelled by all technicians plus any penalty outsourcing costs incurred. Each job must either be served or is outsourced.

7.3 Heuristic Approach

The greedy randomized heuristic comprises of two parts, generating an initial feasible solution, and secondly, iteratively trying to improve the current solution through the use of local operators and evaluating using a simulated annealing with restart metaheuristic.

7.3.1 Greedy Randomized Construction Heuristic

The greedy randomized heuristic behaves in a flexible manner by changing the sorting criteria that decides which job is next to be allocated. There are five insertion methods that have been developed; earliest late window, minimum window size, complex jobs, depot distance, and random. Each of these methods is described in the following subsections. The pseudo code for the greedy randomized construction heuristic is displayed in Figure 7.1.

- **Earliest late window** Each job has a time window $[a_i, b_i]$, and this sorting method orders the jobs into an increasing order of b_i, the end of the time window, the latest time the job can be started. This method tries to ensure that all jobs are allocated before their time window has passed, as they will then be outsourced in order to stay within the feasible solution space, which incurs a cost. By ordering the jobs in this way, the heuristic is maximising its chance of being able to allocate the job.

$$earlylate_i = b_i \tag{7.1}$$

- **Minimum window size** This method sorts the set of jobs by the size of their time window. This method aims to ensure that jobs that have a small opportunity to be started, i.e. the difference between b_i and a_i is small, have a higher chance of being allocated in favour of jobs with a larger difference between b_i and a_i. If the time window is missed, the job will be outsourced which incurs a penalty thus increasing the objective value that can be found.

$$minwindow_i = b_i - a_i \tag{7.2}$$

- **Complex jobs** The set of unallocated jobs are ordered by their difficulty. The difficulty of a job is calculated as the sum of the total skill requirements across each domain and skill level, as in Cordeau et al. (2010). This method aims to allocate jobs which require lots of skill earlier, and jobs that are less difficult to schedule are scheduled later. This method aims to prevent jobs being outsourced due to their skill requirements.

$$complex_i = \sum_{l \in L} \sum_{s \in S} q^i_{l,s} \tag{7.3}$$

- **Depot distance** The depot distance method orders the set of jobs in ascending order of distance away from the depot. The distance between a job i located at x_i, y_i, and the depot located at x_0, y_0 is calculated using the Euclidean distance as shown in Equation (7.23).

$$depotdistance_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \tag{7.4}$$

- **Random** The random sorting method orders the unallocated jobs by shuffling the array that contains the jobs. In this research, the level of randomness, r, has been set to 0.08. The tuning experiments for the level of randomness are explained in Section 7.6.1.

Greedy Randomized Heuristic Pseudo Code

Figure 7.1 shows the greedy randomized construction heuristic. This algorithm takes the following variables, a set of jobs V, a set of teams τ, a schedule S, an outsourcing list O, the following sorting methods ELW: earliest late window, MWS: minimum window size, CJ: complex jobs, DD: depot distance, and R: Randomly, i the job selected for allocation, and τ_i the team selected to be allocated job i.

First, an empty schedule S is initialised. While jobs can be allocated to the schedule S, a random number r is generated on the interval $\{0,1\}$. Dependent on the value of r, a sorting method is chosen; ELW earliest late window, MWS minimum window size, CJ complex jobs, DD depot distance, or R randomly. On line 5, the set of remaining unallocated jobs V is sorted by the chosen sorting method, then a job i is selected belonging to V. Job i is then assigned to a team (if one is available in terms of time windows and skill requirements) and removed from the set of unallocated jobs. The while loop is iterated through until no more job allocations can be made to the teams.

Variables: V: set of jobs, τ: the set of teams , S: the schedule, O: outsource list, ELW: earliest late window, MWS: minimum window size, CJ: complex jobs, DD: depot distance, R: Randomly, i: job selected for allocation, τ_i: team selected to serve job i

```
1: $initialise \quad S$
2: while jobs can be allocated do
3:     $r \leftarrow random\{0,1\}$
4:     $technique \leftarrow Choosesortingmethod(r, ELW, MWS, CJ, DD, R)$
5:     $V \leftarrow method(V)$
6:     $i \leftarrow selectjob(V)$
7:     $\tau_i \leftarrow selectteam(i,S)$
8:     $assign(S, \tau_i, i)$
9:     $remove(V,i)$
10: end while
11: $O \leftarrow addOutsourced(V)$
12: return $S$
```

Figure. 7.1 Figure showing the pseudo code for the greedy randomized construction heuristic used to solve the service technician routing and scheduling problems with time windows

On line 11, any remaining unallocated jobs are added to the outsource list O. Lastly, on line 12, the initial solution S is output.

The cost of outsourcing a job is shown in equation (7.24).

$$o_i = 200 + \sum_{l \in L} \sum_{s \in S} q^i_{l,s} \tag{7.5}$$

7.4 Improvement Phase

7.4.1 Local Operators

A variety of local operators have been used to explore the search space. Some of the local operators used were featured in previous chapters of this thesis (Chapters 3, 4, 5 and 6): *Swap two jobs*, *Move a job*, *Remove a team*, *Remove N jobs* and *Remove N teams*. Other operators used have been featured in other research by Cordeau et al. (2010) which was based on the local operators proposed in Shaw (1998) for solving the vehicle routing problem. In this research, only feasible solution space is explored. This means that when an operator is applied, skill compatibility, time window constraints and route length are checked to make sure the search remains within feasible solution space.

- *Swap order:* two jobs *j1* and *j2* are randomly selected belonging to the same team τ, with positions $p1$ and $p2$ respectively. The operator then reassigns the jobs such that *j2* is now positioned at $p1$ and *j1* is positioned at $p2$ in team τ's route.

- *Remove related jobs:* a number N is randomly chosen, which represents how many jobs that will be removed. Next, a single job is selected at random, job i, and is removed. The remaining jobs are then ranked in terms of how similar they are to the removed job in terms of skill requirements as in Cordeau et al. (2010). The calculation for the relatedness of two jobs in shown in Equation (7.25). The highest scoring job is selected and removed until N jobs have been removed, and added to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.

$$rel_{i,i'} = \sum_{l \in L} \sum_{s \in S} |q^{i}_{l,s} - q^{i'}_{l,s}| \tag{7.6}$$

- *Remove close jobs:* a number N is selected, which defines how many jobs will be removed. Next, a single job is selected at random, job i, and is removed. The remaining jobs are then ranked in terms of how similar they are to the removed job in terms of geographical location. The closeness of two jobs is calculated as the Euclidean distance between them, as illustrated in Equation (7.26). The highest scoring job is selected and removed until N jobs have been removed, and added to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.

$$close_{i,i'} = \sqrt{(x_i - x_{i'})^2 + (y_i - y_{i'})^2} \tag{7.7}$$

- *Remove chains:* This operator iterates through each team belonging to the schedule. If the size of the team's route is greater than two, then the team becomes a candidate to have jobs removed. A number N is chosen between one and the number of candidate teams. A portion of the each of the team's routes is removed and each job is added into the outsourcing list until N team's routes have been changed. The heuristic is then used to try and allocate the jobs from the outsourced list back into the schedule.

7.5 Metaheuristic

7.5.1 Simulated Annealing with Restart

In this work, a simulated annealing metaheuristic with a restart mechanism has been implemented. Simulated annealing was chosen due to its success, in terms of being able to find high quality solutions in reasonable computational times, in other types of combinatorial optimisation problems, (Kundu et al., 2008) and (Cordeau et al., 2010). The implementation of this metaheuristic is shown in Figure 7.2.

The variables associated are: S the initial solution generated by the greedy randomized construction heuristic, S' the neighbouring solution generated by applying a local operator to S, S_{Best} the best solution found, O the set of local operators which perturb the solution S, T the initial temperature, δT the decrement rate, $StepSize$ the maximum number of steps before restarting from the best solution, and lastly, *count* which counts the number of iterations.

The initial solution S, generated by the greedy randomized heuristic, is saved as the best solution S_{Best} on line 1, and *count* is set to 0. Whilst the termination criterion is not met, there is computational time remaining, a local operator is selected on line 4. This local operator o is applied to the solution S on line 5 generating a neighbouring solution S'. On line 6 this solution S' is evaluated. If it has a lower objective function than S, then it replaces S. Next, on line 8 the solution S is evaluated against the best solution S_{Best} and if better, the best solution is updated and the *count* is set to zero.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, T : initial temperature, δT: the cooling rate, $StepSize$: maximum steps before beginning from best solution, *count*: counter for iterations,

```
1: $S_{Best} \leftarrow S$
2: $count \leftarrow 0$
3: **while** termination criteria not met **do**
4:   $randomly \quad choose \quad o \in O$
5:   $S' \leftarrow o(S)$
6:   **if** $S' \leq S$ **then**
7:     $S \leftarrow S'$
8:     **if** $S \leq S_{Best}$ **then**
9:       $S_{Best} \leftarrow S$
10:      $count \leftarrow 0$
11:    **end if**
12:  **else**
13:    $r \leftarrow random\{0,1\}$
14:    $p \leftarrow exp^{(}S' - S)/T$
15:    **if** $p \geq r$ **then**
16:      $S \leftarrow S'$
17:    **end if**
18:  **end if**
19:  $T \leftarrow T \cdot \delta T$
20:  $count \leftarrow count + 1$
21:  **if** $count = StepSize$ **then**
22:    $S \leftarrow S_{Best}$
23:    $count \leftarrow 0$
24:  **end if**
25: **end while**
26: return $S_{Best}$
```

Figure. 7.2 Figure showing the implementation of the simulated annealing with restart metaheuristic on the service technician routing and scheduling problems with time windows

However, if solution S' is not better than the current solution S, then it is evaluated using the simulated annealing criterion and compared to a random number r generated on the interval $\{0,1\}$. If the probability p of accepting this solution is greater than r, then solution S is updated. After every iteration, the simulated annealing temperature is reduced and the *count* is incremented by one. Once the *count* has reached its maximum value, $StepSize$, solution S is set to S_{Best} on line 22, and the *count* is set back to 0. Once the termination criterion has been met, the best solution S_{Best} is output on line 26.

7.6 Computational Experiments

A series of computational experiments have been performed using the greedy randomized heuristic. The aim of these experiments is to ensure the greedy randomized heuristic performs well on the datasets and finds competitive results. The performance of the greedy randomized heuristic is compared against the best known, average, and maximum scores achieved by Kovacs et al. (2012) who used a parallel adaptive large neighbourhood search heuristic.

7.6.1 Tuning the Greedy Randomized Heuristic

The first set of experiments has been used to tune the level of randomness within the greedy randomized heuristic. In order to minimise the number of tuning experiments to be performed, it was decided that the chance of selecting a sorting method, other than the random sorting method, will each have an equal probability. The experiments range from using an equal level of random sorting in comparison to the other sorting methods (0.2), to using no randomness (0) to find the optimal value of randomness. A percentage gap comparison, the difference in objective values divided by the objective value of Kovacs et al. (2012) solution, is presented and compared against the results presented in Kovacs et al. (2012). The percentage gap has been calculated on many criteria; average gap (across all of the datasets used), average gap from the *01* instances (where 100% of the jobs have time windows), average gap on the *03* instances (where 50 % of the jobs have time windows), the *NoTeam* instances (no outsourcing needed), the *ReducedNoTeam* instances (where outsourcing is needed), and lastly, a comparison across the different numbers of domains and skills, 5×4, 6×6 and 7×4.

The 12 datasets chosen from the service technician routing and scheduling problem with time windows instances for the tuning experiments are shown in Table 7.1. The name of the dataset describes its characteristics. For example in $R103\_6 \times 6\_NoTeam$, it can be deduced that the jobs are randomly located by the *R* (*C* is clustered and *RC* randomly clustered), the jobs have 50% time windows defined by the *03* (*01* means

Table 7.1 Table showing the datasets chosen for tuning the greedy randomized heuristic

Datasets
$C101\_5 \times 4\_NoTeam$
$C103\_5 \times 4\_NoTeam$
$C201\_5 \times 4\_ReducedNoTeam$
$C203\_5 \times 4\_ReducedNoTeam$
$R101\_6 \times 6\_NoTeam$
$R103\_6 \times 6\_NoTeam$
$R201\_6 \times 6\_ReducedNoTeam$
$R203\_6 \times 6\_ReducedNoTeam$
$RC101\_7 \times 4\_NoTeam$
$RC103\_7 \times 4\_NoTeam$
$RC201\_7 \times 4\_ReducedNoTeam$
$RC203\_7 \times 4\_ReducedNoTeam$

100% time windows), there are 6 domains and 6 levels within each domain represented by 6×6, and, lastly, the instance is a no team instance represented by $NoTeam$.

Table 7.2 displays the percentage gap achieved from the best known score as found by Kovacs et al. (2012). The first column displays the level of randomness, r, in each implementation and the second column shows the average percentage gap achieved across all of the datasets used in the tuning experiments. The third column shows the average gap from the best known score for the *01* datasets, and column 4 displays the average gap from best known score for the *03* datasets. Columns 5 and 6 show the average gap from best known score achieved across the $NoTeam$ and $ReducedNoTeam$ datasets respectively. Lastly, columns 7, 8 and 9 display the average gap from the best known score across the 5×4, 6×6 and 7×4 datasets.

The experiments have shown that the gap from best known score is minimised overall when using a randomness level $r = 0.08$. Interestingly, these experiments have also highlighted characteristics within the datasets. For example, it seems that the *01* dataset experiments produced a much smaller gap from best known score than the *03* instances. However, the average gaps from best known score seem to be equal when

Table 7.2 Table showing the tuning experiment results for the greedy randomized heuristic on the chosen datasets

r	All	*01*	*03*	*NoTeam*	*ReducedNoTeam*	5×4	6×6	7×4
0.2	0.0634	0.0217	0.1052	0.0602	0.0666	0.0379	0.0575	0.0759
0.16	0.0622	0.0258	0.0985	0.0674	0.0569	0.0353	0.0499	0.0837
0.12	0.0605	0.0253	0.0957	0.0614	0.0595	0.0383	0.0543	0.0696
0.08	**0.0587**	0.0206	0.0968	0.0543	0.0630	0.0370	0.0516	0.0688
0.04	0.0599	0.0248	0.0949	0.0563	0.0634	0.0386	0.0521	0.0696
0.00	0.0605	0.0239	0.0970	0.061	0.0599	0.0394	0.0511	0.0712

comparing the *ReducedNoTeam* and *NoTeam* instances. In addition, there also seems to be a distinction between the number of domains and skill levels, where the larger the number of domains and skills, the larger the gap from optimal results. The tuning experiments performed have shown it is important to have an element of randomness within the algorithm, in order to minimise the gap from best known score.

7.6.2 Tuning the Simulated Annealing with Restart Metaheuristic

The second set of experiments aimed to find the optimal parameter values for the simulated annealing with restart metaheuristic. The restart metaheuristic has three parameters; the $Temperature$, the $Decrement$, and the $StepSize$. The $Temperature$ value controls how likely it is to accept a worse quality solution. *Decrement* controls the rate of decrease in accepting a worse solution and, lastly, the $StepSize$ controls how frequently the search is reverted back to the best solution. Each parameter had two levels, and therefore 2^3 tests had to be undertaken. A series of experiments has been performed using datasets $C101\_5 \times 4\_NoTeam$ and $C103\_5 \times 4\_NoTeam$ (Kovacs et al., 2012), using the parameter values shown in Table 7.3 to find the main and interaction effects.

Table 7.3 Table showing the parameter settings for the implementations of the simulated annealing with restart metaheuristic

Experiment	$StepSize$	$Temperature$	$Decrement$
1	10,000	25	0.9999
2	25,000	25	0.9999
3	10,000	50	0.9999
4	25,000	50	0.9999
5	10,000	25	0.99999
6	25,000	25	0.99999
7	10,000	50	0.99999
8	25,000	50	0.99999

These parameter values have been chosen with careful consideration. First, the number of iterations performed on a timed run was calculated which ranged between 86,000 and 311,000 iterations dependent on the dataset. The initial starting temperatures were then chosen along with decrement values to ensure that the metaheuristic would reach its ground state within the number of iterations that would be performed in a run as shown in Figure 7.3.

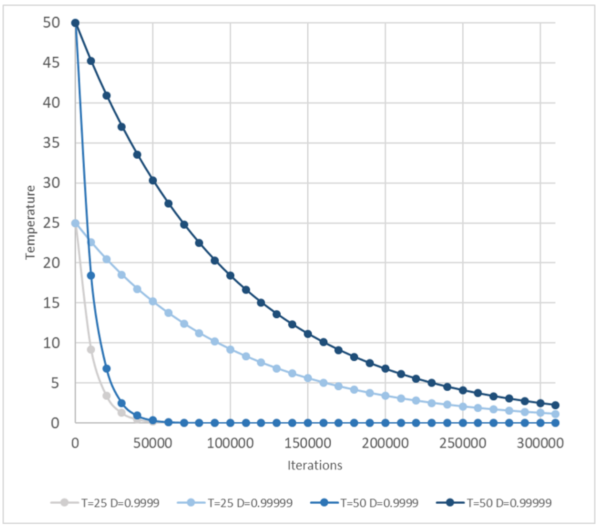


Figure. 7.3 Graph showing temperature over time using different starting temperatures and decrement rates

The simulated annealing with restart metaheuristic was run 10 times for each experiment, and the average objective value obtained was recorded. The main and interaction effects of each parameter are explained in the following subsections for each dataset tested.

$C101\_5 \times 4\_NoTeam$

Figure 7.4 shows the results obtained for dataset $C101\_5 \times 4\_NoTeam$. The main interactions plot illustrates that *Decrement* seems to have the strongest effect on solution quality due to the steepness of the line, i.e. the higher the decrement value the lower mean objective results that are produced. *Temperature* also has a significant effect on the quality of solution obtained, as using a higher temperature produces better quality results. *StepSize* has a smaller yet still significant impact, preferring a smaller step size.

From the interactions plot, shown in Figure 7.4, it seems that *Decrement* and *StepSize* appear parallel, and therefore there is no interaction between these factors. However, factors *StepSize* and *Temperature* do have an interaction as they intersect each other. The most significant interaction is between *Temperature* and *Decrement*.

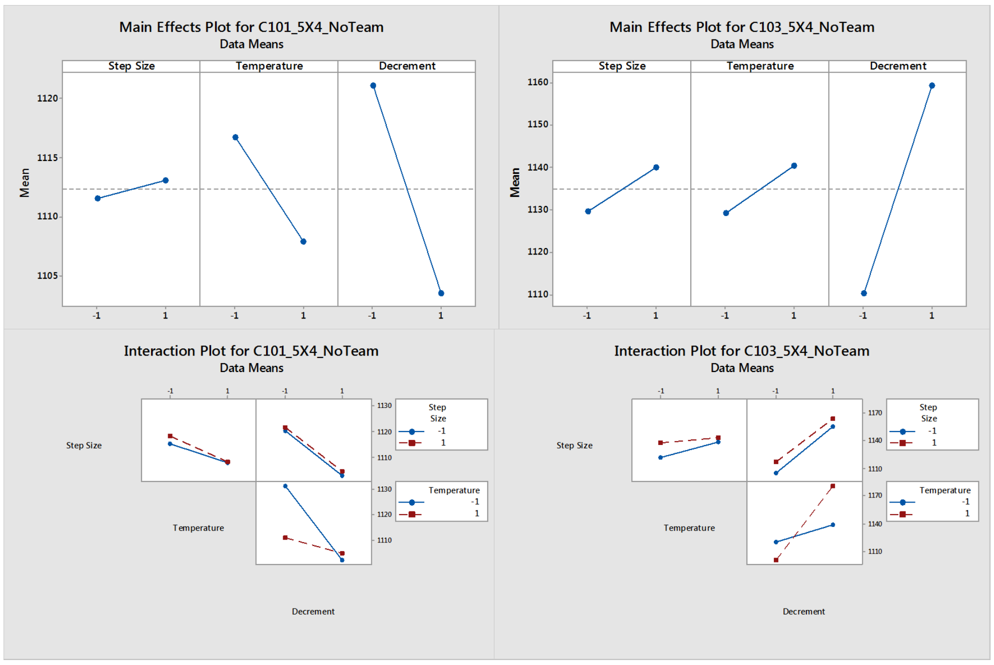


Figure. 7.4 Chart showing the main and interaction effects of the $StepSize$, $Temperature$ and $Decrement$ parameters on datasets $C101\_5 \times 4\_NoTeam$ and $C103\_5 \times 4\_NoTeam$

$C103\_5 \times 4\_NoTeam$

The results for $C103\_5 \times 4\_NoTeam$ are displayed in Figure 7.4. Again, the main effects plot shows that each parameter, $Temperature$, $StepSize$ and $Decrement$ has an effect on the quality of solution found. $Decrement$ has the strongest effect on solution quality in this dataset, but in this dataset, the lower the decrement factor the lower mean objective results that are produced. The other parameters $StepSize$ and $Decrement$ have a similar impact on solution quality evidenced from equal gradients. In this dataset, $C103\_5 \times 4\_NoTeam$, using a lower $Temperature$ and $StepSize$ leads to better quality results.

In Figure 7.4, the interactions plot for $C103\_5 \times 4\_NoTeam$ shows again that there is no interaction between $Decrement$ and $StepSize$. The most significant interaction is between $Decrement$ and $Temperature$, although $Temperature$ and $StepSize$ also have an interaction.

Summary of Tuning Experiments

Figure 7.4 demonstrates that the two types of datasets are affected differently by the parameter values set. In the 01 instances, a higher temperature and decrement lead to better objective values, whereas in the 03 datasets a lower temperature and decrement produced better quality results. This suggests the highly constrained datasets with 100% time windows have fewer combinations and therefore need more freedom to travel through the solution space, whereas less constrained datasets with 50% time windows need a more focused search as there are so many combinations to consider, indicating a range within the datasets.

7.7 Experimental Results

The greedy randomized heuristic was programmed in Java and tested on an HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ with 12GB of RAM. Each run on the *NoTeam* instances lasted 80 seconds and each run on the *ReducedNoTeam* instances lasted 60 seconds, as in Kovacs et al. (2012) for comparison purposes. The greedy randomized heuristic was run 5 times per data instance, and the best, average and worst results obtained are shown in Tables 7.4 and 7.5.

Column 1 shows the name of the dataset, columns 2-4 show the best, average and maximum objective value achieved by Kovacs et al. (2012) with the parallel adaptive large neighbourhood search (pALNS), and columns 4-7 display the best, average and maximum objective values found by the greedy randomized heuristic. The highlighted rows indicate where the greedy randomized heuristic has found a lower objective value than was achieved by Kovacs et al. (2012) with the pALNS.

Table 7.4 Table showing the minimum, maximum and average objective results for the service technician routing and scheduling problem on the *NoTeam* instances

Dataset		pALNS			GREEDY	
	min	avg	max	min	avg	max
C101_5×4	**1098.71**	**1111.08**	**1128.02**	**1096.85**	**1135.03**	**1180.95**
*C*103_5 × 4	1018.61	1037.33	1049.41	1075.36	1119.66	1195.76
C201_5×4	**1158.97**	**1180.93**	**1228.99**	**1157.65**	**1163.1**	**1183.31**
*C*203_5 × 4	1046.93	1049.3	1052.83	1228.23	1297.39	1337.33
R101_5×4	**1678.68**	**1685.85**	**1697.2**	**1672.55**	**1682.17**	**1692.81**
*R*103_5 × 4	1238.67	1249.91	1282.28	1288.48	1312.95	1339.13
*R*201_5 × 4	1440.3	1448.93	1462.62	1526.43	1563.53	1599.98
*R*203_5 × 4	1098	1106.12	1123.08	1281.83	1334.01	1378.83
RC101_5×4	**1708.51**	**1716.07**	**1729.75**	**1676.57**	**1721.95**	**1760.88**
*RC*103_5 × 4	1337.99	1354.11	1388.13	1454.46	1482.46	1507.06
*RC*201_5 × 4	1601.89	1607.25	1610.75	1650.66	1698.03	1727.49
*RC*203_5 × 4	1161.53	1166.5	1178.64	1373.44	1430.32	1467.19
C101_6×6	**989.21**	**1004.82**	**1029.72**	**973.05**	**1002.15**	**1029.72**
*C*103_6 × 6	893.94	897.86	907.62	1075.26	1181.12	1239.93
*C*201_6 × 6	821.55	821.55	821.55	821.55	847.22	868.72
*C*203_6 × 6	689.6	703.1	750.12	831.51	908.91	970.66
*R*101_6 × 6	1658.27	1667.43	1672.57	1662.69	1666.02	1675.24
*R*103_6 × 6	1223.63	1231.49	1243.49	1243.7	1264.54	1286.5
*R*201_6 × 6	1261.94	1270.26	1279.81	1335.66	1375.56	1417.77
*R*203_6 × 6	932.35	951.84	964.54	1104.75	1153.24	1200.86
RC101_6×6	**1679.13**	**1683.96**	**1690.06**	**1672.85**	**1686.62**	**1693.34**
*RC*103_6 × 6	1281.55	1310.95	1331.46	1354.14	1381.79	1400.85
*RC*201_6 × 6	1395.4	1403.95	1411.48	1494.14	1547.03	1613.75
*RC*203_6 × 6	1001.04	1016.71	1030.15	1176.41	1236.67	1291.41
*C*101_7 × 4	1357.05	1398.95	1462.16	1357.05	1416.19	1553.71
*C*103_7 × 4	1215.7	1239.22	1264.17	1263.67	1295.83	1335.95
C201_7×4	**1256.56**	**1282.18**	**1302.56**	**1256.3**	**1264.26**	**1302.56**
*C*203_7 × 4	1150.85	1151.27	1152.94	1288.96	1354.81	1474.64
R101_7×4	**1776.46**	**1793.95**	**1813.53**	**1771.56**	**1791**	**1807.89**
*R*103_7 × 4	1346.8	1375.09	1399.95	1402.04	1423.96	1456.49
*R*201_7 × 4	1398.14	1410.9	1427.95	1427.56	1458.63	1474.29
*R*203_7 × 4	1164.9	1166.94	1169.27	1285.35	1334.17	1407.76
*RC*101_7 × 4	1821.9	1844.37	1859.17	1832.75	1903.58	1980.33
*RC*103_7 × 4	1435.63	1455.33	1477.84	1547.33	1610.13	1679.64
*RC*201_7 × 4	1697.82	1701.25	1705.48	1771.25	1793.66	1811.06
*RC*203_7 × 4	1239.45	1241.65	1249.72	1422.35	1459.22	1527.29

Table 7.5 able showing the minimum, maximum and average objective results for the service technician routing and scheduling problem on the *ReducedNoTeam* instances

Dataset	pALNS			GREEDY		
	min	avg	max	min	avg	max
C101_5×4	**5656.63**	**5733.75**	**5806.55**	**5572.99**	**5798.25**	**6286.36**
$C103\_5 \times 4$	2644.65	2782.2	2869.64	2941.77	3461.57	3861.02
$C201\_5 \times 4$	2755.52	2755.52	2755.52	2755.52	2755.52	2755.52
$C203\_5 \times 4$	2389.37	2392.5	2393.62	2591.11	2680.16	2907.03
$R101\_5 \times 4$	5582.58	5895.38	6181.52	5663.64	6030.76	6400.05
$R103\_5 \times 4$	1710.25	1845.25	2020.48	2034.5	2378.57	2664.23
$R201\_5 \times 4$	2838.5	2854.3	2865.75	2895.78	3014.47	3250.06
$R203\_5 \times 4$	2332.23	2332.23	2332.23	2544.48	2595.48	2802.92
RC101_5×4	**5127.79**	**5164.84**	**5262.36**	**5103.33**	**5428.46**	**5830.71**
$RC103\_5 \times 4$	2170.57	2348.06	2490.12	2661.07	2984.91	3500.22
$RC201\_5 \times 4$	3088.23	3091.67	3093.56	3107.26	3217.95	3282.21
$RC203\_5 \times 4$	2516.16	2540.35	2550.62	2672.52	2761.53	2828.82
C101_6×6	**7731.07**	**7762.94**	**7791.08**	**7660.86**	**7763.51**	**8151.45**
$C103\_6 \times 6$	4980.7	5028.83	5136.21	5242.58	5771.19	6394.29
$C201\_6 \times 6$	3278.07	3299.56	3328.01	3283.84	3405.21	3603.53
$C203\_6 \times 6$	2460.17	2465.9	2468.71	2743.44	2923.61	3101.72
$R101\_6 \times 6$	5955.17	6152.29	6322.82	6174.57	6453.63	6701.15
$R103\_6 \times 6$	2251.64	2329.28	2404.57	2485.9	2839.6	3228.95
$R201\_6 \times 6$	3503.4	3536.7	3574.97	3635.03	3857.49	4072.21
$R203\_6 \times 6$	2437.28	2446.18	2481.77	2649.38	2806.12	2955.39
RC101_6×6	**5276.34**	**5466.18**	**5771.99**	**5231.5**	**5513.1**	**5664.35**
$RC103\_6 \times 6$	2263.83	2349.57	2522.71	2704.04	3212.48	3945.85
$RC201\_6 \times 6$	4422.86	4519.95	4656.79	4973.47	5310.02	5753.73
$RC203\_6 \times 6$	2649.51	2673.72	2730.78	2825.65	3239.56	3615.14
$C101\_7 \times 4$	5208	5257.9	5307.12	5256.49	5438.5	5732.91
$C103\_7 \times 4$	2020.4	2117.44	2173.39	2205.58	2569.24	2804.6
$C201\_7 \times 4$	2773.41	2779.37	2803.21	2773.41	2784.02	2820.23
$C203\_7 \times 4$	2261.37	2282.15	2301.73	2450.03	2566.85	2755.9
R101_7×4	**5239.81**	**5381.35**	**5437.66**	**5232.6**	**5580.65**	**6018.08**
$R103\_7 \times 4$	2104.93	2215.84	2314.3	2338.84	2427.95	2666.58
$R201\_7 \times 4$	2672.96	2679.38	2682.23	2706.8	2764.65	2936.23
$R203\_7 \times 4$	2199.1	2209.8	2229.67	2318.15	2368.62	2443.63
$RC101\_7 \times 4$	5531.06	5799.77	6367.47	5627.79	5959.56	6467.22
$RC103\_7 \times 4$	2586.03	2676.54	2820.48	3127.24	3633.13	3963.97
$RC201\_7 \times 4$	2919.83	2936.28	2945.46	2930.78	3033.64	3244.95
$RC203\_7 \times 4$	2277.62	2285.17	2301.26	2459.13	2541.01	2675.67

7.8 Discussion

7.8.1 Performance of Greedy Randomized Heuristic on *NoTeam* Instances

Table 7.4 displays the results found on the *NoTeam* problem instances. In these datasets, the sequential greedy randomized heuristic is able to find a lower minimum objective value than the parallel adaptive large neighbourhood search in 8 out of 36 datasets and is able to find the same minimum objective value in two datasets $C201\_6\times6\_NoTeam$ and $C101\_7\times4\_NoTeam$.

