Bi-level optimisation and machine learning in the management of large service-oriented field workforces.

AINSLIE, R.T.

2022

Bi-Level Optimisation and Machine Learning in the Management of Large Service-oriented Field Workforces

Russell Thomas Ainslie

PhD	2022

Bi-Level Optimisation and Machine Learning in the Management of Large Service-oriented Field Workforces

Russell Thomas Ainslie

Computing Science and Digital Media

Robert Gordon University

A thesis submitted in partial fulfilment of the
requirements of
Robert Gordon University
for the degree of Doctor of Philosophy

This research programme was carried out
in collaboration with British Telecommunications (BT)

October 2022

Abstract

The tactical planning problem for members of the service industry with large multi-skilled workforces is an important process that is often under looked. It sits between the operational plan which involves the actual allocation of members of the workforce to tasks and the strategic plan where long term visions are set. An accurate tactical plan can have great benefits to service organisations and this is something we demonstrate in this work. Sitting where it does it is made up of a mix of forecast and actual data which can make solving the problem effectively difficult. In members of the service industry with large multi-skilled workforces it can often become a very large problem very quickly as the number of decisions scale quickly with the number of elements within the plan.

In this study we first update and define the tactical planning problem to fit the process currently undertaken manually in practice. We then identify properties within the problem that identify it as a new candidate for the application of bi-level optimisation techniques. The tactical plan is defined in the context of a pair of leader-follower linked sub-models which we show can be solved to produce automated solutions to the tactical plan. We further identify the need for the use of machine learning techniques to effectively find solutions in practical applications where limited detail is available in the data due to its forecast nature. We develop neural network models to solve this issue and show that they provide more accurate results than the current planners. Finally we utilise them as a surrogate for the follower in the bi-level framework to provide real world applicable solutions to the tactical planning problem. The models developed in this work have already begun to be deployed in practice and are providing significant impact. This is along with identifying a new application area for bi-level modelling techniques.

Key Words: Bi-level, Genetic Algorithm, Machine Learning, Neural Network, Real-world, Large Problem, Service Industry

Acknowledgements

Firstly I would like to thank my supervisory team for providing inspiration and feedback throughout this work. They are Prof. John McCall (RGU), Dr. Sid Shakya (BT) and Dr. Gilbert Owusu (BT). For the latter I would also thank him for being foolish enough to offer me a job! Secondly thanks to my family for all their support over the years. I would also like to give a mention to the RGU research degrees office, most specifically Nick Anderson, for being very accommodating and understanding during the final write up process.

Contents

Notation List

Notation	**Definition**
r	Resource r which is a member of the workforce
s	Skill required to complete a task. Resources have skills they can perform and tasks have skills required to complete them
t	Index of a period in the plan – a period being a duration of time e.g. a day so t=0 is today and t=1 is tomorrow
a	Area a where an area is a geographical area within which a task may be required or resource may be based
x_{rast}	Allocation of capacity from resource r to skill s, in area a in period t
ω_{ras}	Cost of allocating 1 unit of time from resource r to skill s in area a
c_{rt}	Total costs associated with resource r in period t
k_{jt}	Total costs associated with task j in period t
ϵ_{ras}	Efficiency of resource r at using skill s in area a
D_{sat}	Total time allocated to complete tasks requiring skill s in area a in period t
x'_{rast}	Allocation of overtime from resource r to skill s in area a and period t
σ_{rt}	Total time available to resource r in period t
ϑ	Proportion of a resources time available for use as overtime
V	Cost for use of one time unit of overtime
g	Resource group which is a collection of resources that can perform the same skills
g_a	Resources from group g that are based in area a
$x_{g_a st}$	Allocation of time from resource group g in area a to skill s in period t
ω_{gs}	Cost of allocating a unit of time from resource group g to skill s

$m_{g_a g_{a'} t}$	Movement of time from resource group g in area a to area a’ in period t
$M_{g_a g_{a'}}$	Cost of moving a unit of resource group g’s time from area a to area a’
$\sigma_{g_a t}$	Total time available to resource group g in area a in period t
$x'_{g_a t}$	Overtime added to resource group g in area a in period t
θ	Total amount of overtime available
$i_{g_a t}$	Intake/recruitment of a new resource to group g in area a in period t
$w_{g_a t}$	Wastage/reduction of a resource from group g in area a in period t
I_c	Cost of adding a new resource
W_c	Cost of removing a resource
I	Total number of new resources available to add to the plan
W	Total number of reductions allowed to the plan
G'_{g_a}	Set of resource groups that members of resource group g in area a can be trained to move to by adding the relevant skill(s)
$\tau_{g_a g'_a t}$	Training of a member of resource group g in area a to move them to resource group g’
$v_{gg'}$	Cost of training a member of group g to move to group g’
P	Total number of training moves allowed within the plan
y_{jat}	Allocation of time to complete task of type j in area a in period t
φ_j	Cost for not completing a task of type j
ρ_{jat}	Total time required for tasks of type j in area a in period t
d_{jat}	Decision to delay tasks of type j in area a in period t to the next period
e_{jat}	Decision to do tasks of type j in area a in period t a period early
δ_j	Cost to delay time of task type j

η_j	Cost to do time required by tasks type j early
f_{jat}	Boolean indicating failure of the service level agreement for task type j in area in period t, where the SLA indicates that a percentage of tasks must be completed within a certain time of being due
F_{jat}	Cost associated with failing the SLA for task type j in area a in period t
L_j	Proportion of task type j that must be completed within the time defined by the SLA
l_j	Number of days defined in the SLA that a task of type j is allowed to be late

Chapter 1

Introduction

In recent decades, with the advent of privatisation and the increase in competition, there has been a drive in the service operations sector to improve the sophistication of supporting applications up to the level of the far more mature supply sector (Voudouris, 2008). Although similarities exist between the fields, which has directed some of the early developments, it has been found that differences are significant enough to prevent the wholesale utilisation of current supply chain management solutions within the service sector. Both essentially involve getting resoures to the right place to cover expected demand, however in the service chain these are often people and thus there can be no stockpiling utilised to provide any buffer against variations. Within the service industry waiting times are also paramount, with many situations where customers expect a near immediate response (e.g. calls to a call centre or repairs to a vital service).

Along with the requirement for a quality forecast to correctly anticipate future requirements, this also relies on accurate planning to ensure that there is adequate resourcing available at all times to cover the expected levels of demand. Research shows that there is a current drive to improve outcomes through the introduction of automation into this service chain planning process. (Owusu & O'Brien, 2013)

In the remainder of this section we will define the overall service chain planning process described in the literature before highlighting a specific level of that process that contains the problem that is the motivation for this thesis. This will be followed by a brief statement of the impacts already achieved as outcomes of this research, including both publications and through being used in real world applications within the industrial partner. This chapter then ends with the definition of the outline for the remainder of this work.

1.1 Service Chain Planning

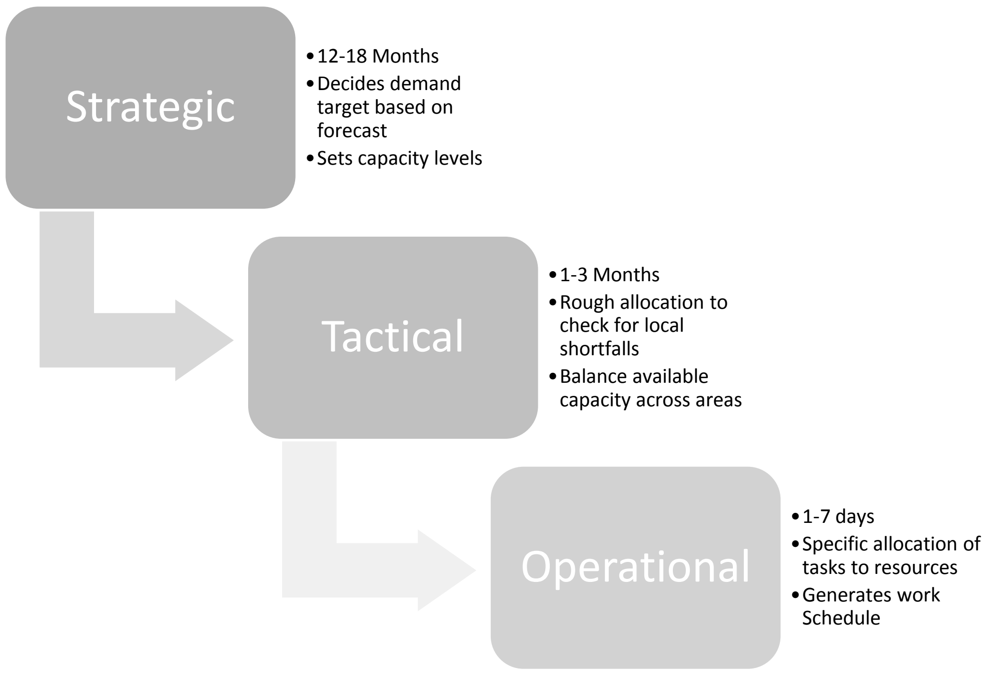
Strategic
•12-18 Months
•Decides demand target based on forecast
•Sets capacity levels
Tactical
•1-3 Months
•Rough allocation to check for local shortfalls
•Balance available capacity across areas
Operational
•1-7 days
•Specific allocation of tasks to resources
•Generates work Schedule

Fig. 1 Service Chain Planning Hierarchy

The current service chain planning model initially evolved from supply chain planning as early work attempted to draw on the similarities between the two processes (Schmidt & Wilhelm, 2000). Both contain three planning stages, strategic, tactical and operational, with the planning horizon reducing while the level of detail increases as it descends through the levels (see Fig 1.). Within the service chain planning field these levels are roughly defined as:

- Strategic Planning – Performed for the next 12-18 months. Takes unconstrained demand forecasts for products and services and decides, based on financial constraints and business targets, how much capacity to make available to complete a target portion of that demand. This produces an output that is a constrained capacity and demand profile used in the next planning stage (Owusu, et al., 2008).
- Tactical Planning – Undertaken for the next 1-3 months. Breaks down the constrained demand from the strategic plan into more fine grained detail, splitting it by activity, geographical area and time period (e.g. daily instead of weekly/monthly). At this level the demand and capacity are a mixture of forecast and known data. This finer grained demand is matched against capacity to identify any more local shortfalls or surpluses. Actions are

then taken to attempt to find a greater balance, such as by moving resources between skills or areas, to ensure that on the day there is enough capacity to cover demand. The outputs of this process are the finer grained capacity deployment input to the operational plan and the expected demand levels that can be covered sent to the reservation system to open bookings (Kern & Owusu, 2008).

- Operational Planning – Covers a window of 1-7 days. Allocates the resources that have been made available in specific geographies and skills to jobs, from customer orders to planned work, in an optimal manner. The aim of this process being to allocate the right job to the right resource at the right time to improve customer satisfaction, increase efficiency and provide robustness (Liret & Dorne, 2008).

Of these stages the tactical plan is often overlooked despite it being an important process to help ensure good customer service levels (Shakya, et al., 2013). As such it is this stage that will be the focus of this thesis.

There is however a lack of consensus in the literature about precisely what falls into this level. In (Mohamed, et al., 2012) and (Mohamed, et al., 2013) tactical planning is defined as the matching of capacity to demand across skills and areas as above, however the scope is only for the next couple of days. It also predominantly deals with known data rather than forecast, involving the matching of actual resources to actual tasks. As such it is closer to an operational plan than a traditional tactical plan. In (Kassem, et al., 2012) the boundary between tactical planning and operational planning (what they call scheduling) is better defined with the definitions more closely matching the early literature. However there are also some cases where the blurring occurs in the other direction, between the strategic and the tactical layers. For example, (Ross, 2017) defines a new aggregate planning layer sitting between tactical and operational over the horizon of around 90 days. Individual workers are aggregated into groups with the same skill loadout and the time from each bucket is allocated to task completions based on the group's efficiency/preference for each skill to identify surpluses or shortfalls in capacity vs demand. This actually is very close to the traditional tactical planning process. The tactical plan in their view however covers a longer period than standard, 12 – 18 months, and contains some of the strategic level capacity decisions. The inclusion of capacity decisions does match what can be seen within real world planning applications however the time horizon is still that of the traditional tactical plan (Kern, et al., 2009). The reason for the inclusion in practice is that a lot of capacity flexing decisions, such as overtime, can be actioned closer to the day so the decision can be delayed to take advantage of the identification of shortfalls coming out of the traditional tactical planning process. This all leads to the updated definition of the tactical plan defined in section 1.2 that is the focus of the work for the rest of this thesis.

1.2 The Tactical Planning Problem

In a member of the service industry with a large multi-skilled workforce the tactical planning process consists of optimising a plan with two main components, the capacity and the demand.

The capacity for the plan is made up of the members of the workforce available on each period of the plan, called the resources. Each resource has a number of different skills they can perform as well as an amount of available time in each period of the plan. At the tactical planning level these are most often used at an aggregated level. This means a specific resource would actually be an aggregation of all individual members of the workforce with the same skill set. Additionally, for the purposes of this thesis a plan period is defined as a day.

The demand portion of the plan consists of the jobs that are to be completed. Similarly to the capacity, the jobs are aggregated for the purposes of the tactical plan. They are grouped by the skill required to complete them into workstacks. In the tactical plan there are generally two different types of workstacks, those where the planner has no control over the number of jobs needing completed and those where the planner does have some control. An example of the former type involves repairing of faults where the number of new jobs added is set by the future forecast of the levels of faults expected. The latter type is one where the planner can set the number of jobs available, such as setting the number of appointments to make available for deliveries, a new connection, etc. For the rest of this thesis to give clarity these will be defined using the telecoms nomenclature, where the former shall be referred to as faults and the latter as installation. Other service industries may use slightly different terminology however the underlying definitions are the same.

The fault workstacks consist of the number of jobs waiting to be completed (backlog), the number of new jobs expected to appear each day of the plan (intake) and the target number to complete each day. In the majority of situations, the target is not to clear the entire backlog, as this may be economically infeasible and may not even be possible (e.g. if the fault job is within a property it will only be possible to complete it when access is available). The target is more usually to complete a certain percentage of the current backlog. That percentage target is set, often dynamically for different days of the plan, to ensure target levels of service are met. A service level target being to successfully repair a certain percentage of faults within a set time period of them being reported.

The installation workstacks do not contain a backlog nor an intake, instead the planner sets the number of jobs to make available each day of the plan. This provides the plan output to the reservation system that opens up appointments for customers to book. There is no guarantee that all appointments will be picked up, thus planners will usually adjust these values each new day as the current levels of uptake are updated.

Both workstack types share a minimum number of jobs requiring completion on each day, these are jobs that have already been appointed such as an installation job that a customer has already booked or fault tasks that have been appointed where property access is required.

The task of the planner, and the definition of the tactical planning problem, is to match the capacity to the demand in such a way that the demand targets are met as closely as possible. This is a two-part process; the first part is in optimally assigning the resources to jobs based on their available time and the skills they can perform, called the capacity-demand matching process; the second part is the attempt to bridge the gap between the capacity and the demand through the use of various levers.

Fig. 2 shows the levers available to the tactical planner. The demand lever mentioned earlier is to decide the number of installation jobs to make available for booking, this modifies the demand profile entering the capacity-demand matching phase. The productivity lever defines the amount of time it takes a resource to complete a job requiring a given skill. This is used to model planned efficiency improvements or to allow the planner to estimate potential variations in output levels. This is directly input into the capacity-demand matching as it sets how an amount of resource time translates into a job completion for each skill. The remaining levers all relate to the capacity and will modify the amount of resource available for the capacity-demand matching process. These capacity levers are:

- Recruitment/Reductions – The moving of resources in or out of the plan, this could be through movements such as from or to other parts of the business or recruiting from outside.
- Training – The training of resources in new skills to increase the number of job types they are able to cover.
- Loans – the temporary movement of resources between areas or parts of the business
- Overtime – Additional time added to a resources day, or bringing a resource in on a day they are rostered off. This lever is effected by the recruitment/reductions, training, and

loans levers in that they change the number of resources available which will modify the maximum constraint on the value for the overtime lever.

- Shrinkage – This models the expected absence of resources, whether unplanned through illness or planned through training, meetings, etc.
- Contractors – The deployment of external resource to increase capacity. This is usually attached to just a single skill and will just be directly deducted from the demand required for each skill contractors are attached to during the capacity-demand matching phase.

For the purpose of the problem explored in this thesis we are taking the productivity and shrinkage levers as fixed, using forecast values rather than opening them up as a planning decision. This is done because they are not usually something the planner has much practical control over. For example, a plan may dictate the completion of 10 jobs requiring a specific skill per hour, but the actual amount that will be completed are more dependent on the reality of the situation. A further point to note is that recruitment/reductions, training and loans all effectively are modelling resource movements. Recruitment/reductions models movement in and out of the business unit, training models movements of an amount of resource to another resource type with a different skill set, and loans model moving resource to another area. This can mean that these levers can often be combined for the purpose of model building.

The outputs from the planning process are the values for these levers along with the number of jobs of each type to be completed by each resource on each day of the plan.

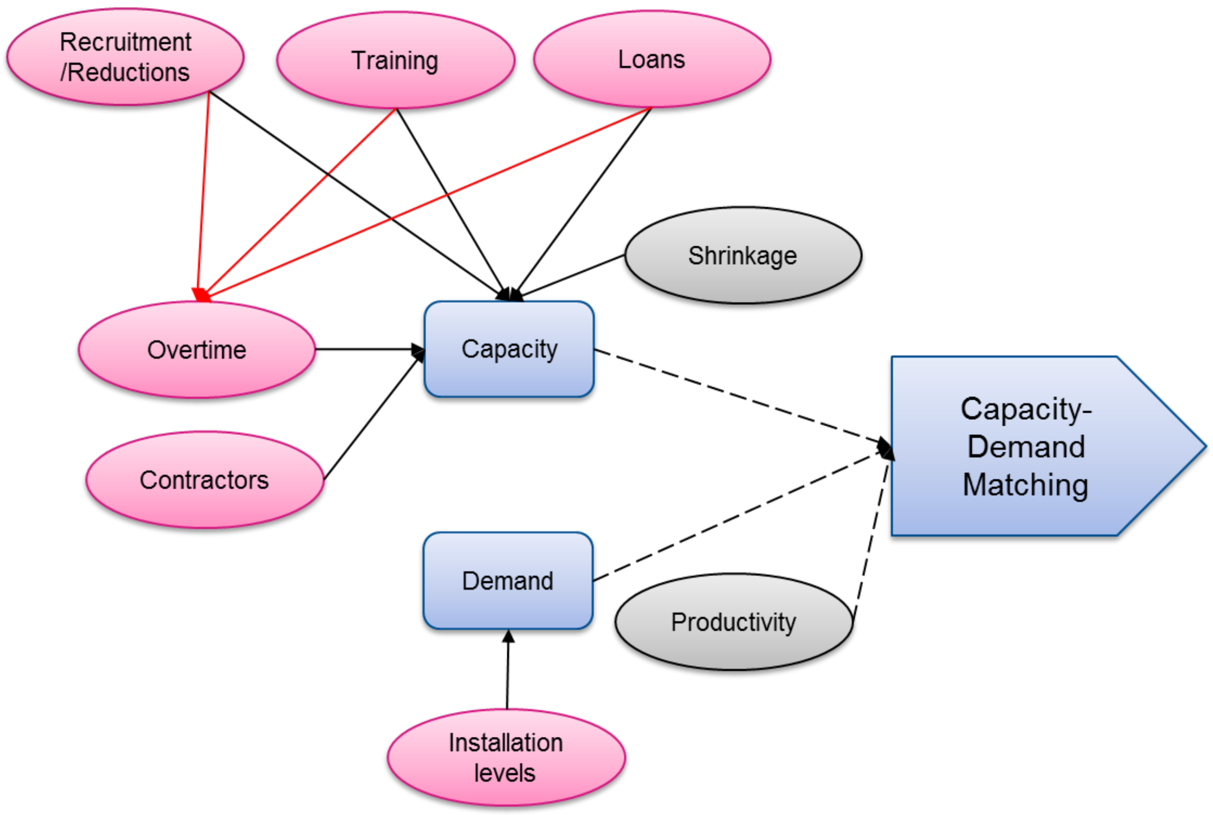

Fig. 2 Tactical planning decision levers

1.3 Case Study

For the purposes of this thesis some real planning data is utilised from a telecoms company with a large multi-skilled workforce. The data was obtained for 6 months in 2017 to provide real world evaluation of the models produced. 6 months was gathered to give some historical data when required. This is commercially sensitive data and thus is presented in anonymised form wherever it appears within this thesis. The data contains 52 different planning areas. The total number of resources number greater than 10000 and are spread across the 52 geographies. The total number of skills defined in the data are 13 - 6 fault skills and 7 installation skills. Three of those installation skills were able to field contractors. Within the data we have values for all the planning variables outlined in section 1.2 as well as some additional data on the achieved success rate for tasks. This data is utilised throughout the thesis and provides a motivation for the attempts to develop the models capable of handling the large scale real world data.

1.4 Research Aims

This work aims to investigate the complex problem that is the underrepresented tactical planning layer of the service chain planning process. In section 1.1 it has been identified that the tactical planning process defined in the literature does not exactly match that which is undertaken in practice and often is not even undertaken at all. This led to the updated definition provided in section 1.2. This research will explore this real world problem, initially defining it to effectively capture all of its features. Then methods are investigated to effectively solve it. During this work, real world considerations add further complexity, with computational time constraints as well as imperfect data. Ultimately we intend to show that undertaking this tactical planning process, more specifically by solving the problem effectively, has real world benefits for a member of the service industry. The research in the remainder of this thesis is driven by the questions picked out in 1.4.1 and guided by the objectives defined in 1.4.2.

1.4.1 Research Questions

(RQ1) What is an effective method of optimising a complex tactical planning situation within a service industry, such as a telecoms company, where there are a number of competing objectives and solution time is an additional objective?

(RQ2) How can we produce feasible real world solutions in typical situations where not all data-points are available in the required level of detail?

1.4.2 Research Objectives

To answer the questions posed in 1.4.1 this thesis has the following objectives. These objectives are shaped by the hypothesis that the complex tactical planning problem may be broken down into sub-problems that can be solved as linked problems.

(O1) Develop an initial formulation for the tactical planning problem to capture the key features

(O2) Determine if this could be modelled as a combination of linked sub-problems to improve solution time

(O3) Investigate suitable algorithms to solve these sub-problems

(O4) Develop a suitable framework to optimise the linked sub-problems

(O5) Investigate potential solutions to deal with the real world situations where not all data-points are available

(O6) Determine if these can be used within the linked optimisation framework to produce feasible solutions to the planning problem in real world scenarios

Here O1 to O4 are intended to answer RQ1 by creating a linked sub-problem model to solve the complex tactical planning problem. RQ2 will then be addressed by O5 and O6 as we aim to develop a real world ready model to utilise the linked sub-problem framework. It should be noted that O6 will also partly cover the solution time portion of RQ1 as a feasible real world solution to the planning problem will include the answer coming in a timely fashion.

1.5 Current Research Outcomes and Thesis Outline

During the course of this work various papers have been produced. As such some of the work seen in this thesis has appeared in the following publications:

- (Chapter 4) Ainslie, R.T., Shakya, S., McCall, J. and Owusu, G., 2015, December. Optimising Skill Matching in the Service Industry for Large Multi-Skilled Workforces. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence* (pp. 231-243). Springer, Cham.
- (Chapter 5) Ainslie, R., McCall, J., Shakya, S. and Owusu, G., 2018, July. Tactical Plan Optimisation for Large Multi-Skilled Workforces using a Bi-Level Model. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). IEEE.
- (Chapter 6) Ainslie, R., McCall, J., Shakya, S. and Owusu, G., 2016, July. Predictive planning with neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)* (pp. 2110-2117). IEEE.
- (Chapter 7) Ainslie, R., McCall, J., Shakya, S. and Owusu, G., 2017, December. Predicting Service Levels Using Neural Networks. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence* (pp. 411-416). Springer, Cham.

Some portions of this work have also already been deployed as practical solutions within the industrial partner. The outcome from chapter 6, the predictive planning model, has been deployed as a core component of a next generation tactical planning application that is used daily across the entirety of the UK. The service level prediction model created in chapter 7 is also being utilised to give a future view of potential service level impacting issues allowing decisions to rectify this to be taken ahead of time. The deployment of both of these have already had demonstrable impacts in performance that will be covered in chapter 9

The remainder of the thesis is arranged as follows. In chapter 2 we pick out bi-level modelling as a method to define our linked problems and identify one of those sub-problems as having features similar to allocation problems in the literature. These two fields are thus the focus of the literature review undertaken in this chapter. This is followed by defining the tactical planning problem mathematically to formalise all of the decision variables and constraints in chapter 3. Methods to solve the allocation sub problem are investigated in chapter 4 where we test a genetic algorithm vs. a linear solver as well as a standard planning algorithm from the literature. Chapter 5 we then utilise the linear model developed for the allocation sub-problem to create a linked bi-level model with a GA leader and linear follower to apply to the tactical planning problem. In chapter 6 we start to investigate methods to cope with real world scenarios where there may be incomplete data by developing a neural network model to solve the allocation problem. Chapter 7 we continue this work with real world scenarios to build a further NN that predicts expected success rates for tasks to allow evaluation of plan decisions. Finally we bring the work together by developing a surrogate follower model for the bi-level framework using the neural network models designed to handle real world scenarios in chapter 8. We then conclude the thesis and define future work in chapter 9.

Chapter 2

Literature Review

An initial search of the literature at the onset of this work was unable to uncover any precise match for the tactical planning problem defined in section 1.2. However, taking just the capacity-demand matching sub-problem, whereby only the capacity allocation and the task completion decision variables are considered, and comparing that to the literature it can be seen that it has elements of an assignment problem (Pentico, 2007). Sometimes, despite the same root mathematical formulations, when dealing with people these are also referred to as human resource assignment problems (Bouajaja & Dridi, 2017). More specifically this sub-problem is similar to a hybrid of the generalized assignment problem (GAP), where multiple tasks are assigned to agents and each agent has a capacity constraint (Cattrysse & Van Wassenhove, 1992), and the β-assignment problem, where not every agent has the skills/qualification to complete every task but agents are not limited by capacity (Chang & Ho, 1998). This similarity to GAP is to be expected as the capacity-demand matching problem has elements of a scheduling problem which is often modelled as a GAP in the literature (Cattrysse & Van Wassenhove, 1992).

The remaining elements of the tactical planning problem not covered by the capacity-demand matching sub problem are the decisions that modify the quantity of available capacity (e.g. application of overtime). Thus the overall tactical planning problem can be modelled as a combination of linked problems where the decision variables for the new problem are setting the capacity constraints that limit the solution space of the capacity-demand matching problem. Within the literature this closely resembles a bi level model. A bi-level optimisation problem being a class of problem that contains two optimisation models, one nested inside the other. The outer or higher level problem is often referred to as the leader, whereby the inner or lower level problem is designated the follower (Bard, 2013). Both models have their own decision variables, objectives and constraints and are linked through some (or all) of the leaders decision variables acting as parameters for the follower. In this case the leader decisions act as the constraints for the follower.