The results suggest that the greedy randomized heuristic finds a smaller gap from best known score on the *01* instances compared to the *03* instances. The difference between these datasets is the proportion of time windows, the *01* instances are more constrained (contain 100% time windows) compared to the *03* instances (contain 50% time windows) and therefore, there are fewer feasible configurations. In the 5×4, 6×6, and 7×4 datasets, the gap from minimum objective results in the *01* instances is 1.08%, 1.98% and 1.12%. This increases to 11.77%, 14.03% and 8.82% in the *03* problem instances.

Another trend within the results occurs in the *203* datasets. These datasets achieve the highest gap from best known score overall, regardless of the distribution of job locations i.e *C* clustered, *R* randomly, or *RC* randomly clustered. This pattern occurs across each set of instances 5×4, 6×6, and 7×4.

In the 5×4 and 7×4 datasets, the gap from best known score increases in relation to the way the set of jobs are located, clustered (C), random (R), and randomly clustered (RC). In the 5×4 datasets the gap is 5.75%, 6.59% and 7.03% and in 7×4 the gap is 3.98%, 4.07% and 6.86%.

7.8.2 Performance of Greedy Randomized Heuristic on *ReducedNoTeam* Instances

Table 7.5 displays the results achieved for the *ReducedNoTeam* problem instances. In these datasets, the greedy randomized heuristic is able to find a lower minimum objective value than the parallel adaptive large neighbourhood search in 5 out of 36 datasets; and is able to find the same minimum objective value in two datasets $C201\_5 \times 4\_ReducedNoTeam$ and $C201\_7 \times 4\_ReducedNoTeam$.

The results again suggest that the greedy randomized heuristic finds a smaller gap from best known score on the *01* instances compared to the *03* instances with the minimum average gap equal to 0.36%, 3.05% and 0.7% across the *01* instances for the 5×4, 6×6 and 7×4 problems. However, the gap from minimum objective value results increases for the *03* instances to 12.5%, 10.33% and 10.49%.

The trend within the distribution of jobs is the same across each level of domains and skills. In each type of dataset, 5×4, 6×6 and 7×4, the gap from minimum results is the smallest when the distribution of jobs is clustered (C), 4.55%, 4.01% and 4.61%. The gap is largest when the jobs are distributed in random clusters (RC), 7.24%, 9.42% and 7.76%.

In these datasets, there is again a pattern between the 5×4 and 7×4 datasets. The pattern occurs in the *203* datasets. These datasets achieve the highest gap from best known score overall, regardless of the distribution of job locations i.e *C* clustered, *R* randomly, *RC* randomly clustered. This could be attributed to the constrained nature of the problems and perhaps skill sparsity amongst the workforce.

7.8.3 Summary of Performance

Overall, considering the results presented on the 72 datasets, the gap from minimum, average and maximum results are calculated to be 6.38%, 10.07% and 14.04%. This can be split in to the performance gap on the *01* instances as 1.39%, 3.27% and 5.82%, and on the *03* instances as 11.36%, 16.86% and 22.27%.

The results presented in section 7.7 illustrate that the sequential greedy randomized heuristic approach can in some cases, 13 out of 72, find a better quality solution than the parallel adaptive large neighbourhood search approach presented by Kovacs et al. (2012). This is the first time that another heuristic approach has been tested on these datasets, allowing a comparison with the parallel adaptive large neighbourhood search by Kovacs et al. (2012). The greedy randomized heuristic approach has not performed as well on the *03* problem datasets which include only 50% of datasets with time windows. It is believed that this is because the datasets are less constrained, resulting in an increased number of configurations that are possible within these datasets, balanced against the short computational run times that are permitted.

In addition, the parallel adaptive large neighbourhood search approach uses adaptive operators that change their chance of selection dependent on performance so far within the search phase. These datasets can be split by many factors such as the percentage of time windows *01* and *03*, how customers are geographically located *C* clustered, *R* randomly, *RC* randomly clustered, and the number of domain skill areas 5×4, 6×6 and 7×4, which suggests that an adaptive approach would perform well.

7.9 Summary

In this chapter, the greedy randomized heuristic has been designed and implemented to solve the service technician routing and scheduling problem with time windows datasets, which have not been tested since Kovacs et al. (2012), with a parallel adaptive large neighbourhood search. It has been demonstrated that the presence of time windows can greatly affect the solution approach and that the algorithmic performance is also heavily dependent on characteristics of the datasets tested, the proportion of time windows within the datasets.

The contributions of this chapter are: (1) the creation of a computationally efficient greedy randomized heuristic, (2) a comparative analysis of the greedy randomized heuristic and the parallel adaptive large neighbourhood search, (3) the implementation of a simulated annealing with restart metaheuristic with tuning experiments, and lastly, 13 new best known results have been discovered for these datasets.

The results found in this work can be readily applied to other scheduling problems with common constraints such as the home healthcare problem. In this problem, trained professionals (skill complexity) travel to patient locations (travel time) to administer medications under strict guidelines (time windows). Furthermore, this work also has the potential to make an environmental impact, since vehicles are used for transportation between locations, and the smaller the distances travelled, the fewer emissions are produced.

Chapter 8

Discussion

A discussion of the research undertaken throughout this thesis in Chapters 2 - 7 is now presented. This thesis has focused on some of the key challenges that are faced by industry, such as the sponsor Service Power PLC, in the field of technician and task scheduling problems.

8.1 Literature Review

A review of the literature based in the field of technician and task scheduling problems has been carried out in Chapter 2. The review highlighted the complexity of the problems under consideration, these problems are NP hard optimisation problems for which no polynomial time algorithm is known. The review also discussed the relatedness of scheduling problems, such as the similarities between vehicle routing, home healthcare, and technician and task scheduling problems. This chapter then discussed the available datasets, problem definitions, heuristic solution approaches, and metaheuristics used in the field. It was highlighted that the gaps in the field appear to be in the investigation and exploration of: 1) large scale problems indicative of the problems faced in the real world 2) problems that include teaming and location since many application areas require multiple members of a workforce to complete a job, 3) problems that include scheduling over multiple days because some problems proposed in the literature were adapted from vehicle routing problems and 4) problems that include precedence constraints, a common occurrence in many application areas.

8.2 Technician and Task Scheduling Problem

Chapter 3 focused on designing heuristic approaches to solve the real world technician and task scheduling problems proposed by the ROADEF 2007 challenge. These datasets were based on France Telecom's optimisation problem and included three sets of data each containing ten problem instances. This problem included a wide range of constraints applicable to many industries such as teaming, precedence, priority, unavailability, multiple days, and outsourcing.

Two heuristic procedures were developed to solve these problem instances, the intelligent decision heuristic and the look ahead heuristic. The intelligent decision heuristic considers multiple scenarios before making an allocation decision, considering the utilisation of the teams that could be made and the feasible job assignments. This heuristic takes a rather global view of the problem and was coupled with an iterative local search. The intelligent decision heuristic performed well in most instances but did struggle on the Set X instances which were the most complex.

The look ahead heuristic was then designed which outperformed the intelligent decision heuristic and matched the performance of the other approximate approaches from the literature such as those described by Estellon et al. (2009), Cordeau et al. (2010), and Hashimoto et al. (2011). This approach rather than considering the utilisation of a single team that was about to be formed instead focused on the impact a team configuration would have on the rest of the idle teams that were left and the further assignments which could be made. This heuristic was coupled with a simulated annealing metaheuristic to guide the search.

8.3 Large Scale Technician and Task Scheduling Problem

In Chapter 4, a set of large scale instances were generated due to the lack of available large scale datasets within the literature. The sponsor of this research, Service Power PLC, is faced with the problem of scheduling thousands of jobs to a large crew of technicians over a multi day period. These problem instances were created, using a data generator developed to ensure the characteristics present in the ROADEF 2007 challenge problems. The problems contain constraints such as outsourcing, teaming, priority and precedence, but range from scheduling 1000-2500 jobs as opposed to 800 jobs.

The aim of creating these datasets was to test the scalability of the solution approaches developed in Chapter 3, the intelligent decision and look ahead heuristic. A simple greedy heuristic approach was also created and implemented on the datasets for comparative purposes. The experiments were conducted under the guidelines of the ROADEF 2007 challenge using a 20 minute computational time and comparing the best result found for each problem instance. Due to the scale of the problem, in these instances, and the way the objective function is calculated, a hill climbing metaheuristic was coupled with each heuristic procedure.

On these datasets, both the intelligent decision heuristic and the look ahead heuristic outperformed the simple greedy heuristic approach on every problem instance. The look ahead heuristic found the best result in all but one of the instances and on average found a solution 3.7% less costly than the intelligent decision heuristic. The experimental results illustrate that both heuristics are scalable approaches, able to solve large problems in reasonably short computational times.

8.4 Precedence Constrained Technician and Task Scheduling Problem

Chapter 5 concentrated on creating some precedence constrained problem instances. The previous chapters, 3 and 4, both contained the complexity of precedence constraints, where a job may not begin until one or more other jobs have been completed. This is an important consideration for many application areas such as technician and task scheduling, utility services, housing developments, and food production. The effect of precedence constraints has not previously been investigated in terms of the effect on solution quality that can be obtained and the use of heuristic approaches.

The datasets were created using a data generator that was designed. Within a set of jobs to be scheduled the level of precedence constraints varies from 0% to 100% whilst all other attributes remain the same such as priority, outsourcing cost, and skill requirements. The datasets created range from scheduling 100 to 1000 jobs over a multi day scheduling period. Again the intelligent decision heuristic, greedy heuristic and look ahead heuristic, were implemented on these datasets. Each heuristic was coupled with a multi start metaheuristic. The datasets were tested using a 10 minutes computational time with restarts every 30 seconds, again recording the best result obtained.

In the experiments conducted the intelligent decision heuristic and the look ahead heuristic found better quality results than the greedy heuristic on every problem instance. Overall, the look ahead heuristic found a lower mean objective overall, but marginally. It appears that as the size of the problem instances increases the look ahead heuristic is likelier to find better quality results. This chapter has confirmed that both the intelligent decision heuristic and the look ahead heuristic are robust approaches successfully adapting to varying levels of precedence constraints to find high quality results.

8.5 Multi Period Technician Routing and Scheduling Problem

In Chapter 6, the ROADEF 2007 challenge problem was extended to include the complexity of location and travel time. This created a set of multi period technician routing and scheduling problem instances. Routing is an important aspect of technician and task scheduling problems which have not been included in the problems studied in chapters 3, 4 and 5. However, most of the problems in the literature that feature location and travel time have been adapted from vehicle routing problem instances and contain scheduling up to 100 jobs over a single day. The literature review in Chapter 2 highlighted that there are no medium scale problems, scheduling up to 800 jobs, over multiple days that contain routing and location, alongside other important constraints such as teaming, technician unavailability, skill requirements, outsourcing, and priority levels.

To extend the original ROADEF 2007 challenge datasets a central depot and location for each customer had to be generated. A grid was created, which had a centralised depot (that teams depart from and return to each day) and each customer was given unique coordinates within the grid. The length of a working day was also increased to accommodate travel time between locations. The objective function remained the same, a priority weighted sum of the latest end times of jobs for each priority class.

The experiments used a 5 minutes run time, to asses the heuristic performance on short computational run times, with the best results obtained recorded. Each heuristic approach, the intelligent decision heuristic, the greedy heuristic, and the look ahead heuristic had to be adapted to deal with the extra constraints added to the problem definition. Simulated annealing was used due to its success on the technician and task scheduling problem instances in Chapter 3.

The performance on these datasets contrasts the performance on the datasets tested in previous chapters. On these datasets, the intelligent decision heuristic overall finds the best results particularly due to its performance on the set X problem instances. This could be due to the more global view this heuristic takes which may benefit the solution quality obtained due to how heavily routing influences the objective function value. The greedy heuristic, as expected, finds the most inferior solutions overall. This chapter has demonstrated that each of the heuristics can be adjusted to deal with extra constraints and can find high quality results in short computational times.

8.6 Service Technician Routing and Scheduling Problem with Time Windows

The last problem studied in this thesis was the service technician routing and scheduling problem with time windows in Chapter 7. This problem came from the literature and was originally adapted by Kovacs et al. (2012) from vehicle routing problem instances proposed by Solomon (1987). These datasets had only been tested by Kovacs et al. (2012) who used a parallel adaptive large neighbourhood search heuristic. Time windows are an important constraint within the field of technician and task scheduling. Allowing a customer to choose a preferred time slot, can influence the customer's satisfaction, business reputation and can ensure repeat business.

Due to the short run times used to solve these 72 problem instances a new approach had to be designed. The datasets were allowed a 60 or 80 second computational time depending on whether the instance was *NoTeam* or *ReducedNoTeam*. The locations of customers were also varied in distribution, for example, clustered, random or randomly clustered, and the percentage of time windows varied from 50% to 100%.

A greedy randomized heuristic was designed in order to solve these instances which have flexible allocation criteria in order to cope with the variability amongst the datasets. This heuristic was coupled with a simulated annealing with restart metaheuristic which was tuned to account for the differences in datasets, particularly the percentage of time windows present, which dictated the freedom needed to travel through the search space. The greedy randomized heuristic performed well on most of the problem instances, finding high quality results on the 01 instances and new best known results in 18% of datasets.

8.7 Summary

Table 8.1 shows the problems and constraints that have been studied throughout this thesis. The original aim of this research was to investigate the complexities of technician and task scheduling problems and how they can be dealt with effectively.

From the outset of this research, it was identified that one of the most significant complexities is the sheer scale of the problem which arises in the real world in terms of problem size. For this reason, this research has created some large scale problems, scheduling up to 2500 jobs, available to researchers in order to test the scalability of heuristic solution approaches to identify whether they could be used in practice.

The lack of problems available has prohibited the experimentation of applying heuristic approaches to multiple problems with different characteristics and challenges to overcome. This research has generated 12 large scale problems to investigate scalability, 25 precedence constrained instances to analyse the effect of precedence relationships and 30 multi period technician routing and scheduling problems to explore the robustness of the heuristics developed.

Figure 8.1 shows that within this research 9 constraints have been studied: skill, outsourcing, priority, precedence, location, time windows, multiple days, teaming, and unavailability. Investigations have been made throughout the previous chapters into how best to deal with outsourcing, how local operators perform, priority levels, precedence constraints, routing, and time windows.

Another complexity that arises is the use of metaheuristics due to the scale of the problem which typically requires a heuristic approach. Each of the heuristics developed, the intelligent decision heuristic (Chapter 3), the look ahead heuristic (Chapter 3) and greedy randomized heuristic (Chapter 7) has been two stage and tuning experiments to control the diversification and intensification of the metaheuristics have been undertaken. Initially, an iterative local search heuristic (Chapter 3) was implemented which moves the search at times. Next, a simulated annealing metaheuristic was implemented (Chapters 3 and 6) which allows the search to move into worse areas of solution space controlled by a temperature parameter. In addition, a hill climber (Chapter 4) and a more sophisticated multi start hill climber (Chapter 5) have also been implemented. Lastly, the most successful metaheuristic used in this research, finding new best known results, has been implemented, simulated annealing with restart (Chapter 6). This is a technique which composes of two key features featured in two other metaheuristics i.e. the ability to accept a worse solution from simulated annealing, and the ability to move the search from a multi start.

	Problem Size	Instances	Skill	Outsourcing	Priority	Precedence	Location	Time Windows	Multiple Period	Teaming	Unavailability
ROADEF Challenge Problem (Chapter 3)	800	30	✓	✓	✓	✓			✓	✓	✓
Large Scale Technician and Task Scheduling Problem (Chapter 4)	2500	12	✓	✓	✓	✓			✓	✓	✓
Precedence Constrained Technician and Task Scheduling Problem (Chapter 5)	1000	25	✓	✓	✓	✓			✓	✓	✓
Multi Period Technician Routing and Scheduling Problem (Chapter 6)	800	30	✓	✓	✓				✓	✓	✓
Service Technician Routing and Scheduling Problem with Time Windows (Chapter 7)	100	72	✓	✓			✓	✓			

Figure. 8.1 Chart showing the problems studied in this research, the problem sizes, number of instances and complexities

Chapter 9

Conclusions and Future Work

The research presented in this thesis has focused on the design and development of optimisation heuristics for solving technician and task scheduling problems and closely related variations of the problem. The research has been sponsored by Service Power PLC due to the need to find robust and scalable solution approaches to solve these problems which arise in industrial settings in a range of application areas.

This section concludes on the research undertaken in this project, discussing the original research question, how each of the objectives have been met, and directions for further investigation.

9.1 Research Question

What are the complexities associated with technician and task scheduling problems and how can they be dealt with effectively?

9.2 Research Outcomes

Numerous complexities can be associated with technician and task scheduling problems as a direct result of their occurrence in a number of real world industrial settings.

Finding efficient, robust, and scalable solution approaches has far-reaching benefits to both the employer, the employee, the customer, and even the environment. The research presented in this thesis illustrates the array of complexities and constraints that must be considered when designing heuristic solution approaches to solve technician and task scheduling problems in time constrained environments. Characteristics investigated in this research include skill requirements, scheduling over multiple days, technician unavailability, teaming, outsourcing, priority levels, precedence relationships, location and travel time, and time windows. Other characteristics common to the field not explored in this research include tools and spare parts and mandatory breaks which will be addressed in further work.

The heuristics proposed in this research have been designed and developed in order to be scalable (able to tackle problems of varying size efficiently) and robust (can be adapted easily to solve closely related problems). One of the key challenges faced was the availability of datasets, and so, as well as using literature problems, datasets also had to be developed or extended. The heuristics created have been applied both to real world data instances, instances from the literature, generated problem instances (large scale and precedence constrained) and extended real world instances (including location and travel time). A data generator was developed in order to generate problem instances in order to study specific constraints and aspects of technician and task scheduling problems outlined in the literature review. All generated datasets are available to researchers for further experimentation and comparison.

The problems studied throughout this research have included solving small problem instances including 5 jobs to allocate, to solving large scale industrial sized problems, with up to 2500 jobs to allocate. Overall, this research has studied 169 technician and task scheduling problems.

Throughout this research a range of metaheuristics have been used, to guide the lower level heuristic procedures that were designed. Iterative local search, simulated annealing, hill climbing, multi start hill climbing and simulated annealing with restart have been implemented.

The research presented in this thesis has demonstrated that the solution approaches applied to solve technician and task scheduling problems are heavily influenced by the framework and scale of the problem under consideration. When faced with satisfying numerous constraints simultaneously, consideration must be given to not only how a feasible solution can be constructed but also how it can be modified/manipulated with local operators and evaluated using metaheuristics in time constrained conditions.

9.3 Objectives

The following subsections detail how each of the objectives of this research have been achieved. Further to the initial objectives, as the research progressed two other key areas were highlighted for exploration, the work detailed in Chapter 5 on the occurrence and effect precedence constraints and the research presented in Chapter 7 on time windows.

Objective 1

A review of the literature based in the field of technician and task scheduling is illustrated in Chapter 2, which highlights the requirement for heuristic approaches to solve technician and task scheduling problems. There are four key areas the literature review addresses: the varying problem definitions of technician and task scheduling problems, the datasets currently available to researchers, the heuristic and exact solution approaches applied, and the metaheuristics used. The review discusses the current gaps in the field and areas for investigation in this research such as solving large scale problems, medium scale problems with location constraints, precedence constraints, teaming, and multi period problems.

Objective 2

Two heuristic approaches have been designed and developed to solve the ROADEF 2007 challenge problem, a technician and task scheduling problem, from real world data, discussed in Chapter 3. The first approach, the intelligent decision heuristic, considers multiple seed jobs and possible team configurations before making an allocation decision, thereby making more intelligent allocation decisions. The look ahead heuristic, the second approach, focuses on considering the consequences of an allocation decision to the idle teams which are left after a job allocation has been made, ensuring that decisions do not have negative consequences on subsequent stages of scheduling. Both heuristics performed competitively in regards to the performance of other solution approaches, however, the look ahead heuristic was superior to the intelligent decision heuristic, finding better quality solutions in shorter computational run times.

Objective 3

Whilst studying the technician and task scheduling problems in this research it has become clear that there are numerous complexities that can be associated with the problems. Some of the constraints and complexities featured in the problems studied are outsourcing, skill requirements, precedence and successor relationships, teaming, location and travel time, routing, priority levels, technician unavailability, and time windows. In this thesis, many individual aspects of the problems featured such as the choice of outsourcing strategy (Section 3.4.4), the performance of local operators (Section 3.4.4), and priority permutations (Section 3.4.4) have been investigated to determine how they affect the scheduling process of the heuristics developed. Furthermore, the heuristics have been compared on different types of problems such as large scale problems (Chapter 4), to assess scalability, and also tested on different problem frameworks to examine how well they handle additional constraints such as location (Chapter 6), or different strengths of complexities such as the percentage of precedence constraints (Chapter 5).

Objective 4

Within each technician and task scheduling problem studied, a metaheuristic has been implemented. A metaheuristic is used to guide the lower level heuristic during the improvement phase. In this research a range of metaheuristics have been implemented such as: iterative local search (in Chapter 3), simulated annealing (in Chapters 3 and 6), hill climbing (in Chapter 4), multi start hill climbing (in Chapter 5), and simulated annealing with restart (in Chapter 7) in conjunction with the heuristics designed. The parameters for these metaheuristics have been tuned in order to improve their performance in time constrained environments. Using factorial experimentation, the main effects and interaction effects of parameters have been explored, as well as empirically adjusting parameters within the implementations of simulated annealing.

Objective 5

A set of multi period technician routing and scheduling problems have been created by extending the ROADEF 2007 challenge problem in Chapter 6. Location and travel time is a key feature of technician and task scheduling problems not contained in the original ROADEF 2007 challenge problem framework. To do this, location requirements have been added on to each customer and a centralised depot was generated. A mathematical formulation for instances is presented, the multi period technician routing and scheduling problem. This problem was chosen due to the lack of medium scale multi day problems that contain routing constraints alongside other common constraints such as teaming, unavailability, outsourcing, and priority. Each of the heuristics, the intelligent decision heuristic, the look ahead heuristic, and greedy heuristic was modified to tackle this problem proving their robustness as a solution approach, with a comparative analysis of performance undertaken.

Objective 6

As evidenced from the literature, the constraints that a problem features and the size of the problems are directly related to the solution approach used. It is essential that solution approaches are scalable, that they can tackle large as well as small scale problems and produce competitive results whilst balancing time efficiency. For this reason, the computational expense of heuristic solution approaches and indeed the local operators, as the size of the problem increases, are significant. In the literature, there are few real world data instances or available artificial problems that are representative of industrial scale. In addition, there has been no investigation into the scalability of existing approaches on large scale problems that have been shown to work well on small scale instances. A set of large scale technician and task scheduling problems has been generated using a data generator (Chapter 4). These datasets have been solved using the intelligent decision and look ahead heuristic, described in Chapter 4, providing a comparative performance analysis.

Extra Objective 1

Precedence constraints are an important feature in many scheduling problems as they arise commonly in many application areas such as housing development and utility services etc. The occurrence of these constraints and their effect on heuristic approaches and solution quality has not previously been studied. For this reason, 25 precedence constrained scheduling problems were generated, described in Chapter 5, using the data generator, containing 100-1000 jobs to schedule. Again, both the intelligent decision and look ahead heuristic were used to solve these datasets to assess the robustness of the heuristics and to compare their performance.

Extra Objective 2

Finally, a set of data instances from the literature, the service technician routing and scheduling problem with time windows, featured in Chapter 7, was explored. These datasets were originally solved by Kovacs et al. (2012) who adapted the problems from vehicle routing problem instances. A greedy randomized heuristic was developed to solve this problem, which has multiple allocation criteria to provide a flexible heuristic approach, due to the short computational run times permitted. The greedy randomized heuristic found new best known results in 18% of the 72 datasets tested.

9.4 Review of Contributions

1. **A review of the constraints, datasets and solution approaches used in the field of technician and task scheduling problems (Chapter 2).** The review highlights the need to design and develop approximate solution approaches that are both robust and scalable to solve these real world problems that occur in a range of industrial settings. The review demonstrates that previous solution approaches have not been applied to multiple problems, of varying nature and size, in order to prove their efficiency/ability to be applied in commercial settings.

2. **The intelligent decision heuristic characterised by its ability to consider multiple seed jobs and possible team configurations simultaneously (Chapters 3 to 6).** The heuristic considers the utilisation of each possible team by considering the potential further allocations, before making an allocation decision. This heuristic has been used to solve a diverse range of problems in this research such as the technician and task scheduling problem, the large scale technician and task scheduling problem, the precedence constrained technician and task scheduling problem and, lastly, the multi-period technician routing and scheduling problem. The heuristic has produced competitive results on each set of data tested and proved its validity as a solution approach.

3. **A look ahead heuristic that has a preprocessing phase to calculate the underlying indirect precedence relationships present between jobs (Chapters 3 to 6).** This heuristic considers the subsequent impact of an allocation decision to the scheduling process in regards to the idle teams which are left and the allocations they may be given. This heuristic has also been applied to a range of problems: the technician and task scheduling problem, the large scale technician and task scheduling problem, the precedence constrained technician and task scheduling problem and, lastly, the multi-period technician routing and scheduling problem. The look ahead heuristic has proved to be both a robust (handling extra constraints such as location and travel time) and scalable (problem

size) solution approach and has generally outperformed the intelligent decision heuristic.

4. **A data generator which has been designed and developed in order to generate technician and task scheduling problems (Chapters 4 to 6).** In this research, novel datasets have been generated in order to explore certain aspects of the problems, which occur in the real world but are not featured within datasets available in the literature. In this thesis, 12 new large scale technician and task scheduling problems, 25 technician and task scheduling datasets containing varying levels of precedence constraints and 30 multi-period technician routing and scheduling problem instances have been created. These datasets have addressed the problems highlighted in Chapter 2, the need for more datasets featuring a range of constraints and problem sizes available publicly to researchers, for further investigation into the field.

5. **New mathematical formulation of the multi-period technician routing and scheduling problem (Chapter 6).** Due to the extension of the ROADEF 2007 challenge problem which now contains location and travel time constraints, the mathematical formulation had to be updated to account for the extra complexities added. In this thesis, a mathematical formulation for the multi-period technician routing and scheduling problem is described. This is a novel problem in the field as it most importantly includes scheduling over multiple days, constructing teams subject to technician unavailability, and scheduling over a geographical area. This problem also includes other constraints such as priority levels, skill requirements, and outsourcing.

6. **The greedy randomized heuristic which is a flexible scheduling approach with multiple allocation criteria (Chapter 7).** This heuristic is the first sequential heuristic to be tested on the service technician routing and scheduling problem with time windows Kovacs et al. (2012), a single day problem extended from vehicle routing instances. This heuristic was able to find new best known

results in 18% of the 72 data instances tested. This is an impressive result due to the complexity of the problems under consideration which include routing, skills, time windows and outsourcing, and the datasets are subject to very short computational run times of less than 90 seconds.

9.5 Future Research

This research presented in this thesis has explored some of the many research areas within the field of technician and task scheduling. In such a vast field there will always be areas that need further investigation. It is clear that the most important direction for research in this field is to include more real world constraints in medium to large scale problems solved using heuristic approaches. Solving problems that are comparable to real world problems will inevitably aid the field.

9.5.1 Site Management

On reflection, a new challenge inspired by industry has been identified. In the construction trade, often a large job such as building renovation or extension or perhaps a housing development will actually be comprised of many hundreds of smaller jobs. The value of the overall job may cost hundreds of thousands of pounds, in material costs, labour costs, and equipment. A set back in the time delivery for the job will be costly and will eat into any potential profits. For this reason, it is common to have a site manager, a single member of the workforce to be present at all times to coordinate the site thus reducing the chance of incurring penalties through late delivery. The site manager will be responsible for ensuring the security of the site, keeping the site closed off to the public and keeping costly equipment, often hired, safe, as well as coordinating other workers on the site, and ordering materials. To the authors knowledge no such problem, where a business is running multiple large scale jobs, comprised of many hundreds of smaller jobs has been studied in the literature.

9.5.2 Tools and Spare Parts

In the service maintenance sector there may be tools that are shared between the workforce, and so must be collected from the central depot or another technician/team. Additionally, there may also be spare parts that are needed in order to complete a job. These resources may be stored in the technician's/team's vehicle but if they run out they may need to be collected from the central depot or a nearby hardware store. Work including these constraints has been studied by Pillac, Gueret and Medaglia (2013) using single day problem instances adapted from Solomon (1987) containing up to 100 jobs. It is proposed that future research including tools and spare parts should use larger scale problems, covering multiple days, indicative of industrial problems that arise in practical settings.

9.5.3 Varying Travel Times

Furthermore, there are also aspects of real world problems that have not yet been investigated, such as the varying time it may take to travel between two customers. It is realistic to assume that the time it takes to travel between locations changes dependent on: the time of day, the day of the week, time of year that the journey is made, and of course, if there are road works or road incidents. For example, travelling during rush hour, 7.30 am to 9 am, will usually incur a larger travel time than a journey made at 11 am. In addition, a journey made on a weekend will take less time on average than a journey made on a weekday, which is due to the volume of traffic being decreased on a weekend. Furthermore, journeys made in the winter months may take longer due to poor weather conditions, decreased visibility and decreased average speed as compared to the summer months with better weather conditions and more visibility. These conditions may influence the decision making processes of a heuristic approach.

9.5.4 Mandatory Breaks

An important aspect of scheduling missing from many problems studied in the literature is the mandatory breaks that employees must receive during their shifts. The more real world aspects/constraints that are studied and included within heuristic approaches, the more certain researchers can be in the translation of solution approaches into commercialised software. For this reason, including the allocation of breaks to employees within the execution of their shifts seems essential for the continued effort to study more representative scheduling problems. Breaks have been included again in some smaller scale problems such as Tricoire et al. (2013) (containing up to 100 jobs) that have used exact and hybrid solution approaches but their inclusion in larger scale problems solved with heuristic methods has yet to be investigated.

9.5.5 Individual Employee Preferences

Ernst et al. (2004) recommended that research also included/focused on employees individual preferences and needs. However, it seems that investigation into this area has not yet begun. It is clear that across workforces there will be individuals with specific requirements different from their fellow workers. These needs could include, starting work later in order to drop off children at school or nursery, or finishing earlier to pick them up, regular hospital visits for workers with preexisting medical conditions, or providing flexibility in working patterns for those with caring responsibilities. The inclusion of a heterogeneous workforce in terms of working patterns not just in terms of skill is an area that has a significant impact on the contentedness of a workforce, and may help reduce the staff turnover (as people are working in conditions that suit their needs) which will impact training costs, vacancy advertising, and interview processes.

9.6 Further Advice and Guidance

This research has focused on solving a range of technician and task scheduling problems using heuristic approaches coupled with metaheuristics. From the experiments performed, a list of recommendations to future researchers has been formulated depending on the type of technician and task scheduling problem being studied.

For example when solving large scale problems the results show that using local operators which are able to change the team configurations as well as job distributions perform better aiding the search for high quality solutions. Conversely, when faced with finding solutions in short computational times, simple flexible heuristic methods perform well, as they are able to perform more iterations maximising the chance of finding improvements. This work has also shown that the simulated annealing metaheuristic and adapted versions perform well both in short and medium computational time scales. Overall, it seems that the look ahead heuristic is appropriate when solving problems with medium computational run times, and when constraints such as team building, precedence and priority are included. The intelligent decision heuristic is more appropriate to use when routing is included in the problem framework. Lastly, the greedy randomized heuristic is most appropriate when faced with very short computational run times.

Lastly, the aim of solving these problems is to find a good quality solutions in reasonable computational times. Whilst the constraints a problem includes will shape the solution approach, consideration must also be given as to what constitutes a reasonable computational time. Using extra computational time must be balanced against the level of improvement that can be made.

References

Abraham, I., Delling, D., Fiat, A., Goldberg, A. V. and Werneck, R. F. (2016), 'Highway dimension and provably efficient shortest path algorithms', *Journal of the ACM (JACM)* **63**(5), 41.

Ahuja, R. K., Mehlhorn, K., Orlin, J. and Tarjan, R. E. (1990), 'Faster algorithms for the shortest path problem', *Journal of the ACM (JACM)* **37**(2), 213–223.

Aickelin, U. and Dowsland, K. A. (2000), 'Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem', *Journal of scheduling* **3**(3), 139–153.

Aickelin, U. and Dowsland, K. A. (2004), 'An indirect genetic algorithm for a nurse-scheduling problem', *Computers & Operations Research* **31**(5), 761–778.

Akjiratikarl, C., Yenradee, P. and Drake, P. R. (2007), 'Pso-based algorithm for home care worker scheduling in the uk', *Computers & Industrial Engineering* **53**(4), 559–583.

Alsheddy, A. and Tsang, E. P. (2011), 'Empowerment scheduling for a field workforce', *Journal of Scheduling* **14**(6), 639–654.

Asensio-Cuesta, S., Diego-Mas, J., Canós-Darós, L. and Andrés-Romano, C. (2012), 'A genetic algorithm for the design of job rotation schedules considering ergonomic and competence criteria', *The International Journal of Advanced Manufacturing Technology* **60**(9), 1161–1174.

Asta, S., Ozcan, E. and Curtois, T. (2016), 'A tensor based hyper-heuristic for nurse rostering', *Knowledge-Based Systems* **98**, 185–199.

Atkinson, J. B. (1994), 'A greedy look-ahead heuristic for combinatorial optimization: an application to vehicle scheduling with time windows', *Journal of the Operational Research Society* **45**(6), 673–684.

Badeau, P., Guertin, F., Gendreau, M., Potvin, J.-Y. and Taillard, E. (1997), 'A parallel tabu search heuristic for the vehicle routing problem with time windows', *Transportation Research Part C: Emerging Technologies* **5**(2), 109–122.

Baker, B. M. and Ayechew, M. (2003), 'A genetic algorithm for the vehicle routing problem', *Computers & Operations Research* **30**(5), 787–800.

Baker, K. R. and Magazine, M. J. (1977), 'Workforce scheduling with cyclic demands and day-off constraints', *Management Science* **24**(2), 161–167.

Barketau, M. and Pesch, E. (2016), 'An approximation algorithm for a special case of the asymmetric travelling salesman problem', *International Journal of Production Research* **54**(14), 4205–4212.

Beasley, J. E. (1983), 'Route first—cluster second methods for vehicle routing', *Omega* **11**(4), 403–408.

Bektaş, T., Demir, E. and Laporte, G. (2016), Green vehicle routing, *in* 'Green Transportation Logistics', Springer, pp. 243–265.

Bertels, S. and Fahle, T. (2006), 'A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem', *Computers & Operations Research* **33**(10), 2866–2890.

Blum, C. and Roli, A. (2003), 'Metaheuristics in combinatorial optimization: Overview and conceptual comparison', *ACM Computing Surveys (CSUR)* **35**(3), 268–308.

Bonomi, E. and Lutton, J.-L. (1984), 'The n-city travelling salesman problem: Statistical mechanics and the metropolis algorithm', *SIAM review* **26**(4), 551–568.

Bouzid, M. C., Haddadene, H. A. and Salhi, S. (2017), 'An integration of lagrangian split and vns: The case of the capacitated vehicle routing problem', *Computers & Operations Research* **78**, 513–525.

Braekers, K., Hartl, R. F., Parragh, S. N. and Tricoire, F. (2016), 'A bi-objective home care scheduling problem: Analyzing the trade-off between costs and client inconvenience', *European Journal of Operational Research* **248**(2), 428–443.

Burke, E., Curtois, T., Hyde, M., Kendall, G., Ochoa, G., Petrovic, S., Vázquez-Rodríguez, J. A. and Gendreau, M. (2010), Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms, *in* 'Evolutionary Computation (CEC), 2010 IEEE Congress on', IEEE, pp. 1–8.

Burke, E. K., De Causmaecker, P., Berghe, G. V. and Van Landeghem, H. (2004), 'The state of the art of nurse rostering', *Journal of scheduling* **7**(6), 441–499.

Carlson, J. A., Jaffe, A. and Wiles, A. (2006), *The millennium prize problems*, American Mathematical Society, Providence, RI.

Carrasco, R. A., Iyengar, G. and Stein, C. (2013), 'Single machine scheduling with job-dependent convex cost and arbitrary precedence constraints', *Operations Research Letters* **41**(5), 436–441.

Castillo-Salazar, J. A., Landa-Silva, D. and Qu, R. (2012), A survey on workforce scheduling and routing problems, *in* 'Proceedings of the 9th international conference on the practice and theory of automated timetabling', Citeseer, pp. 283–302.

Castillo-Salazar, J. A., Landa-Silva, D. and Qu, R. (2016), 'Workforce scheduling and routing problems: literature survey and computational study', *Annals of Operations Research* **239**(1), 39–67.

Cheang, B., Li, H., Lim, A. and Rodrigues, B. (2003), 'Nurse rostering problems—-a bibliographic survey', *European Journal of Operational Research* **151**(3), 447–460.

Chen, X., Thomas, B. W. and Hewitt, M. (2016), 'The technician routing problem with experience-based service times', *Omega* **61**, 49–61.

Chen, Y., Cowling, P., Polack, F., Remde, S. and Mourdjis, P. (2017), 'Dynamic optimisation of preventative and corrective maintenance schedules for a large scale urban drainage system', *European Journal of Operational Research* **257**(2), 494–510.

Cheng, E. and Rich, J. L. (1998), A home health care routing and scheduling problem, Technical report, Technical Report TR98-04, Department of Computational And Applied Mathematics, Rice University.

Christofides, N., Mingozzi, A. and Toth, P. (1981), 'Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations', *Mathematical programming* **20**(1), 255–282.

Clarke, G. and Wright, J. W. (1964), 'Scheduling of vehicles from a central depot to a number of delivery points', *Operations research* **12**(4), 568–581.

Cook, S. (2003), 'The importance of the p versus np question', *Journal of the ACM (JACM)* **50**(1), 27–29.

Cook, S. (2006), 'The p versus np problem', *The millennium prize problems* pp. 87–104.

Cook, S. A. (1983), 'An overview of computational complexity', *Communications of the ACM* **26**(6), 400–408.

Cordeau, J.-F., Gendreau, M. and Laporte, G. (1997), 'A tabu search heuristic for periodic and multi-depot vehicle routing problems', *Networks* **30**(2), 105–119.

Cordeau, J.-F., Laporte, G., Pasin, F. and Ropke, S. (2010), 'Scheduling technicians and tasks in a telecommunications company', *Journal of Scheduling* **13**(4), 393–409.

Cortés, C. E., Gendreau, M., Rousseau, L. M., Souyris, S. and Weintraub, A. (2014), 'Branch-and-price and constraint programming for solving a real-life technician dispatching problem', *European Journal of Operational Research* **238**(1), 300–312.

Crispin, A. and Syrichas, A. (2013), Quantum annealing algorithm for vehicle scheduling, *in* '2013 IEEE International Conference on Systems, Man, and Cybernetics', IEEE, pp. 3523–3528.

Dantzig, G. B. and Ramser, J. H. (1959), 'The truck dispatching problem', *Management science* **6**(1), 80–91.

Dantzig, G., Fulkerson, R. and Johnson, S. (1954), 'Solution of a large-scale traveling-salesman problem', *Journal of the operations research society of America* **2**(4), 393–410.

Davari, M., Demeulemeester, E., Leus, R. and Nobibon, F. T. (2013), 'Exact algorithms for single-machine scheduling with time windows and precedence constraints', *Journal of Scheduling* pp. 1–26.

Defraeye, M. and Van Nieuwenhuyse, I. (2016), 'Staffing and scheduling under nonstationary demand for service: A literature review', *Omega* **58**, 4–25.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992), 'A new optimization algorithm for the vehicle routing problem with time windows', *Operations research* **40**(2), 342–354.

Dijkstra, E. W. (1959), 'A note on two problems in connexion with graphs', *Numerische mathematik* **1**(1), 269–271.

Dongala, S. G. P. (2006), The problem of scheduling technicians and interventions in a telecommunications company, Technical report, Technical report,Instituto Superior Technico, Universidade Tecnica de Lisboa.

Dorigo, M. and Gambardella, L. M. (1997), 'Ant colonies for the travelling salesman problem', *biosystems* **43**(2), 73–81.

Drucker, N., Penn, M. and Strichman, O. (2010), Cyclic routing of unmanned air vehicles, Technical report, Tech. Rep. IE/IS-2014-02, Faculty of Industrial Engineering and Management, Technion.

Dutot, P.-F., Laugier, A. and Bustos, A.-M. (2006), Technicians and interventions scheduling for telecommunications (roadef challenge subject), Technical report, Technical report, France Telecom R&D.

Eksioglu, B., Vural, A. V. and Reisman, A. (2009), 'The vehicle routing problem: A taxonomic review', *Computers & Industrial Engineering* **57**(4), 1472–1483.

Erkoc, M. and Ertogral, K. (2016), 'Overhaul planning and exchange scheduling for maintenance services with rotable inventory and limited processing capacity', *Computers & Industrial Engineering* **98**, 30–39.

Ernst, A. T., Jiang, H., Krishnamoorthy, M. and Sier, D. (2004), 'Staff scheduling and rostering: A review of applications, methods and models', *European journal of operational research* **153**(1), 3–27.

Estellon, B., Gardi, F. and Nouioua, K. (2009), High-performance local search for task scheduling with human resource allocation, *in* 'Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics', Springer, pp. 1–15.

Fikar, C. and Hirsch, P. (2017), 'Home health care routing and scheduling: A review', *Computers & Operations Research* **77**, 86–95.

Fırat, M. and Hurkens, C. (2012), 'An improved mip-based approach for a multi-skill workforce scheduling problem', *Journal of Scheduling* **15**(3), 363–380.

Fischetti, M., Salazar González, J. J. and Toth, P. (1997), 'A branch-and-cut algorithm for the symmetric generalized traveling salesman problem', *Operations Research* **45**(3), 378–394.

Fisher, M. L. (1994), 'Optimal solution of vehicle routing problems using minimum k-trees', *Operations research* **42**(4), 626–642.

Fukasawa, R., Longo, H., Lysgaard, J., de Aragão, M. P., Reis, M., Uchoa, E. and Werneck, R. F. (2006), 'Robust branch-and-cut-and-price for the capacitated vehicle routing problem', *Mathematical programming* **106**(3), 491–511.

Garey, M. R., Johnson, D. S. and Tarjan, R. E. (1976), 'The planar hamiltonian circuit problem is np-complete', *SIAM Journal on Computing* **5**(4), 704–714.

Gartner, D. and Padman, R. (2017), Mathematical programming and heuristics for patient scheduling in hospitals: A survey, *in* 'Handbook of Research on Healthcare Administration and Management', IGI Global, pp. 627–645.

Gendreau, M., Hertz, A. and Laporte, G. (1994), 'A tabu search heuristic for the vehicle routing problem', *Management science* **40**(10), 1276–1290.

Gendreau, M., Laporte, G. and Semet, F. (1998), 'A tabu search heuristic for the undirected selective travelling salesman problem', *European Journal of Operational Research* **106**(2-3), 539–545.

Gérard, M., Clautiaux, F. and Sadykov, R. (2016), ‘Column generation based approaches for a tour scheduling problem with a multi-skill heterogeneous workforce’, *European Journal of Operational Research* **252**(3), 1019–1030.

Glover, F. (1986), ‘Future paths for integer programming and links to artificial intelligence’, *Computers & operations research* **13**(5), 533–549.

Glover, F. and Taillard, E. (1993), ‘A user’s guide to tabu search’, *Annals of operations research* **41**(1), 1–28.

Golden, B. L., Raghavan, S. and Wasil, E. A. (2008), *The vehicle routing problem: latest advances and new challenges*, Vol. 43, Springer Science & Business Media.

Golden, B. L. and Yee, J. R. (1979), ‘A framework for probabilistic vehicle routing’, *AIIE Transactions* **11**(2), 109–112.

Gu, J., Purdom, P. W., Franco, J. and Wah, B. W. (1999), Algorithms for the satisfiability (sat) problem, *in* ‘Handbook of Combinatorial Optimization’, Springer, pp. 379–572.

Hashimoto, H., Boussier, S., Vasquez, M. and Wilbaut, C. (2011), ‘A grasp-based approach for technicians and interventions scheduling for telecommunications’, *Annals of Operations Research* **183**(1), 143–161.

Held, M. and Karp, R. M. (1970), ‘The traveling-salesman problem and minimum spanning trees’, *Operations Research* **18**(6), 1138–1162.

Hernández-Pérez, H., Rodríguez-Martín, I. and Salazar-González, J.-J. (2016), ‘A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem’, *European Journal of Operational Research* **251**(1), 44–52.

Hiermann, G., Prandtstetter, M., Rendl, A., Puchinger, J. and Raidl, G. R. (2015), ‘Metaheuristics for solving a multimodal home-healthcare scheduling problem’, *Central European Journal of Operations Research* **23**(1), 89–113.

Hoffman, K. L., Padberg, M. and Rinaldi, G. (2013), Traveling salesman problem, *in* ‘Encyclopedia of operations research and management science’, Springer, pp. 1573–1578.

Hoogeveen, J. A., Lenstra, J. K. and Veltman, B. (1996), ‘Preemptive scheduling in a two-stage multiprocessor flow shop is np-hard’, *European Journal of Operational Research* **89**(1), 172–175.

Hurkens, C. A. (2009), ‘Incorporating the strength of mip modeling in schedule construction’, *RAIRO-Operations Research* **43**(04), 409–420.

Ioannou, G., Kritikos, M., Prastacos, G. et al. (2001), ‘A greedy look-ahead heuristic for the vehicle routing problem with time windows’, *Journal of the Operational Research Society* **52**(5), 523–537.

Jain, A., Datta, U. and Joshi, N. (2016), ‘Implemented modification in dijkstra’s algorithm to find the shortest path for ‘n’nodes with constraint’, *International Journal of Scientific Engineering and Applied Science* **2**(2), 420–426.

Jaskowski, W. and Wasik, S. (2007), ‘Efficient greedy algorithm with hill climbing for technicians and interventions scheduling problem, french operational research society’.
URL: *http://challenge.roadef.org/2007/files/abstracts¡ndex.html*

Jensen, T. R. and Toft, B. (2011), *Graph coloring problems*, Vol. 39, John Wiley & Sons.

Kang, S., Kim, S.-S., Won, J. and Kang, Y.-M. (2016), 'Gpu-based parallel genetic approach to large-scale travelling salesman problem', *The Journal of Supercomputing* **72**(11), 4399–4414.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. et al. (1983), 'Optimization by simulated annealing', *Science* **220**(4598), 671–680.

Korteweg, P. (2007), 'When to hire the a-team, french operational research society'. **URL:** *http://roadef.proj.info-ufr.univ-montp2.fr/2007/files/abstract*$_r$*oadef*08.*pdf*

Kovacs, A. A., Parragh, S. N., Doerner, K. F. and Hartl, R. F. (2012), 'Adaptive large neighborhood search for service technician routing and scheduling problems', *Journal of scheduling* **15**(5), 579–600.

Krishnamoorthy, M., Ernst, A. T. and Baatar, D. (2012), 'Algorithms for large scale shift minimisation personnel task scheduling problems', *European Journal of Operational Research* **219**(1), 34–48.

Kundu, S., Mahato, M., Mahanty, B. and Acharyya, S. (2008), Comparative performance of simulated annealing and genetic algorithm in solving nurse scheduling problem, *in* 'Proceedings of the International MultiConference of Engineers and Computer Scientists', Vol. 1, pp. 96–100.

Lalla-Ruiz, E., Expósito-Izquierdo, C., Taheripour, S. and Voß, S. (2016), 'An improved formulation for the multi-depot open vehicle routing problem', *OR spectrum* **38**(1), 175–187.

Laporte, G. (1992), 'The vehicle routing problem: An overview of exact and approximate algorithms', *European Journal of Operational Research* **59**(3), 345–358.

Leigh, J. M., Jackson, L. M. and Dunnett, S. J. (2016), Police officer dynamic positioning for incident response and community presence, *in* 'Proceedings of the 5th International Conference on Operations Research and Enterprise Systems (ICORES 2016)', INSTICC/SCITEPRESS, p. 261–270.

Lenstra, J. K. and Kan, A. (1976), 'On general routing problems', *Networks* **6**(3), 273–280.

Lenstra, J. K. and Kan, A. (1981), 'Complexity of vehicle routing and scheduling problems', *Networks* **11**(2), 221–227.

Lenstra, J. K., Kan, A. R. and Brucker, P. (1977), 'Complexity of machine scheduling problems', *Annals of discrete mathematics* **1**, 343–362.

Lenstra, J. K. and Rinnooy Kan, A. (1978), 'Complexity of scheduling under precedence constraints', *Operations Research* **26**(1), 22–35.

Lesaint, D., Voudouris, C. and Azarmi, N. (2000), 'Dynamic workforce scheduling for british telecommunications plc', *Interfaces* **30**(1), 45–56.

Lewis, R. (2016), Advanced techniques for graph colouring, *in* 'A Guide to Graph Colouring', Springer, pp. 55–77.

Li, H. and Alidaee, B. (2016), 'Tabu search for solving the black-and-white travelling salesman problem', *Journal of the Operational Research Society* **67**(8), 1061–1079.

Lourenço, H. R., Martin, O. C. and Stützle, T. (2003), *Iterated local search*, Springer.

Mahvash, B., Awasthi, A. and Chauhan, S. (2017), 'A column generation based heuristic for the capacitated vehicle routing problem with three-dimensional loading constraints', *International Journal of Production Research* **55**(6), 1730–1747.

Martí, R., Moreno-Vega, J. M. and Duarte, A. (2010), Advanced multi-start methods, *in* 'Handbook of metaheuristics', Springer, pp. 265–281.

Martoňák, R., Santoro, G. E. and Tosatti, E. (2004), 'Quantum annealing of the traveling-salesman problem', *Physical Review E* **70**(5), 057701.

Mathlouthi, I., Gendreau, M. and Potvin, J.-Y. (2016), Mixed integer programming for a multi-attribute technician routing and scheduling problem, Technical report, Technical report, Universite of Montreal.
URL: *https://www.cirrelt.ca/DocumentsTravail/CIRRELT-2016-23.pdf*

Minsky, M. (1961), 'Steps toward artificial intelligence', *Proceedings of the IRE* **49**(1), 8–30.

Mısır, M., Smet, P. and Berghe, G. V. (2015), 'An analysis of generalised heuristics for vehicle routing and personnel rostering problems', *Journal of the Operational Research Society* **66**(5), 858–870.

Montané, F. A. T. and Galvao, R. D. (2006), 'A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service', *Computers & Operations Research* **33**(3), 595–619.

Montoya, C., Bellenguez-Morineau, O., Pinson, E. and Rivreau, D. (2015), Integrated column generation and lagrangian relaxation approach for the multi-skill project scheduling problem, *in* 'Handbook on Project Management and Scheduling Vol. 1', Springer, pp. 565–586.

Osman, I. H. (1993), 'Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem', *Annals of operations research* **41**(4), 421–451.

Paessens, H. (1988), 'The savings algorithm for the vehicle routing problem', *European Journal of Operational Research* **34**(3), 336–344.

Panwalkar, S. et al. (2016), 'The proportionate two-machine no-wait job shop scheduling problemauthor-name: Koulamas, christos', *European Journal of Operational Research* **252**(1), 131–135.

Paraskevopoulos, D. C., Laporte, G., Repoussis, P. P. and Tarantilis, C. D. (2017), 'Resource constrained routing and scheduling: Review and research prospects', *European Journal of Operational Research* .

Park, Y., Khosiawan, Y., Moon, I., Janardhanan, M. N. and Nielsen, I. (2016), Scheduling system for multiple unmanned aerial vehicles in indoor environments using the csp approach, *in* 'Intelligent Decision Technologies 2016', Springer, pp. 77–87.

Pillac, V., Gendreau, M., Guéret, C. and Medaglia, A. L. (2013), 'A review of dynamic vehicle routing problems', *European Journal of Operational Research* **225**(1), 1–11.

Pillac, V., Guéret, C. and Medaglia, A. (2012), On the dynamic technician routing and scheduling problem, *in* 'Proceedings of the 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS 2012), Mikonos, Greece'.

Pillac, V., Gueret, C. and Medaglia, A. L. (2013), 'A parallel matheuristic for the technician routing and scheduling problem', *Optimization Letters* **7**(7), 1525–1535.

Pinheiro, R. L., Landa-Silva, D. and Atkin, J. (2016), A variable neighbourhood search for the workforce scheduling and routing problem, *in* 'Advances in Nature and Biologically Inspired Computing', Springer, pp. 247–259.

Pokutta, S. and Stauffer, G. (2009), 'France telecom workforce scheduling problem: a challenge', *RAIRO-Operations Research* **43**(4), 375–386.

Qu, R. and He, F. (2009), A hybrid constraint programming approach for nurse rostering problems, *in* 'Applications and innovations in intelligent systems XVI', Springer, pp. 211–224.

Rabadi, G. (2016), 'Heuristics, metaheuristics and approximate methods in planning and scheduling-2016.', *International Series in Operations Research & Management Science (ISSN 2214-7934)* **236**.

Rasmussen, M. S., Justesen, T., Dohn, A. and Larsen, J. (2012), 'The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies', *European Journal of Operational Research* **219**(3), 598–610.

Reid, K. N., Li, J. and Swan, J. (2016), Variable neighbourhood search: A case study for a highly-constrained workforce scheduling problem, *in* 'IEEE SSCI 2016: IEEE Symposium Series on Computational Intelligence'.

Rest, K.-D. and Hirsch, P. (2016), 'Daily scheduling of home health care services using time-dependent public transport', *Flexible Services and Manufacturing Journal* **28**(3), 495–525.

Santos, H. G., Toffolo, T. A., Gomes, R. A. and Ribas, S. (2016), 'Integer programming techniques for the nurse rostering problem', *Annals of Operations Research* **239**(1), 225–251.

Shaw, P. (1998), Using constraint programming and local search methods to solve vehicle routing problems, *in* 'International Conference on Principles and Practice of Constraint Programming', Springer, pp. 417–431.

Shi, P. and Landa-Silva, D. (2016), Dynamic programming with approximation function for nurse scheduling, *in* 'International Workshop on Machine Learning, Optimization and Big Data', Springer, pp. 269–280.

Shi, Y., Boudouh, T. and Grunder, O. (2017), 'A hybrid genetic algorithm for a home health care routing problem with time window and fuzzy demand', *Expert Systems with Applications* **72**, 160–176.

Sitompul, D. and Randhawa, S. (1989), 'Nurse scheduling models: a state-of-the-art review.', *Journal of the Society for Health Systems* **2**(1), 62–72.

Society, F. O. R. (2007), 'What is the roadef 2007 challenge'.
URL: *http://challenge.roadef.org/2007/en/*

Solomon, M. M. (1987), 'Algorithms for the vehicle routing and scheduling problems with time window constraints', *Operations research* **35**(2), 254–265.

Szelepcsényi, R. (1988), 'The method of forced enumeration for nondeterministic automata', *Acta informatica* **26**(3), 279–284.

Szeto, W. Y., Wu, Y. and Ho, S. C. (2011), 'An artificial bee colony algorithm for the capacitated vehicle routing problem', *European Journal of Operational Research* **215**(1), 126–135.

Taillard, É., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.-Y. (1997), 'A tabu search heuristic for the vehicle routing problem with soft time windows', *Transportation science* **31**(2), 170–186.

Tanomaru, J. (1995), Staff scheduling by a genetic algorithm with heuristic operators, *in* 'Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on', Vol. 3, IEEE, pp. 1951–1956.

Titiloye, O. and Crispin, A. (2011*a*), Graph coloring with a distributed hybrid quantum annealing algorithm, *in* 'KES International Symposium on Agent and Multi-Agent Systems: Technologies and Applications', Springer, pp. 553–562.

Titiloye, O. and Crispin, A. (2011*b*), 'Quantum annealing of the graph coloring problem', *Discrete Optimization* **8**(2), 376–384.

Todosijević, R., Hanafi, S., Urošević, D., Jarboui, B. and Gendron, B. (2017), 'A general variable neighborhood search for the swap-body vehicle routing problem', *Computers & Operations Research* **78**, 468–479.

Tricoire, F., Bostel, N., Dejax, P. and Guez, P. (2013), 'Exact and hybrid methods for the multiperiod field service routing problem', *Central European Journal of Operations Research* **21**(2), 359–377.

Tsang, E. and Voudouris, C. (1997), 'Fast local search and guided local search and their application to british telecom's workforce scheduling problem', *Operations Research Letters* **20**(3), 119–127.

Turing, A. M. (1936), On computable numbers, with an application to the entscheidungs problem, *in* 'Proceedings of the London Mathematical Society', Vol. 2(42), pp. 230–265.
URL: *Retrieved from https://www.cs.virginia.edu/ robins/Turing*$_{paper}$*$_1$936.pdf*

Van den Bergh, J., Beliën, J., De Bruecker, P., Demeulemeester, E. and De Boeck, L. (2013), 'Personnel scheduling: A literature review', *European Journal of Operational Research* **226**(3), 367–385.

Voudouris, C. and Tsang, E. (1999), 'Guided local search and its application to the traveling salesman problem', *European journal of operational research* **113**(2), 469–499.

Weigel, D. and Cao, B. (1999), 'Applying gis and or techniques to solve sears technician-dispatching and home delivery problems', *Interfaces* **29**(1), 112–130.