This chapter thus focuses on these fields, GAP and Bi-level problems, that are the focus of research for the remainder of this thesis. In section 2.1 the GAP literature is explored followed by that of bi-level problems in section 2.2. Section 2.3 then concludes with a summary of the findings from the literature and identifies the areas for research covered in the remainder of this work.

2.1 Generalized Assignment Problems (GAP)

Assignment problems (AP) are a class of problems first introduced in Kuhn's 1955 work published on the Hungarian method for their solution (Kuhn, 1955), as such they are quite a mature field. An AP is a problem where n agents are matched with n tasks. Each agent and task are matched in a 1-1 manner with the optimal solution being that which minimises the cost of these allocations. The mathematical formulation for this problem is shown in equation (1). Here $x_{ij} = 1$ denotes the allocation of task j to agent i, with c_{ij} being the related cost for that allocation, and $x_{ij} = 0$ meaning the task is not allocated to that agent.

$$min\colon \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \quad (1)$$

Such that:

I. $\sum_{i=1}^{n} x_{ij} = 1 \; \forall \, j$
II. $\sum_{j=1}^{n} x_{ij} = 1 \; \forall \, i$
III. $x_{ij} \in 0, 1$

Many variations to this base problem have appeared over the years. Some by adding additional constraints, such as in the k-cardinality assignment problem where only k of the tasks can be assigned to agents and k is less than n (Dell'Amico & Martello, 1997). Another example appears in (Caron, et al., 1999) where constraints were added to limit which tasks can be performed by which agent. Some other variations explored have been to the objectives. In the bottleneck assignment problem (Ravindran & Ramaswami, 1977) the objective is to minimise the maximum cost of all the assignments instead of the sum. This models situations where the problem is to minimise the time it takes all tasks to complete rather than the overall cost. Other variations to the objective are seen in the balanced assignment problem (Martello, et al., 1984), and the Σ_k assignment problem (Grygiel, 1981). In the former, the goal is to minimise the distance between the maximum and minimum assignment cost value, in the latter the goal being to minimise the sum of the k largest assignment costs.

However, the variation to the AP that most fits a portion of the tactical planning problem outlined in 1.2, namely the sub problem relating to the matching of capacity to demand, are those that involve a one-many allocation rather than the one-one of the base AP. This more closely fits with the capacity demand allocation portion of the tactical planning problem where each of the resources are allocated to multiple different jobs. The base version of this modification is called the Generalized Assignment Problem (GAP) with an early description in (Ross & Soland, 1975). In a GAP each task is still assigned to just one agent, however each agent can be allocated multiple tasks. These agents each have a total available capacity and each task an amount of capacity required for completion. This updated formulation can be seen in equation (2) below. Now there are m agents and n tasks with x_{ij} = 1 still denoting the allocation of agent i to task j, 0 meaning not allocated and c_{ij} the associated cost. Each agent now also has a total available capacity of b_i and each b_{ij} indicates how much capacity agent i would use if allocated to task j. Constraint one limits each task to being allocated to just one agent and constraint two states the total capacity used by tasks assigned to an agent cannot exceed the capacity available to that agent.

$$min\colon \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \ (2)$$

Such that:

I. $\sum_{i=1}^{m} x_{ij} = 1 \, \forall \, j$
II. $\sum_{i=1}^{m} a_{ij}x_{ij} \leq b_i \, \forall \, i$
III. $x_{ij} \in 0, 1$

The GAP is a combinatorial problem that has been shown to be NP-hard, thus solving large problems, such as the tactical planning problem, is generally not feasible using exact methods, requiring the application of heuristic approaches (Sahni & Gonzalez, 1976). Even finding just a feasible solution has been shown to be NP-complete showing the problems scaling issues (Fisher & Jaikumar, 1981). Despite that the GAP has been widely applied in the literature in a number of various areas. The principal concern for work in the field of GAPs during their application being the development of heuristic solutions to solve the NP-hard problem when problem sizes become too large for exact methods.

The remainder of this section explores the various applications of the GAP already explored and the different methods that have been applied to solve it. The section then concludes with a brief discussion of the current status of the GAP research and how that relates to the tactical planning problem explored in the remainder of this thesis.

2.1.1. GAP Applications

Many different application areas have been found where the GAP formulation can be applied. Some applications are in the field of network optimisation, in papers such as (Barbas & Marin, 2004) where the goal is to assign terminals to base stations to achieve optimal coverage or in (Bressoud, et al., 2003) which is using a GAP to optimise traffic routing through a network. Similar, yet more physical, versions of these applications can be found in transportation problems such as (Fisher & Jaikumar, 1981) where a GAP formulation was used to set the seed positions for a transportation network. Alternately GAP has been used to model the mix and size of the fleet used in a transportation problem (Golden, et al., 1984). Finally for transport, a GAP based approach can even been used to allocate passengers to flights such as for transporting patients in (Ruland, 1999).

The GAP has also been applied in many situations where the goal is the optimal setting or selection of locations for various elements. Once example being the optimal placement of public facilities solved in (Ross & Soland, 1980) where the objective is the minimal selection of sites such that each client can be served by one facility. Other location problems solved include the single source capacitated facility problem which is an extension whereby the capacity of the facility is also considered (Ahuja, et al., 2004) and the capacitated concentrator location problem whereby the goal is the placement of concentrated supply points such as distribution centres (Gouveia & Saldanha-da-Gama, 2006).

It has also been applied to many problems within production planning, such as machine assignment (Cheng, et al., 1996), storage layout (Lee, 1992), facility location (Ross & Soland, 1977), multi-period order selection (Lee & Kim, 1998) and batch loading and scheduling (Dobson & Nambimadom, 2001).

Some further application examples are also in the field of regional planning to optimise land use (Cromley & Hanink, 1999) or to minimise energy dissipation through the optimal allocation of tasks to processing elements (Yu & Prasanna, 2003). A GAP has also been used to optimally allocate distributed computing resources, such as database partitions to processors and processors to users in (Pirkul, 1986).

Some applications of GAPs in an area closer to the problem explored in this thesis are in the field of Supply Chain planning. Problems such as demand allocation seen in (Benjaafar, et al., 2004), where products are the tasks which are allocated to suppliers who are the agents with the goal to find the minimum cost allocation such that each product is coming from only one supplier and their capacities are not broken. Another stage in the chain, where customers are allocated to warehouses in a similar manner, can also be modelled as a GAP (Freling, et al., 2003). Optimising resource supply decisions has also been implemented as a GAP in (Higgins, 1999). However, as mentioned in chapter 1, although supply chain planning is somewhat analogous to service chain planning there are enough differences that current solutions do not directly translate.

An area GAP has been applied to in the literature that does resemble portions of the tactical planning problem, more specifically the capacity-demand allocation sub-problem, is scheduling. Scheduling problems are often modelled as a GAP in the literature (Cattrysse & Van Wassenhove, 1992), covering a wide range of fields. Applications such as in (Harvey, et al., 2006) to model a machine time allocation problem or for job assignment in (Drexl, 1991) or in (Nowakovski, et al., 1999) for scheduling the use of a space telescope.

Closer to resource allocation is in (Campbell, 1999) and (Campbell & Diaby, 2002) where shift allocation is modelled as a GAP to allocate workers to departments. For a given day multi skilled workers are allocated taking account of the demand levels in each department for that day and their capabilities with each skill. The goals being the maximisation of service and the satisfaction of the workers themselves. A specific example of its use is stated as the allocation of nurses to departments within healthcare.

2.1.2. GAP Solution Methods

A wide range of techniques have been applied to solve various different GAP formulations across the literature, from approximation methods through constraint relaxations and heuristics to exact methods.

Polynomial time approximation methods such as rounding based and local search algorithms have been used to solve a bin packing problem with bin capacities and variable benefits for adding items to different bins and a more specific practical application to distributed caching decisions in (Fleischer, et al., 2006). There are also approximation algorithms that take given solution costs or other objective values and return a solution, if there is a feasible solution, with that objective value or better. One such is applied to solve problems such as the machine allocation problem in (Shmoys & Tardos, 1993).

Greedy hill climber style algorithms, such as (Martello & Toth, 1981) where they allocate items to a knapsack one at a time. Each time it chooses the one with the biggest gap between the best and second best allocation location based on cost change of the solution. The GAP can also been remodelled as a set partitioning problem through the use of a column generation heuristic (Cattrysse, et al., 1994).

Various relaxation methods can also be deployed. A common example is linear programming relaxation, where specific constraints are removed to reduce the problem complexity. Some repair then occurs afterwards, such as applying a hill climber, to fix any of the constraints that are not being met in the relaxed solution (Wilson, 1997). Alternately a heuristic has been applied to allocate the split jobs to specific agents when an integer constraint has been relaxed (Benders & Van Nunen, 1983). This technique has also been expanded through the addition of a heuristic that removes the redundant variables one by one to create the linear relaxation (Trick, 1992).

Lagrangian relaxation and decomposition are two other relaxation style techniques. In the former some of the more difficult to handle constraints are removed and instead placed within the objective function and in the latter the problem is split into sub problems each containing a portion of the constraints (V, 2009). Lagrangian relaxation has been shown to be applicable to the GAP in work such as that by Fisher (Fisher, 1981) (Fisher, 2004), the main methods being by relaxing the capacity (Ross & Soland, 1975) or the semi-assignment constraints (Fisher, et al., 1986). Whereby examples of uses of the decomposition technique can be found in (Jörnsten & Näsberg, 1986) and (Barcia & Jörnsten, 1990).

These techniques have also been combined with heuristic algorithms to provide approximate solutions to more complex problems such as solving a capacity constraint relaxation using a sub-gradient heuristic in (Lorena & Narciso, 1996) or applying a two stage heuristic to a Lagrangian decomposition of the problem through the substitution of specific variables in (Haddadi, 1999).

Many standard metaheuristics have also been utilised to solve the GAP, such as:

- Tabu search in (Dıaz & Fernández, 2001). Also enhanced versions including improvements through the use of an ejection chain to add potential compound moves to the search in (Yagiura, et al., 2004) or the further addition of path-relinking in (Yagiura, et al., 2006).
- Simulated annealing in (Osman, 1995).
- Genetic algorithms in (Chu & Beasley, 1997). Including taking a constructive approach in (Lorena, et al., 1999).
- Greedy random heuristic in (Lourenço & Serra, 1998).
- Ant colony optimisation is also applied in (Lourenço & Serra, 1998).
- Variable depth search heuristics (Racer & Amini, 1994) (Yagiura, et al., 1999).

Finally exact procedures have also been used, such as branch and bound (Haddadi & Ouzia, 2004) and branch and cut algorithms (Pigatti, et al., 2005) though given the np-hard nature of GAP problems these are only applicable in cases where the specific GAP is not overly large.

None of the above were used in tackling particularly large problem sizes, at least not whilst also focussing on quick solution times. Some more recent work on GAP has been focussed on this area however with one example exploiting the potential parallelisation of some GA operations and evaluations to distribute the processing during the optimisation of a GAP across multiple processors (Liu & Wang, 2015).

2.1.3 GAP Discussion

It is clear from the literature that GAP is quite a mature field, having first arisen around 1975 as a special case of the assignment problem that had appeared in the literature about 20 years earlier. A vast range of solution techniques have been utilised to solve this problem ranging from exact methods through to heuristics and metaheuristics. As such it can be concluded that solution methods for the GAP are already well covered within the research making them a strong tool to apply to practical problems where they are applicable.

On the application side the GAP has been used in a wide range of fields over the years from allocation of physical resources such as base stations, through the placement of facilities and production planning to the more relevant applications in actual human resource allocations in scheduling problems. Although some problems covered are close to the demand-allocation sub problem of the tactical planning problem there isn't anything in the literature that could be found that falls within the tactical planning level of the service chain plan. The closest that could be found was either within the fields of supply chain planning or scheduling within the service chain plan. Thus the GAP shows promise in potentially being applicable to a sub-problem of the tactical plan whilst also being a novel new application for this approach.

2.2 Bi-level Optimisation Problems

A bi-level optimisation problem is a class of problem containing two linked optimisation models, one nested inside the other. The outer, or upper, problem is commonly referred to as the leader where the inner, or lower, problem is called the follower. Each level has their own decision variables, objective function and constraints. The leader's decision variables defined as x in decision space X, and the followers as y in decision space Y. The objective functions are then F(x, y) for the leader with constraints G(x, y). The followers corresponding components being f(x, y) for the objective function with g(x, y) for the constraints. The resulting linked problem can be seen in equation (3) (Bard, 2013).

$$\begin{aligned}
&\min_{x\in X} F(x,y)\\
&subject\ to: G(x,y) \leq 0 \qquad\qquad (3)\\
&\qquad\qquad \min_{y\in Y} f(x,y)\\
&\qquad\qquad subject\ to\ g(x,y)\ \leq 0
\end{aligned}$$

When searching for an optimal solution the leader first sets the values for x to attempt to minimise its objective function F. The follower now sets the values for y, given the now known values for x, to minimise its own objective function f(x y) subject to constraints g(x, y). These set values for y are now used to complete the calculation for the leader's objective function which gives the current objective value for the currently selected values for x as well as whether these combined with y meet the constraints G(x, y). The leader may then select different values for x and the process continues till the leader has minimised its objective function value.

Some additional complexity arises when, given a leader decision vector of x, there is more than one option for y that will minimise the lower level objective function f(x, y). This leads to two possible bi-level systems. One where the follower cooperates with the leader, in that given multiple options for y it will select the values that also minimises the leader fitness function. This is defined in the literature as an optimistic position (Dempe, 2002). The alternative is found when in that scenario the follower is non-cooperative, in that it could select any of the options for y that optimise the followers objective function f(x, y) but not necessarily the leaders function F(x, y). In this case the leader optimises for the worst case, where the y selected maximises F(x, y) while minimising f(x, y). This leads to this system being defined as a pessimistic position in the literature (Wiesemann, et al., 2013). The optimistic position is generally easier to handle and guarantees the existence of an optimum solution in a wider number of cases than the pessimistic position (Dempe, et al., 2007). For the remainder of this review we will be focussing on the optimistic position as this would be the case in a bi-level formulation of the tactical planning problem covered in this thesis.

Even when limiting to the optimistic case however the bi-level problem has been shown to be strongly NP-Hard (Hansen, et al., 1992). Not only that, but due to the nature of the linked problems even proving a solution is local optimal is NP-Hard (Vicente, et al., 1994). Thus, although the bi-level formulation maps well to many real world applications the principal concern in the field is developing algorithms and techniques to overcome the complexity involved in solving the model.

In the remainder of this section some current applications of bi-level models will be highlighted showing the applicability of bi-level models to real world situations. This is followed by exploring the current solution methods from the literature, showing the current efforts to solve this complex problem as well as picking out a new potentially emerging area in the use of surrogates. Finishing with a discussion to summarise these findings and identify areas applicable to this work.

2.2.1 Bi-Level Model Applications

Many real world scenarios have been modelled as a bi-level problem. The linked yet separate problems, where one element does not necessarily have full control over all aspects, e.g. in a decentralised situation, can be found in a lot of real world scenarios such as economics, management, and logistics, to name a few. In the remains of this section some of the examples from the literature are highlighted with a particular focus on those close to the tactical planning problem towards the end of the section.

One application of bi-level modelling is in network design, including in the analogous areas of computer and road networks. Examples being that of (Camacho-Vallejo, et al., 2015) where the problem of defining a topological network by grouping clusters into groups and allocating users to clusters within those groups is modelled as a bi-level problem.

In (Ceylan & Bell, 2004) traffic light timings are optimised using a bi-level model where the follower optimises the traffic flow based on the current timings set by the leader, similarly in (Chen, et al., 2010) the leader models decisions of the road network planner with the follower modelling the users of the network planning their journeys where the planner is optimising for safety, cost, congestion, etc. and the users of the network try to plot the cheapest routes in the network. Some other similar examples are in transit priority allocation in (Mesbah, et al., 2011) where the objective is setting the road priority between private cars and other transit methods and in (Yamada, et al., 2009) to find solutions for a multi-modal transport network.

A further frequent application in the transport field found in the literature is that of using bi-level models to solve a class of problem called toll-setting problems. Here the goal of the leader is to set tolls in private roads within the network with various goals such as revenue or traffic control while the follower, as with most transport problems, acts as the users of the network attempting to minimise their costs whilst also taking account of travel times. These can be seen in papers such as (Kalashnikov, et al., 2016) where the leader is the regulator setting tolls and the follower is the transportation companies using the network. Or in (Sinha, et al., 2015) with more general road users as the follower. Further objectives are sometimes also considered such as environmental impacts or health impacts in the work of (Wang, et al., 2014).

Environmental impacts are also the key consideration in a number of bi level applications based around regulation, taxing and fine setting. Examples of this are (Whittaker, et al., 2017) where the leader is setting environmental policy and the followers attempt to comply whilst also targeting their own financial objectives. Further examples are in the water industry where the trade-off between water quality and agriculture is modelled in (Bostian, et al., 1-12), or for mining regulation where the leader tries to maximise tax income while minimising the environmental impact of the mining companies modelled as the followers (Sinha, et al., 2013).

Similarly to the GAP literature, bi-level models have also been applied to facility location problems where additional features are accounted for, such as the location rivals may choose to place their own facilities (Küçükaydin, et al., 2011), or the addition of competitive pricing decisions as in (Panin, et al., 2014). Another example of a location problem in the literature is that of using a bi level model to solve the ring star problem where facilities are placed on some nodes and the remaining nodes are linked to these, such as in concentrator placement in a network or distribution centre locations. This has been solved using a bi level model where the leader selects the locations and the follower then produces the optimal path to link up the remaining uncovered nodes to those locations (Calvete, et al., 2013). The capacitated facility location problem has also been modelled as a bi level problem in (Caramia & Mari, 2016) with the additional features over the GAP version being the splitting into two decision makers with the leader deciding on facilities to open and their capacities and the follower then attempting to make optimal use of those capacitated facilities to give an overall solution that is maximising profits.

Another class of problems that translates well into a bi-level model is the principal-agent problem class (Cecchini, et al., 2013). In these problems the principal does not have direct control over the agent's actions, instead the agent will act to benefit themselves as much as they can. In this case the principal just wields indirect control through use of incentives. This has many practical applications such as that between a doctor and their patient, or employers and their employees, or when a firm subcontracts to another firm (Van Ackere, 1993).

Further applications where as well as not having direct control over the other party bi-level models are also used in cases where the other party is actively attempting to defeat or harm the decision maker who is looking to optimise defence against various situations. This could be in the literal sense where bi-level models have been implemented to optimise missile interception from an attack. The leader is placing missile defence platforms while the follower attempts to maximise damage of an attack (Brown, et al., 2005). However the same approach can be used to guard against worst case scenarios such as optimising network coverage to guard against failures where the leader is building the network and the follower is selecting the optimal x facilities to fail to maximally disrupt that coverage (O'Hanley & Church, 2011).

Bi-level problems are also commonly found in design tasks, such as structural design where the leader is producing the design of the object with the follower then minimises the potential energy problem to allow the calculation of the stresses and loads to test the breaking of the material property constraints defined for the leader (Christiansen, et al., 2001). As well as in structural design bi-level models have also been applied in the field of chemistry where the goal is the optimal reaction or outcome from some chemical process. The leader is controlling the quantities and mixes of chemical components and the follower is an optimisation problem that simulates the resulting chemical reaction (Halter & Mostaghim, 2006).

A further application of bi-level modelling is within the field of model development itself. Metaheuristics and machine learning models require tuning of certain parameters to achieve optimal results. The combinations for these are often vast thus this has motivated some applications to utilise a bi-level model for this purpose. Here the leader controls the parameters and the follower is the model being tuned with the goal to achieve the optimal results from the resulting model. An example of a bi-level model used to tune a machine learning model can be seen in (Bennett, et al., 2008) with a corresponding example for an optimisation algorithm in (Sinha, et al., 2014). In a more practical sense, similarly this approach has been applied to reverse optimal control problems whereby the goal is to generate the best reward function for a given dataset for use in training robotics and similar fields (Suryan, et al., 2016).

The final application area found in the literature, and that of most interest for this thesis, is that of supply chain planning as this is close to the field of service chain planning in which the tactical planning problem resides. Most examples in the supply chain management area were covering facility location already explored above but there were a few examples of bi-level models being implemented for planning and scheduling.

One example is a machine scheduling problem introduced in (Lukač, et al., 2008). Here the leader is scheduling products to machines. Each machine can only handle one product at a time and switching products incurs an additional set up cost. The leader tries to minimise this set up time while the followers are optimising the output and storage associated with each machine. The paper cites a real world application in the pharmaceutical industry however the model developed only contains 2 machines potentially indicating an issue with scaling as the problem gets larger.

Another example is the integration of the planning and scheduling processes in (Chu, et al., 2015). In this paper the supply chain planning problem is modelled as the leader with the schedule the follower. The leader sets the planned quantities for production in the various time periods in the plan and the follower produces a schedule for each period that attempts to meet these targets. The goals of the leader are to set production levels such that they maximise economic gain through meeting customer demand. The goal of the follower is to make optimal use of the machinery available to generate the planned production levels. Again however, the case study example is not that large. The leader has 10 decision variables per period for the 10 products with 12 periods in the plan for 120 variables. Each follower also only has 10 units on which to schedule these products within their 7 days per planning period.

2.2.2 Bi-Level Model Solution Techniques

Bi-level models are complex to solve due to them containing two separate but linked optimisation problems that must be optimised together. The most common methods found in the literature solve this issue by merging the leader and follower into a single model in a process called dimension or level reduction (Rangarajan, 2010). This allows any standard solution techniques for optimisation problems to be applied so long as their requirements are met (e.g. continuous, combinatorial, etc.). This does not work in all cases however as sometimes problem complexities or size make it infeasible to solve this combined single model. In this case the nested approach is used whereby the follower model is solved inside the leader model for each evaluation of different leader solutions. This approach has its own issues when dealing with large sized problems however, since the follower model must be solved for each and every leader evaluation (Oduguwa & Roy, 2002). These evaluations can be computationally intensive and cause solution times to quickly become infeasible. More recent work in the area has begun attempting to mitigate this issue through the use of meta- or surrogate models to replace parts of the bi-level problem.

This section will first give an overview of dimension reduction solution techniques, before then covering the nested approach. Finally some recent work on surrogates is highlighted that links into the work then carried out in this thesis.

2.2.2.1 Dimension Reduction

The most common techniques found in the literature to solve the bi-level problems are that which involve dimension reduction. In this case some method is used to combine the leader and follower to turn the bi-level problem into a single level problem, with a range of different approaches taken. Once the single level version is created this is then solved using any range of available techniques.

The most frequently encountered method to achieve this is through the use of the Karush-Kuhn-Tucker (KKT) conditions of the follower model. In cases where the problem space is sufficiently regular the KKT conditions can be derived such that they are the necessary conditions for a solution to that problem to be optimal. When derived for the follower model these can then be added to the leader model as additional constraints to merge the two models together, reducing the dimensionality of the problem to a single level problem. This problem can then be solved using a number of different techniques.

Some examples seen in the literature involve using standard linear programming techniques such as branch and bound (Shi, et al., 2006) or complimentary pivot algorithms (Önal, 1993). Another common approach is to exploit the fact that an optimal solution of the problem will fall on a vertex of the constraint space by finding a solution using vertex enumeration (Tuy, et al., 1993).

Single level reduction does not always guarantee an easily solvable problem however as the initial problem may have already contained a leader with decision variables or constraints that are not regular or the process of reduction may have generated a problem with a non-convex solution space even if both source problems were themselves convex (Shi, et al., 2005). In these cases standard techniques cannot be deployed.

One solution found in the literature is to remove the constraints and instead replace them with penalty functions. The resulting unconstrained optimisation is easier to solve and the search is directed towards valid space through the use of the penalty term. This term would be zero for valid solutions but any that are breaking constraints would incur a cost, positive for minimisation problems and negative for maximisation. This technique is combined with the KKT approach to dimension reduction to generate solutions in one of two ways. In (Lv, et al., 2007) the problem is first reduced to a single level using the KKT conditions of the follower before the constraints of the new single level problem are replaced by the penalty function and the resulting problem solved. The alternate approach found was to first replace the constraints of the leader and follower with the penalty functions and then use the KKT approach to reduce the simplified leader and follower into a single level such as in (Ishizuka & Aiyoshi, 1992).

Another was through the use of a trust region to approximate a portion of the objective function. A trust region that was itself a bi level model was used to approximate a bi level model in (Colson, et al., 2005), they then reduced it to a single level using the KKT approach and solved that to produce a solution to the problem.

Finally in cases where the leader problem's decision variables or constraints are not regular, and thus the resulting single layer problem is also not regular, then evolutionary algorithms such as a GA (Wang, et al., 2008) or particle swarm (Wan, et al., 2013) have been utilised to find a solution to the reduced dimension problem using KKT conditions.

A few alternate methods for dimension reduction have also been explored, such as including the follower in the leader's objective function through use of its penalty term (Anandalingam & White, 1990) and in (Ye & Zhu, 2010) where the follower is replaced by its optimal value function. The optimal value function is one which takes the leaders decision variables as a parameter and returns the corresponding optimal objective value from the follower. The follower can then be replaced in the leader by a constraint stating that value of the followers objective function must be less than or equal to the optimal value. In cases where the optimal value function is not known, an approximation can be utilised instead based on current members of the population when solving using an evolutionary approach such as in (Sinha, et al., 2020).

2.2.2.2 Nested Approach

The alternate to dimension reduction is the nested approach where the levels of the problem are left intact and the leader and follower are solved together as separate but linked problems. This involves solving the lower level problem for each different higher level solution being explored (Sinha, et al., 2014) to obtain the followers objective value decisions for use in the objective function and constraint calculations of the leader. This can require intense computation however, especially for large problems, due to the number of times the lower level model is evaluated. However as the models for the two levels remain separate they can have different solution methods applied. This can allow exploitation of specific features of the leader and follower models. There are generally two approaches to solving using the nested approach. In the first the leader is solved using an evolutionary algorithm while the follower is solved using an exact method. In cases where the follower is also suitably complex however then the nested approach requires solving both the leader and the follower using evolutionary methods.

A representative example of an evolutionary algorithm that is utilised a lot in the literature as the leader model in nested solution methods for the bi-level model is the genetic algorithm. In a genetic algorithm based solution the follower is imbedded within the fitness function with solving this problem becoming part of the fitness evaluation for the leader. As discussed this is computationally intensive however, particularly in a pop based algorithm like the GA since there will be a large number of these calculations required for each and every generation (Mesbah, et al., 2011).

As such, the follower has often been a simpler problem in these approaches, such as the GA leader with a linear follower (Mathieu, et al., 1994) where the follower was reasonably simple with a single constraint per decision variable and the largest test number of decision variables was 17. In this case solving the linear follower was relatively quick and didn't impact the solution evaluations of the leader GA too much.

Another common option paired with a GA leader is a GA follower as in (Sinha, et al., 2014). Here the follower GA is solved for every evaluation of a leader solution as such the paper only handles smaller problem sizes up to around 40 variables. Even so the function evaluations for the follower models are shown to be vast, reaching into the 10's of millions for only thousands of leader evaluations.