Weisstein, E. W. (2017), '"np-problem." from mathworld–a wolfram web resource'.
URL: *http://mathworld.wolfram.com/NP-Problem.html*

Wright, C., McCartt, P., Raines, D. and Oermann, M. H. (2017), 'Implementation and evaluation of self-scheduling in a hospital system', *Journal for Nurses in Professional Development* **33**(1), 19–24.

Xu, J. and Chiu, S. Y. (2001), 'Effective heuristic procedures for a field technician scheduling problem', *Journal of Heuristics* **7**(5), 495–509.

Zamorano, E. and Stolletz, R. (2017), ‘Branch-and-price approaches for the multi-period technician routing and scheduling problem’, *European Journal of Operational Research* **257**(1), 55–68.

Appendix A

Table A.1 Table showing the ROADEF 2007 technician and task scheduling problem instances

Dataset	Jobs	Techs	Budget	Domains	Levels
A1	5	5	0	3	2
A2	5	5	0	3	2
A3	20	7	0	3	2
A4	20	7	0	4	3
A5	50	10	0	3	2
A6	50	10	0	5	4
A7	100	20	0	5	4
A8	100	20	0	5	4
A9	100	20	0	5	4
A10	100	15	0	5	4
B1	200	20	300	4	4
B2	300	30	300	5	3
B3	400	40	500	4	4
B4	400	30	300	40	3
B5	500	50	900	7	4
B6	500	30	300	8	3
B7	500	100	500	10	5
B8	800	150	500	10	4
B9	120	60	100	5	5
B10	120	40	500	5	5
X1	600	60	50	15	4
X2	800	100	500	6	6
X3	300	50	1000	20	3
X4	800	70	50	15	4
X5	600	60	50	15	4
X6	200	20	500	6	6
X7	300	50	1000	20	3
X8	100	30	150	15	7
X9	500	50	50	15	4
X10	500	40	500	15	4

Table A.2 Table describing the precedence constrained technician and task scheduling problem instances

Dataset	Jobs	Precedence	Techs	Domains	Levels	Budget
P1	100	0%	15	3	2	100
P2	100	25%	15	3	2	100
P3	100	50%	15	3	2	100
P4	100	75%	15	3	2	100
P5	100	100%	15	3	2	100
P6	200	0%	25	2	3	200
P7	200	25%	25	2	3	200
P8	200	50%	25	2	3	200
P9	200	75%	25	2	3	200
P10	200	100%	25	2	3	200
P11	400	0%	50	3	3	400
P12	400	25%	50	3	3	400
P13	400	50%	50	3	3	400
P14	400	75%	50	3	3	400
P15	400	100%	50	3	3	400
P16	800	0%	80	4	2	800
P17	800	25%	80	4	2	800
P18	800	50%	80	4	2	800
P19	800	75%	80	4	2	800
P20	800	100%	80	4	2	800
P21	1000	0%	100	3	4	1000
P22	1000	25%	100	3	4	1000
P23	1000	50%	100	3	4	1000
P24	1000	75%	100	3	4	1000
P25	1000	100%	100	3	4	1000

Table A.3 Table showing the service technician routing and scheduling problem $NoTeam$ instances including the number of available technicians

Dataset	Available Technicians
$C101\_5\times4\_NoTeam$	17
$C103\_5\times4\_NoTeam$	15
$C201\_5\times4\_NoTeam$	8
$C203\_5\times4\_NoTeam$	7
$R101\_5\times4\_NoTeam$	25
$R103\_5\times4\_NoTeam$	20
$R201\_5\times4\_NoTeam$	7
$R203\_5\times4\_NoTeam$	8
$RC101\_5\times4\_NoTeam$	22
$RC103\_5\times4\_NoTeam$	18
$RC201\_5\times4\_NoTeam$	9
$RC203\_5\times4\_NoTeam$	7
$C101\_6\times6\_NoTeam$	16
$C103\_6\times6\_NoTeam$	13
$C201\_6\times6\_NoTeam$	7
$C203\_6\times6\_NoTeam$	9
$R101\_6\times6\_NoTeam$	26
$R103\_6\times6\_NoTeam$	19
$R201\_6\times6\_NoTeam$	7
$R203\_6\times6\_NoTeam$	7
$RC101\_6\times6\_NoTeam$	24
$RC103\_6\times6\_NoTeam$	18
$RC201\_6\times6\_NoTeam$	8
$RC203\_6\times6\_NoTeam$	8
$C101\_7\times4\_NoTeam$	17
$C103\_7\times4\_NoTeam$	17
$C201\_7\times4\_NoTeam$	8
$C203\_7\times4\_NoTeam$	8
$R101\_7\times4\_NoTeam$	28
$R103\_7\times4\_NoTeam$	22
$R201\_7\times4\_NoTeam$	10
$R203\_7\times4\_NoTeam$	9
$RC101\_7\times4\_NoTeam$	23
$RC103\_7\times4\_NoTeam$	19
$RC201\_7\times4\_NoTeam$	9
$RC203\_7\times4\_NoTeam$	8

Table A.4 Table showing the service technician routing and scheduling problem *ReducedNoTeam* instances including the number of available technicians

Dataset	Available Technicians
$C101\_5 \times 4\_ReducedNoTeam$	8
$C103\_5 \times 4\_ReducedNoTeam$	8
$C201\_5 \times 4\_ReducedNoTeam$	4
$C203\_5 \times 4\_ReducedNoTeam$	4
$R101\_5 \times 4\_ReducedNoTeam$	12
$R103\_5 \times 4\_ReducedNoTeam$	12
$R201\_5 \times 4\_ReducedNoTeam$	4
$R203\_5 \times 4\_ReducedNoTeam$	4
$RC101\_5 \times 4\_ReducedNoTeam$	11
$RC103\_5 \times 4\_ReducedNoTeam$	11
$RC201\_5 \times 4\_ReducedNoTeam$	5
$RC203\_5 \times 4\_ReducedNoTeam$	5
$C101\_6 \times 6\_ReducedNoTeam$	8
$C103\_6 \times 6\_ReducedNoTeam$	8
$C201\_6 \times 6\_ReducedNoTeam$	4
$C203\_6 \times 6\_ReducedNoTeam$	4
$R101\_6 \times 6\_ReducedNoTeam$	13
$R103\_6 \times 6\_ReducedNoTeam$	13
$R201\_6 \times 6\_ReducedNoTeam$	4
$R203\_6 \times 6\_ReducedNoTeam$	4
$RC101\_6 \times 6\_ReducedNoTeam$	12
$RC103\_6 \times 6\_ReducedNoTeam$	12
$RC201\_6 \times 6\_ReducedNoTeam$	4
$RC203\_6 \times 6\_ReducedNoTeam$	4
$C101\_7 \times 4\_ReducedNoTeam$	9
$C103\_7 \times 4\_ReducedNoTeam$	9
$C201\_7 \times 4\_ReducedNoTeam$	4
$C203\_7 \times 4\_ReducedNoTeam$	4
$R101\_7 \times 4\_ReducedNoTeam$	14
$R103\_7 \times 4\_ReducedNoTeam$	14
$R201\_7 \times 4\_ReducedNoTeam$	5
$R203\_7 \times 4\_ReducedNoTeam$	5
$RC101\_7 \times 4\_ReducedNoTeam$	12
$RC103\_7 \times 4\_ReducedNoTeam$	12
$RC201\_7 \times 4\_ReducedNoTeam$	5
$RC203\_7 \times 4\_ReducedNoTeam$	5

Table A.5 Table summarising the simulated annealing with restart performance metrics on the *NoTeam* problem instances

Dataset	Average Restarts	Average Iterations
$C101\_5\times4$	21	241,350
$C103\_5\times4$	13	213,761
$C201\_5\times4$	13.6	159,699
$C203\_5\times4$	7	149,073
$R101\_5\times4$	26.5	311,760
$R103\_5\times4$	15.5	235,029
$R201\_5\times4$	12.4	163,032
$R203\_5\times4$	6.2	139,696
$RC101\_5\times4$	22.6	284,923
$RC103\_5\times4$	16.8	237,933
$RC201\_5\times4$	13.6	172,695
$RC203\_5\times4$	5.6	137,529
$C101\_6\times6$	18.8	222,147
$C103\_6\times6$	11.7	199,725
$C201\_6\times6$	7.4	97,845
$C203\_6\times6$	3.5	97,611
$R101\_6\times6$	23.3	268,915
$R103\_6\times6$	12.9	212,172
$R201\_6\times6$	9	124,864
$R203\_6\times6$	2.4	95,175
$RC101\_6\times6$	19.8	247,406
$RC103\_6\times6$	12.2	210,023
$RC201\_6\times6$	8.7	129,974
$RC203\_6\times6$	3.1	101,370
$C101\_7\times4$	24.7	286,337
$C103\_7\times4$	16.1	256,830
$C201\_7\times4$	16.6	188,114
$C203\_7\times4$	11.2	189,899
$R101\_7\times4$	29.5	354,903
$R103\_7\times4$	18.1	265,872
$R201\_7\times4$	15.1	193,354
$R203\_7\times4$	7.8	166,176
$RC101\_7\times4$	24.8	313,719
$RC103\_7\times4$	18.3	275,076
$RC201\_7\times4$	15.6	196,620
$RC203\_7\times4$	8.8	175,336

Table A.6 Table summarising the simulated annealing with restart performance metrics on the *ReducedNoTeam* problem instances

Dataset	Average Restarts	Average Iterations
$C101\_5 \times 4$	19.4	237,760
$C103\_5 \times 4$	11	208,798
$C201\_5 \times 4$	11.6	136,426
$C203\_5 \times 4$	4.8	127,628
$R101\_5 \times 4$	22.4	270,029
$R103\_5 \times 4$	17.4	244,082
$R201\_5 \times 4$	8.9	129,428
$R203\_5 \times 4$	3.6	111,218
$RC101\_5 \times 4$	22.3	274,143
$RC103\_5 \times 4$	12.2	233,929
$RC201\_5 \times 4$	10.8	157,719
$RC203\_5 \times 4$	4.9	127,146
$C101\_6 \times 6$	18.9	224,364
$C103\_6 \times 6$	8.5	187,398
$C201\_6 \times 6$	7.3	101,494
$C203\_6 \times 6$	2.2	101,176
$R101\_6 \times 6$	21.6	255,237
$R103\_6 \times 6$	13.3	213,570
$R201\_6 \times 6$	5	103,062
$R203\_6 \times 6$	2.9	86,080
$RC101\_6 \times 6$	19.6	248,300
$RC103\_6 \times 6$	13.1	210,875
$RC201\_6 \times 6$	5.7	110,432
$RC203\_6 \times 6$	2.7	88,949
$C101\_7 \times 4$	25.3	282,559
$C103\_7 \times 4$	16.4	249,775
$C201\_7 \times 4$	13.5	145,236
$C203\_7 \times 4$	6.7	136,232
$R101\_7 \times 4$	29.4	331,041
$R103\_7 \times 4$	21.5	282,724
$R201\_7 \times 4$	11.2	154,580
$R203\_7 \times 4$	5.6	125,351
$RC101\_7 \times 4$	27.4	315,893
$RC103\_7 \times 4$	20.5	281,064
$RC201\_7 \times 4$	12.2	161,558
$RC203\_7 \times 4$	5.9	138,643

Appendix B

Solving Technician and Task Scheduling Problems with an Intelligent Decision Heuristic

Amy Khalfay, Alan Crispin, and Keeley Crockett

School of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, Manchester, M1 5GD, UK
{a.khalfay, a.crispin, k.crockett}@mmu.ac.uk

Abstract. This paper proposes a new approach, an intelligent decision (ID) heuristic, to solve a technician and task scheduling problem (TTSP) defined by the ROADEF 2007 challenge. The ID heuristic is unlike other approaches because at each stage the heuristic considers multiple scenarios of team configurations and job assignments. Within the ID heuristic, novel operators have been designed which focus on flexibility in team configurations. Furthermore, outsourcing is a sub-problem of the ROADEF 2007 challenge, so computational experiments have been performed to evaluate various strategies of outsourcing to utilize the ID heuristic. Results obtained using the ID heuristic have been compared against other researchers who have tackled this problem.

Keywords: Technician and task scheduling problem (TTSP), intelligent decision (ID) heuristic and outsourcing

1 Introduction

Technician and task scheduling problems are present in many sectors such as telecommunications, health care and public utilities [1]. The difficulty in finding quality solutions to these constrained problems is recognized in [2] and highlights the benefits of optimized staff scheduling such as; customer and workforce satisfaction and economic savings. The ROADEF 2007 challenge, organized by the French Operational Research Society, encouraged researchers to find efficient ways of solving France Telecom's optimization problem [3]. France Telecom wished to protect their market share and maintain a high level of customer service whilst limiting the growth of their workforce [4]. Characteristics featured in the ROADEF 2007 challenge problem are still applicable to the problems faced today in many organisations [5].

The ROADEF 2007 challenge problem can be summarized as follows: each job has domain skill requirements that need a team to be built in order to satisfy the demand. A domain may be a particular area of expertise i.e. electrician and a skill level will represent the proficiency within that domain. Teams are made up of technicians who have intrinsic skill domain levels and days where they are unavailable. In addition, there are also dependency relationships between jobs, prohibiting some jobs being started until others have been completed. Jobs have a priority level, representing how important it is to serve the job as early as possible. Jobs have a completion time and must be started and finished on the same day. Furthermore, in some instances, there is an outsourcing budget available (outsourced jobs do not contribute to the objective function). The ROADEF 2007 challenge comprised of three sets of data; Set A, Set B and Set X, increasing in complexity. Set A instances range from 5-100 jobs, include precedence relationships but no outsourcing budget. Set B and Set X instances range from 110-800 and 100-800 jobs respectively, include precedence relationships and outsourcing budgets.

The ROADEF 2007 challenge attracted a lot of research attention and a number of solution approaches such as; adaptive large neighbourhood search [6], mixed integer programming [7] [8], local search heuristics [9] [10], greedy algorithms [11-13], and greedy randomized adaptive search algorithms [14] have been proposed. All of these approaches use procedures we would classify as single scenario (SS) heuristics. The approaches select a single seed job according to some criteria, and create a team able to service the seed job. The ID heuristic is a new approach, which at each stage evaluates many team and job configuration scenarios. The heuristic selects multiple seed jobs and creates a team able to service each job if possible. For each constructed team, the heuristic then checks which jobs could also be allocated to the team. A utility score is calculated for each scenario, which represents the quality of the job assignments to the team. The highest scoring scenario is then selected, the team is configured and job assignments are made.

In other work on the ROADEF 2007 challenge, team configurations have been rigid and difficult to alter. Three operators presented in this paper were developed to provide flexibility in team configurations using the ID heuristic. The first operator, move with team build, allows a single job to be moved onto a different day by creating a team to service the job. The second operator, decompose and rebuild, allows a single day to be rebuilt within the scheduling horizon. The last operator, decompose and rebuild N, allows N days in the scheduling horizon to be rebuilt. These operators allow distant solutions to be evaluated as they have the ability to change not only team configurations but also the allocation position of a job. Experiments into strategies for outsourcing have previously not been carried out in the evaluation of other heuristics to determine whether the quality of the solution produced is dependent on the choice of outsourcing strategy.

The remainder of this paper is organized as follows. Section 2 presents the formulation of the problem. Section 3 describes the ID heuristic. Section 4 presents the outsourcing strategy testing and results. Section 5 outlines the local operators used within the ID heuristic and section 6 shows the results obtained by the ID heuristic against other researchers work. Lastly, section 7 draws conclusions about the performance of the ID heuristic and directions for further work are identified.

2 Problem Formulation

The aim of the ROADEF 2007 challenge problem is to construct teams over a scheduling horizon in order to service a set of jobs. A set of N jobs must be completed.

Each job i has certain properties, a priority level p where $p = \{1...4\}$, an execution time d_i, a domain skill requirement matrix ${s^i}_{\delta\alpha}$, an outsourcing cost c_i and a set of successor jobs σ_i. All jobs belonging to σ_i may not begin until job i has been completed. There are a set of technicians T, each technician t also has attributes; unavailable days and domain skill levels ${v^t}_{\delta\alpha}$. There are $D = \{ 1...\delta\}$ domains and $A = \{1...\alpha\}$ skill levels within each domain. The scheduling horizon $K = \{1...k\}$ represents an entire solution and each k represents a working day/schedule. Each day is limited to 120 time units with no overtime allowed. Each day k has a set of available technicians T_k, who make up teams $M = \{1...m\}$ and each team m will have job assignments and must stay together for the day. Let $x_{t,k,m} = 1$ if technician t belongs to team m on day k. Let $y_{i,k,m} = 1$ if job i is assigned to team m on day k. The start times of jobs are denoted as b_i. Let $u_{i,i'}$ =1 if jobs i and i' are assigned to the same team on the same day and i' begins after i is completed.

In the Set B and X instances there is an outsourcing budget available, C. Outsourced jobs must adhere to precedence constraints, so if a job is outsourced then so are all successor tasks. Let $z_i = 1$ if job i is outsourced. The objective function of a solution is calculated using Equation (1), a weighted sum of the latest ending times of each priority type. The weightings in the objective function are given by $w_p = \{28, 14, 4, 1\}$ for $p = \{1...4\}$ and e_p is the ending time of the latest job in the scheduling horizon of priority p. However, e_4 is the ending time of the latest job in the scheduling horizon over all priority types.

$$\sum_{p=1}^{4} w_p * e_p \tag{1}$$

Subject to the following constraints;

$$e_p \geq b_i + d_i \qquad \forall\, p \in \{1,2,3\}, i \in N_p \tag{2}$$

$$e_4 \geq b_i + d_i \qquad \forall\ i \in N \tag{3}$$

$$\sum_{m\in M} x_{t,k,m} \leq 1 \qquad \forall\, k \in K, t \in T_k \tag{4}$$

$$\sum_{m\in M} x_{t,k,m} = 0 \qquad \forall\, k \in K, t \in T \setminus T_k \tag{5}$$

$$z_i + \sum_{k\in K}\sum_{m\in M} y_{i,k,m} = 1 \qquad \forall\ i \in N \tag{6}$$

$$y_{i,k,m} * {s^i}_{\delta\alpha} \leq \sum_{t\in T_k} {v^t}_{\delta\alpha} * x_{t,k,m} \qquad \forall\, i \in N, k \in K, m \in M, \alpha \in A, \delta \in D \tag{7}$$

$$b_i + d_i \leq b_{i'} \qquad \forall\, i \in N, i' \in \sigma_i \tag{8}$$

$$120(k-1) * \sum_{m\in M} y_{i,k,m} \leq b_i \qquad \forall\, i \in N, k \in K \tag{9}$$

$$120(k) * \sum_{m\in M} y_{i,k,m} \geq b_i + d_i \qquad \forall\, i \in N, k \in K \tag{10}$$

$$b_i + d_i - (1 - u_{i,i'})M \leq b_{i'} \qquad \forall\, i, i' \in N, i \neq i' \tag{11}$$

$$y_{i,k,m} + y_{i',k,m} - u_{i,i'} - u_{i',i} \leq 1 \qquad \forall\, i, i' \in N, i \neq i', k \in K, m \in M \tag{12}$$

$$\sum z_i * c_i \leq C \qquad \forall\, i \in O \tag{13}$$

$$|\sigma_i| * z_i \leq \sum_{i'\in\sigma_i} z_{i'} \qquad \forall\, i \in N \tag{14}$$

With variables;

$$x_{t,k,m} = \{0,1\} \qquad \forall\, k \in K, m \in M, \mathrm{t} \in T \tag{15}$$
$$y_{i,k,m} = \{0,1\} \qquad \forall\, i \in N, k \in K, m \in M \tag{16}$$
$$u_{i,i'} = \{0,1\} \qquad \forall\; i, i' \in N, i \neq i' \tag{17}$$
$$z_i = \{0,1\} \qquad \forall\, i \in N \tag{18}$$
$$e_p \geq 0 \qquad \forall\; p \in \{1,2,3,4\} \tag{19}$$
$$b_i \geq 0 \qquad \forall\, i \in N \tag{20}$$

Equation (2) states that the latest ending time for each priority group, N_p, must be greater than, or equal to, the start time of every job plus the duration of the job. Equation (3) ensures the latest ending time overall for all jobs is greater than, or equal to, the start time of every job belonging to the set of all jobs N, plus the duration of the job . Equation (4) guarantees that if a technician is available to work, then the technician may only be a member of one team that day. Conversely, Equation (5) confirms that if a technician may not work, then the technician is not a member of any team. Equation (6) ensures that either a job is outsourced or a team completes it during the scheduling horizon. Equation (7) states that if a team completes a job, the team collectively has the skills necessary to service the job. Equation (8) shows that if a job is a successor it may not begin until the predecessor job has been completed. Equations (9&10) ensure that the start and end times of a job lie within the eligible working times of the day that the job is scheduled on. Equation (11) ensures time continuity. If a job is scheduled to be started after another job it does not begin until the other job has been completed. Equation (12) states that if two jobs are to happen sequentially then they must be both scheduled to be completed by the same team on the same day. Equation (13) ensures that the total sum of the outsourced jobs does not exceed the outsourcing budget available. Equation (14) guarantees that if a job i is outsourced, then all successor jobs belonging to σ_i are also outsourced. Equations (15-18) show that variables; $\mathrm{x_{t,k,r}}$, $\mathrm{y_{i,k,r}}$, $\mathrm{u_{i,i'}}$ and $\mathrm{z_i}$ are binary. Lastly, Equations (19 and 20) show that the starting and ending times of jobs are non-negative.

3 Intelligent Decision Heuristic

The variables associated with this heuristic are; the scheduling horizon K, which holds all schedules and is an entire solution, an array $AllJobs$ containing all jobs that need to be scheduled, a schedule k that represents a day within the scheduling horizon K, a set of technicians T who are available for schedule k, an array set containing jobs of priority p that are under consideration for allocation, an array called $hypoteams$ which contains a list of teams that could be made to service jobs belonging to set, an array called $OtherJobs$ which contains further allocations that could be made to each hypothetical team, a team $T1$ which is the best team configuration to make according to a utilization score and lastly, a $PrecedenceArray$, which contains jobs which may not yet be allocated.

Fig 1 shows the pseudo code for the ID construction heuristic. A scheduling horizon K is initialized, which holds individual schedules and makes up an entire solution to the TTSP. The ID heuristic iterates through all jobs until they have been allocated to teams (i.e. the $AllJobs$ array is empty).

```
Create Scheduling Horizon K
While (AllJobs>0)
  Create Schedule k, Add Techs T
  p=1
  While (p<= 4)
    set = AllJobs(p)
    hypoteams = MakeTeams (set)
    if (hypoteams != null)
        Otherjobs= findjobs (hypoteams)
        T1= HighestUtility(hypoteams)
        MakeTeam (t1)
        AddJobs(set)
        Update PrecedenceArray
    Else
        p=p+1
    End While
End While
Initial Solution Created K
```

Fig. 1. Pseudo code for the ID construction heuristic

A new day/schedule k is created by initializing all available technicians as single technician teams. The inner while loop is entered and the set of jobs with priority p is found and stored in an array set. For each job belonging to set, a hypothetical team is made if possible, who if constructed has the time and skills to complete the job. In this heuristic, teams are made in a greedy fashion, at each step, the team member who covers the most skill and wastes the least skill is added as the next member of the team until the job requirements are fulfilled or no members can be added to the team.

A check is performed, if no teams can be created for jobs belonging to the set, then p is incremented and the loop is iterated through again. However, if a team can be created for any job belonging to the set of jobs with priority p, then the heuristic checks which other jobs from set could also be added onto each job list belonging to a hypothetical team. A utility score is calculated for each possible hypothetical team, Equation (21). The utility function is made up of two components, the average over-skill of the team to the jobs they would be allocated Equation (22), and the wasted time Equation (23). The highest scoring utility function is selected; the best hypothetical team is recorded as $T1$. Team $T1$ is constructed and added to the schedule k.

$$Utility = Skill(Scen) * \frac{1}{SlackTime(Scen)} \tag{21}$$

$$Skill\ (Scen) = \frac{1}{size} * \sum_{i=1}^{size} \frac{1}{overskill(hypotheticalteam, i)} \tag{22}$$

$$SlackTime(Scen) = 120 - \sum_{i=1}^{size} d_i \tag{23}$$

All job assignments from set are made to the team and the heuristic then checks whether any jobs are now eligible for allocation due to satisfied precedence constraints. Once no more jobs can be allocated to the current schedule, the heuristic checks whether all jobs have now been allocated, if not another schedule is created and the heuristic iterated through again. The construction heuristic terminates once all jobs have been allocated, and an initial solution has been created. The ID heuristic is used for the initial solution construction and it is used in the improvements phase. The ID heuristic is used by the local operators, which remove a job or selection of jobs and then reallocates them.

4 Outsourcing

In the ROADEF 2007 challenge, outsourcing is treated as a sub-problem and is solved before the scheduling process begins. The selection of outsourced jobs is final i.e. outsourced jobs do not enter the schedules. Experiments into outsourcing strategies have previously not been performed; therefore, the strategies featured in this work are "initial investigation" strategies. The strategies contain features such as the duration of a job, the skill requirements and outsourcing cost. Multiple outsourcing strategies were designed as shown in Table 1. Multiple instances were chosen for experiments from the Set B and Set X datasets, shown in Table 2, as one outsourcing may not be suitable for all problem instances.

Table 1. Outsourcing strategies

Strategy Number	Description
1	Duration
2	Skill Requirements
3	Outsourcing Cost
4	Duration + Skill Requirements
5	Duration + Skill Requirements + Outsourcing Cost
6	Duration + Outsourcing Cost
7	Skill Requirements + Outsourcing Cost

Table 2. Datasets chosen for outsourcing experiments

Set	Data Number	Jobs	Technicians
B	4	400	30
B	8	800	150
X	2	800	100
X	7	300	50
X	10	500	40

Fig.2 shows that the strategy used for outsourcing can greatly affect the quality of the solution produced by the ID heuristic; it also shows that some datasets are more affected by the use of outsourcing strategy than other datasets. For example, the results produced for dataset B8 appear to be consistent; suggesting that the quality of solution produced is independent of the outsourcing strategy chosen. However, for dataset B4 the objective function appears to be heavily dependent on the outsourcing strategy used. It appears that the best strategies are 2, 4 and 5.

The results suggest the most important factors are the skill requirements and the duration of a job. In addition, for dataset X2 there seems to be a relationship between the strategy used and the objective value produced. Strategies 1 and 4 appear to produce the best results suggesting that the most important factors are duration of job followed by the skill requirements of a job.

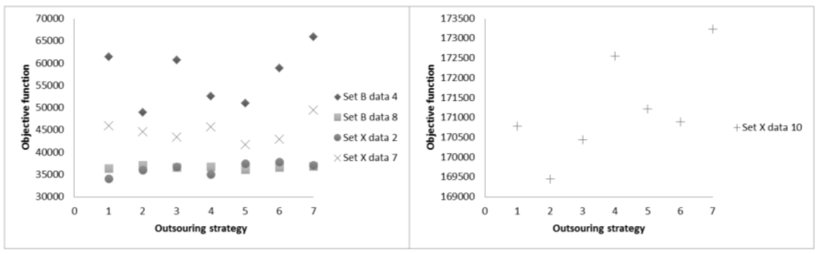


Fig. 2. Mean objective outsourcing strategy testing results

Results produced by dataset X7 also appear to be dependent on the strategy used for outsourcing. The objective values found appear to be of better quality when using strategies 5 and 6, which both include duration time and outsourcing cost of the job in combination. Lastly, for dataset X10, strategy 2 appears to be the best strategy, which considers the skill requirements of the job.

As expected, these results agree with our original assumption that a single outsourcing strategy is not suitable for all datasets. Multiple outsourcing strategies are used on the ROADEF 2007 challenge datasets.

5 Local Operators

A variety of operators were used in this work, some from other combinatorial optimization work such as; move a job, swap two jobs, shuffle a job list, swap three jobs and swap job lists. The following operators where designed for this research and use the ID heuristic; move with team build, decompose and rebuild and decompose and rebuild N. The operator move with team build aims to reallocate a single job. The operator begins by selecting a job at random to reallocate. The ID heuristic then iterates through each day/schedule in the scheduling horizon and checks whether a team can be constructed in order to service the job

The operator decompose and rebuild allows the structure of a day/schedule in the scheduling horizon to change. All jobs assigned to this day/schedule are removed and all team configurations are deconstructed. Each team member is assigned to his or her own individual team. The ID heuristic is then used to reallocate the removed jobs and to make team configurations able to satisfy the jobs. This operator is focused on exploring different team configurations.

The operator decompose and rebuild N allows the structure of a partial amount of the scheduling horizon to change. A value of N is selected which represents how many days/schedules in the scheduling horizon will be decomposed and rebuilt. The schedules are selected at random and ordered from the earliest to latest schedule. All jobs are removed from the schedules and teams are decomposed into single technician teams. The ID heuristic then iterates through the days/schedules, reallocating the jobs and constructs teams. This operator allows not only team configurations to change but also jobs to move across the scheduling horizon.

During the improvement phase of the ID heuristic, an operator is picked randomly (with equal probability) and applied to the current solution. The candidate solution is then evaluated using an Iterative Local Search (ILS) metaheuristic. ILS can be classified as a multi start technique and is conceptually simple [15]. ILS has similarities to hill climbing, however it contains a mechanism to escape locally optimal points within the search space. The ILS metaheuristic has two parameters, the step size N and the kick type. The step size N determines how many non-improving moves will be accepted before the heuristic is moved to another area of the search space. Once N non-improving moves have been reached, a kick is applied that transports us to another area of the search space. When implementing ILS many decisions are left to the developer in regards to the step size used and the type of kick to be performed [16].

6 Challenge Comparison

In the ROADEF 2007 challenge, a 20 minute computational time limit was set. The heuristics were run on each dataset five times and the best score was recorded. The columns of Table 3. represent the datasets used, the best-known score, the results achieved by [8] (Hu), [6] (C), [9] (E), [7] (F), [10] (D), [13] (P), [11] (K), [14] (Ha) and the ID heuristic respectively.

For Sets B and X, in most instances outsourcing strategy 5 was used (time, skill and outsourcing cost) as testing showed this was generally the best strategy. However for some datasets B4, B5, B6, X3 and X10 strategy 2 was used (skill requirements) as this resulted in better quality solutions.

The ID heuristic appears to perform well on the Set A instances and matches the performance of the other researchers. In datasets A1-A4 and A6 the ID heuristic also finds the best reported values from the literature.

In the ROADEF 2007 challenge Set B instances there is variation in the results produced by the ID heuristic. In some datasets, the ID heuristic performs competitively B1-B3 and B7-B10 with regards to the solutions found by the other researchers. In other instances, particularly B4-B6, the ID is unable to perform competitively. Interestingly after examination it appears that these datasets have a high volume of precedence and successor relationships which form chains and result in producing elongated solutions. This suggests that a complex aspect of the ROADEF 2007 challenge problem is the interrelationships between jobs (successor and precedence).

The ROADEF 2007 challenge Set X instances were not tackled by [9-12]. For some datasets, the ID heuristic performs well, finding solutions that are competitive (X1, X7 and X9). For other datasets (X2, X3, X4 and X6), the ID heuristic found worse quality results than the other researchers, also suggesting there are further complexities to this problem that need to be studied.

Table 3. ROADEF 2007 Challenge Comparison

Data	BKS	Hu	C	E	F	D	J	P	K	Ha	ID
A1	2340	2340	2340	2340	2340	2340	2490	2340	2340	2340	2340
A2	4755	5580	4755	4755	4755	4755	4755	4755	4755	4755	4755
A3	11880	12600	11880	11880	11880	13068	12600	11880	11880	11880	11880
A4	13452	13620	13452	14040	13452	13620	14040	14760	13452	13452	13452
A5	28845	30150	29335	29700	29335	31236	32400	33480	29335	28845	29040
A6	18795	20280	18795	18795	20005	21576	21120	22380	19935	18870	18795
A7	30540	32520	30540	30540	30960	40116	32520	33360	31050	30840	30660
A8	16920	18960	17700	20100	17355	23115	19380	21180	17587	17355	20100
A9	27348	29328	27692	28020	28280	34056	28280	30000	28028	27692	28020
A10	38296	40650	38636	38296	39300	52348	41580	42740	40350	40020	39000
B1	33900	34710	37200	34395	34575	58968	46995	44025	43620	43860	34410
B2	15870	17970	17070	15870	16775	28989	19890	21240	20010	20655	18600
B3	16005	18060	18015	16020	16275	34368	20340	20280	19575	20565	18210
B4	23775	26115	23775	25305	23925	56382	29460	31815	35385	26025	45855
B5	88680	94200	117540	89700	88920	N/A	100080	122760	119160	120840	119820
B6	26955	30450	27390	27615	28785	N/A	24230	37965	32760	34215	37755
B7	31620	33300	33900	38200	31620	N/A	36060	38820	41220	35460	37140
B8	33030	35490	33240	37440	35520	N/A	35550	34440	39240	33030	36000
B9	28080	28200	29760	32700	28080	N/A	29460	33360	30000	29550	33360
B10	34680	34680	35640	41280	35040	N/A	36960	44640	38040	34920	40680
X1	146220	151140	159300	188595	146220	N/A	N/A	N/A	N/A	181575	178560
X2	7260	9090	8280	8370	7740	N/A	N/A	N/A	N/A	7260	32925
X3	48720	50400	50400	50100	48720	N/A	N/A	N/A	N/A	52680	52920
X4	64600	65400	66780	68120	64600	N/A	N/A	N/A	N/A	72860	74880
X5	144750	147000	157800	183700	144750	N/A	N/A	N/A	N/A	172500	182820
X6	9480	10320	9900	10440	9690	N/A	N/A	N/A	N/A	9480	13020
X7	32040	33240	47760	37200	32040	N/A	N/A	N/A	N/A	46680	40320
X8	23220	23460	24060	25480	23220	N/A	N/A	N/A	N/A	29070	27420
X9	122800	134760	152400	159660	122800	N/A	N/A	N/A	N/A	168240	159600
X10	120330	137040	140520	152040	120330	N/A	N/A	N/A	N/A	178560	160860

7 Conclusion

The ID heuristic has matched some of the best-known solutions to the ROADEF 2007 challenge problem. In 23 out of 30 datasets, the ID heuristic has produced a competitive solution with regard to solutions found by other researchers.

This research has highlighted that there are many complexities in the ROADEF 2007 datasets which arise due to the real-world nature of the technician and task scheduling problem especially relating to the constraints and their relationships [2]. Future work will investigate the precedence relationships within the datasets to ascertain if results can be improved in instances which have a high number of precedence and successor relationships.

Our contributions in this paper to the field are; (i) the ID heuristic, which behaves in a different manner to the heuristics proposed by other researchers, (ii) outsourcing strategy testing which has shown dependency between the strategy chosen and the quality of result produced and (iii) novel operators designed to provide flexibility in team configurations.

Acknowledgments

This research is sponsored by ServicePower Technologies PLC, a worldwide leader at providing innovative mobile workforce management solutions, in cooperation with MMU and KTP.

References

1. Pillac, V., Guéret, C., Medaglia, A.: On the Technician Routing and Scheduling Problem. In: The IX Metaheuristics International Conference, pp. S2-40, Italy (2011)
2. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review of applications, methods and models. EJOR. 153(1), pp. 3-27 (2004)
3. The ROADEF 2007 Challenge, http://challenge.roadef.org/2007/
4. Dutot, P.F., Laugier, A., Bustos, A.M.: Technicians and interventions scheduling for telecommunications (2007) , Available at, http://challenge.roadef.org/ 2007
5. Montoya, C., Bellenguez-Morineau, O., Pinson, E., Rivreau, D.: Integrated column generation and Lagrangian relaxation approach for the multi-skill project scheduling problem. In: Handbook on Project Management and Scheduling Vol. 1, pp. 565-586. Springer, (2015)
6. Cordeau, J.F., Laporte, G., Pasin, F., Ropke, S.: Scheduling technicians and tasks in a telecommunications company. JOS. 13(4), pp. 393-409 (2010)
7. Fırat, M., Hurkens, C.A.J.: An improved MIP-based approach for a multi-skill workforce scheduling problem. JOS. 15(3), pp.363-380 (2012)
8. Hurkens, C.A.: Incorporating the strength of MIP modeling in schedule construction. RAIRO OP RES. 43(04), pp. 409-420 (2009)
9. Estellon, B., Gardi, F., Nouioua, K.: High-performance local search for task scheduling with human resource allocation. In: Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics, pp. 1-15. Springer, (2009)
10. Dongala, S.G.P.: The Problem of Scheduling Technicians and Interventions in a Telecommunications Company (2008)
11. Korteweg, P.: When to hire the A-Team. (2007) ROADEF, Available at, http://challenge.roadef.org/ 2007
12. Jaskowski, W., Wasik, S.: Efficient Greedy Algorithm with Hill Climbing for Technicians and Interventions Scheduling Problem (2007), Available at, http://challenge.roadef.org /2007
13. Pokutta, S., Stauffer, G.: France Telecom workforce scheduling problem: a challenge. RAIRO OP RES. 43(04), pp. 375-386 (2009)
14. Hashimoto, H., Boussier, S., Vasquez, M., Wilbaut, C.: A GRASP-based approach for technicians and interventions scheduling for telecommunications. ANN OP RES. 183(1), pp. 143-161 (2011)
15. Martí, R., Moreno-Vega, J.M., Duarte, A.: Advanced multi-start methods.' In: Handbook of metaheuristics, pp. 265-281. Springer, Boston (2010)
16. Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search. Springer, Heidelberg, Boston (2003)

Appendix C

Intelligent Systems Conference 2017
7-8 September 2017 | London, UK

A Review of Technician and Task Scheduling Problems, Datasets and Solution Approaches

Amy Khalfay
Manchester Metropolitan University
Manchester
United Kingdom
Email: a.khalfay@mmu.ac.uk

Alan Crispin
Manchester Metropolitan University
Manchester
United Kingdom
Email: a.crispin@mmu.ac.uk

Keeley Crockett
Manchester Metropolitan University
Manchester
United Kingdom
Email: k.crockett@mmu.ac.uk

***Abstract*—This paper aims to provide a review of technician and task scheduling problems. A technician and task scheduling problem requires the creation of timetables for employees in order to serve customers, who each have skill requirements. Optimised scheduling can reduce the cost of a workforce and save the employer significant amounts of money whilst maintaining customer and workforce satisfaction. This problem is an NP-hard combinatorial optimisation problem. Over the years, many technician and task scheduling problems have been studied. This is due to the real world nature of the problem, that occurs in many industries such as service maintenance, home healthcare, call centres, forest management and housing development projects. Each problem studied has included a diverse range of constraints such as teaming, priority levels, outsourcing, precedence, routing, time windows and tools and spare parts. We present a study of the problems in the literature, datasets and solution approaches. This review aims to identify promising areas for further research such as focusing on multi-period problems, technician unavailability, teaming and precedence constraints.**

***Keywords*—Technician and task scheduling problems; constraints; combinatorial optimisation; datasets; solution approaches**

I. Introduction

This paper aims to provide a review of the literature based in the field of technician and task scheduling problems (TTSP). A TTSP requires a schedule to be constructed for technicians who are allocated jobs which each require certain levels of skill proficiency in order to be completed. In the last few decades, personnel scheduling problems have become a widely studied research area [1]. This is due to the real world nature of the problem that occurs in a vast number of organisations and businesses. A scheduling problem involves the creation of timetables for employees that satisfy an organisation's demand for services or goods [2]. There have been many personnel scheduling problems studied of which each include varying constraints that must be adhered to. In some problems, there are additional constraints to skill compatibility, such as travel time and location, time windows, teaming etc. which add to the complexity of the problem, and more intelligent solution approaches have to be designed. This paper aims to identify which constraints are usually associated with the TTSP, review the limitations of the current datasets, highlight promising solution approaches and identify areas for further investigation.

The importance of personnel scheduling problems was highlighted by [3] as a way for industries to maintain market share and ensure repeat business. In an increasingly competitive market, service providers are in competition not only in terms of the product they provide but also the quality of customer service provided and after-care. They also predicted this research area to grow as we become more reliant on machinery and technology in the daily running of our lives, and so the maintenance requests of machinery will naturally rise.

The cost of a workforce is usually one of the largest costs incurred by a business. Optimised scheduling can minimize these costs, and therefore save significant amounts of money. Optimised scheduling also has the potential to ensure that shifts are distributed evenly amongst employees and employees work their preferred shifts [2], which can result in a contented workforce, and, in turn, a more productive workforce.

Scheduling problems are NP-hard combinatorial optimisation problems. This means there are no known polynomial time algorithms that can solve them to optimality within a discrete time period [4]. In the literature exact techniques are used to solve problems that contain up to 29 jobs. For this reason, exact techniques are prohibitive for many real world and large scale scheduling problem instances and approximate techniques must be used [5]. Approximate techniques have no guarantee of finding the optimal solution, but generally are able to find quality solutions in short computational times. Approximate techniques have been used to solve problems that contain up to 1000 jobs [6].

There are also many closely related problems to the TTSP. The most notable related problem is the vehicle routing problem (VRP), which evolved from the truck dispatching problem (TDP) studied by [7] in the 1950's. The VRP requires scheduling a fleet of vehicles to serve a set of customers [8]. Each vehicle leaves the depot and visits a subset of customers before returning to the depot. The constraints of the problem are that each customer is visited exactly once. The objective of the VRP is to minimize the sum of the distance travelled by the vehicles [9]. The Euclidean distance between customers is used.

The VRP has also been studied with various side constraints. Constraints such as the capacity of the vehicles [10], multiple depots [11] and time windows in which customers must be visited [12]. The TTSP can be thought of as a generalization of the VRP. The main differences between these scheduling problems are the homogeneousness of the vehicles (capacity) in the VRP compared to heterogeneousness of the workers (skill set) in the TTSP. The VRP is usually concerned with making a schedule for a single day, whereas some TTSPs require scheduling for multiple days.

Another related problem is the home healthcare problem (HHP). The HHP requires the scheduling of skilled personnel to travel to patients based in different locations to administer medication or provide care. This problem can be thought of as a generalization of the VRP with time windows, multiple depots and compatibility constraints [13]. This problem is becoming more prevalent in society as our population ages and private companies begin to work in this area [14]. The main purpose of this problem is to allow elderly patients to be treated in their own homes for as long as possible [15]. The care given can be anything between cleaning and making food, to changing bandages and dressings to administering medication. There are also many side constraints to be considered such as patient preference and travel time dependent on the mode of transportation [16].

As demonstrated the TTSP is related to other personnel scheduling problems. There are shared constraints such as skill compatibility (care providers to patients), travel time (between patients/customers) and time windows (between patient medications). It is reasonable to assume that approaches to solve a TTSP could be adapted in order to solve the VRP or the HHP since problems share common constraints.

The large number of real world domains in which TTSPs occur has attracted many researchers [17]. These types of personnel scheduling problems have occurred in industrial settings for many decades. TTSPs arise in industries such as utility services, gas and electrical services, and general maintenance [18]. The field of TTSPs includes many types of problem, each of which has different sets of constraints and objectives. Usually the objective is to serve all customers in the least costly manner. There are also many similarities between TTSPs, such as the use of time windows or team building, and therefore results obtained in one area may be applied to another [19]. Due to the wide ranging types of problem and the number of areas in which the problems occur, finding efficient solution approaches can not only benefit the employer, the employee and the customer but also has the potential to impact on the environment.

The main purpose of this paper is to;

- review the constraints that are associated with TTSPs in order to find out which constraints are used most frequently, and which constraints need further investigation
- review the TTSP datasets available in order to identify if there are any additional constraints to be considered, or constraints that have not been studied simultaneously
- review the solution approaches used to solve TTSPs to highlight promising solution approaches and scalability and robustness
- determine key areas for future research in the field of TTSPs

This paper is structured as follows; section II describes the constraints associated with the TTSP, section III presents the datasets used in this field, section IV discusses the solution approaches that have been used to solve TTSPs, section V describes the metaheuristics used, section VI identifies the current gaps and limitations within the literature and lastly, section VII concludes on this paper and our contributions.

II. TTSP AND VARIATIONS

There have been many constraints featured in TTSPs. However, we believe the main constraint that makes a personnel scheduling problem a TTSP is;

"the allocation of skilled personnel to tasks, each of which have skill requirements that must be satisfied"

The following subsections will explain each additional constraint that has been featured throughout the literature in variations of the TTSP that have been studied.

A. Routing

The complexity of routing is an important aspect of many TTSPs. In most settings, the set of customers will each be positioned in their own location, and so skilled personnel must travel between these locations. The occurrence of routing also usually implies that there is a central depot, from which skilled personnel depart and return to at the beginning and end of the day. The travel times are accounted for between these locations and the depot and are usually calculated as Euclidean distance. However, in many practical situations travel time can be variable dependent on the day of the week or the time of the day that the journey between two customers is made. In addition, the use of a SAT NAV can also estimate the journey time which takes into account traffic incidents, roadworks and congestion etc.

Problems featuring the complexity of routing in the literature include [20], [21] and [22] who studied technician routing and scheduling problems (TRSP), [23] who studied a service technician routing and scheduling problem (STRSP), [24] who studied the multi-period field service routing problem (MFSRP), and [25] who studied the multi-period technician routing and scheduling problem (MTRSP).

B. Teaming

Teaming is also an important constraint particularly in the utility and service maintenance sector. Some jobs require skills which cannot be fulfilled by a single skilled worker. Therefore, a team must be made where the cumulative skills of the team satisfy the job's skill requirements. In all problems featuring the complexity of teaming in the literature, the team is configured at the beginning of the working day and will stay together for the whole day. For this reason, team formation decisions require much consideration in order to make sure the team's skills are utilized. In the real world, a manual planner may also take into account other information such as knowing which technicians work well together. Problems featuring the complexity of teaming include the ROADEF 2007 challenge [26] which was a TTSP and [23] which was a TRSP.

C. Single period/ multi-period

Furthermore, there are also differences in the size of the scheduling horizon, i.e. the number of working days available. Some TTSPs in the literature have been adapted from VRPs and so, have only one scheduling day, because in the VRP the objective is to minimize the distance. However, there are some multi-period problems, where typically the schedule can cover over a month of working days. Considerations must then be given to the acceptable amount of time allowed to find feasible solutions. Typically, a schedule that will cover a month should be given more computational time than a single day problem as the impact of a bad solution will be more costly. Single day problems have been studied by [20] and [23]. The ROADEF 2007 challenge problem was a multi-period problem [27]. Multi-period problems were also studied by [21] and [25].

D. Time windows

Time windows are an emerging constraint in TTSPs. Service providers are seeking to differentiate themselves from each other to maintain market share. Allowing a customer to choose their own preferred time slot may result in a better customer experience, customer satisfaction and result in repeat business. Time windows have been used by [20], [21], [23], [24] and [25]. The size of the time window can vary depending on the industry. Time windows may span half a day, a few hours or, in some delivery services, a single hour.

E. Precedence

The complexity of precedence and successor relationships can also be a common occurrence in some types of scheduling problem. A precedence relationship between jobs i and i' implies that job i' may not begin until job i has been completed. We can say that i' is a successor of i, and i precedes i'. Precedence and successor constraints occur in sectors such as utility and electrical maintenance and housing projects. For example, in housing projects the decorations may not begin until all the plastering is completed, or in the utility services, fixing a fault may not begin until the road has been dug up and pipes can be accessed. Furthermore, chains of precedence relationships can develop between a set of jobs which adds to the complexity of the scheduling problem. Precedence constraints have been used in the ROADEF 2007 challenge, which was tackled by [28] and [29], and in other scheduling problems by [25] and [30].

F. Priority

Priority levels are also a consideration in some TTSPs. A priority level quantifies how important it is to serve a job as early as possible in comparison to other jobs. These scenarios can arise where there are customers who place a lot of orders to a business so they will be seen as more important than customers with smaller orders. Priority levels can also be used to classify the seriousness of a fault in the maintenance sector, for example the water being off in a town compared to a minor leak in a single customer's house. Priority levels have been included in the ROADEF 2007 challenge [26] and research by [21].

G. Tools and spare parts

Tools and spare parts are an important aspect in the maintenance and repair sector. Generally there will be a finite number of tools that must be shared between the technicians. A complicating factor is that the tools are usually located at a central depot, and so may need to be collected and returned. Furthermore, spare parts are non replenishable, and therefore consideration must be given to stock levels and so on. A problem where technicians began their routes with a set of tools and spare parts was studied by [20].

H. Unavailability

In some problems, there are also days when some technicians may not be available. This is also an emerging constraint, as in law consideration must now be given to people with special circumstances in relation to the working patterns and hours. Unavailability of resources particularly affects the process of building a team as all members must be available and there may be a trade off between sending someone over qualified against waiting for a technician to become available. Most work does not consider the unavailability of resources [1], however the ROADEF 2007 challenge does [31].

I. Dynamic

In some industries there may also be dynamic jobs. A dynamic job is a job that will need to be fitted into the schedule and arrives in real time. It is common in the water, electricity and gas services, where there are faults that must be fixed quickly. In addition the emergency service is also a sector where jobs continually arrive and must be allocated. When emergency jobs arrive in real time, any jobs that have begun are fixed in position. Only jobs that have not yet begun can be moved in order to fit in the emergency job. A dynamic scheduling problem was studied by [32].

III. DATASETS

There are numerous sets of data based on the TTSP. In fact, some of the datasets for technician and task scheduling have actually been adapted from vehicle routing problem datasets proposed by [12]. Each problem contains a different set of constraints. For example some problems include the complexity of teaming, routing, job relationships, priority levels and tools and spare parts etc. Table I shows the datasets that have been studied in this field, along with the constraints each problem featured and the size of the problems.

A. Technician and task scheduling problem

A TTSP was the basis of the ROADEF 2007 challenge. The ROADEF challenge is a bi-annual competition proposed by the French Operational Research Society, that invites researchers to compete to find efficient ways of solving combinatorial optimisation problems. In 2007, the problem was a TTSP, and used real world datasets provided by France Telecom, containing data instances ranging from 5 to 800 jobs and 5 to 150 technicians. The aim of the problem is to allocate a set of jobs over a scheduling horizon to a set of teams. Teams are made up of technicians, each with intrinsic skill domain levels and days within the scheduling horizon when they are

TABLE I. TECHNICIAN AND TASK SCHEDULING PROBLEMS AND VARIATIONS

Problem Type	Constraints	No. of Customers
TTSP [26]	Teaming Resource unavailability Priority levels Outsourcing Skill Job relationships	800
TRSP [20]	Routing Time windows Skill Tools and spare parts	100
TRSP [23]	Teaming Routing Time windows Skill	100
FTSP [6]	Routing Time windows	1000
MTRSP [25]	Routing Time windows Skill Teaming	25/27
MFSRP [24]	Routing Time windows Skill Resource unavailability Breaks	100
MTRSP [21]	Routing Time windows Skill Tools and spare parts Priority	25
DTRSP [32]	Routing Time windows Skill Tools and spare parts Dynamic	100

not available. Each job has a priority level indicating how important it is to serve the job as early as possible. In some problem instances there is an available outsourcing budget that can be utilized. Jobs can also have relationships with other jobs, which can be precedence or successor relationships, where a job may not begin until another has been completed. Jobs have skill requirements which must be satisfied by the team which serves the job.

B. *Technician routing and scheduling problem*

TRSPs in the literature include problems studied by [20] and [23]. The work by [23] extended the vehicle routing problem instances proposed by [12] into a TRSP. The problem generated skill requirements for each job based on the skill requirements present in the ROADEF 2007 datasets. In this problem, teams leave the depot and travel to service customers. This version of the problem negated some constraints that are present in the ROADEF 2007 challenge problem, such as precedence and successor relationships, priority levels, technician unavailability, length of a working day and outsourcing budgets, but did contain the complexity of routing.

The work by [20] extended the instances in [12] by generating skills, tools and spare parts information. This problem, the technician routing and scheduling problem, did not include the complexity of teaming or precedence. However, this is the first work we are aware of that included the complexity of tools and spare parts, an important aspect of service maintenance problems.

C. *Field technician scheduling problem*

A field technician scheduling problem (FTSP) was proposed by [6], where jobs had to be serviced at different locations within a time window. This research did not treat skill compatibility as a hard constraint [19] which is the under pinning constraint that makes a scheduling problem a TTSP or a variation of the TTSP. The objective was to maximise the number of served jobs with a predefined time period whilst minimizing the cost of the workforce. This paper tested problem instances with up to 1000 jobs, which is representative of the scale of real world problems that occur in industry.

D. *Multi-period technician routing and scheduling problem*

Artificial datasets were used by [21] to solve the multi-period technician routing and scheduling problem (MTRSP). The datasets contained up to 25 jobs and CPLEX was used to solve the mixed integer programming model. The problem included complexities such as skill requirements, priority levels, time windows, breaks and overtime. This work found that CPLEX could only solve all instances with 10 jobs to allocate within reasonable computational times. This paper demonstrated how the computational time rapidly increases with problem size and complexity, and the need for approximate scalable solution approaches.

Both artificial and real world datasets were used by [25]. The artificial datasets contained up to 25 jobs, and the real data instances contained up to 27 jobs. This paper again emphasized the difficulties faced with scalability and robustness of solution approaches, and the need for hybridized approaches.

E. *Multi-period field service routing problem*

The multi-period field service routing problem (MFSRP) studied by [24] used both exact and hybrid solution approaches. The datasets used in this research were artificial and contained two sets, C4 small instances up to 40 jobs and C1 up to 100 jobs. This paper highlighted that even after 7 days of computational time, the branch and price technique was not able to find the optimal solution for 2 out of 5 instances. However, when using some heuristic techniques within branch and bound, a solution can be found for all instances within 24 hours of computational time.

F. *Dynamic technician routing and scheduling problem*

A dynamic technician routing and scheduling problem (DTRSP) was studied by [32]. In this problem new job requests appear as the schedule is implemented in real time. This is another aspect of a real world situation faced by industry. These datasets were created extending VRP instances from [12].

IV. Solution Approaches

Many approaches have been applied to solve the TTSP and its variations. Both exact and approximate approaches have been used. Table II shows some of the solution approaches that have been applied to TTSPs and their variants.

A. Approximate Mixed Integer Programming

Approximate mixed integer programming (MIP) techniques have been used by [28] and [33] to solve the TTSP proposed by the ROADEF 2007 challenge [26]. In these approaches, a construction algorithm was designed that broke down the overall scheduling problem into smaller sub problems. These sub problems then used the CPLEX library to solve these smaller integer programming (IP) and integer linear programming (ILP) problems. These approaches did not contain an improvement phase, but the construction algorithm produced high quality solutions, and solved problems with up to 800 jobs.

B. Adaptive Large Neighbourhood Search

Adaptive large neighbourhood search heuristics have been used by both [23] and [34] to solve a TTSP and a TRSP. The ALNS heuristic is based on the work by [39] with the large neighbourhood search (LNS) heuristic. The ALNS uses several destroy and repair operators. A destroy operator removes a portion of the current solution, and the repair operator reinserts the removed tasks back into the solution. In the ALNS, the effectiveness of each destroy and repair operator is tracked such that operators that have performed well so far are more likely to be selected, which will aid the search for quality solutions.

C. Greedy Randomized Adaptive Search Procedure

A greedy randomized adaptive search procedure (GRASP) was used by [35] to solve the ROADEF 2007 challenge [26]. This approach proved to be a successful one, [35] won 1st place in the student category of the ROADEF 2007 challenge. The GRASP comprises of two components, a greedy heuristic and then a local search phase. A solution is made using a greedy algorithm, iteratively selecting the best decision at each stage of the scheduling process. Local search is then used to try to improve the solution for a short amount of computational time. This process is run multiple times and the best solution found is recorded.

D. Local Search

Local search (LS) has been used by [31] to solve the ROADEF 2007 challenge. This heuristic came joint 2nd place with the ALNS used by [34]. In this heuristic, an initial solution is constructed using a greedy heuristic. Next, in the improvement phase, operators are iteratively applied in order to reduce the objective function.

E. Mathheuristic

The mathheuirtsic is a novel heuristic proposed by [20]. The heuristic was created in order to solve a TRSP that included many constraints. The mathheuirtsic is composed of three phases, a construction algorithm, a parallel ALNS and a mathematical programming post optimisation phase. The parallel ALNS takes advantage of the parallel architectures which results in a significant computation speed up.

F. Intelligent Decision heuristic

The intelligent decision heuristic is a novel heuristic developed to tackle the ROADEF 2007 challenge problem instances [37]. This heuristic considers many different scenarios of team configurations at each stage of the allocation process. Furthermore, this heuristic contained operators that provided flexibility within team configurations.

G. Branch and price

Branch and price is an exact solution technique. This method has proved popular as many authors have used this technique [24], [25] and [38]. However, exact solution approaches are only suitable for small sized problems. Branch and price combines branch and bound with column generation techniques in order to solve large IP problems.

H. MIP Programming

MIP programming has been used by [21] to solve a multi skill technician routing and scheduling problem. Again, this solution approach is only suitable on small sized problems and so would not be suitable on an industrial scale. In MIP, variables are restricted to take integer values.

V. Metaheuristics

Most approximate solution approaches to solve TTSPs are two phase. In the first phase, an initial solution is generated. In the second phase, local operators are applied which perturb the current solution generating a neighbouring solution. This neighbouring solution has to be evaluated, using a metaheuristic. According to [5]

"metaheuristics are high level strategies
for exploring search spaces by using different methods"

The following subsections will describe three popular trajectory metaheuristics, hill climbing, iterative local search and simulated annealing. Each of these metaheuristics has been used successfully to solve TTSPs and their variants.