Some work around improving performance when using a GA leader with more complex followers have evolved around utilising simpler heuristics at the follower level, where the problem is too complex for an exact method, or attempting to utilise some feature of the problem to guide the algorithm. An example of the former is seen in (Camacho-Vallejo, et al., 2015) where the GA leader contains a greedy constructive heuristic as a follower. In this case the problem is optimising a LAN configuration with the GA leader attempting to minimise connection cost while the constrictive follower is building paths minimising message time. Another example of utilising a simpler follower to reach a solution is also seen in (Yin, 2000) where a gradient descent algorithm is used to solve the follower for each evaluation of the GA leader.

An example of utilising the features of a problem to assist the algorithm is seen in (Li & Wang, 2007) where the solutions to the follower model are utilised in a hybrid simplex method crossover operator within the GA leader to better guide the evolution of the population towards optimal solutions.

Other methods used to model the leader seen in the literature are in (Li, et al., 2006) where a particle swarm (PSO) model was used as both the leader and the follower to solve various toy problems. The algorithms performed well in the tests but there was no analysis performed on their computational performance. Or in (Camacho-Vallejo, et al., 2015) where a scatter search algorithm was utilised as the leader with linear followers solved using CPLEX.

Another evolutionary technique seen in the literature is that of differential evolution (DE). An example of this is (Angelo, et al., 2013) where a DE was used as both the leader and follower to solve test problems from the literature. A DE has also been utilised as leader in (Zhu, et al., 2006). In this case the follower was implemented using an interior point algorithm with the authors able to utilise testing the follower problem for solvability to screen out infeasible solutions and reach some performance improvement.

Finally, a mix of evolutionary algorithms has also been used in some situations in order to take advantage of specific features of the problem. A specific example of this is in (Angelo & Barbosa, 2015) where they solved a distribution planning problem through use of an ant colony optimisation model as the leader to take advantage of their advantages in solving problems involving route building and then a DE at the lower level to provide solutions to the follower problem.

Despite often being computationally expensive the performance of the nested approach can also be improved through parallelising the lower level solution evaluations or by further exploiting the separate model approach to replace the lower model with a surrogate. These are covered in the next section.

2.2.2.3 Surrogates

Surrogates, also referred to as meta-models, are often used where problems are too large or complex making them too computationally expensive to solve in a reasonable time frame. This situation frequently arises when dealing with real world problems where problem sizes are often much larger than the toy problems used in academia (Wang & Shan, 2006). To solve this problem computationally expensive processes or calculations are replaced with a surrogate or meta-model that is easier to compute. These models effectively act as approximations or predictors for the behaviour of the more complex model they are built to replace. The surrogate is generally trained using samples from the model they are replacing, however in cases where that model is more complex the surrogate is often updated as the search progresses. They have been widely used already in various fields such as simulation, process modelling and control, parameter estimation and optimisation to name a few (Bhosekar & Ierapetritou, 2018). Some common types of surrogate models utilised in these areas include:

- polynomial response surface (Hosder, et al., 2001) – Model based on the least squares
- Kriging (Hu & Mahadevan, 2016) – An approximation model for a function based on the weighted average of points local to the value being calculated
- Radial basis function (Ozcanan & Atahan, 2021) – based on a system of linear equations
- Support vector regression (Clarke, et al., 2005) - Supervised learning algorithm with configurable acceptable error levels when fitting to the training data
- Artificial Neural Network(ANN) (Fyfe, 2005) – A model of a group of nodes connected with attached transmission functions and weights that are tuned to fit training data using a learning algorithm

For the purposes of this literature review the focus is on their application to bi-level models. However, despite the prevalence of work done on surrogates, utilising these to reduce the computational complexity of bi level models has been relatively scarce in the literature (Islam, et al., 2017).

The main use of surrogate methods used in bi level modelling that could be found in the literature is that of reaction set mapping. The reaction set for a bi-level model is the mapping between the leader's objective values and the resulting follower objective values upon solving the follower objective function with those given leader objective values. If that entire set was known then there would be no need to calculate the follower objective function at each iteration and thus the problem would effectively collapse to a single level. However in any practical problem it is not possible to know the entire reaction set in advance. Thus the reaction set is approximated by a surrogate. This surrogate is iteratively updated as the optimisation progresses.

The current techniques found that have been used to build a surrogate model for the reaction set mapping in the literature so far are that of k-NN approximation, such as used by (Angelo, et al., 2014) or quadratic approximation, such as in (Sinha, et al., 2017). In both cases, when evaluating an offspring during evolution if there are enough members nearby, through some distance measure, that have calculated their fitness fully by optimising the follower then the surrogate for the reaction set mapping is used, if not enough then the follower is optimised fully. In this way the surrogates are updated as the evolution progresses whilst computational time is saved whenever the surrogate is used instead of the follower's objective function.

2.2.3 Bi-Level Model Discussion

In this section bi-level models were explored with their two level behaviour effectively modelling many real world applications, as evidenced by the wide range of problems that have currently been tackled in the literature. Applications for this technique in real world problems are varied such as the road and computer network design, environmental regulation, facility location, to name a few. This list also including the field of supply chain management. The latter of these is one of the motivating factors behind exploring bi level modelling to apply to the tactical planning problem in this thesis due to the similarities between supply and service chain management.

Many challenges remain in this field however brought about primarily by the inherent complexity involved in solving two connected models together, particularly in real world scenarios where large data sets or problem sizes are often involved. Standard solution techniques involving reducing the bi-level problems dimensions to a single level are generally not applicable in these scenarios where the problems are already overly complicated. The nested approach show some promise with the ability to bring separate techniques to the different parts of the problem as well as introducing opportunity for parallelisation. There is also opportunity for novel work in the area around the use of surrogate models within portions of the bi-level process as this still seems fairly rare in the literature.

Finally, as with GAP, although this has been applied to some similar problems, such as in service chain management, there are also no examples of bi level models being used for the tactical planning problem within service chain planning.

2.3 Conclusion

In this section an overview of the literature was conducted, focussing on the areas of GAP and bi-level problems that will be the focus of the remainder of this thesis.

In the field of GAP it can be seen that although this is a mature field there is still scope for some new work there in applying it to a new field in the case of the tactical planning problem. The selection of the GAP for this purpose can also be seen as a logical decision based on the similarity between the demand allocation sub-problem of the tactical plan and a scheduling problem. These have been covered by GAPs already in the literature so should also be applicable to that sub-problem. This is explored in chapter 4 where the capacity demand sup problem is picked out and a GA and linear model are used to solve it. These were chosen due to them both being common solution methods seen in both the GAP and bi-level literature.

From the bi-level literature, although also a reasonably mature field with a wide range of applications and solutions already demonstrated, there are still some key gaps in the knowledge. Issues remain about computational complexity, particularly when applying to real world problems. Again the technique has not been deployed in the tactical planning phase of service chain management. Some applications can be seen in the analogous supply chain field but those solutions are not directly applicable due to the differences between the fields. The closeness though does motivate the exploration of bi-level modelling as a potential solution for this problem and guarantees some novel results will be produced. Chapter 5 covers this, where a bi-level model is investigated as a potential solution for the tactical planning problem.

The final piece of interesting work that can be approached is to explore methods to potentially mitigate, at least partially, some of the computational complexity issues of the bi-level optimisation process. Surrogate models have shown promise for this in other areas and are as yet not very common in the bi-level literature. This, along with the issues with incomplete data discovered, motivates their investigation in chapters 6 and 7 and finally integrated into the bi-level model in chapter 8 to test their impact on performance there.

All of this work is also carried out in a real world problem utilising real world data that produced some real world novel impact in the production of optimisation algorithms that have been deployed in the heart of a couple of applications being used across the UK.

Chapter 3

Tactical Planning Problem Formulation

3.1 Introduction

The first step to solving the very large tactical planning problem defined in section 1.2 involves formally defining it mathematically. This involves the initial capturing of the decision variables (or levers) available in the planning model and the defining of their constraints. Once the model is defined then solution algorithms, either exact or heuristic, can be applied to find the optimal (in the case of exact) or near optimal solutions to the problem. In this chapter we model the problem using linear programming notation as it is commonly used to solve similar problems. Additionally it is a good way to mathematically define the tactical planning problem, given it involves multiple decision levers that are subject to constraints in their use. The fixed levers from the tactical planning model, namely shrinkage and productivity, are simply treated as constants and do not require modelling here.

The remainder of this chapter will progress as follows. First the overall framework is defined outlining the overall costs in section 3.2. Then section 3.3 expands the resource costs to add the resource based decision variables and their constraints. Section 3.4 then expands the demand cost portion of the equation, adding the demand decisions and constraints. The remaining constraints are defined in section 3.5 where the model is completed. This is followed by a brief analysis of the problem complexity in the context of the applied problem in section 3.6. Finally the chapter concludes in section 3.7.

3.2 Model Framework

The overall model framework is rather simple in concept.

First, calculate the base cost, this is a constant. It includes the cost for resources, and the cost if no tasks are completed. Resources represent members of the workforce which have a set of skills, where having a specific skill indicates this resource has the capability to complete tasks requiring the matching skill to complete. Each skill a resource has is given a specific cost or efficiency with resources generally starting allocated to their cheapest or most efficient skill, this is called their primary skill. Tasks are then jobs which require a specific skill to complete and thus can only be fulfilled by resources with a matching skill. Both resources and tasks also have an area which indicates a work area in which they are located. Resources generally can only complete tasks within the same area with an additional cost incurred to pick up tasks in different areas. Each resource has an amount of time available, sometimes called capacity, and each task has an amount of time required to complete. Multiple tasks can be assigned to the same resource so long as there is still time available to use.

Then, for each period in the plan, add any additional costs incurred by resources:

- Skill moves - where resources time is applied to a skill other than their primary.
- Area moves – the cost associated with moving a resource away from their current location to work in a different geographical area
- Recruitment – the introduction of new resources with a specific skill set
- Training – adding an additional skill to the set of skills a resource is able to complete
- Wastage/reduction – the reduction in the number of resources with a specific skill set
- Overtime – additional time applied to a specific resource usually at a premium cost

Finally subtract the cost of any tasks completed as well as adding any additional costs incurred. Each task has a due date which is the period in the plan in which it should be completed, e.g. have the time allocated to it. Costs can be incurred by doing this task early or late by moving the task earlier or later. There are also costs for failing to meet the tasks service level agreement (SLA). This is an agreement whereby a certain percentage of that task have to be completed no more than a certain number of days late. The percentage and the allowed number of days both vary depending on the specific agreement.

Mathematically, let t, an integer, denote an individual period of the plan and T, a natural number, the total number of periods. Here a period being an individual chunk of time. For example in a daily plan where each period indicates a day then period 0 would be today and period 1 would be tomorrow. Each individual resource is denoted by r, an integer index for a resource from the resource set. R is a natural number indicating the size of that set. Finally j is the integer index of a task out of natural number J total tasks. Using these we define the initial framework as:

$$min\colon \sum_{t=1}^{T}\left[\sum_{r=1}^{R} c_{rt}\right] + \sum_{j=1}^{J} k_{jt} \quad (4)$$

Where c_{rt} is a real number denoting the additional costs incurred by resource r in period t (e.g. through overtime or skill/area moves) and k_{jt} is the real number that denotes the costs associated with the task type j completed in period t.

3.3 Additional Resource Costs $\boldsymbol{c_{rt}}$

In this section the additional costs incurred by resources within a period are expanded. This includes additional costs for the resource time allocation, overtime assignment, recruitment, reduction, and training decision variables. Initially the time allocation and overtime decision variables are added in 3.3.1 including a refactoring of resources into groups to reduce the number of decision variables. This is followed by the addition of the recruitment and reduction decision variables in section 3.3.2, and training in section 3.3.3.

3.3.1 Resource time allocation

First step in defining the model is designing how to represent a particular allocation of a resource's available time. In this sub-section how to model the allocation of a resource's base time and their overtime is first defined in 3.3.1.1 and 3.3.1.2 respectively. In 3.3.1.3 a concept of "resource groups" is created to reduce the number of variables produced by this allocation by aggregating similar resources into buckets. This is followed by a refactoring of the allocation and overtime decision variables to fit the resource groups in 3.3.1.4 before ending this sub-section with a final point on the restrictions from this grouping in 3.3.1.5.

3.3.1.1 Base Resource Time

The first component to model is the allocation of a resource's time to a skill, integer index s, within an area, integer index a. As a first step we define a decision variable for each combination a resource could cover.

For example, if a resource could perform 3 different skills and could work in 3 different areas, 9 decision variables are produced (one for each combination of skill + area).

Let A be the natural number of areas and S the natural number of skills, then this gives the cost for the allocation of resource r in period t as:

$$b_{rt} = \sum_{a=1}^{A} \left[\sum_{s=1}^{S} \omega_{ras} x_{rast} \right] \quad (5)$$

Where x_{rast} is the decision variable indicating the real value amount of resource r's time that is allocated to perform skill s in area a in period t. ω_{ras} is the real value cost associated with the use of that resource in this skill and area.

This requires a related constraint:

I. $\sum_{a=1}^{A}\left[\sum_{s=1}^{S} x_{rast}\right] \leq \sigma_{rt} \; \forall\, r, t$

Where σ_{rt} is the real value amount of time available to resource r in period t. Essentially this is limiting the total amount of time drawn from a particular resource to be less than the amount of time that resource has available.

At this stage we need to also start to construct the constraint that links our used capacity to the met demand.

II. $\sum_{r=1}^{R}[\epsilon_{ras} x_{rast}] - D_{sat} = 0 \; \forall\, a, s, t$

Here we introduce a new real value constant, ϵ_{ras}, which indicates the efficiency of resource r, using skill s in area a. This is used to model a resource taking more time when using an unfamiliar skill or losing time to travelling when being utilised outside their primary area. The second factor, D_{sat}, is there to indicate the total time used meeting demand of skill s in area a in period t. This variable is expanded in section 3.3 when the demand side of the equation is defined. Currently this is a real valued placeholder for the expansion done in that section.

This constraint is therefore specifying that the total time drawn from resources for a particular skill in a particular area in a particular period has to match the total time used on the same type of tasks.

Putting this all together, by the end of this sub-section we now have the model:

$$min\colon \sum_{t=1}^{T}\left[\sum_{r=1}^{R}\left[\sum_{a=1}^{A}\left[\sum_{s=1}^{S} \omega_{ras} x_{rast}\right] + c'_{rt}\right] + \sum_{j=1}^{J} k_{jt}\right] \quad (6)$$

Such that:

I. $\sum_{a=1}^{A}\left[\sum_{s=1}^{S} x_{rast}\right] \leq \sigma_{rt} \; \forall\, r, t$

II. $\sum_{r=1}^{R}[\epsilon_{ras} x_{rast}] - D_{sat} = 0 \; \forall\, a, s, t$

Where c'_{rt} indicates the real value sum of remaining additional resource costs other than the cost of their allocation. The notation to expand it to define overtime, recruitment/reductions, and training decisions is developed in subsequent sections.

3.3.1.2 Overtime

To model overtime use for a resource we have to take account of the two main constraints. Firstly, there is a total limit of overtime allowed across the entire model. Secondly, there is an individual limit applied to each resource, e.g. a resource may only be allowed to have 8% additional overtime on top of their standard hours. Finally the overtime decision variables are required with the cost associated with using overtime for the associated resource.

To add the decision variable a similar process as the resource time allocation is followed. A decision variable is generated for each skill a resource has in each area they can cover for each period of the plan to indicate overtime use by that resource on that skill in that area in that period. This gives the cost equation for overtime use on a specific resource as:

$$o_{rt} = \sum_{a=1}^{A} \left[\sum_{s=1}^{S} [(\omega_{ras} + V) x'_{rast}] \right] \text{ (7)}$$

Here the additional cost, V, is the real value cost for the use of overtime and x'_{rast} is the real value decision variable indicating allocation of overtime for resource r to skill s in area a in period t.

To take account of the overtime constraints, first a limit to the overtime use by a single resource is added:

III. $\sum_{a=1}^{A} \left[\sum_{s=1}^{S} x'_{rast} \right] \leq \vartheta \sigma_{rt} \ \forall \, r, t$

Here ϑ is the real value proportion of a resources time that they are allowed as additional overtime, thus multiplying it by their time available in period t, σ_{rt}, gives the maximum overtime they have available in period t ($\vartheta \sigma_{rt}$). This essentially constrains the total overtime used by a resource within a period to be less than or equal to the maximum overtime they have available.

Along with this we need to also constrain the total overtime used by all resources:

IV. $\sum_{a=1}^{A} \left[\sum_{s=1}^{S} [\sum_{r=1}^{R} [\sum_{t=1}^{T} x'_{rast}]] \right] \leq \theta$

Here $\theta$is the real value indicating the maximum overtime budgeted for the entire plan. This constraint effectively limits all overtime allocated within the objective function to be less than or equal to the maximum allowed.

Finally, we need to modify constraint II to take account of the extra capacity for a skill within an area drawn from overtime:

II. $\sum_{r=1}^{R}[\epsilon_{ras}(x_{rast} + x'_{rast})] - D_{sat} = 0 \; \forall \; a, s, t$

As before this states that total capacity drawn for a skill within an area within a period must equal the total demand fulfilled for the same. In this case the capacity drawn now includes that drawn from overtime also.

Adding this overtime into the main equation we now have:

$$min: \sum_{t=1}^{T}\left[\sum_{r=1}^{R}\left[\sum_{a=1}^{A}\left[\sum_{s=1}^{S}[\omega_{ras}x_{rast} + (\omega_{ras} + V)x'_{rast}\,]\right] + c''_{rt}\right] + \sum_{j=1}^{J}k_{jt}\right] \quad (8)$$

Such that:

I. $\sum_{a=1}^{A}\left[\sum_{s=1}^{S} x_{rast}\right] \leq \sigma_{rt} \; \forall \; r, t$

II. $\sum_{r=1}^{R}[\epsilon_{ras}(x_{rast} + x'_{rast})] - D_{sat} = 0 \; \forall \; a, s, t$

III. $\sum_{a=1}^{A}\left[\sum_{s=1}^{S} x'_{rast}\right] \leq \vartheta\sigma_{rt} \; \forall \; r, t$

IV. $\sum_{a=1}^{A}\left[\sum_{s=1}^{S}\left[\sum_{r=1}^{R}\left[\sum_{t=1}^{T} x'_{rast}\right]\right]\right] \leq \theta$

Where c''_{rt} indicates the remaining resource costs yet to be modelled. There are now just recruitment, reductions, and training.

3.3.1.3 Resource groups

Up to this point, resources were thought of as individuals. Each resource was modelled individually. However not only is this inefficient (it creates a large number of variables) it also makes it difficult to model the concept of recruitment and training. In order to allow for the modelling of the final resource costs we instead need to start thinking about resources as groups.

In this definition, resource group g is the integer index of the set of resources with a particular ordered list of skill preferences. E.g. if two resources both have skill1 as their primary skill and skill2 as their secondary skill, then they would be described as both being of the same resource group. A further refinement to this is to say that area group g_a is the integer index of the subset of resources from resource group g whose primary area is a.

We now have the tools available to model recruitment of a resource belonging to a specific group in a specific area. With these changes in place we can now further refine the model in an attempt to reduce the number of required variables before moving on to completing the resource costs section.

3.3.1.4 Base time and Overtime refactoring

If A_r indicates the set of areas resource r can cover and S_r is the set of skills they can perform, then currently to produce the base time and overtime allocation for a single resource we are producing $|S_r| * |A_r| * 2$ decision variables for every period of the plan.

This is a variable for each skill and area combination that resources can cover multiplied by 2 as each allocation variable also has an associated overtime variable.

Defining resources as groups we can instead use decision variables to model an area group's allocation within their primary area. Allocation to different areas are then modelled using decision variables for movement between area groups. Add a variable for each of the resource group's possible alternate areas to indicate movement of time of area group g_a to area group $g_{a'}$. This would produce only an additional $|A_r| - 1$ variables for the area moves with $|S_r|$ variables generated for the time allocation within an area.

This has already reduced the number of variables as $|S_r| + |A_r| - 1 \leq |S_r| * |A_r| \forall S_r, A_r$

In order to add this to the model we introduce area movement decision variables to indicate movement of an area group's time to a different area. Applying these changes to the resource time allocation cost we get the new equation:

$$a_{g_a t} = \sum_{s \in S_g} [\omega_{gs} x_{g_a st}] + \sum_{a' \in A_{g_a}} \left[M_{g_a g_{a'}} m_{g_a g_{a'} t}\right] \quad (9)$$

Now $x_{g_a st}$ is indicating the real value allocation of time from resource group g in area a to skill s in period t, with ω_{gs} being the real value cost for resource group g to use skill s.

The second sum in the equation is the extraction of the area choice to now model an area movement instead. Here $m_{g_a g_{a'}}$ is the real value decision variable, indicating the amount of time of resource group g being sent from area a to a' with $M_{g_a g_{a'}}$ being the real value cost associated with this movement.

In order for this refactoring to work the constraints also needs to be modified. With available time being moved between resource groups the first constraint needs to be altered to reflect this. We need to add time that is being loaned out, as that is the same as it being utilised, and subtract time that is being loaned in. We also need to apply the efficiency factor to the time being loaned in to model loss of time when moving a resource's time between areas.

Thus the first constraint becomes:

I. $\sum_{s=1}^{S}\left[x_{g_a st}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] \leq \sigma_{g_a t}\ \forall\, g, a, t$

Here we see the constraint is that the total time used by area group g_a in period t ($\sum_{s=1}^{S} x_{g_a st}$) is added to the total time loaned out minus the total time loaned in. The time loaned in is further modified by a real valued efficiency factor $\epsilon_{g_{a'} a}$indicating the efficiency of moving time from area group $g_{a'}$ to area a. This efficiency factor allows the modelling of time lost to travelling or working in an unfamiliar area. This factor is applied to the time loaned in as it is within this new area that the time would be lost. As before, this is constrained to be less than or equal to the total time that area group has available in that period.

The second constraint also requires some modification as we are now referring to resource groups rather than resources. Also, the efficiency factor for working in different areas is now covered by the area move decision variable and thus the efficiency factor for this constraint is only attached to the skill and resource group. Making the modifications gives us:

II. $\sum_{g=1}^{G}\left[\epsilon_{gs} x_{g_a st}\right] - D_{sat} = 0\ \forall\, a, s, t$

It should be noted that the overtime variable has been removed from this constraint as we have not yet reached that part of the refactoring process. As it turns out overtime will no longer be required within this constraint and we will see the reason why in the coming paragraphs.

Following on from this, it is now also possible to model the overtime as a single decision variable for each area group. The decision being modelled is then how much overtime to allocate to that area group. Then similarly to the area moves that time is added to the available time of the area group within the constraint. This removes the need for one variable for each possible allocation and instead reduces the number of variables required to indicate overtime for a single area group to one per period.

Making this change the overtime cost in the objective function becomes:

$$o_{g_a t} = V x'_{g_a t} \quad (10)$$

Where V is still the real valued cost associated with overtime and the decision variable is now a real value setting how much overtime to give to resource group g in area a in period t.

Next, to ensure that the overtime is being allocated to the available time for that area group, we must again modify the first constraint by subtracting the amount of overtime allocated from the left hand side. This gives us:

I. $\sum_{s=1}^{S}\left[x_{g_a st} - x'_{g_a t}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] \le \sigma_{g_a t}\ \forall\, g, a, t$

This ensures that any additional time allocated to overtime is made available to the relevant resource group in the relevant area and period.

Finally a slight re-factoring of constraints 3 and 4 is required due to the change in implementation:

III. $x'_{g_a t} \le \vartheta \sigma_{g_a t}\ \forall\, g_a, t$

IV. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} x'_{g_a t}\right]\right] \le \theta$

Finally, as mentioned above, an overtime factor is no longer required within constraint II. This is because overtime allocation is now being added to the available time for a resource within the model, rather than being directly assigned to a skill/area/period coordinate as before.

Bringing this altogether the improved model in its current state is now:

$$min\colon \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}\left[\omega_{gs} x_{g_a st}\right] + \sum_{a'\in A_{g_a}}\left[M_{g_a g_{a'}} m_{g_a g_{a'} t}\right] + V x'_{g_a t} + c''_{g_a t}\right]\right] + \sum_{j=1}^{J} k_{jt}\right] \quad (11)$$

Such that:

I. $\sum_{s=1}^{S}\left[x_{g_a st} - x'_{g_a t}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] \le \sigma_{g_a t}\ \forall\, g, a, t$

II. $\sum_{g=1}^{G}\left[\epsilon_{gs} x_{g_a st}\right] - D_{sat} = 0\ \forall\, a, s, t$

III. $x'_{g_a t} \le \vartheta \sigma_{g_a t}\ \forall\, g_a, t$

IV. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} x'_{g_a t}\right]\right] \le \theta$

3.3.2 Recruitment and Reduction

Recruitment is the addition of new resources into the plan, they will appear on a given period and exist for all future periods thereafter. Reductions is the corollary to this where resources are removed. As such with the similarity between these two they will be discussed together. In fact, if the cost for both is the same then they could be modelled with a single variable. However for the purposes of this model it will be assumed that the costs will be different and thus a separate variable will be added to the objective function for each.

Since recruitment and reductions are the permanent adding or removal of resources it is essentially the decision to add or reduce the available time for a particular area group for that period and any future periods. Thus the first step is adding a variable to each area group within each period to indicate recruitment and reduction of a resource from that group during that period.

The cost calculation for these variables is slightly more complex than before as aside from the flat cost associated with each, the cost of a resource to the plan must be added (or removed) for that period and all future periods.

Looking at recruitment first, the cost just from having a new resource in the plan is going to be the number of periods left plus one (the additional one is for the current period). If the periods go from period 1 to period T then the additional cost for hiring a resource in period t is T + 1 – t or the number of periods minus t. Thus a new recruit in the last period, T, will cost 1, as the resource must be paid for that period.

On top of this we add a flat real valued cost for recruitment I_c, so the cost to recruit 1 resource in period t is I_c + T + 1 - t.

Similarly for wastage/reduction, removing a resource in period t removes their cost from the plan for that and each subsequent period (as the cost has already been added in the constant C). Thus, removing a resource in period t means a reduction in the plan cost of T + 1 - t. Again a real valued flat cost for reductions is added to give an overall cost to reduce resource levels by 1 in period t as W_c + t - T – 1.

This gives the equation to calculate the cost of intake and wastage for area group g_a in period t as:

$$p_{g_a t} = (I_c + T + 1 - t)i_{g_a t} + (W_c + t - T - 1)w_{g_a t} \quad (12)$$

The new decision variables added are both integer values indicating the number of new resources from group g in area a have been added (i_{gat}) or reduced (w_{gat}) in period t. Similar to overtime, we now need to reflect the additional and reduced time available through recruitment and reduction in the time available constraint. Additional time available in period t for area group g_a is given by summing recruitment of that group from period 1 to t minus the sum of reduction of that resource to date from period 1 to t. Thus additional time available from recruitment and reduction for area group g_a in period t is given by:

$$f_{g_a t} = \sum_{t'=1}^{t} \left[i_{g_a t'} - w_{g_a t'} \right] \quad (13)$$

This extra time needs to be applied to the base time constraint (I) and the resource overtime constraint (III). In the base time constraint the extra time is simply subtracted from the left hand side (equivalent to adding it to the right hand side, the total time available). In the overtime constraint this time has to also be multiplied by the allowed overtime proportion ϑ. Applying these to both constraints gives:

I. $\sum_{s=1}^{S}\left[x_{g_a st} - x'_{g_a t}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] - \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'}\right] \leq \sigma_{g_a t}\ \forall\, g, a, t$

III. $x'_{g_a t} - \vartheta \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'}\right] \leq \vartheta \sigma_{g_a t}\ \forall\, g_a, t$

Finally, two new constraints are added to limit the total amount of recruitment and reductions allowed over the course of the plan:

V. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} i_{g_a t}\right]\right] \leq I$

VI. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} w_{g_a t}\right]\right] \leq W$

Where I is the integer maximum amount of recruitment and W is the integer maximum amount of wastage allowed over the course of the plan.