A. Hill Climbing

Hill climbing is the most simple and easy to implement metaheuristic. Figure 1 illustrates the hill climbing metaheuristic. First an initial solution is generated using a construction heuristic. It is recorded as the best solution on line 1. On each iteration a local operator o is selected randomly applied to solution S which generates neighbouring solution S' on line 4. If solution S' is of better quality than solution S then it is accepted and becomes the new current solution S on line 6. If solution S is now of better quality than the best solution, then it replaces it on line 8. Once the termination criterion has

TABLE II. SOLUTION APPROACHES

Type	Approach	Problem
Heuristic	Approximate Mixed Integer Programming	TTSP [28], [33]
	ALNS	TTSP [34] TRSP [23]
	GRASP	TTSP [35] FTSP [6]
	Local Search	TTSP [31], [36]
	Parallel Matheuristic	TRSP [20]
	Intelligent Decision Heuristic	TTSP [37]
Exact	Branch and Price	MFSRP [24] MTRSP [25] TRSP [38]
	Mixed Integer Programming	MTRSP [21]

been met, the best solution is output. Hill climbing has been proved to be successful in solving a range of combinatorial optimisation problems and was used by [31]. One of the drawbacks of the hill climbing metaheuristic is that it does not incorporate an escape mechanism. For this reason, it can get stuck in local optima.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators

```
1: $S_{Best} \leftarrow S$
2: while termination criteria not met do
3:     $randomly \quad choose \quad o \in O$
4:     $S' \leftarrow o(S)$
5:     if $S' \leq S$ then
6:         $S \leftarrow S'$
7:         if $S \leq S_{Best}$ then
8:             $S_{Best} \leftarrow S$
9:         end if
10:    end if
11: end while
12: return $S_{Best}$
```

Fig. 1. Hill Climbing

B. Iterative Local Search

Iterative local search (ILS) is an extension to the hill climbing metaheuristic and does include an escape mechanism. Iterative local search is classed as a multi start technique [40] and contains a diversification quality [41]. There is one parameter associated with this metaheuristic, the step size N. The implementation of ILS is shown in Figure 2. Firstly, the variable *count* is assigned the value 0. On each iteration a local operator o is randomly selected and applied to solution S which generates neighbouring solution S' on line 5. If solution S' is of better quality than solution S then it is accepted and becomes the current solution S on line 7.

If solution S is now of better quality than the best solution, then it replaces it on line 9, and the *count* variable is reset to 0 as an improving move has been found. If the neighbouring solution is not of better quality then the variable *count* is incremented by one. On line 15 a check is performed, if the number of non improving moves is equal to the maximum step size N, then a local operator is applied to S generating S' on line 16. The solution S' then replaces S on line 17 to become the current solution and the search will continue from this point

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, N: maximum steps before beginning from best solution, *count*: counter for iterations,

```
1: $S_{Best} \leftarrow S$
2: $count \leftarrow 0$
3: while termination criteria not met do
4:     $randomly \quad choose \quad o \in O$
5:     $S' \leftarrow o(S)$
6:     if $S' \leq S$ then
7:         $S \leftarrow S'$
8:         if $S \leq S_{Best}$ then
9:             $S_{Best} \leftarrow S$
10:            $count \leftarrow 0$
11:        end if
12:    else
13:        $count \leftarrow count + 1$
14:    end if
15:    if $count = N$ then
16:        $S' \leftarrow o(S)$
17:        $S \leftarrow S'$
18:        $count \leftarrow 0$
19:    end if
20: end while
21: return $S_{Best}$
```

Fig. 2. Iterative Local Search

in the solution space. This feature of the ILS metaheuristic allows the algorithm to escape local minima when it becomes stuck. ILS can ensure that previously unvisited regions of the solution space are searched.

C. Simulated Annealing

Simulated annealing is one of the most widely used metaheuristic techniques. This metaheuristic was first proposed by [42] and has shown to be able to produce quality results not only in the field of TTSPs [34], but in other optimisation problems.

This metaheuristic has two parameters, the temperature T and the decrement δT. Simulated annealing has the ability to escape local optima, through the ability to accept a worse quality solution. The chance of accepting a worse quality solution is controlled by the temperature parameter. The simulated annealing metaheuristic is shown in Figure 3. On each iteration

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, T : initial temperature, δT: the cooling rate

```
1: $S_{Best} \leftarrow S$
2: while termination criteria not met do
3:     $randomly \quad choose \quad o \in O$
4:     $S' \leftarrow o(S)$
5:     if $S' \leq S$ then
6:         $S \leftarrow S'$
7:         if $S \leq S_{Best}$ then
8:             $S_{Best} \leftarrow S$
9:         end if
10:    else
11:        $r \leftarrow random(0, 1)$
12:        $p \leftarrow exp^(S' - S)/T$
13:        if $p \geq r$ then
14:            $S \leftarrow S'$
15:        end if
16:    end if
17:    $T \leftarrow T \cdot \delta T$
18: end while
19: return $S_{Best}$
```

Fig. 3. Simulated Annealing

a local operator is randomly selected and applied to solution S which generates S'. If S' is of better quality then it replaces S on line 6. If solution S is better than the best found solution so far then it replaces S_{Best} on line 8. However, if solution S' is of worse quality than solution S then a probability p is calculated on line 12. The probability p is the negative exponential of the difference in solution quality divided by the temperature parameter. If the probability p is greater than a random number generated on the interval [0,1] then S' is accepted and replaces S on line 14. After each iteration the temperature parameter is decremented using δT, reducing the likelihood of accepting a worse quality solution. Once the termination criterion has been met the best solution is output.

D. Comparison of metaheuristics

Each of these metaheuristics has been applied successfully to solve TTSPs or their variants. A hill climbing metaheuristic was used by [31] to solve the ROADEF 2007 challenge. A simulated annealing metaheuristic was used by [34] to also solve the ROADEF 2007 challenge. Both [31] and [34] ranked 2nd place in the competition. An iterative local search metaheuristic was also used by [37] to solve the ROADEF 2007 challenge and produced competitive results.

It seems that one of the important qualities of a metaheuristic is the ability to escape local optima. In such complex problems the solution space is full of peaks and troughs. It can be easy to end up in a trough and so, there is a need to be able to move through the solution space by accepting a worse solution in the hope of being able to find a better quality one.

VI. Discussion

As demonstrated, the field of TTSPs is a vast research field that needs continued investigation. There is a wealth of industrial applications which means that breakthroughs in this field have the potential to make both environmental and financial impacts in closely related fields.

The complexity of teaming has been studied on relatively small problem sizes, for example in [20] and [23] the problem instances contain at most 100 jobs that were adapted from VRPs. The ROADEF 2007 challenge does deal with teaming with larger instances, up to 800 jobs, but routing is not part of the problem definition. The complexity of routing has been studied in many problems, with usually up to 100 jobs. Routing was considered in [6] with 1000 jobs, but skill compatibility was not treated as a hard constraint.

Precedence constraints are featured in the ROADEF 2007 challenge but are absent from many other problems that have been studied. Precedence constraints can occur in many application areas such as home health care. In the field of TTSPs precedence constraints are featured in housing developments, utility and electrical services problems. Precedence constraints add significant complexity to the problem and can influence the design of a solution approach. It is vital that research is undertaken that considers precedence constraints because they can occur in many sectors and pose significant challenges when designing a solution approach.

Furthermore, the largest problems studied have included up to 1000 jobs in [6] although skill was not treated as a hard constraint, which we believe is the fundamental characteristic that makes a TTSP. A problem with up to 800 jobs was studied in [26]. However, in most problems up to 100 jobs are considered. In many industrial scenarios there will be many jobs to schedule over a larger geographical area and realistically sized problem instances are needed in order to validate solution approaches.

The lack of multi-period problems has also meant that the complexity of the unavailability of resources needs further research and investigation. Typically many TTSP problems and their variants are either adapted from VRP instances or are artificial. The use of exact solution approaches has also prohibited the use of multi-period problems as the problem size has had to be kept relatively small. The unavailability of resources is also directly linked to the complexity of teaming, as all team members must be available to join the team. In large organisations, it is necessary to schedule a workforce over multiple days, accounting for technician unavailability and, in some organisations, create teams to complete jobs.

VII. Conclusion

This paper has given an outline of the TTSPs and variations featured in the literature. We have also presented the datasets available for researchers and the solution approaches used, both exact and approximate.

Our contributions are; (i) a comprehensive study of the constraints that have been associated with the TTSP and variants, and an analysis of which constraints have been studied simultaneously, (ii) review of the datasets used which has highlighted the limitations of the datasets available (in terms of problem size, number of scheduling days and the source of the data i.e. artificial and adapted) and (iii) a review of solution approaches which has shown the limitations of exact approaches and the need for heuristic approaches to solve industry sized problems.

We conclude that future research in the field would benefit from being focused on multi-period problems, technician unavailability, teaming and precedence constraints.

Acknowledgment

This research is sponsored by ServicePower Technologies PLC, a worldwide leader in providing innovative mobile workforce management solutions, in cooperation with MMU and KTP.

References

[1] J. Van den Bergh, J. Beliën, P. De Bruecker, E. Demeulemeester, and L. De Boeck, "Personnel scheduling: A literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367–385, 2013.

[2] A. T. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European journal of operational research*, vol. 153, no. 1, pp. 3–27, 2004.

[3] D. L. Haugen and A. V. Hill, "Scheduling to improve field service quality*," *Decision Sciences*, vol. 30, no. 3, pp. 783–804, 1999.

[4] M. Krishnamoorthy, A. T. Ernst, and D. Baatar, "Algorithms for large scale shift minimisation personnel task scheduling problems," *European Journal of Operational Research*, vol. 219, no. 1, pp. 34–48, 2012.

[5] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308, 2003.

[6] J. Xu and S. Y. Chiu, "Effective heuristic procedures for a field technician scheduling problem," *Journal of Heuristics*, vol. 7, no. 5, pp. 495–509, 2001.

[7] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[8] A. Crispin and A. Syrichas, "Quantum annealing algorithm for vehicle scheduling," in *2013 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2013, pp. 3523–3528.

[9] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

[10] R. Fukasawa, H. Longo, J. Lysgaard, M. P. de Aragão, M. Reis, E. Uchoa, and R. F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, vol. 106, no. 3, pp. 491–511, 2006.

[11] J.-F. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30, no. 2, pp. 105–119, 1997.

[12] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Operations research*, vol. 35, no. 2, pp. 254–265, 1987.

[13] E. Cheng and J. L. Rich, "A home health care routing and scheduling problem," *URL http://citeseerx. ist. psu. edu/viewdoc/summary*, 1998.

[14] S. Bertels and T. Fahle, "A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem," *Computers & Operations Research*, vol. 33, no. 10, pp. 2866–2890, 2006.

[15] M. S. Rasmussen, T. Justesen, A. Dohn, and J. Larsen, "The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies," *European Journal of Operational Research*, vol. 219, no. 3, pp. 598–610, 2012.

[16] G. Hiermann, M. Prandtstetter, A. Rendl, J. Puchinger, and G. R. Raidl, "Metaheuristics for solving a multimodal home-healthcare scheduling problem," *Central European Journal of Operations Research*, vol. 23, no. 1, pp. 89–113, 2015.

[17] P. Brucker, R. Qu, and E. Burke, "Personnel scheduling: Models and complexity," *European Journal of Operational Research*, vol. 210, no. 3, pp. 467–473, 2011.

[18] J. A. Castillo-Salazar, D. Landa-Silva, and R. Qu, "A survey on workforce scheduling and routing problems," in *Proceedings of the 9th international conference on the practice and theory of automated timetabling*. Citeseer, 2012, pp. 283–302.

[19] D. C. Paraskevopoulos, G. Laporte, P. P. Repoussis, and C. D. Tarantilis, "Resource constrained routing and scheduling: Review and research prospects," 2016.

[20] V. Pillac, C. Gueret, and A. L. Medaglia, "A parallel matheuristic for the technician routing and scheduling problem," *Optimization Letters*, vol. 7, no. 7, pp. 1525–1535, 2013.

[21] I. Mathlouthi, M. Gendreau, and J.-Y. Potvin, "Mixed integer programming for a multi-attribute technician routing and scheduling problem," 2016.

[22] X. Chen, B. W. Thomas, and M. Hewitt, "The technician routing problem with experience-based service times," *Omega*, vol. 61, pp. 49–61, 2016.

[23] A. A. Kovacs, S. N. Parragh, K. F. Doerner, and R. F. Hartl, "Adaptive large neighborhood search for service technician routing and scheduling problems," *Journal of scheduling*, vol. 15, no. 5, pp. 579–600, 2012.

[24] F. Tricoire, N. Bostel, P. Dejax, and P. Guez, "Exact and hybrid methods for the multiperiod field service routing problem," *Central European Journal of Operations Research*, vol. 21, no. 2, pp. 359–377, 2013.

[25] E. Zamorano and R. Stolletz, "Branch-and-price approaches for the multiperiod technician routing and scheduling problem," *European Journal of Operational Research*, 2016.

[26] F. O. R. Society. (2016) What is the roadef 2007 challenge. [Online]. Available: http://challenge.roadef.org/2007/en/

[27] P.-F. Dutot, A. Laugier, and A.-M. Bustos, "Technicians and interventions scheduling for telecommunications," *France Telecom R&D*, 2006.

[28] M. Fırat and C. Hurkens, "An improved mip-based approach for a multi-skill workforce scheduling problem," *Journal of Scheduling*, vol. 15, no. 3, pp. 363–380, 2012.

[29] P. Korteweg, "When to hire the a-team," 2007.

[30] Y. Park, Y. Khosiawan, I. Moon, M. N. Janardhanan, and I. Nielsen, "Scheduling system for multiple unmanned aerial vehicles in indoor environments using the csp approach," in *Intelligent Decision Technologies 2016*. Springer, 2016, pp. 77–87.

[31] B. Estellon, F. Gardi, and K. Nouioua, "High-performance local search for task scheduling with human resource allocation," in *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*. Springer, 2009, pp. 1–15.

[32] V. Pillac, C. Guéret, and A. Medaglia, "On the dynamic technician routing and scheduling problem," 2012.

[33] C. A. Hurkens, "Incorporating the strength of mip modeling in schedule construction," *RAIRO-Operations Research*, vol. 43, no. 04, pp. 409–420, 2009.

[34] J.-F. Cordeau, G. Laporte, F. Pasin, and S. Ropke, "Scheduling technicians and tasks in a telecommunications company," *Journal of Scheduling*, vol. 13, no. 4, pp. 393–409, 2010.

[35] H. Hashimoto, S. Boussier, M. Vasquez, and C. Wilbaut, "A grasp-based approach for technicians and interventions scheduling for telecommunications," *Annals of Operations Research*, vol. 183, no. 1, pp. 143–161, 2011.

[36] E. Tsang and C. Voudouris, "Fast local search and guided local search and their application to british telecom's workforce scheduling problem," *Operations Research Letters*, vol. 20, no. 3, pp. 119–127, 1997.

[37] A. Khalfay, A. Crispin, and K. Crockett, "Solving technician and task scheduling problems with an intelligent decision heuristic," in *Intelligent Decision Technologies 2016*. Springer, 2016, pp. 63–75.

[38] C. E. Cortés, M. Gendreau, L. M. Rousseau, S. Souyris, and A. Weintraub, "Branch-and-price and constraint programming for solving a real-life technician dispatching problem," *European Journal of Operational Research*, vol. 238, no. 1, pp. 300–312, 2014.

[39] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," in *International Conference on Principles and Practice of Constraint Programming*. Springer, 1998, pp. 417–431.

[40] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search," in *Handbook of metaheuristics*. Springer, 2003, pp. 320–353.

[41] R. Martí, J. M. Moreno-Vega, and A. Duarte, "Advanced multi-start methods," in *Handbook of metaheuristics*. Springer, 2010, pp. 265–281.

[42] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi *et al.*, "Optimization by simmulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

Appendix D

Applying the Intelligent Decision Heuristic to Solve Large Scale Technician and Task Scheduling Problems

( )

Department of Computing, Mathematics and Digital Technology,
Manchester Metropolitan University, Manchester, UK
{a.khalfay,a.crispin,k.crockett}@mmu.ac.uk

Abstract. Scheduling personnel to complete tasks is a complex combinatorial optimisation problem. In large organisations, finding quality solutions is of paramount importance due to the costs associated with staffing. In this paper we have generated and solved a set of novel large scale technician and task scheduling problems. The datasets include complexities such as priority levels, precedence constraints, skill requirements, teaming and outsourcing. The problems are considerably larger than those featured previously in the literature and are more representative of industrial scale problems, with up to 2500 jobs. We present our data generator and apply two heuristics, the intelligent decision heuristic and greedy heuristic, to provide a comparative analysis.

Keywords: Combinatorial optimisation · Large scale technician and task scheduling problems · Data generator · Intelligent decision heuristic

1 Introduction

[ ]

[ ]

[1] [ ]

[1]

[11 1 1] 00

[10]

00 [ ]

[1]

00 00

[1] [1]

( )

100

[11] [1]

00

100

[1]

(00

)

10

00

00

00

(1000 +)

(

00)

00

2 ROADEF 2007 Challenge Mathematical Formulation

00

$= [1...\ ]$ i

N p $p \in [1...\ ]$

d_i $s^i_{\delta\alpha}$ (δ

) c_i $_i$

$= [1...m]$ $T = [1...t]$

(1)

e_p

$_p = [\ \ , 1\ ,\ \ , 1]$ $p = [1,\ \ ,\ \ ,\ \ ]$

$$inimi\ \ e \sum_{p=1}^{4} \ _p * e_p \tag{1}$$

$_i$ ( )

$p \in [1...\ ]$

$$e_p \geq \ _i + d_i \quad \forall p \in 1,\ \ ,\ \ , i \in N_p \quad (\ )$$

( ) e_4

$$e_4 \geq \ _i + d_i \quad \forall, i \in N \quad (\ )$$

$x_{t,k,m} = 1$ t m ( )

T_k

$$\sum_{m \in M} x_{t,k,m} \leq 1 \quad \forall\ \ \in\ \ , t \in T_k \quad (\ )$$

( )

T_k

$$x_{t,k,m} = 0 \quad \forall \quad \in \quad , t \notin T_k \qquad ( )$$
$$m \in M$$

$y_{i,k,m} = 1$ i m ( ) N $_i = 1$

$$_i + \qquad y_{i,k,m} = 1 \quad \forall i \in N \qquad ( )$$
$$k \in K \; m \in M$$

( ) $y_{i,k,m} = 1$

$$y_{i,k,m} * s^i_{\delta\alpha} \leq \qquad {}^t_{\delta\alpha} * x_{t,k,m} \quad \forall i \in N, \quad \in \quad , m \in \quad , \quad \in \quad , \delta \in \qquad ( )$$
$$t \in T_k$$

( )

i' i $_i$ i' i

$$_i + d_i \leq \; {}'_i \quad \forall i \in N, i' \in \; _i \qquad ( )$$

( ) (10) ( )

(10)

$$1 \;\; 0(\quad - 1) * \qquad y_{i,k,m} \leq \; _i \quad \forall i \in N, \quad \in \qquad ( )$$
$$m \in M$$
$$1 \;\; 0(\;) * \qquad y_{i,k,m} \geq \; _i + d_i \quad \forall i \in N, \quad \in \qquad (10)$$
$$m \in M$$

$u_{i,i'} = 1$ i i'

i' i (11)

i

i'

$$_i + d_i - \quad (1 - u_{i,i'}) \leq \; {}'_i \quad \forall i, i' \in N, i \neq i' \qquad (11)$$

(1)

$$y_{i,k,m} + y_{i',k,m} - u_{i,i'} - u_{i',i} \leq 1 \quad \forall i, i' \in N i \neq i', \quad \in \quad , m \in \qquad (1 \;)$$

00

$_i = 1$ i (1)

$$_i * c_i \leq \quad \forall i \in N \tag{1 }$$

(1)

$$| \ _i| * \ _i \leq \sum_{i \in \sigma_i} \ '_i \quad \forall i \in N^\sigma \tag{1 }$$

(1) (1) $x_{t,k,m}$ $y_{i,k,m}$ $u_{i,i'}$ $_i$

$$x_{t,k,m} = [0,1] \quad \forall \ \in \ , m \in \ , t \in T \tag{1 }$$

$$y_{i,k,m} = [0,1] \quad \forall \ \in \ , m \in \ , i \in N \tag{1 }$$

$$u_{i,i'} = [0,1] \quad \forall i, i' \in N, i \neq i' \tag{1 }$$

$$_i = [0,1] \quad \forall i \in N \tag{1 }$$

(1) (0)

$$e_p \geq 0 \quad \forall i \in N_p \tag{1 }$$

$$_i \geq 0 \quad \forall i \in N \tag{ 0}$$

3 Large Scale Technician and Task Scheduling Problem Instances

1

[1]
[11] [1]

( )
( ) ( )
(
)
1000 00
1 10 1

Table 1. Large scale technician and task scheduling problem instances

Dataset	Jobs	Techs	Budget	Domains	Levels
L1	1000	25	500	3	3
L2	1000	50	500	3	3
L3	1000	100	500	3	3
L4	1500	25	1000	4	4
L5	1500	50	1000	4	4
L6	1500	100	1000	4	4
L7	2000	25	1500	3	3
L8	2000	50	1500	3	3
L9	2000	100	1500	3	3
L10	2500	25	2000	4	4
L11	2500	50	2000	4	4
L12	2500	100	2000	4	4

3.1 Generating Large Scale Instances

Generating Job Durations. 00
1 0
1 1 0 1
1
1 0

Generating Skill Domain Requirements.
0
0

Generating Precedence and Successor Relationships.

00

p p

( ) ( )

1 1

1

(1)

( )

( )

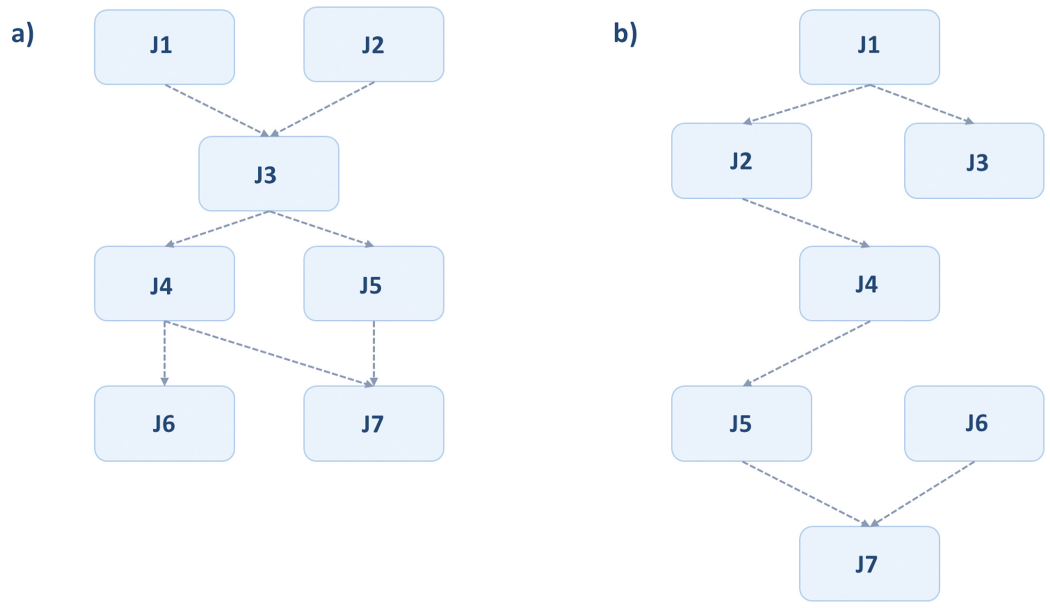


Fig. 1. Example of the dynamic precedence and successor relationship trees

1

(

) 00

4 Heuristic Approaches

4.1 Intelligent Decision Heuristic

[10]

4.2 Greedy Heuristic

5 Experimental Results

00

0

0

10 00 1

( )

6 Discussion

Table 2. Experimental results for the large scale problem instances

Dataset	ID	Greedy	% Gap
L1	192810	203850	5.7
L2	97725	103440	5.8
L3	48330	50700	4.9
L4	296940	315210	6.2
L5	147480	156960	6.4
L6	76110	80880	6.3
L7	420660	445335	5.8
L8	198900	207405	4.3
L9	97080	102870	6
L10	574465	607890	5.8
L11	280260	290745	3.7
L12	140970	144840	2.7
Average	214311	225844	5.3

7 Conclusion

1

( )

( )

( )

Acknowledgements. This research is sponsored by ServicePower Technologies PLC, a worldwide leader in providing innovative mobile workforce management solutions, in cooperation with Manchester Metropolitan University and KTP.

References

1. Burke, E., De Causmaecker, P., Berghe, G.V.: A hybrid tabu search algorithm for the nurse rostering problem. In: Asia-Pacific Conference on Simulated Evolution and Learning, pp. 187–194. Springer (1998)
2. Cordeau, J.F., Laporte, G., Pasin, F., Ropke, S.: Scheduling technicians and tasks in a telecommunications company. J. Sched. **13**(4), 393–409 (2010)
3. Crispin, A., Syrichas, A.: Quantum annealing algorithm for vehicle scheduling. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp. 3523–3528. IEEE (2013)
4. Dutot, P.F., Laugier, A., Bustos, A.M.: Technicians and Interventions Scheduling for Telecommunications. France Telecom R&D, Lannion (2006)
5. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: a review of applications, methods and models. Eur. J. Oper. Res. **153**(1), 3–27 (2004)
6. Estellon, B., Gardi, F., Nouioua, K.: High-performance local search for task scheduling with human resource allocation. In: Engineering Stochastic Local Search Algorithms, Designing, Implementing and Analyzing Effective Heuristics, pp. 1–15. Springer (2009)
7. Hashimoto, H., Boussier, S., Vasquez, M., Wilbaut, C.: A grasp-based approach for technicians and interventions scheduling for telecommunications. Ann. Oper. Res. **183**(1), 143–161 (2011)
8. Haugen, D.L., Hill, A.V.: Scheduling to improve field service quality*. Decis. Sci. **30**(3), 783–804 (1999)
9. Hurkens, C.A.: Incorporating the strength of MIP modeling in schedule construction. RAIRO-Oper. Res. **43**(04), 409–420 (2009)
10. Khalfay, A., Crispin, A., Crockett, K.: Solving technician and task scheduling problems with an intelligent decision heuristic. In: Intelligent Decision Technologies 2016, pp. 63–75. Springer (2016)
11. Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R.F.: Adaptive large neighborhood search for service technician routing and scheduling problems. J. Sched. **15**(5), 579–600 (2012)
12. Krishnamoorthy, M., Ernst, A.T., Baatar, D.: Algorithms for large scale shift minimisation personnel task scheduling problems. Eur. J. Oper. Research **219**(1), 34–48 (2012)
13. Pillac, V., Guéret, C., Medaglia, A.: On the dynamic technician routing and scheduling problem. In: Proceedings of the 5th International Workshop on Freight Transportation and Logistics (ODYSSEUS) (2012)
14. Pillac, V., Gueret, C., Medaglia, A.L.: A parallel matheuristic for the technician routing and scheduling problem. Optim. Lett. **7**(7), 1525–1535 (2013)

15. Society, F.O.R.: What is the roadef 2007 challenge (2016). http://challenge.roadef.org/2007/en/
16. Solomon, M.M.: Algorithms for the vehicle routing and scheduling problems with time window constraints. Oper. Res. **35**(2), 254–265 (1987)
17. Titiloye, O., Crispin, A.: Quantum annealing of the graph coloring problem. Discrete Optim. **8**(2), 376–384 (2011)

Appendix E

Operational Research manuscript No.
(will be inserted by the editor)

Solving the service technician routing and scheduling problem with time windows

Author 1 · Author 2 · Author 3

Received: April 2017 / Accepted: date

Abstract In this paper, we present the first comparative performance analysis on the service technician routing and scheduling problem with time windows. Time window constraints occur in many sectors such as telecommunications, maintenance, call centres, warehouses and healthcare, and is a way of service providers differentiating themselves from each other in a bid to maintain customer satisfaction and ultimately retain market share. We present the first sequential heuristic, the greedy randomized heuristic, to be tested on the service technician routing and scheduling problem with time windows datasets, coupled with a simulated annealing with restart metaheuristic. Our sequential heuristic is tested on 72 known service technician routing and scheduling problem with time windows instances, and compared against a parallel adaptive large neighbourhood search heuristic, presenting new best known results in 18% of the datasets.

Keywords technician routing and scheduling problem · time windows · sequential heuirstic · greedy randomized heuristic · simulated annealing with restart

1 Introduction

The service technician routing and scheduling problem with time windows (STR-SPTW) is an NP-hard combinatorial optimisation problem, meaning that as the problem size increases, exact methods become prohibitive, and, therefore, approximate techniques must be used in order to find feasible and good quality solutions within reasonable computational times (Blum and Roli, 2003). This research is based on a combinatorial optimisation problem applicable to many industries such as the maintenance sector (Fırat and Hurkens (2012) and Pillac et al. (2013)), call centres (Van den Bergh et al. (2013)) and home healthcare (Hiermann et al. (2015) and Paraskevopoulos et al. (2016)).

Address(es) of author(s) should be given

The cost of a workforce is usually the most significant cost a business or organisation incurs. In the context of STRSPTW, it is not solely the cost of the employees, but also the maintenance and repair of the fleet of vehicles that must be considered. Efficient scheduling and routing of employees and vehicles can reduce the cost of a workforce and ensure a balanced workload, and has the potential to reduce the environmental impacts caused by the vehicles used.

The occurrence of time windows is becoming increasingly popular with service maintenance providers, and directly affects the scheduling and routing of employees. From a customer's perspective, knowing that a technician/skilled worker will be arriving between time a_i and b_i can improve his/her customer experience. It may allow the customer waiting for a service to take less time off work, and even choose his/her preferred time slot, providing not only convenience but satisfaction.

In this paper, we propose the first sequential heuristic to solve the STRSPTW data instances that were first proposed by Kovacs et al. (2012). These problem instances were adapted from the vehicle routing instances proposed by Solomon (1987), by combining the datasets with skill domain information taken from the ROADEF 2007 challenge (Society, 2016). The research presented in this paper studies a range of STRSPTW datasets, each with 100 customers, a varying crew size, and different proportions of jobs with time windows. The STRSPTW datasets have only been solved by Kovacs et al. (2012) who used a parallel adaptive large neighbourhood search (pALNS) algorithm. We propose a sequential greedy randomized heuristic on these problem instances to test performance and provide a comparative analysis.