Thus after applying intake and wastage the model now looks like:

$$min\colon \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}\left[\omega_{gs} x_{g_a st}\right] + \sum_{a' \in A_{g_a}}\left[M_{g_a g_{a'}} m_{g_a g_{a'} t}\right] + V x'_{g_a t} + (I_c + T + 1 - t) i_{g_a t} \right.\right.\right.$$
$$\left.\left.\left. + (W_c + t - T - 1) w_{g_a t} + c'''_{g_a t}\right]\right] + \sum_{j=1}^{J} k_{jt}\right] \quad (14)$$

Such that:

I. $\sum_{s=1}^{S}\left[x_{g_a st} - x'_{g_a t}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] - \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'}\right] \leq \sigma_{g_a t}\ \forall\, g, a, t$

II. $\sum_{g=1}^{G}\left[\epsilon_{gs} x_{g_a st}\right] - D_{sat} = 0\ \forall\, a, s, t$

III. $x'_{g_a t} - \vartheta \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'}\right] \leq \vartheta \sigma_{g_a t}\ \forall\, g_a, t$

IV. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} x'_{g_a t}\right]\right] \leq \theta$

V. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} i_{g_a t}\right]\right] \leq I$

VI. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} w_{g_a t}\right]\right] \leq W$

Where $c'''_{g_a t}$ is the real valued cost associated with training for area group g_a in period t.

3.3.3 Training

The final addition to the resource costs is the training cost. Modelling this is like a combination of the new area move formulation with the recruitment/reduction formulation. For a given area group, g_a, a variable is added for each possible training movement (from that area group to a new area group) with an associated cost. This variable is a natural number and indicates movement of a certain number of resources from that area group to the new area group within that period. This gives the equation for the training cost of area group g_a in period t as:

$$c'''_{g_a t} = \sum_{g'_a \in G'_{g_a}} \nu_{gg'} \tau_{g_a g'_a t} \quad (15)$$

Where G'_{g_a} is the set of area groups that resources from area group g_a can be trained to belong to. The integer variable $\tau_{g_a g'_a t}$ indicates the number of members of area group g_a being trained to become a member of area group g'_a in period t. It should be noted that the area, a, is unchanged. Finally $\nu_{gg'}$ is the real valued cost associated with training a resource of resource group g to group g'.

This new addition to the equation requires further modification of the available base and overtime constraints to reflect the lost time for resource group g_a and the gain for group g'_a. A new constraint also needs to be added to limit the amount of training allowed over the plan.

First we formulate the equation indicating the gains and losses to available time due to training movements. Similar to recruitment, all training moves involving that resource from period 1 up to the current period, t, need to be taken into account. Thus the change to time available for area group g_a in period t due to training movements can be given as:

$$f'_{g_at} = \sum_{t'=1}^{t}\left[\sum_{g'_a \in G'_{g_a}} \left[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}\right]\right] \quad (16)$$

This is essentially the sum of all resources trained into area group g_a minus the sum of all resources trained out of area group g_a for all periods from 1 to t. Now similarly to the recruitment and reduction condition changes, this change in available time is applied to the base time constraint (I)and the resource overtime constraint(III) giving:

I. $\sum_{s=1}^{S}\left[x_{g_ast} - x'_{g_at}\right] + \sum_{a'=1}^{A}\left[m_{g_ag_{a'}t} - \epsilon_{g_{a'}a}\, m_{g_{a'}g_at}\right] - \sum_{t'=1}^{t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a \in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \leq \sigma_{g_at} \;\forall\, g, a, t$

III. $x'_{g_at} - \vartheta \sum_{t'=1}^{t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a \in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \leq \vartheta\sigma_{g_at} \;\forall\, g_a, t$

Also, an additional constraint is created to limit the total allowed amount of training, giving:

VII. $\sum_{g=1}^{G}\left[\sum_{g'=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T}\tau_{g_ag'_at}\right]\right]\right] \leq \mathrm{P}$

Where P is the integer indicating the maximum number of training movements allowed over the length of the plan.

Thus, with the resource costs now all modelled, the equation now looks like:

$$min\text{:} \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}\left[\omega_{gs}x_{g_ast}\right] + \sum_{a' \in A_{g_a}}\left[M_{g_ag_{a'}}m_{g_ag_{a'}t}\right] + Vx'_{g_at} + (I_c + T + 1 - t)i_{g_at} + (W_c + t - T - 1)w_{g_at} + \sum_{g'_a \in G'_{g_a}} v_{gg'}\tau_{g_ag'_at}\right]\right] + \sum_{j=1}^{J}k_{jt}\right] \quad (17)$$

Such that:

I. $\sum_{s=1}^{S}\left[x_{g_ast} - x'_{g_at}\right] + \sum_{a'=1}^{A}\left[m_{g_ag_{a'}t} - \epsilon_{g_{a'}a}\, m_{g_{a'}g_at}\right] - \sum_{t'=1}^{t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a \in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \leq \sigma_{g_at} \;\forall\, g, a, t$

II. $\sum_{g=1}^{G}\left[\epsilon_{gs}x_{g_ast}\right] - D_{sat} = 0 \;\forall\, a, s, t$

III. $x'_{g_at} - \vartheta \sum_{t'=1}^{t'=t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a \in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \leq \vartheta\sigma_{g_at} \;\forall\, g_a, t$

IV. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T}x'_{g_at}\right]\right] \leq \theta$

V. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} i_{g_at}\right]\right] \leq I$

VI. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} w_{g_at}\right]\right] \leq W$

VII. $\sum_{g=1}^{G}\left[\sum_{g'=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T}\tau_{g_a g'_a t}\right]\right]\right] \leq \mathrm{P}$

3.4 Cost associated with tasks $\boldsymbol{k_{jt}}$

Now we will expand the last part of the equation, the costs associated with the tasks. First we will add the cost associated with not completing the task in 3.4.1. Then in 3.4.2 we will add the costs for moving the tasks forwards and backwards in time, before finally adding a cost for failing to meet the SLA for a task in 3.4.3.

3.4.1 Task time allocation

The first step to modelling the cost associated with tasks is to add the variable indicating the decision to meet the requirements of the work stacks (tasks requiring completion). These tasks have an associated skill requirement, within an area, within a given period. They also have a task type and a priority level. These factors can be used to define a related cost for not completing that task. As outlined in section 3.1 the cost for completing no tasks is added as a constant to the model equation. Thus as time is allocated to complete a task, the cost associated with not completing that task should be removed from the plan.

The choice at this stage is whether to model each task individually, or attempt some amalgamation similar to the merging of resources into resource groups. In order to attempt to keep the size of the eventual objective function (and thus the solving time) down, attempts are made to merge as many tasks as possible without losing any required data. For example, tasks can be combined based on skill, area and period required that also have the same cost derived from their priority and type, however it may also be required to split these groups further to correctly model associated costs for moving that demand in time (e.g. postponing the task or bringing the work forwards) or even to model that some tasks can be moved and some cannot.

For the case of the mathematical model however this does not matter. We merely need to model tasks as task types. Each type has an associated cost for non-completion. Later in this section we will also add the costs for movements in time (if allowable for that type) and finally the service level agreement (SLA) costs. This could be passed into the model as individual tasks, or some refactoring could occur beforehand to combine similar tasks into types.

With this in mind, we can now being to build up the equation describing the costs associated with the tasks. First we need to model the cost for allocating time to a task type j. This simply involves adding a decision variable for each task type indicating the time allocated to it, multiplied by the associated cost derived from its type, priority and other factors. This gives:

$$b'_{jt} = -\varphi_j y_{jat} \quad (18)$$

Here $y_{jat}$is the real valued decision variable indicating time allocated to this task type, and $\varphi_j$is the real value cost for not completing 1 FTE (or whichever time unit is used to pass data to the model) of this task. The cost is negative as the cost for not completing any of this task has already been added to the cost for the solution. As time is allocated to complete tasks, the cost for not completing that task is subtracted. This allows the model to contain a task priority level (it will be trying to complete the most costly tasks first) whilst still giving a value that is closely related to an actual monetary value as the final cost of any solution.

We now need to construct the associated constraints. The first constraint is that more time cannot be allocated to a task type than is required. Thus we get a new constraint:

VIII. $y_{jat} \leq \rho_{jat} \; \forall \, j, a, t$

Here ρ_{jat} is the real valued requirement for task type j in area a in period t. Thus this constraint simply states that the time allocated to task type j be less than or equal to the time required.

At this stage we need to also define s_j as the skill required by task type j. This is required for the final constraint associated with this variable, which is constraint II that links the capacity used to the demand met. So far we have:

$$D_{sat} = \sum_{s_j = s} y_{jat} \quad (19)$$

This means, the sum of time allocated to tasks in area a in period t where the skill required by the task type j is s.

Plugging this into the constraint we began to construct in the previous section to constrain the time drawn from a resource for a specific skill/area/period coordinate to be equal to the time used for that skill/area/period by tasks we get:

II. $\sum_{g=1}^{G} \left[\epsilon_{gs} x_{g_a st} \right] - \sum_{s_j = s} y_{jat} = 0 \; \forall \, a, s, t$

Adding this all into the model we start to get the demand costs taking shape:

$$min: \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}[\omega_{gs}x_{g_a st}] + \sum_{a'\in A_{g_a}}\left[M_{g_a g_{a'}} m_{g_a g_{a'} t}\right] + Vx'_{g_a t} + (I_c + T + 1 - t)i_{g_a t}\right.\right.\right.$$

$$\left.\left.\left. + (W_c + t - T - 1)w_{g_a t} + \sum_{g'_a \in G'_{g_a}} \nu_{gg'}\tau_{g_a g'_a t}\right] - \sum_{j=1}^{J}\varphi_j y_{jat}\right] + \sum_{j=1}^{J} k'_{jt}\right] \quad (20)$$

Such that:

I. $\sum_{s=1}^{S}[x_{g_a st} - x'_{g_a t}] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] - \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a \in G'_{g_a}}[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}]\right] \leq \sigma_{g_a t}\ \forall\ g, a, t$

II. $\sum_{g=1}^{G}[\epsilon_{gs}x_{g_a st}] - \sum_{s_j = s} y_{jat} = 0\ \forall\ a, s, t$

III. $x'_{g_a t} - \vartheta \sum_{t'=1}^{t'=t}\left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a \in G'_{g_a}}[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}]\right] \leq \vartheta\sigma_{g_a t}\ \forall\ g_a, t$

IV. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} x'_{g_a t}]] \leq \theta$

V. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} i_{g_a t}]] \leq I$

VI. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} w_{g_a t}]] \leq W$

VII. $\sum_{g=1}^{G}[\sum_{g'=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} \tau_{g_a g'_a t}]]] \leq \mathrm{P}$

VIII. $y_{jat} \leq \rho_{jat}\ \forall\ j, a, t$

Where k'_{jt} indicates the real value demand costs for task type j in period t that are still to be added into the model, those for demand movement and SLA costs.

3.4.2 Demand Movement

The next part of the model to develop is the ability to simulate demand being moved forwards or backwards in time. This takes the form of postponing a task to a later period or bringing that work forwards to an earlier period of the plan. In the context of this model, this takes the form of moving time required on a task type in an area in a period either forwards to the next period or backwards to the previous period. Each of these moves will have a cost attached.

It was decided to only model the movement of one period at a time (it is still possible to move multiple periods through these single steps, but it disallows the ability to attach a non-linear cost. E.g. moving 5 periods costs 5* the cost of moving one period, but perhaps the cost would be more than that) in order to keep the already large number of variables within the model from becoming even larger.

To model the demand movement in this way, for each task type two new variables are added to the objective function. One indicating requirement for that task being moved forwards in time and the other indicating movement backwards in time. These two variables are used to calculate the cost for demand movement in the following equation:

$$b''_{jat} = \delta_j d_{jat} + \eta_j e_{jat} \quad (21)$$

Here, e_{jat} indicates the real valued amount of time for task type j in area a moved from period t to the previous period, with associated real valued cost η_j and similarly d_{jat} indicates the real valued amount of movement to the next period with its own associated real valued cost δ_j.

Similarly to area movements for the area groups, we need to calculate the additional (or reduction in) time required by task type j in area a in period t due to demand moving out and into that period. This gives us an equation:

$$f''_{jat} = e_{ja(t+1)} + d_{ja(t-1)} - d_{jat} - e_{jat} \quad (22)$$

Here we are adding the time required for this task type moved into this period from the previous period ($d_{ja(t-1)}$) to the time moved from the next period ($e_{ja(t+1)}$) and subtracting the amount of time requirement moved out of this period in either direction (d_{jat}& e_{jat}). This additional time is added to the requirement for this task type by subtracting it from the left side of the task time constraint (VIII) giving the new constraint:

VIII. $y_{jat} - e_{ja(t+1)} - d_{ja(t-1)} + d_{jat} + e_{jat} \leq \rho_{jat} \; \forall \, j, a, t$

Adding these into the model we now have:

$$\begin{aligned} min: \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s \in S_g}[\omega_{gs} x_{g_a st}] + \sum_{a' \in A_{g_a}}\left[M_{g_a g_{a'}} m_{g_a g_{a'} t}\right] + V x'_{g_a t} + (I_c + T + 1 - t) i_{g_a t}\right.\right.\right. \\ \left. + (W_c + t - T - 1) w_{g_a t} + \sum_{g'_a \in G'_{g_a}} \nu_{gg'} \tau_{g_a g'_a t}\right] \\ \left.\left. + \sum_{j=1}^{J}[\delta_j d_{jat} + \eta_j e_{jat} - \varphi_j y_{jat}]\right] + \sum_{j=1}^{J} k''_{jt}\right] \quad (23) \end{aligned}$$

Such that:

I. $\sum_{s=1}^{S}\left[x_{g_a st} - x'_{g_a t}\right] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'} t} - \epsilon_{g_{a'} a}\, m_{g_{a'} g_a t}\right] - \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a \in G'_{g_a}}\left[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}\right]\right] \leq \sigma_{g_a t} \; \forall \, g, a, t$

II. $\sum_{g=1}^{G}\left[\epsilon_{gs} x_{g_a st}\right] - \sum_{s_j = s} y_{jat} = 0 \; \forall \, a, s, t$

III. $x'_{g_a t} - \vartheta \sum_{t'=1}^{t'=t} \left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a \in G'_{g_a}} \left[\tau_{g'_a g_a t} - \tau_{g_a g'_a t} \right] \right] \leq \vartheta \sigma_{g_a t} \; \forall \; g_a, t$

IV. $\sum_{g=1}^{G} \left[\sum_{a=1}^{A} \left[\sum_{t=1}^{T} x'_{g_a t} \right] \right] \leq \theta$

V. $\sum_{g=1}^{G} \left[\sum_{a=1}^{A} \left[\sum_{t=1}^{T} i_{g_a t} \right] \right] \leq I$

VI. $\sum_{g=1}^{G} \left[\sum_{a=1}^{A} \left[\sum_{t=1}^{T} w_{g_a t} \right] \right] \leq W$

VII. $\sum_{g=1}^{G} \left[\sum_{g'=1}^{G} \left[\sum_{a=1}^{A} \left[\sum_{t=1}^{T} \tau_{g_a g'_a t} \right] \right] \right] \leq \mathrm{P}$

VIII. $y_{jat} - e_{ja(t+1)} - d_{ja(t-1)} + d_{jat} + e_{jat} \leq \rho_{jat} \; \forall \; j, a, t$

Where k''_{jt} is the final real valued cost to be added to the model associated with the tasks, the cost of failure to meet the SLA for that task.

3.4.3 SLA costs

The final factor built into this model is the capacity for penalising the cost due to failure to meet the SLA for a given task.

An SLA for a task states that y% of that task must be completed within x days of schedule. If this agreement is not met then a fine is levied, e.g. a large additional cost to the plan.

In this section we will first outline a method for calculating the number of tasks failing the SLA target within the month using the available data before then showing how to implement SLA costs within the model using that method.

3.4.3.1 SLA method

Defining n as the real valued requirement for a task over the course of a period in this plan and m as the real valued capacity assigned to that task over the period. We can then calculate the amount of demand for that task not being met within a day in the plan as n – m. Thus we can say that the real value indicating demand unmet for that task at the end of day t (R_t) is given by:

$$R_t = (n - m)t$$

Therefore, we can say that the rollover at the end of a period of length T is given by R_T. Assuming that demand that spills over to the next day is met first then the real value amount of demand, L_x that is x days or longer late in completion is:

$$L_x = R_T - (x - 1)m$$

Thus we can calculate the proportion of tasks, F, that are taking x days or longer as:

$$F = \begin{cases} 0, (x-1)m \leq R_T \\ \frac{L_x}{nT}, (x-1)m > R_T \end{cases}$$

So if we have an SLA to complete y% of those tasks within x days, we can check if the SLA is failing by checking if $F > y/100$. This allows us a simple method to estimate the number of a task that may be failing the SLA within a given period.

3.4.3.2 Applying SLA to the model equation

Having formulated an equation to estimate whether a task is failing its SLA within a given period we now need to add SLA into the model. At first glance this doesn't look to be possible, we are attempting to add a non-linear property (cost is 0 up to a certain point then suddenly becomes something) to a linear equation.

There is a workaround to this problem however to allow the application of this non-linear property to the linear model. The first step is to add a constraint to the model that forces the SLA to always be satisfied. Thus from the above equation, we are creating a new constraint that states that $F \leq {}^{y}/_{100}$. Here y/100 is the proportion of that task that must be completed within the allowable time. When translating this into a constraint for our model we introduce a new real valued variable, L_j, which is the proportion of task j that must be completed within the allowable time. This gives us:

$$F \leq L_j$$

The next step for constructing this constraint is to expand the left hand side and translate it into the models terms. Expanding F from the equation that was developed in the previous section we know that: $F = \frac{(n-m)T \; - (x-1)m}{nT}$

Refactoring this we get $F = \frac{nT + \; m(1-x-T)}{nT}$

Here T is the number of sub-periods within the period (e.g. number of days within a month if planning monthly). As we are looking at a multi-period plan however we create a new variable, T_t, which is the number of sub-periods within period t (e.g. the number of days in month t).

Next we look at m. This is the amount of this task type that are being fulfilled, from the model we know that the capacity applied to a task is given by the decision variable y_{jat}. Thus we simply replace m with y_{jat}.

After this we look at x. This is the number of days a task is allowed to be late. As we are applying this to the model with the possibility of multiple tasks we need to replace it with a new integer variable, l_j. This is the number of days a task of type j is allowed to be late.

We will pause at this stage to apply these modifications to the equation as the next step is the most complex. Thus we currently have:

$$F = \frac{nT_t + y_{jat}(1 - l_j - T_t)}{nT_t}$$

The final variable we need to replace is n. This is the demand for the current task type. The base demand for a task is given by the variable ρ_{jat}. However, we also need to take account of any demand that has been moved into or out of this period. Thus we need to subtract any demand that is moved out (given by the variables d_{jat} & e_{jat}) and add any demand that is moved into this period (given by the variables $e_{ja(t+1)}$ & $d_{ja(t-1)}$).

This gives us the equation $n = \rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}$

Bringing this together we get:

$$F = \frac{T_t\big(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}\big) + y_{jat}(1 - l_j - T_t)}{T_t\big(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}\big)}$$

Thus we are ready to add the constraint to the model that forces the SLA for all tasks to be met, giving us the new constraint:

IX. $$\frac{T_t\big(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}\big) + y_{jat}(1 - l_j - T_t)}{T_t\big(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}\big)} \leq L_j$$

Modelling SLA is not complete however as this does not allow for the possibility of an SLA failure. Thus in a case where it is impossible to satisfy all of the SLA's the model will not have a feasible solution.

The final step for applying the non-linear SLA process to the linear equation is to now add a decision variable to the model that indicates failure of the SLA for that task. We make this a Boolean choice, where the model sets 0 if it decides to pass the SLA and 1 if it decides to fail. We multiply this by the cost for failing that SLA. We then subtract the value of this decision variable (which will be 0 or 1) from the left hand side of the new SLA constraint. As both the left and right hand side of the equation currently are proportions that range from 0 to 1, by subtracting 1 from the left hand side it will always be less than or equal to the right.

Since choosing to fail an SLA will have a high cost associated the model will attempt to set all these to 0. However, if it cannot manage to fulfil the SLA constraint it can choose to set the variable to 1 instead. This solves the infeasible solution problem that arose when we first constructed this constraint.

Choosing f_{jat} to represent the Boolean decision to fail the SLA for task j in area a in period t with the associated real valued cost F_{jat} we get the cost from SLA failure for a task as:

$$k''_{jat} = F_{jat} f_{jat}$$

With the new constraint modified to be:

IX. $\frac{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}) + y_{jat}(1 - l_j - T_t)}{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)})} - f_{jat} \le L_j$

Adding this into the model we now have:

$$min: \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}[\omega_{gs}x_{g_ast}] + \sum_{a'\in A_{g_a}}\left[M_{g_ag_{a'}}m_{g_ag_{a'}t}\right] + Vx'_{g_at} + (I_c + T + 1 - t)i_{g_at}\right.\right.\right.$$
$$\left. + (W_c + t - T - 1)w_{g_at} + \sum_{g'_a\in G'_{g_a}} v_{gg'}\tau_{g_ag'_at}\right]$$
$$\left.\left. + \sum_{j=1}^{J}\left[\delta_j d_{jat} + \eta_j e_{jat} - \varphi_j y_{jat} + F_{jat}f_{jat}\right]\right]\right] \quad (24)$$

Such that:

I. $\sum_{s=1}^{S}\left[x_{g_ast} - x'_{g_at}\right] + \sum_{a'=1}^{A}\left[m_{g_ag_{a'}t} - \epsilon_{g_{a'}a}\, m_{g_{a'}g_at}\right] - \sum_{t'=1}^{t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a\in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \le \sigma_{g_at}\ \forall\, g, a, t$

II. $\sum_{g=1}^{G}\left[\epsilon_{gs}x_{g_ast}\right] - \sum_{s_j=s} y_{jat} = 0\ \forall\, a, s, t$

III. $x'_{g_at} - \vartheta\sum_{t'=1}^{t'=t}\left[i_{g_at'} - w_{g_at'} + \sum_{g'_a\in G'_{g_a}}\left[\tau_{g'_ag_at} - \tau_{g_ag'_at}\right]\right] \le \vartheta\sigma_{g_at}\ \forall\, g_a, t$

IV. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} x'_{g_at}\right]\right] \le \theta$

V. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} i_{g_at}\right]\right] \le I$

VI. $\sum_{g=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} w_{g_at}\right]\right] \le W$

VII. $\sum_{g=1}^{G}\left[\sum_{g'=1}^{G}\left[\sum_{a=1}^{A}\left[\sum_{t=1}^{T} \tau_{g_ag'_at}\right]\right]\right] \le \mathrm{P}$

VIII. $y_{jat} - e_{ja(t+1)} - d_{ja(t-1)} + d_{jat} + e_{jat} \le \rho_{jat}\ \forall\, j, a, t$

IX. $$\frac{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}) + y_{jat}(1 - l_j - T_t)}{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)})} - f_{jat} \leq L_j$$

3.5 Finalising Model

In this section we add the final touches to the model. Firstly the final constraints are added to the model before the model is again restated in its final state.

3.5.1 Final Constraints

In order to finalise the model we need to add the final constraints. These involve setting the lower limit for each variable and setting each variables type.

The first of these, setting the lower limits for each decision variable, is rather simple. None of the decision variables can be negative, thus we set all of their lower limits to be zero. This gives us the new constraint:

X. $$x_{g_a st}, x'_{g_a t}, m_{g_a g_{a'} t}, d_{jat}, e_{jat}, y_{jat} \geq 0$$

The final constraint required is the constraint that sets all of the variables types. Most are real numbers and those are the variables already mentioned in the constraint above. However the recruit, reduction and training numbers are natural (including 0), which is why they did not need to be mentioned in the previous constraint as they cannot be negative anyway. Finally the decision to fail an SLA is a Boolean decision.

Putting this together we get the final constraint:

XI. $$x_{g_a st}, x'_{g_a t}, m_{g_a g_{a'} t}, d_{jat}, e_{jat}, y_{jat} \in \mathbb{R}, r_{g_a t}, w_{g_a t}, \tau_{g_a g'_a t} \in \mathbb{N}^0, f_{jat} \in \{0, 1\}$$

3.5.2 Full Linear Model

Adding these final constraints onto the model we get the final version of the model:

$$min: \sum_{t=1}^{T}\left[\sum_{a=1}^{A}\left[\sum_{g=1}^{G}\left[\sum_{s\in S_g}[\omega_{gs}x_{g_a st}] + \sum_{a'\in A_{g_a}}\left[M_{g_a g_{a'}}m_{g_a g_{a'}t}\right] + Vx'_{g_a t} + (I_c + T + 1 - t)i_{g_a t}\right.\right.\right.$$

$$\left. + (W_c + t - T - 1)w_{g_a t} + \sum_{g'_a\in G'_{g_a}} \nu_{gg'}\tau_{g_a g'_a t}\right]$$

$$\left.\left. + \sum_{j=1}^{J}[\delta_j d_{jat} + \eta_j e_{jat} - \varphi_j y_{jat} + F_{jat}f_{jat}]\right]\right] \quad (25)$$

Such that:

I. $\sum_{s=1}^{S}[x_{g_a st} - x'_{g_a t}] + \sum_{a'=1}^{A}\left[m_{g_a g_{a'}t} - \epsilon_{g_{a'}a}\, m_{g_{a'}g_a t}\right] - \sum_{t'=1}^{t}\left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a\in G'_{g_a}}[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}]\right] \leq \sigma_{g_a t}\ \forall\, g, a, t$

II. $\sum_{g=1}^{G}[\epsilon_{gs}x_{g_a st}] - \sum_{s_j=s} y_{jat} = 0\ \forall\, a, s, t$

III. $x'_{g_a t} - \vartheta\sum_{t'=1}^{t'=t}\left[i_{g_a t'} - w_{g_a t'} + \sum_{g'_a\in G'_{g_a}}[\tau_{g'_a g_a t} - \tau_{g_a g'_a t}]\right] \leq \vartheta\sigma_{g_a t}\ \forall\, g_a, t$

IV. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} x'_{g_a t}]] \leq \theta$

V. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} i_{g_a t}]] \leq I$

VI. $\sum_{g=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} w_{g_a t}]] \leq W$

VII. $\sum_{g=1}^{G}[\sum_{g'=1}^{G}[\sum_{a=1}^{A}[\sum_{t=1}^{T} \tau_{g_a g'_a t}]]] \leq \mathrm{P}$

VIII. $y_{jat} - e_{ja(t+1)} - d_{ja(t-1)} + d_{jat} + e_{jat} \leq \rho_{jat}\ \forall\, j, a, t$

IX. $\frac{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)}) + y_{jat}(1 - l_j - T_t)}{T_t(\rho_{jat} - d_{jat} - e_{jat} + e_{ja(t+1)} + d_{ja(t-1)})} - f_{jat} \leq L_j$

X. $x_{g_a st}, x'_{g_a t}, m_{g_a g_{a'}t}, d_{jat}, e_{jat}, y_{jat} \geq 0$

XI. $x_{g_a st}, x'_{g_a t}, m_{g_a g_{a'}t}, d_{jat}, e_{jat}, y_{jat} \in \mathbb{R}, r_{g_a t}, w_{g_a t}, \tau_{g_a g'_a t} \in \mathbb{N}^0, f_{jat} \in \{0, 1\}$

3.6 Problem Size Discussion

It is clear when analysing the above problem that the complexity will scale very quickly due to the number of nested summations. Further to that, even if we ignore different priority orders, the number of potential resource groups will increase exponentially with the number of skills. In practice this is not exactly the case as some skill combinations will not exist in the workforce. For example if there are many skills it is unlikely there are resources that have all or most of them. In corollary it would also be unlikely that any resources had a very low number of skills. A typical area from the real world case study data defined in 1.3 where there are 13 different skills has 52 different combinations for example.