This problem requires tours to be designed for teams, such that all jobs are served or outsourced. The objective of the problem is to minimize the sum of the routing and outsourcing costs. Each team is made up of one or more technicians, who each have intrinsic skills and levels of competency within each skill area. Each team leaves a central depot at the beginning of the working day, travels to service customers, and returns to the depot before the end of the working day. Each job requires a set of skills that must be satisfied by the team who serves the job. Each job also has a time window $[a_i, b_i]$, and the beginning of service B_i must lie within the time window such that $a_i \leq B_i \leq b_i$. This problem is a single period problem as the solution consists of a single working day. Lastly, there is also the option to outsource some jobs (if they are unable to be allocated), which incurs a penalty cost.

Our approach, a greedy randomized heuristic, proposed in this paper has some similarities with the well known greedy randomized adaptive search procedure (GRASP) metaheuristic. The greedy randomized heuristic, like GRASP, uses multiple scoring criteria to decide which job should be allocated next, and includes a degree of randomness in order to avoid deterministic results. In contrast, our heuristic does not generate multiple initial solutions. Instead, the greedy randomized heuristic generates a single initial solution and the rest of the computational time is spent trying to iteratively improve it. During the improvement phase, the chance of selecting a local operator is uniform. We have used a greedy randomized heuristic as opposed to a GRASP heuristic due to the short run times that were used in the initial experiments by Kovacs et al. (2012) as we believe multi start heuristic procedures are suited better to longer run times.

The rest of the paper is organised as follows: section 2 provides a literature review of the research already undertaken in this area of personnel scheduling, section 3 presents the mathematical formulation of the STRSPTW, section 4 describes our heuristic approach, section 5 shows the results of the computational experiments performed, section 6 discusses the results obtained and section 7 concludes on the research undertaken.

2 Literature

Research in the area of personnel scheduling has included many problem definitions; the technician routing and scheduling problem (TRSP), the technician and task scheduling problem (TTSP), the service technician routing and scheduling problem (STRSP), the field technician scheduling problem (FTSP) and the STRSPTW. Each problem has included a diverse range of constraints such as routing, teaming, priority levels, precedence, time windows, multi-period (multiple days), tools and spare parts, and experienced based service times. However, the main characteristic featured in all of these problems is skill complexity. All of these problems require jobs to be completed by a technician/team who collectively posses the necessary expertise to perform the required job.

Many approaches have been applied to solve these types of personnel scheduling problems including both exact and approximate techniques. Approximate techniques such as adaptive large neighbourhood search (Cordeau et al., 2010; Kovacs et al., 2012), local search (Estellon et al., 2009), greedy randomized adaptive search (Hashimoto et al., 2011) and intelligent decision heuristics (Author1 et al., 2016) have been applied to these types of problem. Exact techniques such as mixed integer programming (Hurkens, 2009; Fırat and Hurkens, 2012; Mathlouthi et al., 2016) and branch and price (Zamorano and Stolletz, 2016) have also been used.

An FTSP was studied by Xu and Chiu (2001), which included the complexity of routing, overtime, multiple depots and skill requirements. However, skill requirements were not treated as hard constraints in this problem, and unlike most scheduling problems, the objective was not to serve all jobs in the least costly manner, but to serve as many jobs as possible. This research used data instances with up to 1000 jobs.

A TTSP was the basis of the ROADEF 2007 challenge. The ROADEF challenge is a bi-annual competition proposed by the French Operational Research Society, that invites researchers to compete to find efficient ways of solving combinatorial optimisation problems. In 2007, the problem was a TTSP, and used real world datasets provided by France Telecom, containing data instances ranging from 5 to 800 jobs and 5 to 150 technicians. The aim of the problem is to allocate a set of jobs over a scheduling horizon to a set of teams. Teams are made up of technicians, each with intrinsic skill domain levels and days within the scheduling horizon when they are not available. Each job has a priority level indicating how important it is to serve the job as early as possible. In some problem instances there is an available outsourcing budget that can be utilized. Jobs can also have relationships with other jobs, which can be precedence or successor relationships, where a job may not begin until another has been completed. Jobs have skill requirements which must be satisfied by the team which serves the job. Many approaches such as mixed integer programming (Hurkens, 2009; Fırat and Hurkens, 2012), adaptive large neighbourhood search (Cordeau et al., 2010), local search (Estellon et al., 2009), greedy randomized adaptive search (Hashimoto et al., 2011) and intelligent decision heuristics (Author1 et al., 2016) have been applied to this problem.

A study by Pillac et al. (2012) concentrated on the dynamic technician routing and scheduling problem (DTRSP), in which new job requests appear, an aspect of a real world situation faced by industry. In addition a static TRSP was studied by Pillac et al. (2013), who extended instances from vehicle routing problems proposed by Solomon (1987) by combining them with randomly generated skill requirements and tools and spare parts information. This TRSP used a crew of up to 25 technicians and scheduled up to 100 jobs and required the scheduling of crew over a single day. This research included the complexity of tools and spare parts constraints, an important aspect in the service maintenance sector.

An exact approach was studied by Tricoire et al. (2013) who compared the performance of exact and hybrid solution approaches, concluding the trade off between computational time and solution quality. In addition, Chen et al. (2016) studied a version of the TRSP where the technicians became more experienced throughout the scheduling horizon resulting in a reduction of service times, an aspect of the real world scenario previously unstudied.

Other exact approaches in the literature include papers by Mathlouthi et al. (2016) and Zamorano and Stolletz (2016) who used mixed integer programming (MIP) and branch and price solution approaches respectively. Mathlouthi et al. (2016) used artificial datasets that contained up to 25 jobs and used CPLEX to solve the mixed integer programming model. The problem included the complexities such as skill requirements, priority levels, time windows, breaks and overtime. This paper also illustrated how the computational time needed to solve the problems rapidly increases with problem size and complexity. Zamorano and Stolletz (2016) used both artificial and real world datasets containing up to 29 jobs, again emphasizing the difficulties faced with the scalability of MIP solution approaches.

Lastly, Author1 et al. (2017a) proposed some large scale technician and task scheduling problems, created under the framework of the ROADEF 2007 challenge, that required the use of heuristic approaches. The instances included up to 2500 jobs to schedule over a multi day scheduling horizon. This research compared the performance of an intelligent decision heuristic (Author1 et al., 2016) and a greedy heuristic.

Reviews in the field of personnel scheduling problems have been undertaken by Ernst et al. (2004), Castillo-Salazar et al. (2012) and Author1 et al. (2017b) which outline the many complexities and constraints that are included in the problems such as routing, teaming, skill requirements, priority levels, precedence and successor relationships, unavailability of resources, scheduling over multiple days, tools and spare parts and time windows.

Throughout the literature it is clear that research is needed into approximate approaches in order to tackle medium and large scale problems as exact methods are prohibitive. Furthermore, as the number of machines grow, so do the number of technicians needed to perform skilled jobs, and so does the need for scalable and robust heuristic solution approaches. Our research focuses on designing and implementing approximate approaches that can deal with relevant real world constraints such as time windows. As evidenced, time windows are an important consideration for service providers who seek to maintain customer satisfaction and maintain repeat business through providing a reliable and customer focused service.

3 Problem formulation

In this work, we solve a set of STRSPTW datasets. We present the Mixed Integer Programming (MIP) formulation, as introduced by Kovacs et al. (2012) but do not solve it using MIP as the problems are too large and complex to solve within reasonable computational time. The problem can be defined mathematically as a complete directed graph $G = \{V, A\}$, where V is the set of all vertices i.e the set of jobs, and A a set of arcs between the vertices. The set of jobs that is allocated can be defined as V', and jobs that belong to V but not V' is the set of jobs that is outsourced.

There is a set of technicians $T = \{1, \ldots, t\}$ and a set of teams $\tau \subset \mathscr{T}$, made up of technicians. We denote the starting depot as 0 and the ending depot as N. Each technician has intrinsic skills $s \in S$ and varying levels $l \in L$ within each area of expertise. The fleet of technicians is heterogeneous, each being unique with different skills and levels within each skill area. The technician's skills can be represented by an $L \times S$ matrix where $[p^t_{l,s}]$ denotes the level of expertise the technician has in skill area s to level l. Skill levels are hierarchical so if $p^t_{l,s} = 1$ then $p^t_{l',s} = 1$ for $l' < l$.

Each job belonging to the set V has a service time denoted by d_i, a skill requirement matrix, of size $L \times S$, denoted as $q_{l,s}$ and a time window in which service of the job must begin $[a_i, b_i]$. The aim of the problem is to construct the least costly schedule, by minimizing the total sum of the outsourcing and routing costs. We use the following variables;

B^τ_i = the beginning time of service of job i or the depot by team τ
E^τ_i = the end time of service of job i or the depot by team τ
$x^\tau_{i,j}$ that equals one only if team τ travels from job i to job j
z_i equals one only if job i has been outsourced
y^τ_i equals one if job i is assigned to team τ
v^τ_t equals one if technician t is assigned to team τ

The problem can now be represented as;

$$\min \sum_{t \in \tau} \sum_{i,j \in A} c_{i,j} \cdot x^\tau_{i,j} + \sum_{i \in V'} o_i \cdot z_i \tag{1}$$

Equation (1) shows the objective function of the problem, which is to minimise the total sum of the routing costs ($c_{i,j}$ is the distance between customers i and j) and the outsourcing costs (where o_i is the cost of outsourcing job i). Subject to;

$$\sum_{\tau \in \mathscr{T}} v^\tau_t \leq 1 \quad \forall t \in T \tag{2}$$

Equation (2) ensures that each technician may belong to one team only.

$$\sum_{\tau \in \mathscr{T}} y_i^\tau + z_i = 1 \quad \forall i \in V \tag{3}$$

Equation (3) guarantees that each job is either outsourced or it is allocated to a single team.

$$\sum_{j \in V' \cup \{N\}} x_{0,j}^\tau = 1 \quad \forall \tau \in \mathscr{T} \tag{4}$$

$$\sum_{i \in V' \cup \{N\}} x_{i,N}^\tau = 1 \quad \forall \tau \in \mathscr{T} \tag{5}$$

Equations (4 and 5) ensure each team departs from the central depot at the beginning of the working day and returns to the depot at the end of the working day.

$$\sum_{j \in V' \cup \{0\}} x_{j,i}^\tau = y_i^\tau \quad \forall i \in V', \tau \in \mathscr{T} \tag{6}$$

$$\sum_{j \in V' \cup \{0\}} x_{j,i}^\tau - \sum_{j \in V'} x_{i,j}^\tau = 0 \quad \forall i \in V', \tau \in \mathscr{T} \tag{7}$$

Equations (6 and 7) confirm that if a team is assigned to a job *i*, then the team must travel to the job from another location, and leave the job to travel to another location.

$$B_j^\tau \geq (E_i^\tau + c_{i,j}) * x_{i,j,}^\tau \quad \forall i \in V', \tau \in \mathscr{T} \tag{8}$$

Equation (8) states that if two jobs *i* and *j* happen sequentially, then the start time of *j* must be equal to or greater than the end time of *i* plus that travel time between *i* and *j*, to ensure continuity.

$$B_0^\tau = 0 \quad \forall \tau \in \mathscr{T} \tag{9}$$

Equation (9) sets the beginning time of each team's route.

$$y_i^\tau \cdot q_{l,s}^i \leq \sum_{t \in T_k} p_{l,s}^t \cdot v_t^\tau \quad \forall i \in V', \quad l \in L, \quad s \in S \tag{10}$$

Equation (10) ensures that if a job is allocated to a team, then the technicians who make up the team collectively have the skills necessary to service the job.

$$E_i^\tau = (B_i^\tau + d_i) \cdot y_i^\tau \quad \forall i \in V', \tau \in \mathscr{T} \tag{11}$$

Equation (11) states that the end service time of a job must be equal to the beginning service time plus the service time of the job.

$$a_i \leq B_i^\tau \leq b_i \quad \forall i \in V' \cup \{N\}, \tau \in \mathscr{T} \tag{12}$$

Equation (12) guarantees that the beginning of service of a job *i* is within the time window.

$$\sum_{t \in T} v_t^\tau = 1 \quad \forall \tau \in \mathscr{T} \tag{13}$$

Lastly, in this work we are tackling the instances that contain no team building, i.e each team contains, at most, one technician. This can be modelled as equation (13). We use the following variables;

$$x^{\tau}_{i,j,} \in \{0,1\} \quad \forall (i,j) \in A, \tau \in \mathscr{T} \tag{14}$$

$$z_i \in \{0,1\} \quad \forall i \in V' \tag{15}$$

$$y^{\tau}_i \in \{0,1\} \quad \forall i \in V', \tau \in \mathscr{T} \tag{16}$$

$$v^{\tau}_t \in \{0,1\} \quad \forall t \in T, \tau \in \mathscr{T} \tag{17}$$

$$B^{\tau}_i \geq 0 \quad \forall i \in V' \tag{18}$$

$$E^{\tau}_i \geq 0 \quad \forall i \in V' \tag{19}$$

4 Heuristic approach

Our solution approach comprises of two parts, generating an initial feasible solution, and secondly, iteratively trying to improve the current solution through the use of local operators and evaluating using a simulated annealing with restart metaheuristic.

4.1 Greedy randomized construction heuristic

The greedy randomized heuristic behaves in a flexible manner by changing the sorting criteria that decide which job is next to be allocated. There are five insertion methods; earliest late window, minimum window size, complex jobs, depot distance and random. Each of these methods is described in the following subsections. The pseudo code for the greedy randomized construction heuristic is displayed in Figure 1.

– **Earliest late window** This insertion method orders the set of unallocated jobs into a list. Each job has a time window $[a_i, b_i]$, and the sorting method orders the jobs into an increasing order of b_i, i.e the end of the time window, the latest time the job can be started. This method tries to ensure that all jobs are allocated before their time window has passed, as they will then be outsourced in order to stay within the feasible solution space, which incurs a cost.

$$earlylate_i = b_i \tag{20}$$

- **Minimum window size** The minimum window size method orders the set of unallocated jobs into a list, sorting them by the size of their time window. This method aims to ensure that jobs that have a small opportunity to be started, i.e the difference between b_i and a_i is small, have a higher chance of being allocated in favour of jobs with a larger difference between b_i and a_i.

$$minwindow_i = b_i - a_i \tag{21}$$

- **Complex jobs** The set of unallocated jobs is ordered by the complexity of the jobs that are currently unallocated. The difficulty of a job is calculated as the sum of the total skill requirements across each domain and skill level, as in Cordeau et al. (2010). This method aims to allocate jobs which require lots of skill earlier, and jobs that are less difficult to schedule are scheduled later.

$$complex_i = \sum_{l \in L} \sum_{s \in S} q^i_{l,s} \tag{22}$$

- **Depot distance** The depot distance method orders the set of jobs in ascending order of distance away from the depot. The distance between a job i located at x_i, y_i, and the depot located at x_0, y_0 is calculated using the Euclidean distance as shown in Equation (23).

$$depotdistance_i = \sqrt{(x_0 - x_i)^2 + (y_0 - y_i)^2} \tag{23}$$

- **Random** The random sorting method orders the unallocated jobs by shuffling the array that contains the jobs. In this work, we have set the level of randomness, r, to 0.08. The tuning experiments for the level of randomness is explained in Section 5.1.

4.1.1 Greedy randomized heuristic pseudo code

Figure 1 shows the greedy randomized construction heuristic. This algorithm takes the following variables, a set of jobs V, a set of teams τ, a schedule S, an outsourcing list O, the following sorting methods ELW: earliest late window, MWS: minimum window size, CJ: complex jobs, DD: depot distance, and R: Randomly, i the job selected for allocation, and τ_i the team selected to be allocated job i.

First, an empty schedule S is initialised. While jobs can be allocated to the schedule S, a random number r is generated on the interval $\{0,1\}$. Dependent on the value of r, a sorting method is chosen; ELW earliest late window, MWS minimum window size, CJ complex jobs, DD depot distance or R randomly. On line 5, the set of remaining unallocated jobs V is sorted by the chosen sorting method, then a job i is selected belonging to V. Job i is then assigned to a team (if one is available in terms of time windows and skill requirements) and removed from the set of unallocated jobs. The while loop is iterated through until no more job allocations can be made to the teams. On line 11, any remaining unallocated jobs are added to the outsource list O. Lastly, on line 12, the initial solution S is output.

The cost of outsourcing a job is shown in equation (24). Note, it is always less costly to schedule a job, if there are technicians available rather than to outsource it.

Variables: V: set of jobs, τ: the set of teams , S: the schedule, O: outsource list, ELW: earliest late window, MWS: minimum window size, CJ: complex jobs, DD: depot distance, R: Randomly, i:job selected for allocation, τ_i: team selected to serve job i

```
1: $initialise \quad S$
2: while jobs can be allcoated do
3:     $r \leftarrow random(0,1)$
4:     $technique \leftarrow Choosesortingmethod(r, ELW, MWS, CJ, DD, R)$
5:     $V \leftarrow method(V)$
6:     $i \leftarrow selectjob(V)$
7:     $\tau_i \leftarrow selectteam(i,S)$
8:     $assign(S, \tau_i, i)$
9:     $remove(V,i)$
10: end while
11: $O \leftarrow addOutsourced(V)$
12: return $S$
```

Fig. 1: Greedy randomized construction heuristic

The cost of outsourcing is equal to 200 plus the sum of the domain skill requirements of the job.

$$o_i = 200 + \sum_{l \in L} \sum_{s \in S} q^i_{l,s} \tag{24}$$

4.2 Local operators

In this work we have used a variety of local operators. A local operator is used to perturb the current solution generating a neighbouring solution that can be evaluated using a metaheuristic. Some of the local operators used were featured in our previous work, Khalfay et al. (2016), and some from other work by Cordeau et al. (2010) and Kovacs et al. (2012) which were based on the local operators proposed in Shaw (1998) for the VRP. In this work, we only search through feasible solution space. This means that when an operator is applied, skill compatibility, time window constraints and route length are checked to make sure we remain within feasible solution space.

– **Swap two jobs** This operator randomly selects two jobs $j1$ and $j2$ belonging to different teams $\tau1$ and $\tau2$ respectively and tries to swap them over such that job $j2$ is now allocated to team $\tau1$ and job $j1$ is now allocated to team $\tau2$.
– **Swap order** This operator randomly selects two jobs $j1$ and $j2$ belonging to the same team τ, with positions $p1$ and $p2$ respectively. The operator then reassigns the jobs such that $j2$ is now positioned at $p1$ and $j1$ is positioned at $p2$ in team τ's route.
– **Move a job** This operator randomly selects a job $j1$ belonging to team $\tau1$ and removes it from its current position. The heuristic is then used to reallocate this job back into the schedule. If the job cannot be reallocated it is added into the outsource list.

- **Remove a team** This operator randomly selects a team τ and removes all of the jobs that are currently assigned to the team, placing the jobs in the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule, since it is always less costly to allocate a job than to outsource it.
- **Remove** N **jobs** This operator selects a number N which defines how many jobs are to be removed from the schedule. The heuristic then randomly selects a job and removes from the schedule until N jobs have been removed. The removed jobs are added to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.
- **Remove** N **teams** This operator randomly selects N teams belonging to the schedule and removes all of the jobs that were assigned to the teams, adding each job to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.
- **Remove related jobs** This operator randomly selects a number N, which represents how many jobs that will be removed. Next, a single job is selected at random, job i, and removed. The remaining jobs are then ranked in terms of how similar they are to the removed job in terms of skill requirements as in Cordeau et al. (2010). The highest scoring job is selected and removed until N jobs have been removed, and added to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.

$$rel_{i,i'} = \sum_{l \in L} \sum_{s \in S} |q^{i}_{l,s} - q^{i'}_{l,s}| \tag{25}$$

- **Remove close jobs** This operator randomly selects a number N, which defines how many jobs will be removed. Next, a single job is selected at random, job i, and removed. The remaining jobs are then ranked in terms of how similar they are to the removed job in terms of geographical location. The highest scoring job is selected and removed until N jobs have been removed, and added to the outsourced list. The heuristic is then used to try to allocate the jobs in the outsourced list back to the schedule.

$$close_{i,i'} = \sqrt{(x_i - x_{i'})^2 + (y_i - y_{i'})^2} \tag{26}$$

- **Remove chains** This operator iterates through each team belonging to the schedule. If the size of the team's route is greater than two, then a portion of the route is removed and each job is added into the outsourced list. The heuristic is then used to try and allocate the jobs from the outsourced list back into the schedule.

4.3 Simulated annealing with restart

In this work we have implemented a simulated annealing metaheuristic with a restart mechanism. Simulated annealing was chosen due to its success in other types of combinatorial optimisation problems i.e. Kundu et al. (2008) and Cordeau et al. (2010). The implementation of this metaheuristic is shown in Figure 2.

The variables associated are: S the initial solution generated by the greedy randomized construction heuristic, S' the neighbouring solution generated by applying a local operator to S, S_{Best} the best solution found, O the set of local operators which perturb the solution S, T the initial temperature, δT the decrement rate, $StepSize$ the maximum number of steps before restarting from the best solution, and lastly, *count* which counts the number of iterations.

Variables: S: current solution, S': neighbouring solution, S_{Best}: the best solution, O: the set of local operators, T : initial temperature, δT: the cooling rate, $StepSize$: maximum steps before beginning from best solution, *count*: counter for iterations,

```
1:  $S_{Best} \leftarrow S$
2:  $count \leftarrow 0$
3:  while termination criteria not met do
4:      $randomly \quad choose \quad o \in O$
5:      $S' \leftarrow o(S)$
6:      if $S' \leq S$ then
7:          $S \leftarrow S'$
8:          if $S \leq S_{Best}$ then
9:              $S_{Best} \leftarrow S$
10:             $count \leftarrow 0$
11:         end if
12:     else
13:         $r \leftarrow random(0,1)$
14:         $p \leftarrow exp^{(}S' - S)/T$
15:         if $p \geq r$ then
16:             $S \leftarrow S'$
17:         end if
18:     end if
19:     $T \leftarrow T \cdot \delta T$
20:     $count \leftarrow count + 1$
21:     if $count = StepSize$ then
22:         $S \leftarrow S_{Best}$
23:         $count \leftarrow 0$
24:     end if
25: end while
26: return $S_{Best}$
```

Fig. 2: Simulated annealing with restart metaheuristic

The initial solution S, generated by the greedy randomized heuristic, is saved as the best solution S_{Best} on line 1, and *count* is set to 0. Whilst the termination criterion is not met, i.e. there is computational time remaining, a local operator is selected on line 4. This local operator o is applied to the solution S on line 5 generating a neighbouring solution S'. On line 6 this solution S' is evaluated. If it has a lower objective function than S, then it replaces S. Next, on line 8 the solution S is evaluated against the best solution S_{Best} and if better, the best solution is updated and the *count* is set to zero. However, if solution S' is not better than the current solution S then it is evaluated using the simulated annealing criterion and compared to a random number r generated on the interval (0,1). If the probability p of accepting this solution is greater than r then solution S is updated. After every iteration, the simulated annealing temperature is reduced and the *count* is incremented by one. Once the *count* has reached its maximum value, $StepSize$, solution S is set to S_{Best} on line 22, and the *count* is set back to 0. Once the termination criterion has been met, the best solution S_{Best} is output on line 26.

5 Computational experiments

A series of computational experiments have been performed to ensure the least costly solution is produced using the greedy randomized heuristic. We are comparing the performance of the greedy randomized heuristic against the best known, average and maximum scores achieved by Kovacs et al. (2012) who used a pALNS.

5.1 Tuning the greedy randomized heuristic

The first set of experiments has been used to tune the level of randomness within the greedy randomized heuristic. In order to minimise the number of tuning experiments to be performed, it was decided that the chance of selecting a sorting method, other than the random sorting method, will each have an equal probability. Our experiments range from using an equal level of random sorting in comparison to the other sorting methods (0.2), to using no randomness (0) to find the optimal value of randomness. We present a percentage gap comparison compared against the results presented in Kovacs et al. (2012). We compare the percentage gap based on many criteria; average gap (across all of the datasets used), average gap from the *01* instances (where 100% of the jobs have time windows), average gap on the *03* instances (where 50 % of the jobs have time windows), the $NoTeam$ instances (no outsourcing needed), the $ReducedNoTeam$ instances (where outsourcing is needed) and lastly, a comparison across the different numbers of domains and skills, 5×4, 6×6 and 7×4.

Table 1: Tuning datasets for the greedy randomized heuristic

Datasets
$C101\_5 \times 4\_NoTeam$
$C103\_5 \times 4\_NoTeam$
$C201\_5 \times 4\_ReducedNoTeam$
$C203\_5 \times 4\_ReducedNoTeam$
$R101\_6 \times 6\_NoTeam$
$R103\_6 \times 6\_NoTeam$
$R201\_6 \times 6\_ReducedNoTeam$
$R203\_6 \times 6\_ReducedNoTeam$
$RC101\_7 \times 4\_NoTeam$
$RC103\_7 \times 4\_NoTeam$
$RC201\_7 \times 4\_ReducedNoTeam$
$RC203\_7 \times 4\_ReducedNoTeam$

The 12 datasets chosen from the STRSPTW instances for the tuning experiments are shown in Table 1. The name of the dataset describes its characteristics. For example in $R103\_6 \times 6\_NoTeam$, we can deduce that the jobs are randomly located by the *R* (*C* is clustered and *RC* randomly clustered), the jobs have 50% time windows defined by the *03* (*01* means 100% time windows), there are 6 domains and 6 levels within each domain represented by 6×6, and, lastly, the instance is a no team instance represented by *NoTeam*.

Table 2 displays the percentage gap achieved from the best known score (BKS) as found by Kovacs et al. (2012). The first column displays the level of randomness, *r*, in each implementation and the second column shows the average percentage gap achieved across all of the datasets used in the tuning experiments. The third column shows the average gap from the BKS for the *01* datasets, and column 4 displays the average gap from BKS for the *03* datasets. Columns 5 and 6 show the average gap from BKS achieved across the *NoTeam* and *ReducedNoTeam* datasets respectively. Lastly, columns 7, 8 and 9 display the average gap from the BKS across the 5×4, 6×6 and 7×4 datasets.

Table 2: Tuning experiment results for the greedy randomized heuristic

r	All	*01*	*03*	*NoTeam*	*ReducedNoTeam*	5×4	6×6	7×4
0.2	0.0634	0.0217	0.1052	0.0602	0.0666	0.0379	0.0575	0.0759
0.16	0.0622	0.0258	0.0985	0.0674	0.0569	0.0353	0.0499	0.0837
0.12	0.0605	0.0253	0.0957	0.0614	0.0595	0.0383	0.0543	0.0696
0.08	**0.0587**	0.0206	0.0968	0.0543	0.0630	0.0370	0.0516	0.0688
0.04	0.0599	0.0248	0.0949	0.0563	0.0634	0.0386	0.0521	0.0696
0.00	0.0605	0.0239	0.0970	0.061	0.0599	0.0394	0.0511	0.0712

The experiments have shown that the gap from BKS is minimized overall when using a randomness level $r = 0.08$. Interestingly, these experiments have also highlighted characteristics within the datasets. For example, it seems that the *01* dataset experiments produced a much smaller gap from BKS than the *03* instances. However, the average gaps from BKS seem to be equal when comparing the *ReducedNoTeam* and *NoTeam* instances. In addition there also seems to be a distinction between the number of domains and skill levels, where the larger the number of domains and skills, the larger the gap from optimal results. The tuning experiments performed have shown it is important to have an element of randomness within the algorithm, in order to minimize the gap from BKS.

5.2 Tuning the simulated annealing with restart metaheuristic

The second set of experiments aimed to find the optimal parameter values for the simulated annealing with restart metaheuristic. The restart metaheuristic has three parameters; the *Temperature*, the *Decrement* and the *StepSize*. The *Temperature* value controls how likely it is to accept a worse quality solution. *Decrement* controls the rate of decrease in accepting a worse solution and, lastly, the *StepSize* controls how frequently we revert back to the best solution. Each parameter had two levels, and therefore 2^3 tests had to be undertaken. We have performed a series of experiments using datasets $C101\_5 \times 4\_NoTeam$ and $C103\_5 \times 4\_NoTeam$ (Kovacs et al., 2012), using the parameter values shown in Table 3 to find the main and interaction effects.

Table 3: Main and interaction effects parameter testing

Experiment	*StepSize*	*Temperature*	*Decrement*
1	10,000	25	0.9999
2	25,000	25	0.9999
3	10,000	50	0.9999
4	25,000	50	0.9999
5	10,000	25	0.99999
6	25,000	25	0.99999
7	10,000	50	0.99999
8	25,000	50	0.99999

The simulated annealing with restart metaheuristic was run 10 times for each experiment, and the average objective value obtained was recorded. The main and interaction effects of each parameter are explained in the following subsections for each dataset tested.

5.2.1 $C101\_5 \times 4\_NoTeam$

Figure 3 shows the results obtained for dataset $C101\_5 \times 4\_NoTeam$. The main interactions plot illustrates that *Decrement* seems to have the strongest effect on solution quality due to the steepness of the line, i.e. the higher the decrement value the lower mean objective results that are produced. *Temperature* also has a significant effect on the quality of solution obtained, using a higher temperature produces better quality results. *StepSize* has a smaller yet still significant impact, preferring a smaller step size.

From the interactions plot, shown in Figure 3, we can also deduce that *Decrement* and *StepSize* appear parallel, and therefore there is no interaction between these factors. However, factors *StepSize* and *Temperature* do have an interaction as they intersect each other. The most significant interaction is between *Temperature* and *Decrement*.