Some assumptions are required to calculate a rough estimate of how the problem will scale. One of which relates to the number of allocation variables for resource groups time. Each resource group will have a decision variable for each skill they can perform which is then multiplied by the number of areas and periods in the problem. The number of skills will vary depending on the resource group. Thus to get an estimate we shall multiply the resource groups by the mean number of skills they contain, defined as$|S_g|$. Another approximation required is for the number of valid areas moves for an area group. This could be as high as A-1, e.g. they can move to any area other than their own. However in practice there is usually a limitation so we will define that the mean number of possible area moves as $|A_g|$. Finally, the mean number of potential training moves is defined as $|G'_{ga}|$. Using those approximations we can define an equation to calculate the number of decision variables, N, as:

$$N = TA\big(G\big(|S_g| + |A_g| + \ |G'_{ga}| + 3\ \big) + 4J\big) \ (26)$$

It is clear to see this will become large pretty quickly as T, A, G and J increase along with the increase in number of resource groups caused by the number of skills increasing.

Calculating this using the example data defined in 1.3 we have T=28 and A=52. With 13 skills we can estimate the remaining variables as G = 52, $|S_g|$ = 7, $|A_g|$ = 13, | $|G'_{ga}|$ = 13 and J = 13. Assuming there that resources can travel to, on average, ¼ of the areas each, and that there is a potential training move to add each of the 13 skills and that there is just one task variable for each skill. Calculating equation 22 with these values gives 2801344 decision variables. Even solving for a single area which reduces the problem to 34944 variables proved to be too computationally intensive to solve within a reasonable amount of time for use in a real world scenario. This was found after an initial performance test was performed using CPLEX version 12.8 with default settings being run in a java 1.7 application using the SCPSOLVER library as the wrapper on a windows 8.1 Lenovo ThinkPad P50 laptop running an intel core i7 6820HQ processor with 40GB of RAM. Attempts to solve the problem on this set up were not able to find a solution within the 8 hours the experiment was run for in each case. This matches the findings in the literature as even just the GAP sub-problem contained within this full model cannot be solved in a reasonable time when using exact methods if the problem size becomes sufficiently large.

3.7 Conclusions

In this chapter the tactical planning problem was defined mathematically as a mixed integer linear programming model. This exercise served to formally define the problem outlined in section 1.2 setting down the precise decision variables and constraints.

During the model creation however it was quickly discovered that solving such a large problem all at once using linear programming techniques was infeasible as the solution time, particularly with the addition of some of the integer decision variables, was too long to be useable in a real world situation. A brief investigation of a small example created by utilising only single area from the case study data failed to find a solution within a feasible amount of time. The experiment was ended after 8 hours as by the time the solution was calculated all of the input data would already be out of date. For the purposes of a useable solution to the planning problem the expectation would be to, at the very least, find a solution within a working day in the case where no manual adjustment was required. In the more realistic case however where some values may require tweaking through manual intervention after each run to try out multiple variations of the plan then it would need to complete much quicker than that.

The remainder of this thesis now focuses on searching for a means to solve this problem within the real world context within a useable time frame.

Chapter 4

Solving the capacity-demand allocation problem

4.1 Introduction

In this chapter we define the problem of matching resources and their skills to demand, as faced by planners in service industry organisations with large multi-skilled workforces. Chapter 2 showed that the overall tactical planning problem explored in this thesis could be broken down into two sub-problems. These sub-problems combine to form a bi-level model where a leader is setting the capacity constraints for the capacity-demand allocation follower. The problem defined in this chapter makes up the follower level of that overall model. It was identified that this follower model contained aspects of a generalised assignment problem. Many methods have been used in the literature to solve this problem and in this chapter we compare a few of them to choose a method to use in the follower layer of the nested bi-level problem. Of these we will be investigating both the linear programming and the genetic algorithm approach to solving the problem as these two also are known to be applicable to a level of the nested bi-level model. Along with this we will compare these approaches with a greedy hill climber algorithm that has been used to solve the capacity-demand matching problem in a practical application (Kern, et al., 2006). This approach is also similar to the methods a manual planner would undertake and thus is a good comparator for algorithm performance. By the end of this chapter we aim to have selected an approach to use to solve the follower model within the wider bi-level model.

To meet these goals the capacity-demand allocation sub problem is first defined in section 4.2. In section 4.3 we define the three solution methods we will be comparing. The genetic algorithm, hill climber and linear programming models. In section 4.4 the experiments used to perform this comparison as described before they are analysed in section 4.5. Finally in section 4.6 we conclude this chapter by selecting the algorithm to use to solve the follower model going forwards.

4.2 The Capacity-Demand Allocation Problem

4.2.1 Problem Description

A large multi-skilled workforce contains a number of resources, with each resource having a number of skills. Further to this, each resource has a preference level allocated to the skill they perform. This value is used to indicate which skills they are better practiced at performing and is a factor in producing an optimal plan. The planner should prefer to use a resource's higher preference skills where possible. Each resource has a total time available to it which requires allocation to the skills it can perform. For example a particular employee may have three skills they can perform, skill 1, 2 and 3 with preferences of 1, 2 and 3 respectively. On a given day they have 8 hours available to work. During the allocation it would be preferable to allocate all 8 hours to their first choice skill however they could also be assigned some time on jobs requiring skills 2 and 3. This assignment to secondary and tertiary skills etc. is part of the capacity-demand allocation process where resources are moved to their less favoured skills to better balance the plan.

The organisation also has a *demand,* consisting of a set of jobs requiring completion in a particular period. Each job has a skill that is required to complete them, along with an amount of time that the job needs applied to it. The final factor is the priority of the job. This priority is used to ensure that the planner attempts to fulfil the more important jobs first. For example there may be two jobs requiring completion, one requires skill one and has priority 2 needing 2 hours whereas the other requires skill two and has priority 1 requiring 4 hours to complete. In this case despite needing more time to complete the plan would usually attempt to cover the second job first.

The job of the planner, therefore, is to attempt to best match the time of the resources to the jobs making up the demand, subject to the amount of each resource available in the time period and what skills they can perform.

For a manual planner however, individually matching between resources and jobs is not practical. For example typically in a large service company the workforce can number in the tens of thousands, divided into regional workgroups numbering in the hundreds. This makes it infeasible to individually allocate every resource by hand. In this case, a solution is to plan using the skill variances. The variance for a skill is defined as the total time supplied for that skill from the resources, minus the total time required for that skill from the jobs. A positive variance indicates there is over-allocation to that skill (surplus); a negative variance shows there is not enough time allocated to that skill (deficit). The planner then attempts to move time from skills with a surplus to those with deficits to bring each variance value as close to zero as possible.

This can be defined as the constrained optimisation problem.

$$min\colon \sum_{r=1}^{R}\left[\sum_{s\in S_r}\omega_{rs}x_{rs}\right]+\sum_{j=1}^{J}\left[p_j\left(t_j-y_j\right)\right]+E\sum_{s=1}^{S}z_s \quad (27)$$

subject to:

1. $\sum_{s=1}^{S} x_{rs} = T_r \; \forall \, r$
2. $y_j \leq t_j \; \forall \, j$
3. $\sum_{r=1}^{R} x_{rs} - \sum_{s_j=s} y_j - z_s = 0 \; \forall \, s$
4. $x_{rs}, y_j, z_s \geq 0$
5. $x_{rs}, y_j, z_s \; \in \; \mathbb{R}$

Equation (27) defines the objective function for resource-demand matching in the single planning dimension of resource skills. The equation can be broken down into three key components, denoted by each summation operator.

The first component calculates the cost of the use of resources in the current plan. The decision variable here, x_{rs}, denotes the amount of time from resource r allocated to skill s. This is multiplied by ω_{rs} which is the associated cost for this resource using that skill based on the resource's preference for the skill. The set of skills a resource has is denoted by S_r, thus this component of the equation sums the allocation multiplied by the cost for each skill each resource has.

The second component calculates the cost of any unmet demand. Here y_j is the decision variable, indicating the amount of time allocated to job j. The other variables in this section denote the total time required for the job t_j, and the penalty cost for not allocating enough time p_j. Thus the unmet-demand cost is calculated by summing the difference between the required and the allocated time to each job multiplied by their penalty cost.

The final component is an additional penalty cost for over-allocation to any skills. The decision variable here, z_s, denotes extra time allocated to skill s that hasn't been assigned to complete a job. The cost E is the penalty factor for this over-allocation. For the purposes of this comparison the cost was set to zero as there is no capacity levers available to rectify excess capacity and there is no penalty applied to over allocation in the comparison hill-climber algorithm.

Valid solutions to the equation are defined by the constraints. The first constraint (I) states that the total time allocated to a resource's skills should equal the total time available to that resource (T_r). Similarly, the second constraint (II) limits the amount of time applied to a job to be less than or equal to the time required by that job. The third constraint (III) links the components of the objective function, stating that the total time for a skill supplied from the resource side must equal the total time used by the demand side of the resource-demand matching. The final constraints (IV & V) limit the decision variables to positive real values.

4.3 Solution Methods

This section outlines the methods used to solve the planning problem outlined in section 4.2. The GA and linear approach have been selected as some common methods for solving both the GAP and follower levels of bi-level models. As well as these we will be outlining the hill-climber used as a comparator to the current planning process. The cost of the solution provided by each of these methods will be calculated using equation (27) to provide an equal comparison.

4.3.1 GA

A genetic algorithm is a metaheuristic, more specifically an evolutionary algorithm, inspired by the natural evolution process (Mirjalili, 2019). It is a population based algorithm with the population being made up of a number of current solutions, also referred to as chromosomes. Individual elements of a chromosome, reflecting the parallel with evolution, are defined as alleles or genes. Each population member has a fitness value indicating how good that solution is. This is defined by the fitness function for the problem which takes a chromosome/solution and returns an associated value based on the current values of the individual alleles.

The GA then searches the solution space by iterating though generations. In each generation the population is updated through the process of selection, crossover and mutation. During selection population members are selected based on their fitness value and the selection operator used to be the parents for the next generation.

For each two parents selected, there is a probability that crossover will occur and two children are created. Crossover is where segments of their parent's chromosomes are swapped or merged to create two new solutions. This is usually done through the random selection of one (or many) points at which to chop up both parent chromosomes and the resulting segments are then swapped alternately to give two children that are a mix of their parent's genes. For example in single point crossover if the crossover point was chosen after the second gene then the first child would consist of the first 2 genes from the first parent and the remaining genes from the second parent. Similarly the second child would have the first 2 genes from the second parent and the remaining genes from the first parent.

The final operator is the mutation operator used to model the arising of new traits. This has a configured probability to occur and when it does will perform some form of random change to an allele in a chromosome.

Crossover and mutation are often seen as driving exploitation and exploration respectively during the algorithms search where crossover is combining solutions to drive towards local optima and mutation is jumping solutions around the solution space to explore for other optimal areas. An additional parameter also used is called elitism, this indicates that the top x solution should always be carried through unchanged to the next generation and is often used to ensure good solutions are not lost.

The search usually continues for a number of configured generations before returning the best solution (given by fitness value) or an alternative is to run until stagnation, which is where it will continue iterating through generations till the best solution has not improved for a certain number of generations.

To solve this problem using a GA therefore we must define all of the components. First we design the solution representation. Then define the operators for use with this solution before describing the fitness function. Finally we briefly outline the pseudo code for the algorithm before covering the implementation used for the comparison experiments.

5.3.1.1 Solution Representation

The chromosome for this specific problem is that defining the time allocation of resources in the problem to their skills.

In order to represent an individual resource's available time distribution a list of real values is used. Each value on this list equates to the amount of time that resource spends on each skill they can perform (analogous to x_{rs} in equation (27)). The sum of these values is constrained to equal the time that resource has available (constraint I).

This list of real values is an individual gene of the solution. The entire solution is made up of a list of these genes, one for each resource in the problem. Thus the chromosome is a list of vectors, where each vector is a resource and the values in the vector indicate the distribution of their time across the skills they are able to perform.

Take, for example, a problem containing four resources. The first can perform three different skills and has 8 hours available, the second can perform two different skills and has 10 hours available, the third can perform four skills and has twelve hours available and the fourth can perform two skills and has eight hours available. A valid solution for this problem could be:

```
[6.0, 2.0, 0.0], [5.0, 5.0], [3.0, 2.0, 0.0, 7.0], [1.0, 7.0]
```

Here, the first gene, *[6.0, 2.0, 0.0]*, represents the time allocation for the first resource. Six hours of their time is allocated to their first skill, two to their second and zero to their third.

Similarly the second gene represents the time allocation for the second resource. This resource has higher available time than the first resource with their time equally split with five hours for each skill they can perform. The same follows for the third and fourth.

4.3.1.2 Operators

The operators chosen for use were a uniform crossover (Spears & De Jong, 1995) with the tournament selection operator (Goldberg & Deb, 1991). In uniform crossover there is an equal chance for each individual gene to be swapped during the crossover process. In this case the gene had been defined as the vector indicating an individual resource's time allocation to their skills. Swapping these intact meant there was no issue with the constraint that the sum of the allocation to skills for a resource should equal their time available would be broken. Thus uniform crossover is a good choice.

Tournament selection involves two population members being selected at random and whichever has the highest fitness is selected to become a parent for the next generation. This gives a good balance between selecting the fittest members whilst still having a chance to select some of the less fit solutions to retain some diversity.

Due to the nature of the individual genes however, something other than the standard mutation operators had to be defined. Standard mutation would have changed a single point of the solution causing the resource time allocation constraint to be broken.

Three operators were thus implemented for this GA. In each case a gene is selected at random for mutation based on the current mutation probability. Once selected one of the below operators is applied to that gene.

The first completely re-randomises a gene (and is the same operator used when initially generating the gene values for the solutions in the initial population). This more drastically changes a solution and this promotes exploration. It randomly allocates the resource's available time, and ensures that the generated gene is valid by ensuring all of the resource's time is allocated. Essentially a random skill that the resource can perform is selected and then a random amount from 0 to the resources max time remaining is chosen to be applied to that skill. The amount used is then subtracted from the time available and the process continues with the skill already allocated removed from the potential list. The process continues till all of the resource's time is allocated, or there is only one skill left on the list (in which case that skill is allocated all of the remaining time).

The second involves some minor modifications to the mutated distribution. This more minor adjustment promotes more of an exploitation approach to gradually move towards a local optima. A random amount less than the max time available to that resource is subtracted from one randomly selected value in the gene and added to the other. If the amount to be subtracted is greater than the value selected, then the value is set to zero and its previous value is added to another. For example, given a gene:

```
[4.0, 2.0, 0.0, 2.0]
```

The random modification amount is chosen as 3.0, and the second point is chosen as the source with the first as the destination.

As the second point is 2.0, subtracting 3.0 would cause it to become negative and thus an invalid solution. Instead 2.0 is subtracted from the second point and added to the first, giving the resulting gene:

```
[6.0, 0.0, 0.0, 2.0]
```

In order to retain the effectiveness of this mutation in problems with longer genes the number of changes occurring for any given mutation is chosen randomly (with an upper limit of half the genes length).

The third mutation operator is a hybrid of the first two. For the first few generations (the number tuneable by the operator) the first mutation method that completely re-randomises a gene is performed. After the configured number of generations the second mutation operator is performed instead to just perform minor modifications for the rest of the evolution. This has the advantage of combining the previous two where the solutions will jump around the solution space more in the first x generations to explore the solution space before switching to more minor moves to exploit after the first x generations has passed.

4.3.1.3 Fitness Function

The fitness function for this genetic algorithm calculates the cost of the solution using a similar method to equation 22. As mentioned in the solution representation, the GA solution provides the values for x_{rs}. To speed calculation of this portion of the equation, the fitness function contains an index containing the associated cost for each value in the solution (ω_{rs}). The fitness function need only multiply these together to calculate the resource portion of the cost function.

Unlike the linear model however, the GA does not explicitly allocate the decision variables for the demand side of the equation. To calculate the cost of the unmet demand the fitness function splits the jobs into groups by their skill required and orders them in priority order. The time allocated to each skill by the resources is applied to complete the jobs of the same skill, fulfilling the highest priority job first. This provides the values for y_j in equation 22.

The values for z_s are calculated as any excess time left over after all jobs of a skill are fulfilled. As mentioned in section 4.2 however, for the purposes of these experiments the additional cost was set to zero.

4.3.1.4 Pseudo Code

The final part of the GA required for implementation is the decision around the stopping condition. As the solution complexity depends on the number of resources and possible skills in a data-set, then the precise number of generations required to reach a near optimal solution would vary per problem. Something dynamic was required to allow the GA to run longer for larger problems or stop sooner on smaller ones. The solution implemented here was to stop the evolution upon reaching stagnation. This involves reaching termination when the best fitness has not improved for a tuneable number of generations.

Bringing all the components together we get the GA algorithm as:

```
1.  Generate Initial Population;
2.  Evaluate population;
3.  Set bestSolution = solution with best fitness;
4.  Set Stagnation Generations = 0;
5.  While (stagnation generations < termination limit) {
6.          Select population for breeding;
7.          Perform crossover on selected population;
8.          Perform mutation on new children;
9.          Evaluate population;
10.         If (current best solution has better fitness than best solution) {
11.                 Stagnation Generations = 0;
12.                 Best solution = current best;
13.         }
14.         Else {
15.                 Stagnation Generations++;
16.         }
17. }
18. Return best solution;
```

4.3.1.5 GA Implementation

The previously described GA was implemented in Java 1.7 making use of the Watchmaker framework (Dyer, 2010). Java was utilised as this was a requirement from the industrial partner who at the time did the majority of their development within Java. The Watchmaker framework was chosen to implement the GA as it brought with it the benefits of optimised code to give improved performance, including parallelisation across multiple cores, whilst being configurable enough to allow the implementation of any fitness function, solution encoding, mutation, selection and crossover operators that might be required.

After some initial test runs to tune the parameters the crossover probability was set to 0.6 with a mutation probability of 0.02. This was found to give a good balance between exploration and exploitation for the search. The mutation was set to perform re-randomisation for the first 500 generations then use the gene modification method thereafter to further promote exploration early in the search before focussing more on exploitation in the latter phases. Population size was chosen as 250 being around 1.5 times the length of the solution to give a good spread across the search space at the start, as well as with an elitism value of 3 to ensure the best solutions were retained for future generations. The algorithm was set to terminate after reaching stagnation for 100 generations rather than run for a specific number of generations to give a chance to converge to an optimal in both more complex and simple search scenarios.

4.3.2 Greedy Hill-Climber Heuristic

To provide some real world comparisons, we also solve the problem instances using a method currently utilised in the real world planning process (Owusu, et al., 2006).

The first step assigns all of a resource's available time to their primary (most preferred) skill. This is analogous with how the data is usually first represented in a planning application, before any manual (or otherwise) planning is performed.

The second step applies a simple rule based optimisation to attempt to move surplus time allocated to one skill to any skills with deficits where possible. The rule attempts to move time to the highest priority skills first. All the skills where there is a surplus are iterated through from that with the highest deficit to the lowest. Some of the applied time is then attempted to be moved to the highest priority skill with a deficit. To do so each resource currently applying time to the surplus skill is iterated through in the order of preference for the deficit skill (with those that cannot perform the deficit skill ignored) and the amount of time applied to the surplus skill (where preference is equal). Their allocation is then updated to move time to the deficit skill till either all the resource's available time is moved, the current surplus skill is no longer in surplus, or the current deficit skill is no longer in deficit. The process continues till all surplus skills have been checked for potential moves. This is essentially the heuristic used when performing manual planning so acts as a good comparator for the current process.

4.3.3 Linear Program

The constrained optimisation problem defined in section 2 was also solved using CPLEX (Bliek, et al., 2014). CPLEX was chosen as a state of the art commercial solver with the availability of an academic license allowing use for experiments. Due to the requirements of the industrial partner to develop within java, the model was implemented in java 1.7 using SCPSolver (Planatscher & Schober, 2015) to interface between Java and CPLEX solver version 12.8. The use of SCPSolver was required as a wrapper around CPLEX since it is C based thus requires a wrapper to function in a java application. SCPSolver itself was chosen as it also included the ability to plug in two different open source solvers instead of CPLEX without the need to change any of the java code defining the model. This was seen as an advantage since due to the required licenses it may not be possible to use CPLEX when deploying the model within an application. The disadvantage of the wrapper approach was less easy access to CPLEX parameters thus the default settings were used, though these were found to perform adequately for the purpose of this experiment. Solving using CPLEX is an exact method so will provide the optimal solution, however it was seen to have poor scalability in chapter 3 when applied to the tactical planning problem. However, for the purposes of this sub-problem, it was chosen to test the computational performance vs the GA as the problem size is substantially smaller. If it performs better while providing the optimal solution then it will be a good candidate to use to solve the follower level of the bi-level problem.

4.4 Experiments

In this section we define the experiments to perform a comparison between the methods outlined in section 4.3. First we outline the motivation for the experiments before explaining the method used.

4.4.1 Motivation

The purpose of the experiments conducted was to provide an effective comparison between the formulated GA, the linear approach and the simple planner's heuristic used by current planners. The objective being to discover which approach provides the best option to use as the follower in the tactical planning bi-level model.

To analyse the relative effectiveness of each method, we look both at some skill variance graphs to analyse the allocation of time to each skill as well as looking at the overall cost of the solutions produced.

4.4.2 Experimental Method

At this stage the real world data was not yet available. As such a logic was built to generate randomised data for a specified number of skills, resources and work-stacks (jobs). The specified number of skills were created then the specified number of resources and workstacks.

Each generated resource was allocated a random number of skills (each with an associated preference) and a random amount of time available. First a random skill from the full skill list is selected to add as the resource's primary skill with an even chance of selecting any skill. Then for each remaining skill the resource selected a random Boolean, if true the skill was added and if not the skill was not. Thus there was a 50% chance to be able to perform each additional skill. For each skill added the resource rolled a random number between 1 and 3 to be used as their preference value for that skill. This roughly matched real data where it was observed that most resources could perform on average just over half of the available skills. The resource was then allocated an amount of time randomly between 0 and 10 for each period in the plan by rolling a random double between 0 and 1 and multiplying that by 10.

Each work-stack was generated to require a random skill, require a random amount of time and was allocated a random priority level. The skill was selected randomly from the list with an even chance of selecting any. Priority level was randomly rolled between 1 and 6. For the time required a base calculation as performed of 10 * the number of resources / the number of workstacks. Then a random number between 0 and 1 was rolled for each period in the plan and multiplied by that base calculation to give the time required for each period.

The range for possible values of time available to a resource and time required by jobs were thus set such that selecting the mean in all cases would produce a data-set where the total time available equalled the total time required. This creates a situation where there is a workforce that generally matches the demand, the job of the planner is simply to perform the skill matching between the two.

This logic was used to generate 12 separate data-sets, each with 50 resources, 10 skills and 200 work-stacks, to simulate a typical planning situation for a service company (Voudouris, et al., 2008).

Each of the methods outlined in section 3 was used to solve the planning problems arising from the resulting data-sets. As the genetic algorithm is non-deterministic the model was run for 100 repetitions on each of those 12 problems and the mean results noted. The alternate methods were all deterministic and thus only run once per problem.

As well as the optimisation methods, a 4th dataset was produced to show the results of simply allocating all of the time to each resources' primary skills. This was to provide a baseline solution.

The experiments were run on the same set up as that used in chapter 3, on a windows 8.1 Lenovo ThinkPad P50 laptop running an Intel core i7 6820HQ processor with 40GB of RAM. Java 1.7 was used for all the methods with Watchmaker version 0.7.1 used to implement the GA and CPLEX version 12.8 with default settings running within the SCPSOLVER (no version number) library as the wrapper used to solve the linear program.

4.5 Results

Here we present the results of the comparison experiments. First we look at the graph showing the costs of the solutions produced by each method in each problem. After that we select a few typical examples of skill variance graphs showing a few different situations to illustrate how each handles the allocation of resource skills to the required demand.

4.5.1 Solution Cost

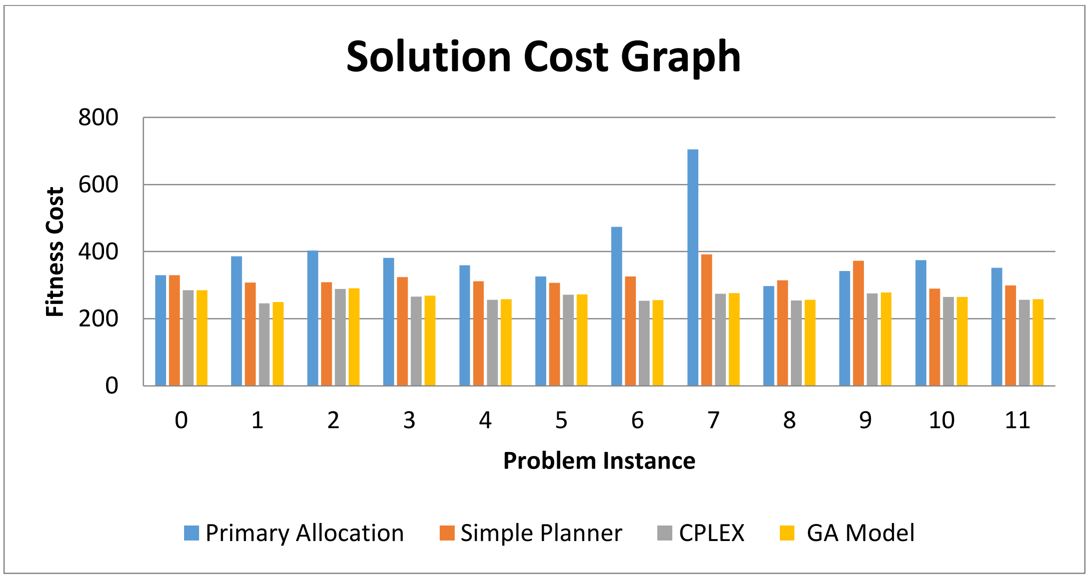

Fig. 3 Solution costs

Fig. 3 shows the cost values achieved by using equation (27) to evaluate the solutions. In this case the lower the value the better the solution. The graph shows that the GA approach is producing lower cost solutions than the simple planner in all cases with an average cost across the 12 instances of 267.76 compared to 323.66. The linear model does slightly outperform the GA however with an average cost of only 266. The GA does match the linear model in 3 of the 12 instances, with the linear model slightly lower cost in the remaining 9.