5.2.2 $C103\_5 \times 4\_NoTeam$

The results for $C103\_5 \times 4\_NoTeam$ is displayed in Figure 3. Again, the main effects plot shows that each parameter, *Temperature*, *StepSize* and *Decrement* has an

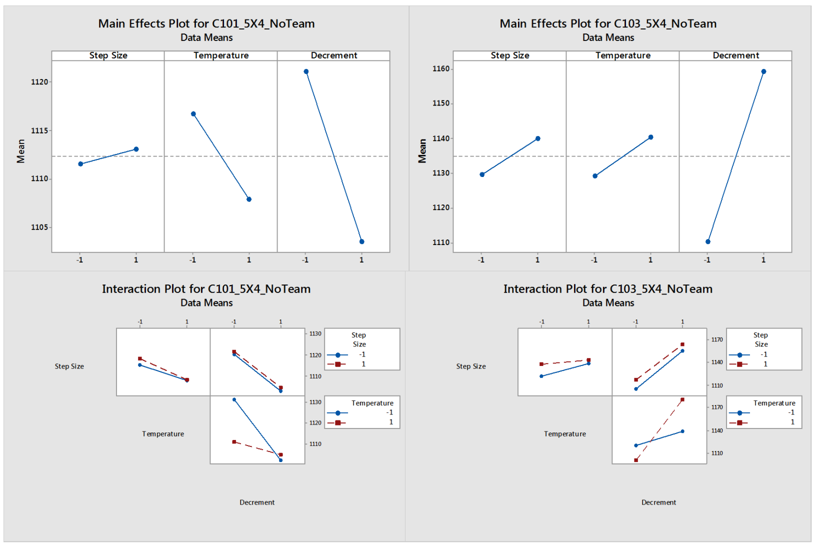


Fig. 3: Plot showing the main and interaction effects of each parameter for $C101\_5 \times 4\_NoTeam$ and $C103\_5 \times 4\_NoTeam$

effect on the quality of solution found. *Decrement* has the strongest effect on solution quality in this dataset, but in this dataset, the lower the decrement factor the lower mean objective results that are produced. The other parameters $StepSize$ and *Decrement* have a similar impact on solution quality evidenced from equal gradients. In this dataset, $C103\_5 \times 4\_NoTeam$, using a lower $Temperature$ and $StepSize$ leads to better quality results.

In Figure 3, the interactions plot for $C103\_5 \times 4\_NoTeam$ shows again that there is no interaction between *Decrement* and $StepSize$. The most significant interaction is between *Decrement* and $Temperature$, although $Temperature$ and $StepSize$ also have an interaction.

5.2.3 Summary of tuning experiments

Figure 3 demonstrates that the two types of datasets are affected differently by the parameter values set. In the 01 instances, a higher temperature and decrement lead to better objective values, whereas in the 03 datasets a lower temperature and decrement produce better quality results. This suggests the highly constrained datasets need more freedom to travel through the solution space, whereas the less constrained datasets need more focus.

5.2.4 Simulated annealing with restart performance

In this research, we have calculated both the average number of iterations and restarts for each dataset over ten runs of the greedy randomized heuristic. Tables 4 and 5 display the results for the $NoTeam$ and $ReducedNoTeam$ datasets respectively.

Due to the results from the tuning experiments performed in section 5.2 we have set the $StepSize$ to 10,000 iterations. This means that if no improvements are found on the best solution for 10,000 iterations, the heuristic resets the current solution S to S_{Best} and the search continues until the computational time limit has been reached.

Table 4: Simulated annealing with restart performance metrics $NoTeam$

Dataset	Average Restarts	Average Iterations
$C101\_5 \times 4\_NoTeam$	21	241,350
$C103\_5 \times 4\_NoTeam$	13	213,761
$C201\_5 \times 4\_NoTeam$	13.6	159,699
$C203\_5 \times 4\_NoTeam$	7	149,073
$R101\_5 \times 4\_NoTeam$	26.5	311,760
$R103\_5 \times 4\_NoTeam$	15.5	235,029
$R201\_5 \times 4\_NoTeam$	12.4	163,032
$R203\_5 \times 4\_NoTeam$	6.2	139,696
$RC101\_5 \times 4\_NoTeam$	22.6	284,923
$RC103\_5 \times 4\_NoTeam$	16.8	237,933
$RC201\_5 \times 4\_NoTeam$	13.6	172,695
$RC203\_5 \times 4\_NoTeam$	5.6	137,529
$C101\_6 \times 6\_NoTeam$	18.8	222,147
$C103\_6 \times 6\_NoTeam$	11.7	199,725
$C201\_6 \times 6\_NoTeam$	7.4	97,845
$C203\_6 \times 6\_NoTeam$	3.5	97,611
$R101\_6 \times 6\_NoTeam$	23.3	268,915
$R103\_6 \times 6\_NoTeam$	12.9	212,172
$R201\_6 \times 6\_NoTeam$	9	124,864
$R203\_6 \times 6\_NoTeam$	2.4	95,175
$RC101\_6 \times 6\_NoTeam$	19.8	247,406
$RC103\_6 \times 6\_NoTeam$	12.2	210,023
$RC201\_6 \times 6\_NoTeam$	8.7	129,974
$RC203\_6 \times 6\_NoTeam$	3.1	101,370
$C101\_7 \times 4\_NoTeam$	24.7	286,337
$C103\_7 \times 4\_NoTeam$	16.1	256,830
$C201\_7 \times 4\_NoTeam$	16.6	188,114
$C203\_7 \times 4\_NoTeam$	11.2	189,899
$R101\_7 \times 4\_NoTeam$	29.5	354,903
$R103\_7 \times 4\_NoTeam$	18.1	265,872
$R201\_7 \times 4\_NoTeam$	15.1	193,354
$R203\_7 \times 4\_NoTeam$	7.8	166,176
$RC101\_7 \times 4\_NoTeam$	24.8	313,719
$RC103\_7 \times 4\_NoTeam$	18.3	275,076
$RC201\_7 \times 4\_NoTeam$	15.6	196,620
$RC203\_7 \times 4\_NoTeam$	8.8	175,336

In Table 4 it is clear that the most iterations and restarts are performed on the *101* instances. The fewest iterations and restarts are performed on the *203* problem instances. This pattern occurs throughout the 5×4, 6×6 and 7×4 instances, regardless of the distributions of jobs i.e clustered, random and randomly clustered. In Table 5, similarly, the most iterations and restarts are performed on the *101* and the fewest on the *203* problem instances.

Table 5: Simulated annealing with restart performance metrics *ReducedNoTeam*

Dataset	Average Restarts	Average Iterations
$C101\_5 \times 4\_ReducedNoTeam$	19.4	237,760
$C103\_5 \times 4\_ReducedNoTeam$	11	208,798
$C201\_5 \times 4\_ReducedNoTeam$	11.6	136,426
$C203\_5 \times 4\_ReducedNoTeam$	4.8	127,628
$R101\_5 \times 4\_ReducedNoTeam$	22.4	270,029
$R103\_5 \times 4\_ReducedNoTeam$	17.4	244,082
$R201\_5 \times 4\_ReducedNoTeam$	8.9	129,428
$R203\_5 \times 4\_ReducedNoTeam$	3.6	111,218
$RC101\_5 \times 4\_ReducedNoTeam$	22.3	274,143
$RC103\_5 \times 4\_ReducedNoTeam$	12.2	233,929
$RC201\_5 \times 4\_ReducedNoTeam$	10.8	157,719
$RC203\_5 \times 4\_ReducedNoTeam$	4.9	127,146
$C101\_6 \times 6\_ReducedNoTeam$	18.9	224,364
$C103\_6 \times 6\_ReducedNoTeam$	8.5	187,398
$C201\_6 \times 6\_ReducedNoTeam$	7.3	101,494
$C203\_6 \times 6\_ReducedNoTeam$	2.2	101,176
$R101\_6 \times 6\_ReducedNoTeam$	21.6	255,237
$R103\_6 \times 6\_ReducedNoTeam$	13.3	213,570
$R201\_6 \times 6\_ReducedNoTeam$	5	103,062
$R203\_6 \times 6\_ReducedNoTeam$	2.9	86,080
$RC101\_6 \times 6\_ReducedNoTeam$	19.6	248,300
$RC103\_6 \times 6\_ReducedNoTeam$	13.1	210,875
$RC201\_6 \times 6\_ReducedNoTeam$	5.7	110,432
$RC203\_6 \times 6\_ReducedNoTeam$	2.7	88,949
$C101\_7 \times 4\_ReducedNoTeam$	25.3	282,559
$C103\_7 \times 4\_ReducedNoTeam$	16.4	249,775
$C201\_7 \times 4\_ReducedNoTeam$	13.5	145,236
$C203\_7 \times 4\_ReducedNoTeam$	6.7	136,232
$R101\_7 \times 4\_ReducedNoTeam$	29.4	331,041
$R103\_7 \times 4\_ReducedNoTeam$	21.5	282,724
$R201\_7 \times 4\_ReducedNoTeam$	11.2	154,580
$R203\_7 \times 4\_ReducedNoTeam$	5.6	125,351
$RC101\_7 \times 4\_ReducedNoTeam$	27.4	315,893
$RC103\_7 \times 4\_ReducedNoTeam$	20.5	281,064
$RC201\_7 \times 4\_ReducedNoTeam$	12.2	161,558
$RC203\_7 \times 4\_ReducedNoTeam$	5.9	138,643

We believe that this pattern, more iterations and restarts, may occur because the *101* instances are the most constrained, so there are fewer feasible insertions for jobs, which speeds up the heuristic's decision making. In contrast, the *203* problem instances are the least constrained, therefore there are more feasible insertions to choose from which slows down the heuristic.

5.3 STRSPTW computational results

The greedy randomized heuristic was programmed in Java and tested on an HP Z210 Workstation, with an i7-2600 CPU with 3.4 GHZ with 12GB of RAM. Each run on the *NoTeam* instances lasted 80 seconds and each run on the *ReducedNoTeam* instances lasted 60 seconds, as in Kovacs et al. (2012) for comparison purposes. The greedy randomized heuristic was run 5 times per data instance, and the best, average and worst results obtained are shown in Tables 6 and 7.

Column 1 shows the dataset, columns 2-4 show the best, average and maximum objective value achieved by Kovacs et al. (2012) with the pALNS, and columns 4-7 displays the best, average and maximum objective values found by the greedy randomized heuristic. The highlighted rows indicate where the greedy randomized heuristic has found a lower objective value than was achieved by Kovacs et al. (2012) with the pALNS.

6 Discussion

6.1 Performance of greedy randomized heuristic on *NoTeam* instances

Table 6 displays the results achieved for the *NoTeam* problem instances. In these datasets, the sequential greedy randomized heuristic is able to find a lower minimum objective value than the pALNS in 8 out of 36 datasets and is able to find the same minimum objective value in two datasets, $C201\_6 \times 6\_NoTeam$ and $C101\_7 \times 4\_NoTeam$.

The results illustrate that the greedy randomized heuristic finds a smaller gap from BKS on the *01* instances compared to the *03* instances. The difference between these datasets is the proportion of time windows, the *01* instances are more constrained (contain 100% time windows) compared to the *03* instances (contain 50% time windows) and therefore, there are fewer options of where and when to allocate a job. In the 5×4, 6×6 and 7×4 datasets, the gap from minimum objective results in the *01* instances is 1.08%, 1.98% and 1.12%. This increases to 11.77%, 14.03% and 8.82% in the *03* instances.

Another trend within the results occurs in the *203* datasets. These datasets achieve the highest gap from BKS overall, regardless of the distribution of job locations i.e *C* clustered, *R* randomly, *RC* randomly clustered. This pattern occurs across each set of domains and levels, 5×4, 6×6 and 7×4.

Furthermore, we can also deduce a pattern across the distribution of customers' locations in the 5×4 and 7×4 datasets. Throughout the distributions, C, R and RC, the gap from BKS increases. In 5×4 the gap is 5.75%, 6.59% and 7.03% and in 7×4 the gap is 3.98%, 4.07% and 6.86%, respective of distributions R,C and RC.

Table 6: *NoTeam*

Dataset		pALNS			GREEDY	
	min	avg	max	min	avg	max
$C101\_5 \times 4\_NoTeam$	1098.71	1111.08	1128.02	1096.85	1135.03	1180.95
$C103\_5 \times 4\_NoTeam$	1018.61	1037.33	1049.41	1075.36	1119.66	1195.76
$C201\_5 \times 4\_NoTeam$	1158.97	1180.93	1228.99	1157.65	1163.1	1183.31
$C203\_5 \times 4\_NoTeam$	1046.93	1049.3	1052.83	1228.23	1297.39	1337.33
$R101\_5 \times 4\_NoTeam$	1678.68	1685.85	1697.2	1672.55	1682.17	1692.81
$R103\_5 \times 4\_NoTeam$	1238.67	1249.91	1282.28	1288.48	1312.95	1339.13
$R201\_5 \times 4\_NoTeam$	1440.3	1448.93	1462.62	1526.43	1563.53	1599.98
$R203\_5 \times 4\_NoTeam$	1098	1106.12	1123.08	1281.83	1334.01	1378.83
$RC101\_5 \times 4\_NoTeam$	1708.51	1716.07	1729.75	1676.57	1721.95	1760.88
$RC103\_5 \times 4\_NoTeam$	1337.99	1354.11	1388.13	1454.46	1482.46	1507.06
$RC201\_5 \times 4\_NoTeam$	1601.89	1607.25	1610.75	1650.66	1698.03	1727.49
$RC203\_5 \times 4\_NoTeam$	1161.53	1166.5	1178.64	1373.44	1430.32	1467.19
$C101\_6x6\_NoTeam$	989.21	1004.82	1029.72	973.05	1002.15	1029.72
$C103\_6 \times 6\_NoTeam$	893.94	897.86	907.62	1075.26	1181.12	1239.93
$C201\_6 \times 6\_NoTeam$	821.55	821.55	821.55	821.55	847.22	868.72
$C203\_6 \times 6\_NoTeam$	689.6	703.1	750.12	831.51	908.91	970.66
$R101\_6 \times 6\_NoTeam$	1658.27	1667.43	1672.57	1662.69	1666.02	1675.24
$R103\_6 \times 6\_NoTeam$	1223.63	1231.49	1243.49	1243.7	1264.54	1286.5
$R201\_6 \times 6\_NoTeam$	1261.94	1270.26	1279.81	1335.66	1375.56	1417.77
$R203\_6 \times 6\_NoTeam$	932.35	951.84	964.54	1104.75	1153.24	1200.86
$RC101\_6 \times 6\_NoTeam$	1679.13	1683.96	1690.06	1672.85	1686.62	1693.34
$RC103\_6 \times 6\_NoTeam$	1281.55	1310.95	1331.46	1354.14	1381.79	1400.85
$RC201\_6 \times 6\_NoTeam$	1395.4	1403.95	1411.48	1494.14	1547.03	1613.75
$RC203\_6 \times 6\_NoTeam$	1001.04	1016.71	1030.15	1176.41	1236.67	1291.41
$C101\_7 \times 4\_NoTeam$	1357.05	1398.95	1462.16	1357.05	1416.19	1553.71
$C103\_7 \times 4\_NoTeam$	1215.7	1239.22	1264.17	1263.67	1295.83	1335.95
$C201\_7 \times 4\_NoTeam$	1256.56	1282.18	1302.56	1256.3	1264.26	1302.56
$C203\_7 \times 4\_NoTeam$	1150.85	1151.27	1152.94	1288.96	1354.81	1474.64
$R101\_7 \times 4\_NoTeam$	1776.46	1793.95	1813.53	1771.56	1791	1807.89
$R103\_7 \times 4\_NoTeam$	1346.8	1375.09	1399.95	1402.04	1423.96	1456.49
$R201\_7 \times 4\_NoTeam$	1398.14	1410.9	1427.95	1427.56	1458.63	1474.29
$R203\_7 \times 4\_NoTeam$	1164.9	1166.94	1169.27	1285.35	1334.17	1407.76
$RC101\_7 \times 4\_NoTeam$	1821.9	1844.37	1859.17	1832.75	1903.58	1980.33
$RC103\_7 \times 4\_NoTeam$	1435.63	1455.33	1477.84	1547.33	1610.13	1679.64
$RC201\_7 \times 4\_NoTeam$	1697.82	1701.25	1705.48	1771.25	1793.66	1811.06
$RC203\_7 \times 4\_NoTeam$	1239.45	1241.65	1249.72	1422.35	1459.22	1527.29

6.2 Performance of greedy randomized heuristic on *ReducedNoTeam* instances

Table 7 displays the results achieved for the *ReducedNoTeam* problem instances. In these datasets, the greedy randomized heuristic is able to find a lower minimum objective value than the pALNS in 5 out of 36 datasets; and is able to find the same minimum objective value in two datasets $C201\_5 \times 4\_ReducedNoTeam$ and $C201\_7 \times 4\_ReducedNoTeam$.

The results again show that the greedy randomized heuristic finds a smaller gap from BKS on the *01* instances compared to the *03* instances with the average minimum gap equal to 0.36%, 3.05% and 0.7% across the *01* instances for the 5×4,

Table 7: *ReducedNoTeam*

Dataset		pALNS			GREEDY	
	min	avg	max	min	avg	max
*C*101_5 × 4_*ReducedNoTeam*	5656.63	5733.75	5806.55	5572.99	5798.25	6286.36
*C*103_5 × 4_*ReducedNoTeam*	2644.65	2782.2	2869.64	2941.77	3461.57	3861.02
*C*201_5 × 4_*ReducedNoTeam*	2755.52	2755.52	2755.52	2755.52	2755.52	2755.52
*C*203_5 × 4_*ReducedNoTeam*	2389.37	2392.5	2393.62	2591.11	2680.16	2907.03
*R*101_5 × 4_*ReducedNoTeam*	5582.58	5895.38	6181.52	5663.64	6030.76	6400.05
*R*103_5 × 4_*ReducedNoTeam*	1710.25	1845.25	2020.48	2034.5	2378.57	2664.23
*R*201_5 × 4_*ReducedNoTeam*	2838.5	2854.3	2865.75	2895.78	3014.47	3250.06
*R*203_5 × 4_*ReducedNoTeam*	2332.23	2332.23	2332.23	2544.48	2595.48	2802.92
*RC*101_5 × 4_*ReducedNoTeam*	5127.79	5164.84	5262.36	5103.33	5428.46	5830.71
*RC*103_5 × 4_*ReducedNoTeam*	2170.57	2348.06	2490.12	2661.07	2984.91	3500.22
*RC*201_5 × 4_*ReducedNoTeam*	3088.23	3091.67	3093.56	3107.26	3217.95	3282.21
*RC*203_5 × 4_*ReducedNoTeam*	2516.16	2540.35	2550.62	2672.52	2761.53	2828.82
*C*101_6 × 6_*ReducedNoTeam*	7731.07	7762.94	7791.08	7660.86	7763.51	8151.45
*C*103_6 × 6_*ReducedNoTeam*	4980.7	5028.83	5136.21	5242.58	5771.19	6394.29
*C*201_6 × 6_*ReducedNoTeam*	3278.07	3299.56	3328.01	3283.84	3405.21	3603.53
*C*203_6 × 6_*ReducedNoTeam*	2460.17	2465.9	2468.71	2743.44	2923.61	3101.72
*R*101_6 × 6_*ReducedNoTeam*	5955.17	6152.29	6322.82	6174.57	6453.63	6701.15
*R*103_6 × 6_*ReducedNoTeam*	2251.64	2329.28	2404.57	2485.9	2839.6	3228.95
*R*201_6 × 6_*ReducedNoTeam*	3503.4	3536.7	3574.97	3635.03	3857.49	4072.21
*R*203_6 × 6_*ReducedNoTeam*	2437.28	2446.18	2481.77	2649.38	2806.12	2955.39
*RC*101_6 × 6_*ReducedNoTeam*	5276.34	5466.18	5771.99	5231.5	5513.1	5664.35
*RC*103_6 × 6_*ReducedNoTeam*	2263.83	2349.57	2522.71	2704.04	3212.48	3945.85
*RC*201_6 × 6_*ReducedNoTeam*	4422.86	4519.95	4656.79	4973.47	5310.02	5753.73
*RC*203_6 × 6_*ReducedNoTeam*	2649.51	2673.72	2730.78	2825.65	3239.56	3615.14
*C*101_7 × 4_*ReducedNoTeam*	5208	5257.9	5307.12	5256.49	5438.5	5732.91
*C*103_7 × 4_*ReducedNoTeam*	2020.4	2117.44	2173.39	2205.58	2569.24	2804.6
*C*201_7 × 4_*ReducedNoTeam*	2773.41	2779.37	2803.21	2773.41	2784.02	2820.23
*C*203_7 × 4_*ReducedNoTeam*	2261.37	2282.15	2301.73	2450.03	2566.85	2755.9
*R*101_7 × 4_*ReducedNoTeam*	5239.81	5381.35	5437.66	5232.6	5580.65	6018.08
*R*103_7 × 4_*ReducedNoTeam*	2104.93	2215.84	2314.3	2338.84	2427.95	2666.58
*R*201_7 × 4_*ReducedNoTeam*	2672.96	2679.38	2682.23	2706.8	2764.65	2936.23
*R*203_7 × 4_*ReducedNoTeam*	2199.1	2209.8	2229.67	2318.15	2368.62	2443.63
*RC*101_7 × 4_*ReducedNoTeam*	5531.06	5799.77	6367.47	5627.79	5959.56	6467.22
*RC*103_7 × 4_*ReducedNoTeam*	2586.03	2676.54	2820.48	3127.24	3633.13	3963.97
*RC*201_7 × 4_*ReducedNoTeam*	2919.83	2936.28	2945.46	2930.78	3033.64	3244.95
*RC*203_7 × 4_*ReducedNoTeam*	2277.62	2285.17	2301.26	2459.13	2541.01	2675.67

6×6 and 7×4 problems. However, the gap from minimum objective value results increases for the *03* instances to 12.5%, 10.33% and 10.49%.

Again, we can find a pattern within the distribution of customers' locations. In the *ReducedNoTeam* problem instances, the pattern occurs within each skill domain area, 5×4, 6×6 and 7×4. The gap from minimum results is the smallest when the distribution of jobs is clustered (C), 4.55%, 4.01% and 4.61%. The gap is largest when the jobs are distributed in random clusters (RC), 7.24%, 9.42% and 7.76%.

In these datasets there is a new pattern present in both the 5×4 and 7×4 datasets. In these datasets, the pattern occurs in the *201* datasets. These datasets achieve the highest gap from BKS overall, regardless of the distribution of job locations i.e C, R and RC. In the 6×6 datasets the *203* achieve the highest gap from BKS overall.

6.3 Summary of performance

Overall, considering the results presented on the 72 datasets, we calculate that the gap from minimum, average and maximum results is 6.38%, 10.07% and 14.04%. We can split this into the performance gap on the *01* instances as 1.39%, 3.27% and 5.82%, and on the *03* instances as 11.36%, 16.86% and 22.27%.

The results presented in this paper illustrate that the sequential greedy randomized heuristic approach can in some cases, 13 out of 72, find a better quality solution than the pALNS approach presented by Kovacs et al. (2012). This is the first time that another heuristic approach, and indeed a sequential approach, has been tested on these datasets, allowing a comparison with pALNS. Our approach has not performed as well on the *03* problem datasets which include only 50% of datasets with time windows. We believe this is due to the number of configurations that are possible, as the datasets are less constrained than the *01* datasets.

In addition the pALNS approach uses adaptive operators that change their chance of selection dependent on performance so far within the search phase. The datasets presented in this paper can be split by many factors such as percentage of time windows, *01* and *03*, how customers are geographically located, *C* clustered, *R* randomly, *RC* randomly clustered, and the number of domain skill areas, 5×4, 6×6 and 7×4, which suggests that an adaptive approach would perform well.

7 Conclusion

In this work we have presented our approach to solving the STRSPTW. We have designed and implemented a sequential greedy randomized construction heuristic to solve the STRSPTW datasets which have not been tested since Kovacs et al. (2012), with a pALNS. Furthermore, we have also implemented a simulated annealing with restart metaheuristic, and tested both the main and interaction effects of parameters across different types of dataset. Lastly, we have also found 13 new best known results for these datasets. We have shown that the presence of time windows can greatly affect the solution approach and that the algorithmic performance is also heavily dependent on characteristics of the datasets tested i.e. the proportion of time windows within the datasets.

The results found in this work can be readily applied to other scheduling problems with common constraints such as the home healthcare problem. In this problem trained professionals (skill complexity) travel to patient locations (travel time) to administer medications under strict guidelines (time windows). Furthermore, this work also has the potential to make an environmental impact, since vehicles are used for transportation between locations, and the smaller the distances travelled, the fewer emissions are produced.

Future work will focus on related service maintenance combinatorial optimisation problems, featuring other common constraints such as tools and spare parts.

Acknowledgements This research is sponsored by ServicePower Technologies PLC, a worldwide leader in providing innovative mobile workforce management solutions, in cooperation with MMU and KTP.

References

Author1, Author 2, and Author 3. Solving technician and task scheduling problems with an intelligent decision heuristic. In *Intelligent Decision Technologies 2016*, pages 63–75. Springer, 2016.

Author1, Author 2, and Author 3. Applying the intelligent decision heuristic to large scale technician and task scheduling problems. In *Intelligent Decision Technologies 2017*, pages xx–xx. Springer, 2017a.

Author1, Author 2, and Author 3. A review of technician and task scheduling problems, datasets and solution approaches. In *SAI Intelligent Systems Conference (IntelliSys), 2017*, page Accepted. IEEE, 2017b.

Christian Blum and Andrea Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, 35(3): 268–308, 2003.

J Arturo Castillo-Salazar, Dario Landa-Silva, and Rong Qu. A survey on workforce scheduling and routing problems. In *Proceedings of the 9th international conference on the practice and theory of automated timetabling*, pages 283–302. Citeseer, 2012.

Xi Chen, Barrett W Thomas, and Mike Hewitt. The technician routing problem with experience-based service times. *Omega*, 61:49–61, 2016.

Jean-François Cordeau, Gilbert Laporte, Federico Pasin, and Stefan Ropke. Scheduling technicians and tasks in a telecommunications company. *Journal of Scheduling*, 13(4):393–409, 2010.

Andreas T Ernst, Houyuan Jiang, Mohan Krishnamoorthy, and David Sier. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research*, 153(1):3–27, 2004.

Bertrand Estellon, Frédéric Gardi, and Karim Nouioua. High-performance local search for task scheduling with human resource allocation. In *Engineering Stochastic Local Search Algorithms. Designing, Implementing and Analyzing Effective Heuristics*, pages 1–15. Springer, 2009.

Murat Fırat and CAJ Hurkens. An improved mip-based approach for a multi-skill workforce scheduling problem. *Journal of Scheduling*, 15(3):363–380, 2012.

Hideki Hashimoto, Sylvain Boussier, Michel Vasquez, and Christophe Wilbaut. A grasp-based approach for technicians and interventions scheduling for telecommunications. *Annals of Operations Research*, 183(1):143–161, 2011.

Gerhard Hiermann, Matthias Prandtstetter, Andrea Rendl, Jakob Puchinger, and Günther R Raidl. Metaheuristics for solving a multimodal home-healthcare scheduling problem. *Central European Journal of Operations Research*, 23(1): 89–113, 2015.

Cor AJ Hurkens. Incorporating the strength of mip modeling in schedule construction. *RAIRO-Operations Research*, 43(04):409–420, 2009.

Amy Khalfay, Alan Crispin, and Keeley Crockett. Solving technician and task scheduling problems with an intelligent decision heuristic. In *Intelligent Decision Technologies 2016*, pages 63–75. Springer, 2016.

Attila A Kovacs, Sophie N Parragh, Karl F Doerner, and Richard F Hartl. Adaptive large neighborhood search for service technician routing and scheduling problems. *Journal of scheduling*, 15(5):579–600, 2012.

S Kundu, M Mahato, B Mahanty, and S Acharyya. Comparative performance of simulated annealing and genetic algorithm in solving nurse scheduling problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume 1, pages 96–100, 2008.

Ines Mathlouthi, Michel Gendreau, and Jean-Yves Potvin. Mixed integer programming for a multi-attribute technician routing and scheduling problem. 2016.

Dimitris C Paraskevopoulos, Gilbert Laporte, Panagiotis P Repoussis, and Christos D Tarantilis. Resource constrained routing and scheduling: Review and research prospects. 2016.

Victor Pillac, Christelle Guéret, and Andrés Medaglia. On the dynamic technician routing and scheduling problem. 2012.

Victor Pillac, Christelle Gueret, and Andrés L Medaglia. A parallel matheuristic for the technician routing and scheduling problem. *Optimization Letters*, 7(7):1525–1535, 2013.

Paul Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *International Conference on Principles and Practice of Constraint Programming*, pages 417–431. Springer, 1998.

French Operational Research Society. What is the roadef 2007 challenge, 2016. URL `http://challenge.roadef.org/2007/en/`.

Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

Fabien Tricoire, Nathalie Bostel, Pierre Dejax, and Pierre Guez. Exact and hybrid methods for the multiperiod field service routing problem. *Central European Journal of Operations Research*, 21(2):359–377, 2013.

Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013.

Jiyang Xu and Steve Y Chiu. Effective heuristic procedures for a field technician scheduling problem. *Journal of Heuristics*, 7(5):495–509, 2001.

Emilio Zamorano and Raik Stolletz. Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 2016.