4.5.2 Skill Variances

In this section we pick out a few representative skill variance graphs highlighting some of the common behaviours seen in the results data.

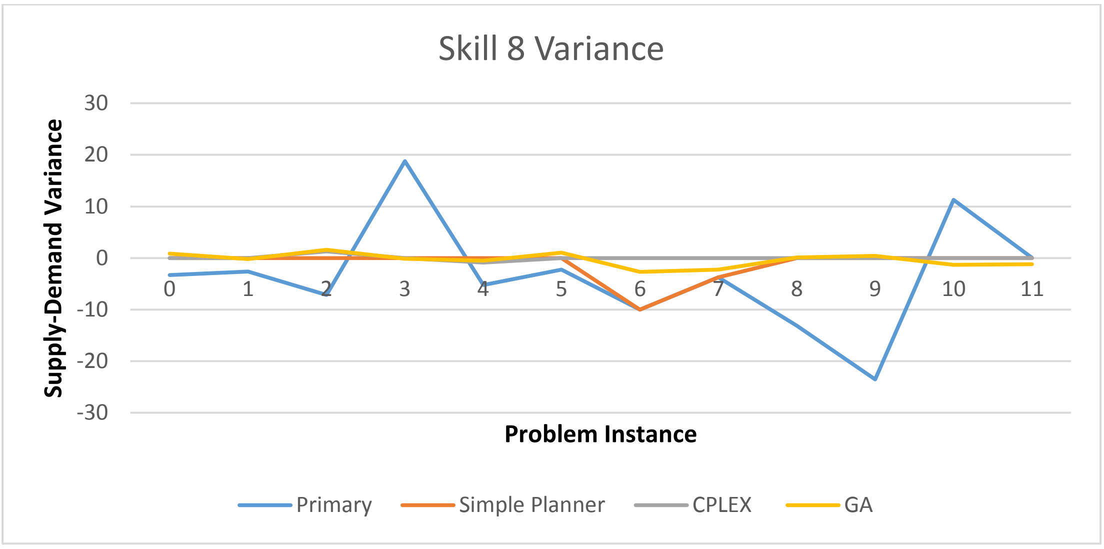

Fig. 4 Typical Variance Improvement

Fig. 4 shows a typical looking variance curve; in this case we are looking at the values for skill 8 across the 12 problems. In all cases the optimisation methods reach values the same as or closer to zero than the base solution. Most methods have managed to get close to zero in all the problems; the simple planner heuristic however shows some under-allocation in problem 6.

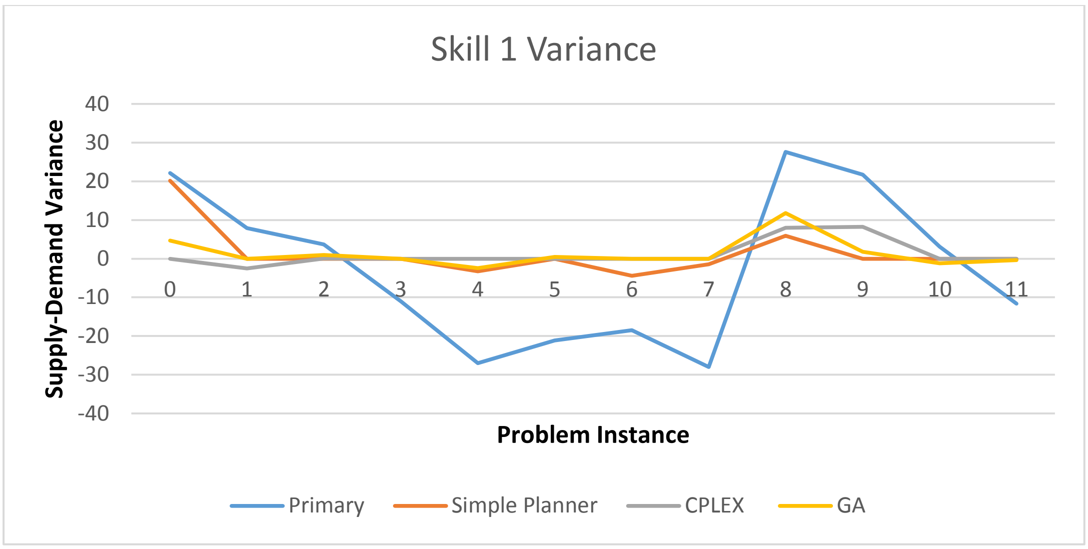

Fig. 5 Higher Variance than Simple Planner in Problem 8

Fig.5 shows a less typical variance graph. This graph is chosen as it highlights a case where the GA has produced a higher variance than the simple planner heuristic in problem 8. The rest of the problems produced typical results however.

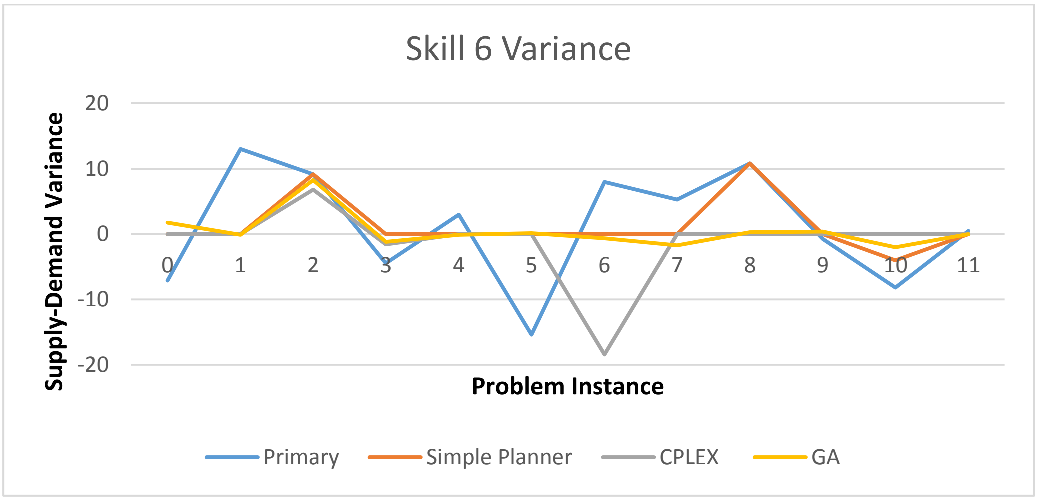

Fig. 6 Lower Variance than Simple Planner in Problem 8

Fig. 6 shows the balancing value for figure 3 in problem 8. Here in problem 8 it is the simple planner heuristic that has over-allocated while the GA has reached a balanced allocation. The rest of the graph shows typical values, with the GA generally producing similar or better variance than the simple planner heuristic.

4.5.3 Solution Run Time

The final result recorded from these experiments was the average time taken to produce a solution by the GA vs. CPLEX. The GA took an average of 0.322 seconds to complete over the 100 repetitions whereas CPLEX was only taking on average 0.012 seconds to complete. This is almost a factor of 27 quicker and shows that for the capacity-demand sub problem the size is not so large yet as to cause problems seen in chapter 2 for the run time of the linear approach.

4.5.4 Analysis

The results of the tests show a significant improvement in using the GA model over current simple planning practices, as can be seen in fig. 3. The GA produces a lower cost solution in all cases. Further illustrated by the same graph is that the GA also performs comparably to CPLEX, reaching optimal or near optimal in all cases.

The one case where the GA produces a worse skill variance, comparatively to the simple planner heuristic, for skill 1 in problem 8 (figure 3), is balanced by a better variance for skill 6 (figure 4). The differing decisions producing a lower cost for the GA solution, as seen in figure 1.

The likely reason for the cost reduction is that the simple planner has a tendency to become stuck within local optima. This is a side effect of starting from an initial point before making moves to improve the solution. A GA, in comparison, performs a more robust exploration of the solution space.

Furthermore, although we can see that the GA is close to the optimal solution in most cases, the run time is still longer than using the linear approach for this sub-problem as it is not yet large enough to start causing the large increases in run time observed when looking at the full tactical planning problem.

4.6 Conclusions

In this chapter the capacity-demand allocation problem, the follower portion of the nested bi-level formulation of the tactical planning problem, has been defined and explored. Three methods to find solutions to the problem were compared. Two typical examples of methods to solve similar problems, a GA and linear programming model, were chosen. The final model was a greedy hill climber algorithm currently applied to solve this problem in a real world planning application.

It was shown that this GA produces superior results to the simple planner heuristic, providing a lower cost solution in all problems they were applied to. However, although it was demonstrated that the GA presented produces near optimal solutions in all the test problems it does, in these experiments, take almost 27 times as long to do so. This proves the GA is potentially a strong alternative to the badly scaling linear programming approach for the overall problem but that for solving this sub-problem the linear programming approach should be used.

Chapter 5

Solving a Subset of the Tactical Planning Problem using a Bi-Level Model

5.1 Introduction

Following the work in chapter 4 to identify a suitable solution method for the capacity-demand matching follower, in this chapter we introduce the leader model to create the nested bi-level model. We continue limiting the model to a single area, which reflects how manual planners operate, however the period dimension is introduced. As such the capacity-demand allocation model is extended to add the modelling of rollover of incomplete tasks to the next period as well as introducing the installation selling lever from the tactical planning problem. To create the bi-level model a representative subset of the capacity decisions are also chosen to be implemented as the leader model. The three levers chosen generally cover key capacity decisions. Overtime models modifying capacity available to resources, contractors model the application of external capacity and reductions models the movement of whole resources. These levers available to the leader will set the capacity constraints for the follower model to then attempt to minimise missed fault tasks whilst creating installation appointment availabilities. The overall cost is then that of the used capacity levers combined with the results of the follower models allocation.

As chapter 4 showed the linear model to be the superior choice for the demand-allocation problem it is chosen to solve the follower model here. Chapter 2 showed that nested bi-level models tend to use metaheuristics for the leader model. Given the performance seen by the GA at solving the demand-allocation problem it was chosen as the metaheuristic to be applied to the leader portion of the problem here.

For this chapter the case study data outlined in 1.3 is also used and the results of the bi-level solutions compared with the actual decisions of the real world planners.

In section 5.2 the Bi-Level model used to solve this problem is defined, both the leader and follower portions. Section 5.3 then covers the results of some experiments performed using this model on some real world data where the resulting outputs are compared with the actual decisions taken by the planners at the time, before the chapter concludes in section 5.4.

5.2 Bi-Level Model

In this section the bi-level model used to solve the subset of the tactical planning problem is specified. A nested model, as seen in chapter 2, is defined. First the general structure is defined, followed by detailed descriptions of the leader and follower models. Finally, the model configuration used to achieve the required behaviour is outlined.

5.2.1 General Structure

To recap, the general structure of a bi-level model is given by the equation:

$$\begin{aligned} &\min_{x\in X, y\in Y} F(x,y) \\ &subject\ to: \quad && G_i(x,y) \leq 0, for\ i \in \{1,2,..,I\}; \\ & && y \in \arg\min_{z\in Y} \left\{ f(x,z): g_j(x,z) \leq 0, j \in \{1,2,\dots,J\} \right\} \end{aligned} \tag{28}$$

Here *F(x,y)* is the leader model with the associated constraints *G(x,y)*, similarly *f(x, z)* denotes the follower model with the associated constraints of *g(x,z)*. In the case of the model developed in this chapter, to solve the tactical planning problem, the leader model's decision variables, *x*, are not included in the fitness function of the follower model. Instead they only affect the constraints for the follower model. Thus, for the purposes of this model, the follower model's fitness function can be simplified to *f(z)*. The leader and follower models for this solution are described in sections 5.2.2 and 5.2.3 respectively.

5.2.2 Leader Model

The leader model controls the capacity levers that set the constraints for the follower model. These encompass the following decision variables

1. Overtime - applied to each resource
2. Contractors - Number of contractors applied to each skill
3. Reduction - Number of resources to remove from the plan

Each of these variables require an entry for each period in the plan. To solve the leader level of the problem a genetic algorithm was used, a detailed description of a GA can be found in section 4.3.1. A GA was selected as the literature showed meta-heuristics are generally used for this level when taking the nested approach to solving a bi-level model. The GA being specifically selected as a common example of a meta-heuristic used that has been shown to be effective in this field already in chapter 4 whilst also being applicable in a practically deployed application in java (as required by the industrial partner) using the watchmaker framework. The following sub-sections of this chapter describe the problem specific components of the GA such as the chromosome structure, followed by the constraints applied to the problem, the evolution methods selected and finally the fitness function.

5.2.2.1 Chromosome Structure

Fig. 7 shows the structure of a solution to the leader model. The chromosome is split into three main sections that define the decisions for contractors, overtime and reductions in the plan similar to the multi-section approach demonstrated in chapter 4. Within the contractor section there is an allele, C_n, for each of the N skills contractors can be applied to. For overtime and reductions there are alleles O_m and R_m respectively for each of the M resources in the plan. Each allele is a list of values, P_t, for each of the T periods in the plan that represent the value for that allele in that plan period. The values within the contractor and overtime alleles were all positive real numbers with the values in the reductions being natural. For example, if the value of P_1 for the allele O_2 is 3.5 then that indicates that resource 2 has been allocated 3.5 additional overtime (in whatever unit the model inputs used, typically FTE) in the first period of the plan.

Decoding the chromosome then gives the amount of overtime and reductions applied to each resource on each period of the plan along with the amount of completions allocated to contractors on each day of the plan. This modifies the available capacity, thus changing the capacity constraints in the follower model.

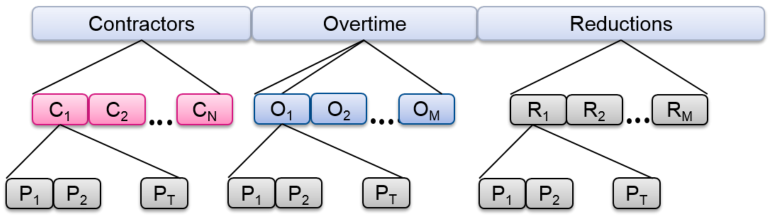

Fig. 7 Leader Model Solution Chromosome

An example chromosome for a simple problem with 2 skills and 2 resources with 2 periods in the plan would look like this:

[[2,1], [0,3]],   [[4.1,1.4], [0.2,5.2]],   [[4,5], [6,7]]

The first section with [2,1] and [0,3] indicating the completions given to contractors for skills 1 and 2 respectively. With 2 being allocated to skill1 in period1, and 1 being allocated to skill1 in period 2. Then the 0 and 3 are allocated to skill2 in periods 1 and 2 respectively. The second section indicating time allocated to resources 1 and 2 in each period. With [4.1,1.4] being time overtime allocated to resource group 1 in periods 1 and 2 respectively and [0.2,5.2] being overtime applied to resource group 2. Finally the last section indicates removal of resources from resource group 1, [4,5], in periods 1 and 2 respectively and similarly from resource group 2 ([6,7]).

5.2.2.2 Leader Model Constraints

Constraints applied to the leader model were of two types, the first to set the budget for the solution and the second to ensure correct behaviour of the model. We define the set of decision variables for the follower model as *Y.* Within the leader model we define the following decision variables. c_{nt} denotes the number of contractors applied to skill n in period t. Overtime and reductions applied to resource m in period t are defined as o_{mt} and r_{mt} respectively. If we further define the variable τ_{mt} as the amount of time available for resource *m* in period *t* then the leader portion of the problem can be defined with its constraints as:

$$min\!: F(c,o,r,y)\ for\ c{\in}C, o{\in}O, r{\in}R, y{\in}Y \quad (29)$$

subject to:

1. $\sum_{t=1}^{T}[\sum_{n=1}^{N} c_{nt}] \ \le C_b$

2. $\sum_{t=1}^{T}\sum_{m=1}^{M} o_{mt} \le O_b$

3. $\sum_{m=1}^{M} r_{mt} \le \ R_b \forall\ t\ \in T$

4. $o_{mt}\ \le O_x \ \forall\ m\ \in M, t\ \in T$

5. $o_{mt} +\ \tau_{mt}\ \le\ \tau_x \ \forall\ m\ \ M, t\ \ T$

The first three constraints for (29) are budget constraints which are simply the maximum allowed contractors, C_b overtime, O_b and reductions, R_b, allowed within the plan. For contractors and overtime, this just sets the maximum total values that could be applied to each of the solution sections. Reductions maximum was slightly different in that it was applied to each specific period of the plan, e.g. a value of 5 for the maximum reductions constraint means that on each period of the plan there cannot be more than a total of 5 reductions applied across all of the *M* resources.

The remaining constraints were to keep the solutions within the feasible bounds for a plan solution. These were constraints to the maximum amount of overtime that could be applied to one resource in a period,O_x, and the maximum total time a resource could have in a period, τ_x. The latter is there to ensure that if a resource already has 8 hours available time on a given day and the max they can work in a day is 10 hours, the model won't allocate more than 2 hours overtime to that resource in that day.

5.2.2.3 Evolution Methods

The GA uses tournament selection (Goldberg & Deb, 1991) to select the parents for subsequent generations chosen as a standard selection operator that gives a good mix of selecting the best current solutions whilst still keeping some lower fitness members for diversity of the population. In tournament selection parents are selected for the subsequent generation by randomly picking two members from the current population and whichever of those has the higher fitness value is selected.

Mutation chance is configured separately for each of the 3 sections of the chromosome to give individual control to each sections exploration rate. Mutation operator used causes a random single value to be increased or decreased by a proportion of the current value for the contractor or overtime sections, with the proportion randomly chosen from a Gaussian distribution with mean 0 and standard deviation of 0.333. A random Boolean is selected to decide if that mutation will be an increase or decrease then the value is modified by adding or subtracting the Gaussian value times the current value. For the reductions section, a mutation just increases or decreases the reduction amount for that resource in that period by 1. All of the mutations are restricted to only allow solutions that will meet the model constraints. If a solution is mutated then a check and repair process is performed that will modify the solution to ensure it remains within the valid solution space if required. It does this by proportionally reducing all of the alleles associated with that lever to bring them back within the max constraint. For example if the total overtime is 11 but the max allowed is 10 then every overtime allele value is multiplied by 10/11.

Crossover is performed using a multi-point crossover process with a configurable number of crossover points and probability of occurring to allow control over the exploitation rate of the algorithm. Crossover is allowed to occur within sections as well as between sections, however crossover that splits a section will go through the same check and repair process applied after mutation to ensure that the max constraint for that section is not broken. Thus some random points on the chromosome are selected and the alternating sections created by these points are swapped between the parents to create the two new child solutions.

5.2.2.4 Fitness Function

The fitness of a solution is calculated based on the cost of the budget use by the leader solution plus the cost achieved by the follower model with those capacity constraints. If the cost for contractors, overtime and reductions use are defined as ω_c, ω_o and ω_r respectively and the cost for the follower model is given by *f(z)* where $z \in Y$ is the optimal solution of the follower model for the current leader solution then the fitness function can be given as:

$$F(c,o,r,z) = \sum_{t=1}^{T}\left[\sum_{n=1}^{N}[\omega_c c_{nt}] + \sum_{m=1}^{M}[\omega_o o_{mt} + \omega_r r_{mt}]\right] + f(z) \quad (30)$$

5.2.3 Follower Model

The follower model matches the available capacity to the demand, attempting to meet the workstacks targets. The matching occurs across the dimensions of skill and time, with the available time for each resource in each period allocated to the skills they can perform to complete jobs in the workstacks associated with those skills. The model is chosen to be built using linear programming as it is both an approach used in the literature and performed well on this problem in chapter 4. Here we first describe the linear formulation of this problem before expanding on the methods used to optimise towards the target workstack levels.

5.2.3.1 Linear Model Formulation

With the addition of the installation job selling lever, the decision variables to indicate job completions are now split into those that are fault jobs and those that are installation jobs. Installation jobs are further subdivided into target jobs and additional jobs to model the non-linear nature of that process. Installation jobs up to target levels getting greater reward with then a much smaller reward for any additional installation jobs applied. Additionally there are extra decision variables associated with fault jobs that indicate any incomplete jobs rolling over to the next period in the plan. These are also split into target and additional to allow modelling of targeting to complete a specific percentage of fault jobs on a given day. This is expanded on in 5.2.3.2 however first we expand *Y*, defining the follower decision variables as *a*, *d*, *d'*, *d''*, *L* and *E*. Here,

a_{mst} is the allocation of time for resource *m* to skill *s* in period *t*,

d_{st} is the completion of a fault job requiring skill *s* in period *t*,

d'_{st} is the completion of a target installation jobs for skill *s* in period *t*,

d''_{st} is completion of an additional installation job for skill *s* in period *t*,

L_{st} is the rollover of target fault completions for skill *s* to period *t+1*

E_{st} is the rollover of the remainder of the fault workstack for skill *s* to period *t+1*.

The mechanics for the workstack rollovers and target installation completions will be explained in the next sub-section. With these decision variables defined, we can now specify the follower model mathematically as:

$$min: f(a, d, d', d'', L, E) = \sum_{t \in T} \left[\begin{array}{c} \sum_{m \in M} \left[\sum_{s \in S_r} \omega_m a_{mst} \right] + \\ \sum_{s \in S_{fault}} \left[\omega_{lt} L_{st} + \emptyset E_{st} - \omega_f d_{st} \right] - \sum_{s \in S_{prov}} \left[\omega_p d'_{st} + \omega_{ap} d''_{st} \right] \end{array} \right] \quad (31)$$

subject to:

1. $\sum_{s=1}^{S} a_{mst} \leq o_{mt} + \tau_{mt} \; \forall \, m, t$
2. $\sum_{m=1}^{M} [p_{mst} a_{mst}] - d_{st} = 0 \; \forall \, t \in T, s \in S_f$
3. $\sum_{m=1}^{M} [p_{mst} a_{mst}] - d'_{st} - d''_{st} = 0 \; \forall \, t \in T, s \in S_i$
4. $d_{st} + L_{st} + E_{st} - L_{s(t-1)} - E_{s(t-1)} = I_{s(t-1)} \; \forall \, t \in T, s \in S_f$
5. $E_{st} + (\rho_{st} - 1) E_{s(t-1)} \leq (1 - \rho_{st}) I_{s(t-1)} \; \forall t \in T, s \in S_f$
6. $L_{st} - L_{s(t-1)} - \rho_{st} E_{s(t-1)} \leq \rho_{st} I_{s(t-1)} \; \forall \, t \in T, s \in S_f$
7. $d_{st} - \sigma_{st} L_{s(t-1)} - \sigma_{st} E_{s(t-1)} \leq \sigma_{st} I_{s(t-1)} \; \forall \, t \in T, s \in S_f$
8. $d'_{st} \leq \rho_{st} \forall \, t \in T, s \in S_i$
9. $(d'_{st} + d''_{st}), d_{st} \geq d_{st}^{min} \; \forall t, s$
10. $a_{mst}, L_{st}, d'_{st}, d''_{st}, E_{st} \geq 0 \; \forall \, t, s, m$

The weight variables ω_m, ω_{lt}, ω_f, ω_p and ω_{ap} are the costs and benefits for using resource time, rolling over target fault completions, completing faults jobs, completing installation tasks up to

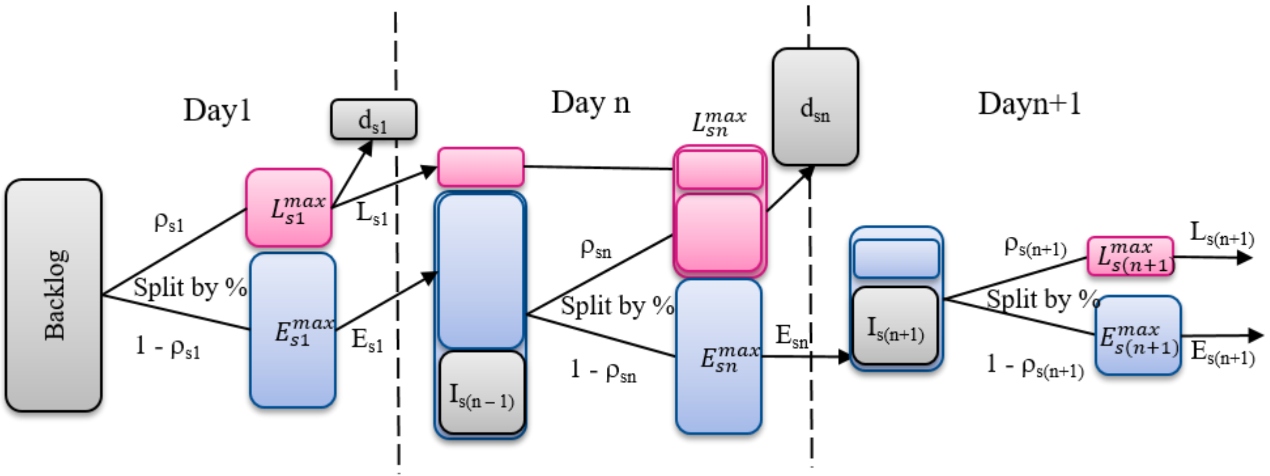

Fig. 8 Fault Workstack Propagation in the Linear Model

the installation target and completing additional installation tasks respectively. The final cost variable $\varnothing$ is a zero cost that is attached to rolling over of additional fault jobs beyond the target. There is no penalty to rolling over the additional jobs so a zero cost is applied but the decision variable is still required as these need to propagate through the periods of the plan still.

The constraints for (31) ensure a valid planning solution is produced and set the skill matching and completion targets behaviour. The 1st constraint simply states that the max amount of time used for each resource *m* in period *t* across all their skills should be less than or equal to the total time they have available as set by their base time, τ_{mt}, plus any addition overtime, o_{mt}, applied by the leader model. The second and third constraints state that the total number of completions by resources of a skill, given by the time applied to that skill, a_{mst}, multiplied by the number of jobs of that skill they can complete per unit of time applied (the productivity p_{mst}), must equal the total number of completions assigned to that skills workstack. This is d_{st} in the case of a fault skill (S_f) or $d'_{st} + d''_{st}$ in the case of an installation skill (S_i). Constraint 4 introduces a new quantity, $I_{s(t-1)}$, which is the intake of new fault jobs on period *t-1*. For the first period of the plan this value would be the starting backlog. Constraint 4 is stating that all jobs must be accounted for, either by being completed or by being rolled over to the next period through L_{st} and E_{st}. The rollover from the previous period is also taken into account in this constraint, ensuring the backlog correctly propagates through the model. The 5th and 6th constraints enforce the fault completions target mechanic by aiming to complete a proportion of the start of day backlog for each skill in each period. This is defined as ρ_{st}. The mechanics for this are explained in the next sub-section.

The 7th constraint is an additional constraint on fault completions which states that only a certain proportion of the start of day backlog for each skill in each period can be completed, defined by σ_{st}. The 8th constraint also uses the ρ_{st} quantity, however in the context of an installation skill, this is the target number of jobs the model should aim to do for installation in each period of the plan. The final two constraints set a minimum number of completions for each skill in each period, $d^{min}{}_{st}$, and also forbid any decision variables from being negative.

5.2.3.2 Optimising towards target levels

A key problem to solve when defining the follower as a linear model was how to replicate the non-linear behaviour caused by planning to target workstack levels. When the backlog is too high (above the target level), a planner puts extra resource time into that skill (if possible) to increase the number of completions and bring the backlog lower. However, once the target backlog level is met, the skill will suddenly become a lower priority. This two-stage behaviour was captured in the model by splitting the fault workstacks backlog into two portions in each period of the plan. The first is the proportion that should be completed in that period for that skill to meet the target, ρ_{st}, and second is the remainder. Constraints 5 and 6 of (31) encode this behaviour into the linear model.

Fig. 8 shows a graphical representation of how these constraints, along with the *L* and *E* decision variables that represent the rollover of jobs above and below the target level respectively, cause the required behaviour. In the first plan day, the backlog is split into two portions. First one represented by L_{s1}^{max} is the target number of completions for skill *s* in the first period, defined as:

$$L_{s1}^{max} = \rho_{s1} * Backlog \qquad (32)$$

The remainder of the backlog is represented by E_{s1}^{max}. These two variables represent the maximum values possible for the two rollover variables L_{s1} and E_{s1} respectively. The value of the decision variable d_{s1} sets the number of completions performed for skill *s* in the first period, which are first taken from the amount of rollover of jobs above the target represented by L_{s1}. In the first period in Fig. 2, this value is less than the max rollover for jobs above the target level. Therefore, some jobs above the target level and all the jobs below the target level roll over to the next period. For the next day the target completions are all the target completions from the previous period. In this case it is L_{s1}, plus the portion of the backlog to complete this period. For period *n* the backlog is the rollover of jobs below the target level, $E_{s(n-1)}$, which in this case is E_{s1}, plus the intake of new jobs from the previous period, $I_{s(n-1)}$ which in this example is I_{s1}. The backlog is then split in the same way as previously described to give the target completions for this period of:

$$L_{sn}^{max} = \rho_{sn}\left(I_{s(n-1)} + E_{s(n-1)}\right) + L_{s(n-1)} \qquad (33)$$

The jobs below the target level are therefore given by the remainder of the backlog for period n:

$$E_{sn}^{max} = \left(1 - \rho_{sn}\right)\left(I_{s(n-1)} + E_{s(n-1)}\right) \quad (34)$$

Fig 2. Further illustrates what happens if the number of completions in the period are above the target level. The transition from day *n* to day *n + 1* shows a value of d_{sn} greater than the target level, so the extra completions are removed from the remaining rollover E_{sn} . This results in a reduced backlog for the next period which continues to be split using the target proportion as before.

5.2.4 Model Configuration

With both the leader and follower model defined, the final part of creating the bi-level model is configuring the weights, ω, for all of the decision variables in order to obtain the required behaviour.

The first fix for the model weights is to address the issue of the different productivities for different skills. Some skills take little time to perform, so 1 hour applied to that skill might produce 5 completions, whereas an hour applied to another skill might only produce 1 completion. This introduces an unintended priority to the linear model as the optimal solution is going to be to apply time to the skills with the higher productivities in order to maximise the number of completions. This is rectified to some extent by introducing a fixing factor to the weights for resource time application. This is done by dividing the weight by the productivity, giving ω_m / p_{mst}. This ensures the cost for the completion of one job of skill s_1 will equal the cost for one completion of s_2 no matter what their productivities are.

With the fix in place, the weights are configured to replicate the priorities of a manual planner. They will attempt to apply extra capacity through the use of overtime and contractors to allow the fault and productivity completions to reach their targets, prioritising faults slightly over installation. For this purpose, the cost for the use of resource time, ω_m was set to 1.0, with the benefit for completing a fault job, ω_f, set to -1.0 and the penalty for rolling over a target fault job to the next period, ω_{lt}, set to 0.1. In this way, there was zero benefit for completing a fault job, if it was below the target but completing any jobs above the target level, would prevent the 0.1

penalty per job rolled over from being applied. Similarly, the benefit for completing an installation job up to the target amount, ω_p, was set to -1.05 and the benefit for any additional jobs, ω_{ap}, was set to -1.0. In this way the model prioritises target fault jobs, then target installation, then gains no further benefit from additional fault and installation.

The weights for the leader model were also set to create the required behaviour. The cost for overtime, ω_o, was set to 0.05 and contractors, ω_c, to 0.01. The reasons for these values is that contractors are generally cheaper to use, but cover less skills, than overtime so should attempt to be applied first. Also, although the cost for overtime matches the benefit for installation, the fact that the productivity value is generally > 1 means that 1 unit of overtime translates into more than 1 unit of completions and thus it still would give a benefit when those completions are used for target installation completions. Finally, the cost for reductions was set to 0.0 as the benefit for removing someone from the plan would be seen by the reduction in the cost of their time being used in the follower model if it wasn't being applied efficiently.

5.3 Results

In this section we test the bi-level model on some real-world data. First the data used is outlined, followed by the experimental technique and finally the results of the experiments are examined.

5.3.1 Experiment Data

For the purposes of this experiment one weeks' worth of real planning data was obtained for a week in October 2017. The data for 10 different areas was used to give us 10 different problem instances for the tests. The total number of skills in the problem were 13 - 6 fault skills and 7 installation skills. Three of those installation skills were able to field contractors. The number of resources varied per area, with the minimum number being 106 and the maximum 186 with an average number across all ten areas of 126. To reduce the problem size to a more manageable size, resources with similar skill sets were grouped together into resource groups creating on average 3 groups per area.

Fault workstacks for each skill in each area varied from 50 up to over 200 with additional jobs arriving per day (intake) ranging from 10 up to over 100 on average. The target percentages for these, to indicate the target amount of the workstack to be completed each day, were set to the same as those used by the planners during that week. These vary daily based on a rolling average of the percentage of jobs available daily used to define fault targets. The variance being quite large from 30% up to over 80% in some cases depending on area, skill and day of the week but with an average of around 50%. Similarly a rolling average by day of the week is also used to define installation selling targets based on the historical selling levels in those areas. Four of the installation skills were lower volume and generally had completion levels per day in the 10s whereas the remaining 3, the ones that allowed contractors, were higher volume with values in the hundreds. The total average installation per day for the different areas ranged from 265 up to 786 with an average of 391. We also obtained the decisions made by the actual planners for those 10 areas during that week to use as a comparison to the models results.

5.3.2 Experimental Method

The model was again implemented using java 1.7 with the watchmaker framework 0.71 used to implement the leader GA and the linear follower model implemented within the fitness function using SCPSolver and CPLEX 12.8. This was then all run on a windows 8.1 Lenovo ThinkPad P50 laptop running an Intel core i7 6820HQ processor with 40GB of RAM.

The model was run on each of the 10 datasets with the evolution set to stop after 10 generations of stagnation. Stagnation was used rather than generations again to allow earlier stopping or longer running depending on the complexity of the specific search space. Stagnation was reduced to 10 generations however as the resulting solution evaluations of this model were more computationally intensive than those of the model in chapter 4. The mutation probability for all 3 sections of the chromosome were set to 0.05 with the crossover probability of 0.85 as some initial parameter turning found these to give a good balance between exploitation and exploration. The overtime budget was set to 500 and the contractors to 1800 which were both slightly higher than the highest amount used by any of the planners in the problem instances to give the model a realistic max constraint that had a bit more space above what was actually utilised to ensure the solution is not constrained to just be at max the same as the actual solution generated by the manual planners. The maximum amount of reductions allowed per day was set to 20 as a reasonable max number that might be loaned out of a given period of a plan. After the models had reached their stopping point the resulting completions and capacity lever values were recorded to be compared with the planners' decisions in those instances.

5.3.3 Comparison to Planners

Table 1 shows the results of the comparison experiments. The values shown in the first column are the difference between the average distance between the target backlog level and the backlog achieved by the plan for each fault skill across each day of the plan by the model and the planner. The values for all areas are negative, which means the bi-level model was closer to the target in all problem instances and by quite a significant margin. Overall the model was 26.1 completions closer to the target on average than the actual planners.

The second column shows the difference between the average amount of installation jobs completed for each installation skill across each day of the plan by the model and the planner. This time the negative value means that the bi-level model was producing less installation completions than the actual planners, however the value is fairly low at only a 1.3 difference on average when compared to the average installation jobs of 391 within the plan.

Table 1 Results comparing model values vs real planners

Area	**Average Per Skill Per Day**		**Average Per Day**		
	Fault Off-Target (jobs)	***Installation Sold (jobs)***	***Overtime (FTE)***	***Contractors (jobs)***	***Resources (FTE)***
Area 1	-24.3	-1.3	2.7	40.5	-6.7
Area 2	-60.8	-0.8	22.6	4.4	-2.3
Area 3	-10.6	-3.3	8.4	-45.8	-2.3
Area 4	-12.6	-2.0	9.2	-29.6	-2.6
Area 5	-12.5	-1.5	-6.6	53.4	-2.3

Area	Average Per Skill Per Day		Average Per Day		
	Fault Off-Target (jobs)	***Installation Sold (jobs)***	***Overtime (FTE)***	***Contractors (jobs)***	***Resources (FTE)***
Area 6	-50.4	-1.6	32.4	-66.2	-2.7
Area 7	-20.6	1.0	17.7	-37.5	-2.9
Area 8	-18.1	-0.5	-0.8	75.2	-2.9
Area 9	-26.2	-1.9	8.0	-3.3	-2.9
Area 10	-24.6	-1.5	18.0	-30.9	-2.3
Overall	-26.1	-1.3	11.1	-4.0	-3.0

The last three columns show the difference between the average overtime, contractors and reductions applied per day by the model and the planner. Here overtime and resources are in FTE and contractors are in number of jobs handled by them. The data shows that on average the model used 11.1 more FTE of overtime daily but gave 4 less jobs to contractors and managed to remove an average of 3 resources from the plan per day. The reason resources were removed while overtime was applied would indicate that some resources didn't have useful skills on specific days and overtime was needed on the resources with the rarer skills instead. These freed resources could be loaned to other areas or given different tasks. The overall gain from the additional completions outweighs the additional cost of the overtime applied if we use the weights applied to configure the model, thus in that sense the model has produced a more optimal solution than the actual planners across all the problem instances. This is particularly true when noting that being closer to the target workstack levels for faults will improve service and reduce any penalty payments. It should also be noted that the high use of overtime by the model is also down to the fact that it is the only resource adjusting lever available. An actual planner would spread those increases around across the other levers such as shrinkage or loans. As such, with the goal being to reach the fault backlog levels and installation completion levels required, the model has shown the optimal way to reach these with the levers it had available in each case.

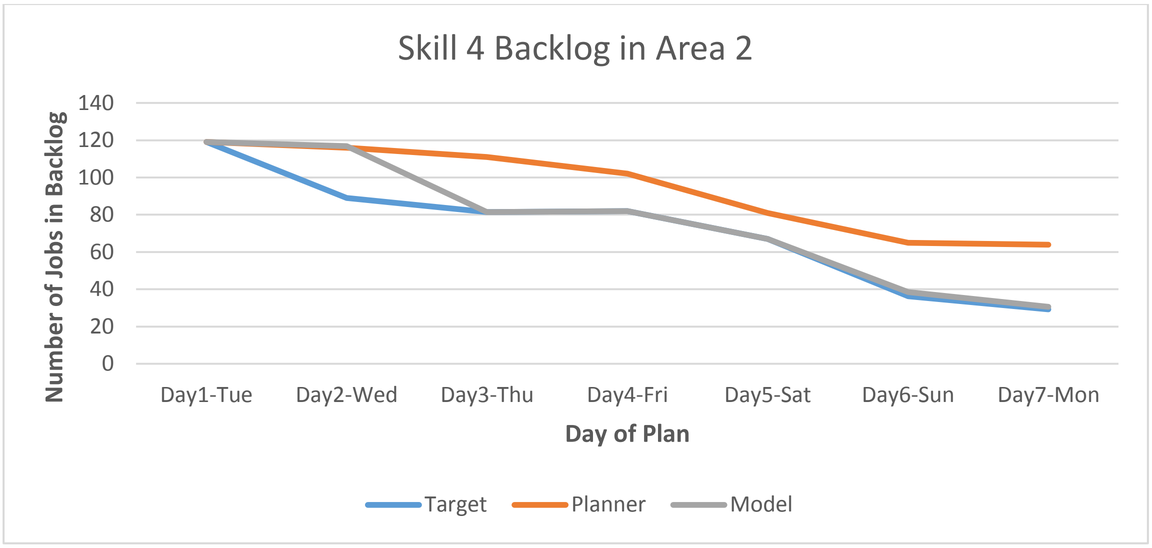

Fig. 9 Target backlog level for skill 4 in area 2 vs. that achieved by the planner and the bi-level model

Looking specifically at the area with the largest difference in the fault off-target value between the planner and the model, area 2, to see how this was achieved, the large difference can be attributed to the area starting with the backlogs above the target equilibrium as illustrated in the graph of a typical skill's backlog shown in fig. 9. Looking at the target backlog line we can see that it is decreasing across the 7 days of the plan. This is due to the target number of jobs to complete each day being a percentage of the start of day backlog, the target backlog levels therefore will

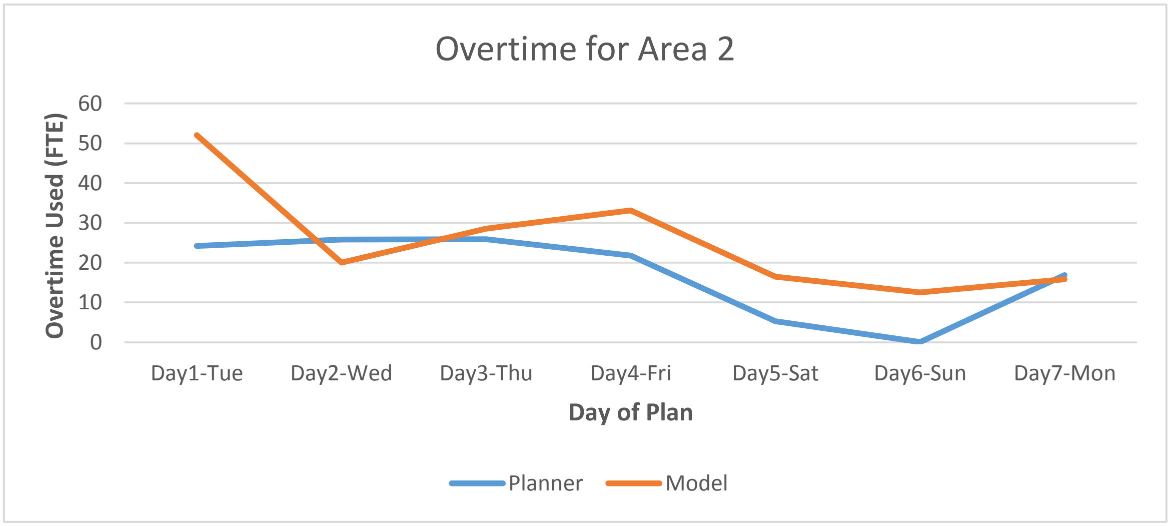

Fig. 10 Overtime applied by the planner and the bi-level model in area 2

naturally increase or decrease to the point where the percentage of the backlog on each day (the target number of jobs to complete) is equal to the intake of new jobs that day. From the planner and model's achieved backlog level lines we can see that the model performs much better in bringing the backlog down to target levels. The planner does achieve some reduction but not at the same rate as the target backlog level reduces and thus falls behind the required reduction rate. Fig. 10 shows where this improvement is mainly achieved by the model, which is in the additional application of overtime early in the plan to bring the backlog down to the target level. We see increased overtime early in the plan with the levels back to normal by the last day of the week. This is a typical pattern seen across all the areas tested, to a greater or lesser extent.

5.3.4 Solution run times

The final factor for consideration in these experiments is the solution time. This was found to vary greatly between areas with the average solution times across the 10 areas when run on a windows 8.1 laptop found to range from 1 hour 6 minutes up to 7 hours 54 minutes. The average time across all runs across all areas being 3 hours 34 minutes. Times of this magnitude bring the solution time into the range of usability in a fully automated planning solution however still far from fast enough to be used in a dynamic scenario with inputs rapidly changing. Whether this is from planners modifying values to test scenarios or through updates to any forecast values.

5.4 Conclusion

In this chapter the capacity-demand matching sub-problem from the previous chapter was further expanded through the addition of a few capacity decision levers and extension to cover additional selling and demand rollover. A bi-level model for solving this problem was defined, using a GA as the leader model that set the capacity constraints for the follower model, which was formulated as an improved linear programming version of the demand matching model. The model weights were configured to mimic the planning priorities of the actual planners, aiming to use the capacity levers to bring the fault backlogs down to target levels, whilst also providing for target levels of installation jobs to be completed. The model was also given the additional goal of attempting to remove excess resources from the plan (where possible) that could be loaned to other areas, used on other tasks or trained to cover new skills.

It was shown using real world data that this model produces solutions which compare favourably with the current planning processes by bringing the fault backlogs far closer to target levels than the planners themselves had managed whilst keeping similar installation completion levels. This is achieved by applying additional overtime to the plan but at the same time managing to reduce the contractor use and free up some resources for use elsewhere. These solutions were following the same priorities that a real planner would be taking and so the model could be used to effectively automate the tactical plan in the future.

However it was also discovered that the solution time for this bi-level model is only within the useable range for a static planning scenario where one plan is created each day. In a dynamic scenario where the problem requires solving on demand the run time is still too long. It should also be noted that the model here requires precise knowledge of available resources and required tasks. In a real tactical planning situation this is not always available. It is these factors that motivate the work conducted in the remainder of this thesis.

Chapter 6

Solving the Capacity-Demand Allocation Problem with Incomplete Data

6.1 Introduction

When looking at real world scenarios, the planning problem is not always thoroughly defined. Some elements of data are incomplete or abstracted, for example it is not possible to know weeks in advance which resources may be unavailable through illness. To still allow planning in this uncertain environment aggregated levels of resourcing are used (e.g. rather than having 10 individual resources they would be counted as a quantity of 10 of an aggregated type). Future illness, to continue the example, is thus modelled as a percentage reduction in this aggregated amount (the value based on predictions or historical trends). This aggregation allows modelling of the problem in this uncertain situation (e.g. which exact resources will be ill etc.) but means that information is lost in the process, such as the precise skill makeup of each resource available as capacity. In the previous chapters historical data was used to test the real world problem, meaning that a lot of things that may usually be unknown were known. Such as the number of resources available on each day of the plan. In reality a plan would be undertaken for future days where there is a lot of uncertainty. This means that the optimisation models defined for demand matching in the previous chapters would not be applicable in this case as there is not enough information available for them to be used. It is not just resource information that can be incomplete and thus require aggregation, demand information is also effected. For example, for future predicted jobs it is not usually possible to know precisely where within an area these will arise and thus how long they will need to complete (Campbell, 2011).

In this uncertain scenario, the process of demand allocation moves away from a precise optimisation problem to become more of a prediction problem. It is not possible to dictate what jobs will be completed on each day, since which jobs will exist and thus where they will be located as well as which exact resources will be available is not known. Instead, given the known aggregated data, the process of demand allocation becomes that of trying to work out the most likely distribution of job completions across the skills each day of the plan by each aggregated resource grouping. Thus demand allocation, where decisions are being made on where to distribute resources, is instead replaced by predictive planning, whereby the goal is to predict with high accuracy on which skills resources will be utilised once the schedule for that day is finally produced (when actual jobs and available resources are known).

As such, in this chapter we explore this predictive planning problem and design and build a neural network model to solve it. The goal being that this automated model could then be used as part of the capacity-demand allocation process when solving the planning problem in the real world situations with incomplete data. A further motivation for this work is the potential to apply it in a practical scenario to greatly improve the accuracy and speed of performing the capacity-demand allocation process of the tactical plan.

The chapter is organized as below. Section 6.2 describes the predictive planning problem and defines the models that are used in predictive planning with justification to use a neural network (NN) (Anderson, 1995) (Haga, et al., 1996). Section 6.3 introduces three different predictive planning models investigated in this chapter. It also describes the data set and the experimental setups. Sections 6.4, 6.5 and 6.6 describe the three models respectively together with the experimental results. We conclude the chapter in section 6.7

6.2 The Predictive Planning Problem

This section of this chapter first outlines the predictive planning problem before possible predictive methods to solve this problem are described along with the justification for choosing the NN which we go on to develop in the remaining sections of this chapter.

6.2.1 Problem Definition

The predictive planning problem is that of predicting the expected number of completions of each task type (tasks grouped by the skill required to complete them) by each resource type on each day of the plan given the available time of each resource type (the capacity) on each day, the number of tasks requiring completion for each skill at the start of the plan (the workstacks) and the expected number of new tasks arriving (the intake) for each skill on each day.

For the purpose of this chapter, the problem can be simplified to that of a single day. The requirement to predict the completions that day, given the current workstack levels and the capacity. The predicted completions on this day can then be used to calculate the expected workstacks for the next day by subtracting them from today's workstacks and adding on the intake. This would fulfil the requirements of the full definition, however some prediction errors would propagate to future days of the plan. This is not a problem however as the plan is updated each day, thus the current day would not contain any additional errors and the future days values are merely used to forecast any potential problems that may arise and set some base expectations for what is likely to occur.

A separate model is required to predict completions for each skill by each resource group as knowledge of who is completing what is required in the final produced plan.

6.2.2 Predictive Methods

Methods that are considered for solving this problem are the use of a rolling average, which is simply taking the average of the past x number of day's completions for that skill by that resource type on that day of the week, a linear regression (Kutner, 1996) or a NN. These were selected as some representative methods used within forecasting with increasing levels of complexity.

The rolling average is only briefly considered, although its accuracy is investigated, as it will not take into account the varying capacity levels on a given day. For example, if resourcing levels are lower than previous weeks the rolling average will be predicting completion levels that are impossible to meet with that capacity. Clearly this is not good enough for a predictive planning model. However, the rolling average does make for a good baseline comparison value to use during the early stages of later model development.

The second method considered is the use of a linear regression. A linear regression is an approach for modelling the relationship between an output variable and one or more input variables. This can be used to produce an equation to be used to calculate the output variables given our known inputs. This will take into account the capacity levels if they are used as an input to the model and thus is an improvement over the rolling average. Some initial testing however shows the linear regression approach is not providing much, if any, improvement over the rolling average. This is likely due to some complex relationships between variables that the regression is unable to capture.

Thus, the third method is considered, that of using a NN. For the purposes of this chapter we are focusing on the multi-layer perceptron (MLP) (Gardner & Dorling, 1998) variant of NN as they are widely used in forecasting problems (Zhang, et al., 1998).

MLP's incorporate three layers, an input layer, an output layer, and between those, the hidden layer. The hidden layer can contain a number of layers within itself. Each layer can contain a number of nodes, each of these nodes is connected to each of the nodes in the next layer. The number of layers within the hidden layer and the number of nodes in each of these layers is referred to as the hidden layer topology for the remainder of this chapter. The number of nodes in the input layer will always equal the number of inputs to the model, and the number of output nodes will always be one. This is because we will be building a separate model to predict each of the required output values. Fig. 11 shows an example of the layout of an MLP with three inputs and a hidden layer topology of two layers each with two nodes (or 2-2). The connections between the nodes in each layer are each assigned a weight, these combine with node biases to allow the network to perform complex non-linear calculations giving an output value based on the values input to the network. This non-linear feature is the last element required by our predictive methods to capture all the elements of our modelling problem. Thus the NN, in particular the MLP, is a suitable candidate for use as the core model of the application and is the focus of the remainder of this chapter.

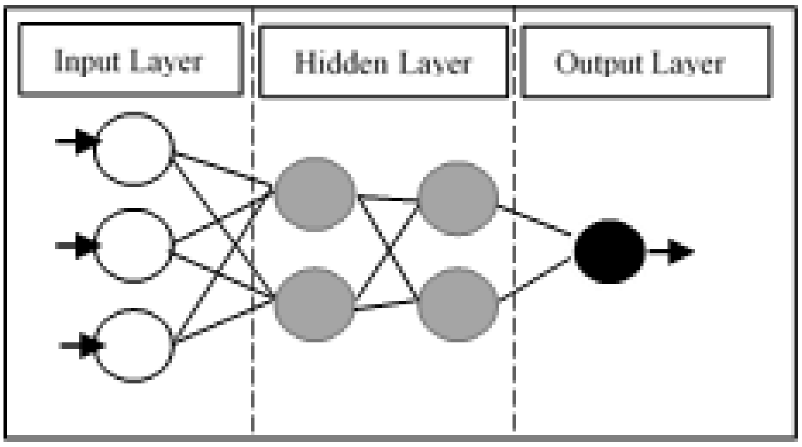

Fig. 11 Multilayer Perceptron Node Layout Example

6.3 Neural Network Model Development

During the model development process a large number of experiments are performed to evaluate and validate decisions made. All of the results presented are generated using the case study data outlined in 1.3. The data obtained contains three workforce types, six fault skills and seven installation skills. The specific data gathered for the model are the daily values for the capacity of the three workforce types and the total workstack levels of the thirteen skills, used as inputs, along the number of tasks completed by each resource type, used as the output of each model. This data was gathered for four separate areas. Two separate historical data sets are available for model building purposes, 38 weeks data for a specific day of the week – Friday and 6 weeks data for all days of the week.

Due to this data limitation we develop three separate NN models to cover three different scenarios.

1. Single day model – to evaluate the full accuracy of a day specific completion prediction model with a large dataset of 38 weeks
2. Full week model – to extend the single day model to predict completions for the whole week with a limited set of data
3. Fault only model – Specific completion prediction model for some products where installation data is not relevant and only fault data is available

Each data set is sourced from similar real world data set used to provide data in chapter 4. This includes 6 fault skills and 7 installation skills. The data in this case had 38 weeks of a single day of the week (Friday) used for scenario 1 with 6 weeks covering all days of the week used for the remaining scenarios. 4 representative areas were sourced with a range from lower volume to higher volume. The resource levels per area in the 100-200 range of available FTE with workstacks per skill ranging from around 50 for lower volume up to near 200 on higher volume skills. Completions per day ranged quite widely from single figures on average for some skills up to around 100 on some of the higher volume skills. There was also a significantly lower volume at the weekends, particularly Sunday when most skills only performed single figure completions.

For each experiment in this chapter, a separate model is created to predict the completions for each of the six fault tasks by each of the three workforce types in each of the four areas, resulting in a total of 72 models per experiment. These models are implemented using Java 1.7 and the Encog NN library (Heaton, 2015) and run on a 64 bit Windows 8.1 based machine comprising 4GB RAM and an Intel Core i5-4300U CPU. Again a java solution was chosen as it is a requirement to develop a deployable solution in the industrial partner with the Encog library being a good java based NN library. One week's worth of data is set aside as the test data set and the models are trained on the remainder. The trained models are then run on the test data and the predicted values compared with the actuals to calculate the accuracy and Pearson's correlation for each task completion in each area. The average of the accuracies and correlations over the four areas are what is presented and analysed in this chapter.

6.4 Single day model

For the single day model, initially a MLP is implemented with two hidden layers, both containing five nodes, using the backpropagation (Chauvin & Rumelhart, 1995) training algorithm as a starting point. All nodes were set to use the sigmoid activation function (Sibi, et al., 2013). Training is set to run for 500 iterations. We then tune this further through two distinct phases.

1. Input analysis: Analysis and decisions relating to the inclusion of possible inputs.
2. Training algorithms comparison

6.4.1 Phase 1: Input Analysis

The first step is analysing potential inputs and deciding on their inclusion. Inputs under consideration are the resource capacities, R, skill workstacks, S, and the historical job completions values for each skill. The resource capacities are the amount of time available for each group of resources on that day of the plan. The workstack for a skill is the number of jobs that currently exist that are awaiting completion which require that skill to complete. Finally the historical completions are the number of jobs of each skill that have been completed daily up to the current day. This is also the value that the models are looking to predict for the future dates.

The first decision to be made with regards to the model inputs is how, if at all, to use the past completion data as an additional input. Two alternatives are tested, inputting these as a time series or as a rolling average. In both cases the number of past data points included is varied from zero (no past completion data used) up to eleven. The accuracy achieved for each of these values can be seen in Table 2. The measure of accuracy used here, and in all subsequent results, is generated by calculating a weighted version of the mean absolute percentage error (MAPE) and inverting it to give the accuracy by subtracting it from 100. The MAPE is a standard measure of error used in statistics and is calculated by first calculating the absolute percentage error of each data point by dividing the difference between the predicted and actual values and dividing that by the actual. These are then summed and divided by the total number of data points to give the mean as seen in equation (35) where A_i is the actual value of data point i of n total data points and P_i is the predicted value.

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{A_i - P_i}{A_i}\right| \quad (35)$$

The problem with using this measure for the error is a bias towards data points with a lower actual, as often seen in real world data where weekend values or specific skills may have lower volumes. Data points where ns the actual is low, such as 5, then being 1 away gives an absolute error of 20%. However if the actual is 50 then a prediction that is 1 away only gives an absolute error of 2%. In both cases the prediction is only 1 away which when calculating something like resource capacity required in a planning problem the values are equivalent. To remedy this a weighted version of the MAPE is used whereby the sum of the absolute error is divided by the sum of the actuals which removes this bias as seen in equation (36).

$$weighted\ MAPE = \frac{\sum_{i=1}^{n} |A_i - P_i|}{\sum_{i=i}^{n} |A_i|} \ (36)$$

It is discovered that the accuracy is similar between the two, with the rolling average slightly more accurate. For this reason we decide to use the rolling average as the past completion input, with a duration of three selected. This gives the best correlation value and also keeps the average length low which keeps the number of lost data points to a minimum whilst still giving a slight accuracy increase over not using the average at all.

The second decision is which of the remaining inputs to use. A correlation analysis between the inputs and outputs is performed with some typical results shown in Fig. 12 and 13. The graphs show the correlation of the stated output with each of the inputs, the capacity for resource 1 to resource 3 and the workstacks of skill 1 to skill 13. Here skills 1 to 6 are the fault skills and 7 to 13 are the installation skills. Fig. 12 shows an example of typical correlation results seen across a large number of outputs. The output of completions of jobs requiring skill 2 by resource 1 is most highly correlated to the workstack for that skill. This is as expected, the more work available the more jobs that may be completed. However between the four areas we can see that the correlation with the remaining inputs varies significantly. Fig. 3 shows a less typical situation where we see that completions of skill 4 by resource 2 are more highly correlated to some of the other skills workstacks than its own workstack. This could be due to this resource only working on tasks of this skill type if there are not enough of other types available or perhaps this task is often performed at the same time as another. Again we see a large variance between correlations with different inputs between the areas, thus the decision of which inputs to use cannot just be made on an output by output basis. In order to accommodate this, and also any possible future correlation differences, a dynamic method of choosing which inputs to use is created. During training, the correlation between each input and the output is calculated and inputs are filtered out if they do not pass a set threshold.

This threshold is the focus of the experiments displayed in Table 3. The first experiment run was using a length 3 rolling average and all of the capacity and demand inputs. This showed an improvement was achieved by dynamically filtering out any inputs below a certain correlation value, with a cut-off point of 0.3 achieving the best results.

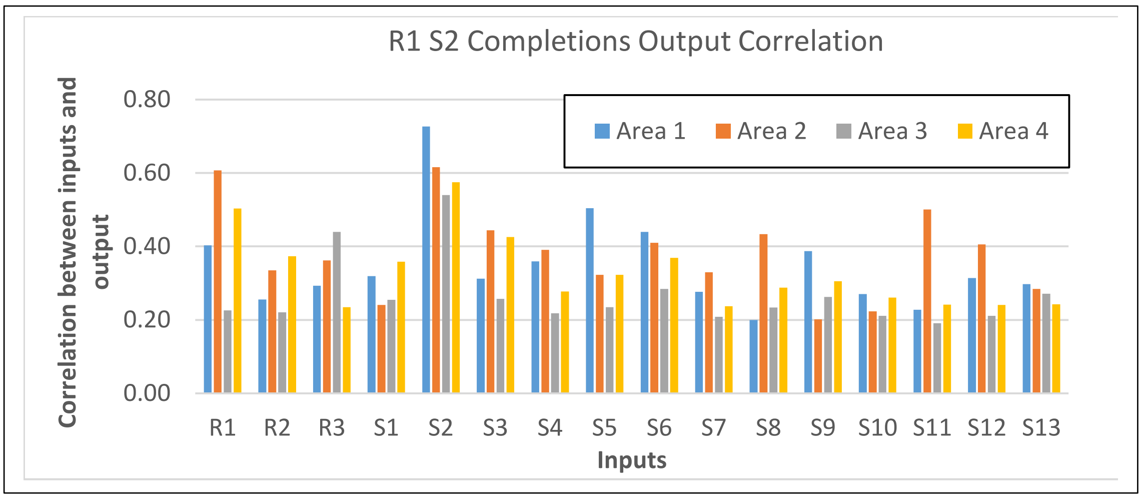

Fig. 12 Typical results where the output is most highly correlated with its own skills workstack

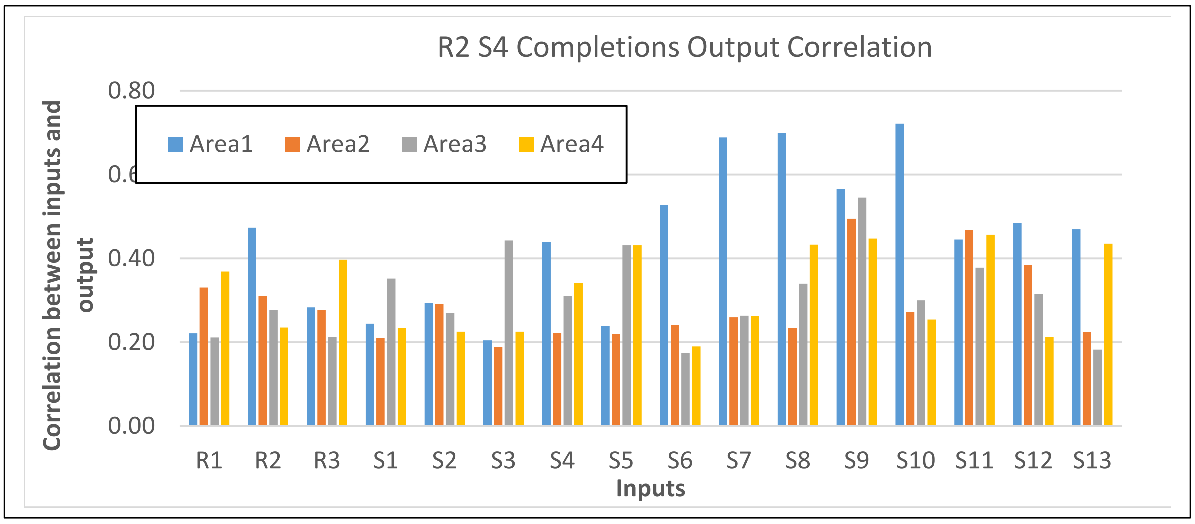

Fig. 13 Case where the output is not most correlated with its own skills workstack

Thus from the input analysis we can conclude that a rolling average of length 3 and a correlation cut-off point of 0.3 provides the best results. However both of these processes do have their cons. The rolling average for completions is only valid for the first week, after that predicted completions values start dropping in which introduces inaccuracies. It also requires additional historic data, which may not be readily available within the plan. For the correlation, it introduces additional storage requirements when saving each model, as which inputs were used in each case would have to be stored along with the model itself. For these reasons, the gains achieved through including these factors will continue to be evaluated through the additional developments in phases two and three to ensure they still outweigh the cons.

Table 2 PAST COMPLETIONS AS INPUTS

Data Points	Average		Time Series	
	Accuracy	***Correlation***	***Accuracy***	***Correlation***
0	83.8%	0.54	83.8%	0.54
1	83.5%	0.54	83.5%	0.54
3	84.4%	0.56	84.2%	0.53
5	84.7%	0.53	84.3%	0.53
7	84.8%	0.51	84.4%	0.52
9	85.1%	0.48	84.5%	0.49
11	85.0%	0.50	84.5%	0.47

Table 3 CORRELATION FILTERING

Correlation Filter	Backpropagation		Resilient Backprop	
	Accuracy	***Correlation***	***Accuracy***	***Correlation***
0	84.4%	0.56	88.3%	0.73
0.1	84.3%	0.54	88.2%	0.72
0.3	85.4%	0.61	88.9%	0.77
0.35	85.2%	0.61	88.6%	0.74
0.4	85.0%	0.60	88.7%	0.71
0.45	85.2%	0.60	87.6%	0.70
0.5	84.9%	0.60	86.6%	0.68

6.4.2 Phase 2: Training Algorithm comparison

The next step undertaken is to explore some alternate training algorithms available within the Encog library. Resilient backpropagation (Kişi & Uncuoğlu, 2005) is selected as a good candidate and tested, the results shown in the second column in Table 3 where the same measure of accuracy, the inverse of the weighted MAPE described in 6.4.1, is still used. It can be seen to produce far superior results to the original backpropagation which is likely due to the variations it introduces to the learning rate. In particular it shows a large improvement in the correlation. Thus it is used throughout the future development. It is also noticed that with the improved training algorithm the improvements from using correlation filtering have been reduced. Thus we decide to remove the correlation filtering step to simplify the training and storage processes as it isn't providing a great enough improvement to justify the complication.

6.5 Full Week Model

The next step is to expand the single day model through the creation of a model to cover an entire week. With only six weeks of data available there are not enough data points to create an individual model for each day of the week. As such we investigate three methods of constructing a model to cover the entire week.

1. Create one NN model to predict any day of the week.
2. Create a composite model combining one NN model to predict weekdays and another to predict weekends as weekends tended to behave differently to weekdays.
3. After observing that Saturdays often only have a slight reduction in completion values over a weekday a third alternative is investigated, to create the composite model using one NN model to predict Monday to Saturday combined with a separate Sunday model.

With the correlation filtering already ruled out during the single day model building process, we continue to evaluate the rolling average to ensure the gains are still enough to overcome the cons. For the full week data this average is constructed for each day of the week individually, so the rolling average input on a Monday is the average of the last x Monday's completions. The results of testing the different model configurations, along with the varying number of data points used for the average, can be seen in Table 4. Again the accuracy displayed is the inverse of the weighted MAPE described in 6.4.1.

These show that the best combination is the third option, to create a NN model for Monday to Saturday with a separate model for Sundays. The inclusion of the average also produces little improvement in accuracy or correlation, with the model producing a very high correlation to the weekday trends (e.g. lower at the weekends) even without the average to give day specific inputs. In some cases the average even reduces the model accuracy. Thus the best decision is to not use the rolling average as an additional input.

6.6 Fault Only Model

Unlike fault workstacks, which can be forecast many months in advance, in many products and services the installation workstack is only fully known on the execution day as they are not forecast for future dates. This means that these workstacks are not available to use as inputs for these products. The rest of this chapter is focused on solving the problem with only the fault workstacks available as it is one of the core requirements from the planning community. The goal is to produce a prediction at least as accurate as the current manual planning process where planners decide the completions numbers only based on the forecast fault workstacks.

The first approach is to simply run the model after removing the installation data set. Initial tests show that removing this data causes a significant dip in the accuracy of the full week model of greater than 10%.

A two phase approach is then investigated to improve the prediction model to achieve better results than the current planning processes.

Table 4 Full Week Model Setups

Data Points	**All Week**		**Weekday + Weekend**		**Mon-Sat + Sun**	
	Acc.	***Corr.***	***Acc.***	***Corr.***	***Acc.***	***Corr.***
0	86.1%	0.93	87.2%	0.93	88.3%	0.94
3	85.7%	0.92	87.6%	0.94	88.0%	0.94
5	85.6%	0.92	87.8%	0.94	88.5%	0.95
7	85.8%	0.93	87.7%	0.94	88.2%	0.94
9	86.5%	0.93	87.8%	0.94	88.1%	0.94

1. Initial Network Improvements: The first phase is an attempt to improve the neural network to produce a greater accuracy. For this we introduce cross validation into the training process, and also investigate different hidden layer topologies.

2. Overfitting avoidance: The second phase is to investigate solutions to the problem of overfitting to the training dataset. This is motivated by an initial test run of an experiment to change the number of training iterations notes an accuracy decrease when the number is initially increased. This highlights that the accuracy is perhaps being impacted by overfitting to the currently small training data set.

We describe the two phases in detail below in sections 6.6.1 and 6.6.2, followed by the final analysis of the results in section 6.6.3.

6.6.1 Phase 1: Initial Network Improvements

As mentioned above, these first experiments are an attempt to increase the accuracy by improving the network. The first of these involves introducing cross validation to the training process. The number of folds are varied, with values ranging from two to six tested, to discover which makes best use of the available training dataset. The best of these are used in the subsequent stages. The second round of experiments involves modifying the topology of the hidden layer. Early on it is discovered that using just one single hidden layer appears to be optimal, with values from three to nine hidden nodes tested, however a few multi-layer results are also included.

6.6.1.1 Cross Validation Results

Fig. 14 shows the results of varying the number of folds when using cross validation. The difference is not very large, however the best result for accuracy and correlation can be seen around three folds where 76.1% is reached. At this stage the model accuracy is still below the planners' accuracy of 82.6%, however if we look at the accuracy breakdown per task type seen in Fig. 15 for the best result of three folds we can see that in the case of task types 2 and 3 the model is producing slightly better results. The model is still underperforming substantially when predicting the remaining task types, particularly task type 4. The results look promising overall however with such a significant improvement achieved already.

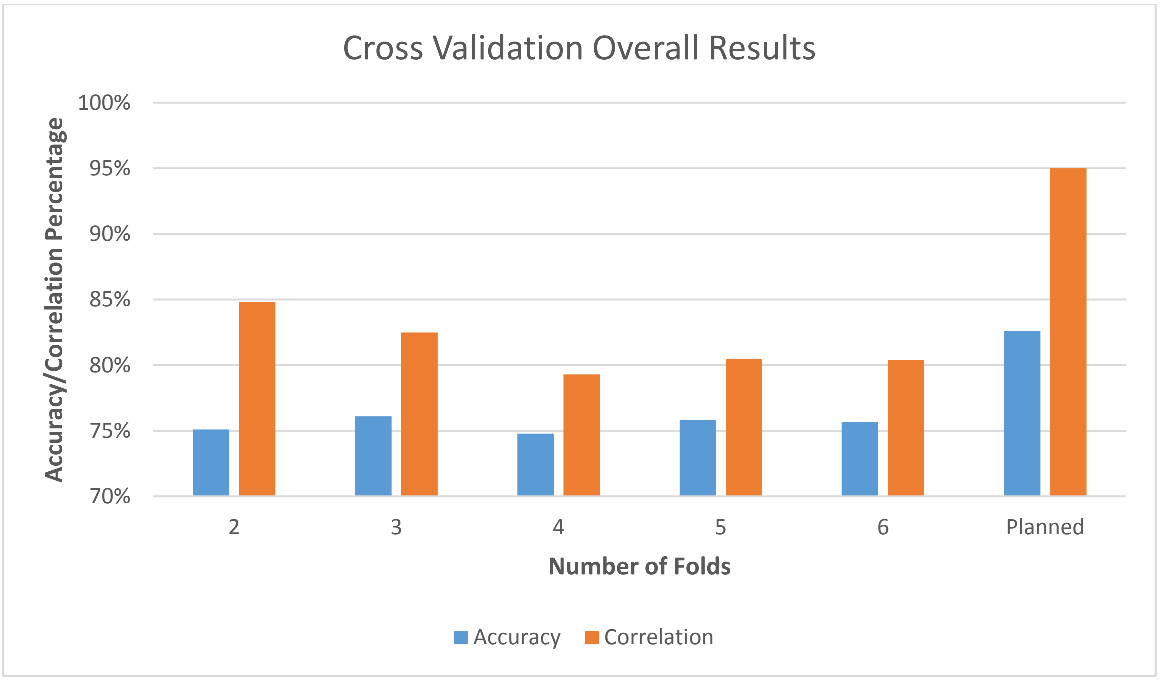

Fig. 14 Cross Validation Experiment Overall Results

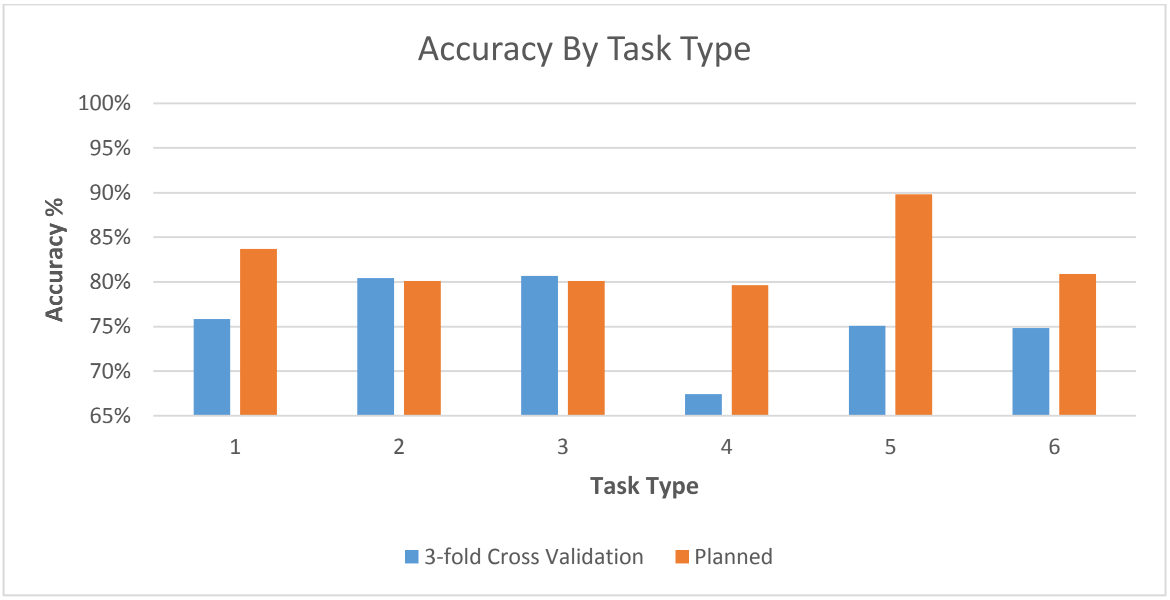

Fig. 15 3-Fold Cross Validation Accuracy Results by Task Type

6.6.1.2 Hidden Layer Topology Results

In Fig. 16 We can see the results of the hidden layer topology tests. A single number for the node topology indicates that number of nodes in a single layer, two numbers indicates that number of nodes in each of two layers. The results show that further improvements have been reached over the 5-5 topology used previously. The best results occur for using only a single hidden layer with the optimum occurring with 8 nodes in that layer producing an accuracy of 79%. This is still lower than the planners' accuracy but has halved the gap previously seen. The correlation has also bridged the gap by a similar magnitude. Looking at the task type breakdown for this improved model (Fig. 17) we see that task type 2's accuracy is further improved compared to the planner, with task types 1, 3, and 6 now reasonably close.

6.6.2 Phase 2: Overfitting avoidance

The next batch of experiments for phase 2 involve attempting to overcome the overfitting to the small dataset. The first solution trialled is using an ensemble of models, rather than a single model. Multiple models are trained for each output and the predicted value given by taking the average of the predicted value of all the models in that cluster. The final method explored to overcome the overfitting issue is the use of a new stopping condition during the training process. Instead of just running the training for x number of generations, something which we could attempt to tune in the same manner as the hidden layer topology, it is instead decided to use a more dynamic stopping condition, that of the early stopping strategy (Prechelt, 1994). During training, instead of using all of the training data set to train the model a further portion is set aside for validation at each training generation. After each training generation the accuracy is tested on the separate validation data, the theory being that as the model is trained the accuracy will initially increase for this validation data. However, once overfitting starts to occur the accuracy will then begin to decrease again on the validation data. Thus the early stopping strategy runs the training process until the accuracy on the validation data begins to decrease, e.g. once overfitting has been detected.

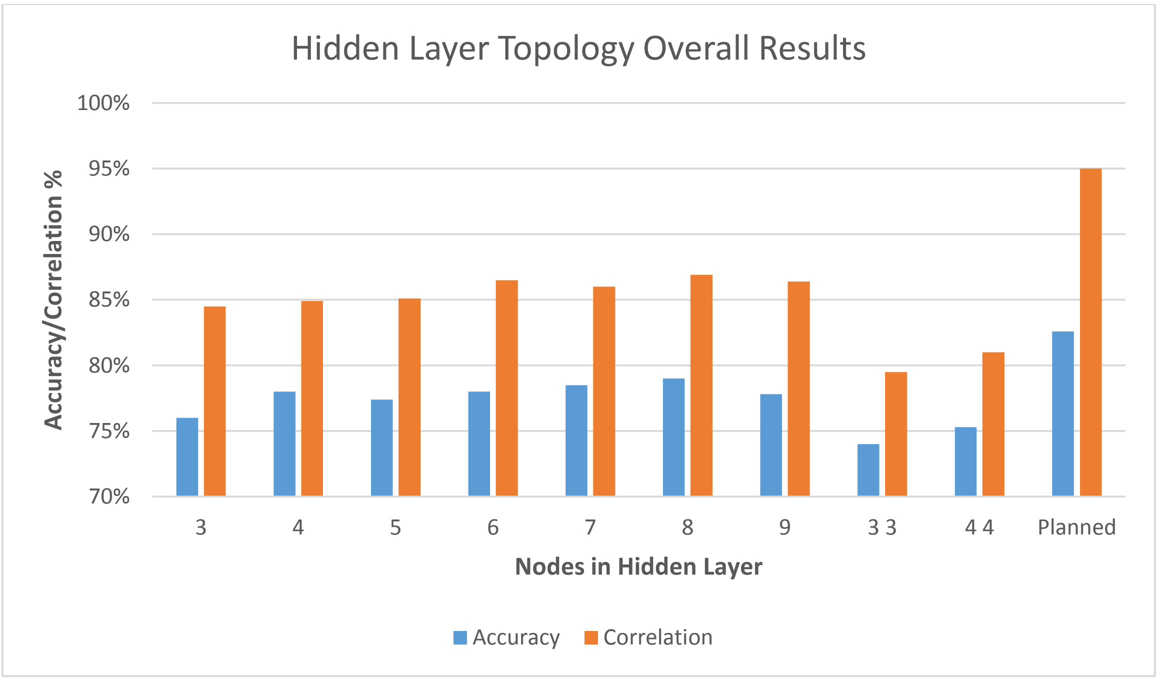

Fig. 16 Hidden Layer Topology Overall Experiment Results

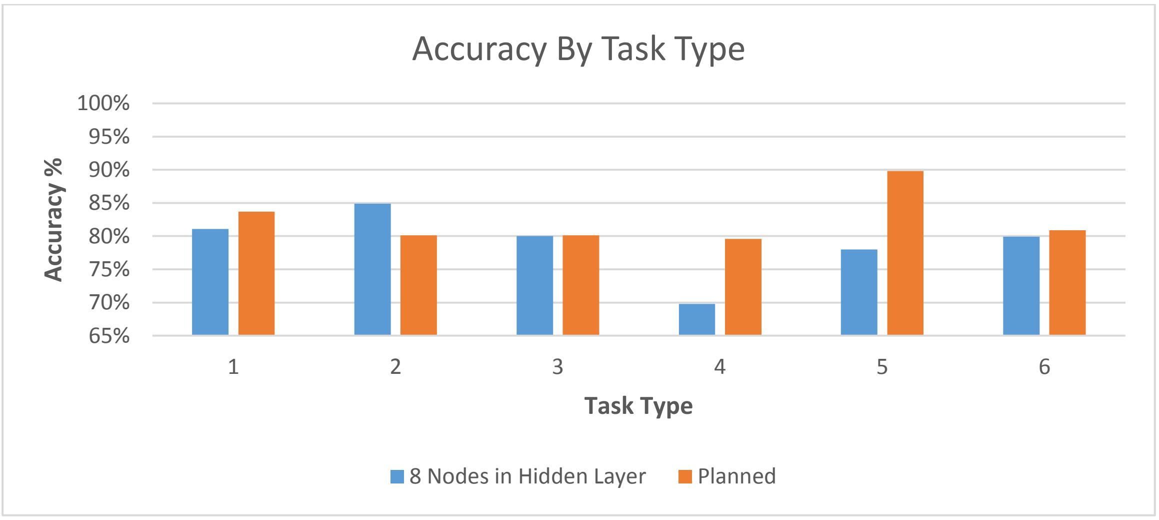

Fig. 17 8 Nodes in Hidden Layer Accuracy by Task Type

6.6.2.1 Model Cluster Results

This next experiment involves training an ensemble of models to attempt to reduce the impact of overfitting by taking their average as the actual prediction. The results in Fig. 18 show that there is a further, slight, improvement achieved of around 1% with the best coming from using a batch of 7 models giving an accuracy of 80.1%. Looking at the individual task type breakdown for this setup in Fig. 19 we can see that it provides better results than the planner in half of them now, however there is still a large deficit in task types 4 and 5.

6.6.2.2 Early Stopping Strategy Results

The final experiment performed is a repetition of the previous experiment, varying the number of models in the prediction cluster, but with the addition of the use of the early stopping strategy during model training to avoid overfitting to the small data set. This produces significant improvements, as seen in Fig. 20, with the best overall accuracy now achieved with a 5 model cluster. This gives an accuracy of 84.1%, which is higher than the 82.6% achieved by the planners in the same period. In fact, even without using clustering (with a cluster size of 1) the model was achieving 83.1% which is still slightly better than the current planning process. The correlation achieved is also up to 0.94, which is comparable to the 0.95 achieved by the planners. Looking at the individual task type breakdown again in Fig. 21 we can see that the model is now outperforming the planner in five of them, however there is still a significant deficit in task type 5. This may highlight this task type as one that is a problem for the model to predict accurately or the manual planners may understand something about it that has not been captured by the inputs being used.

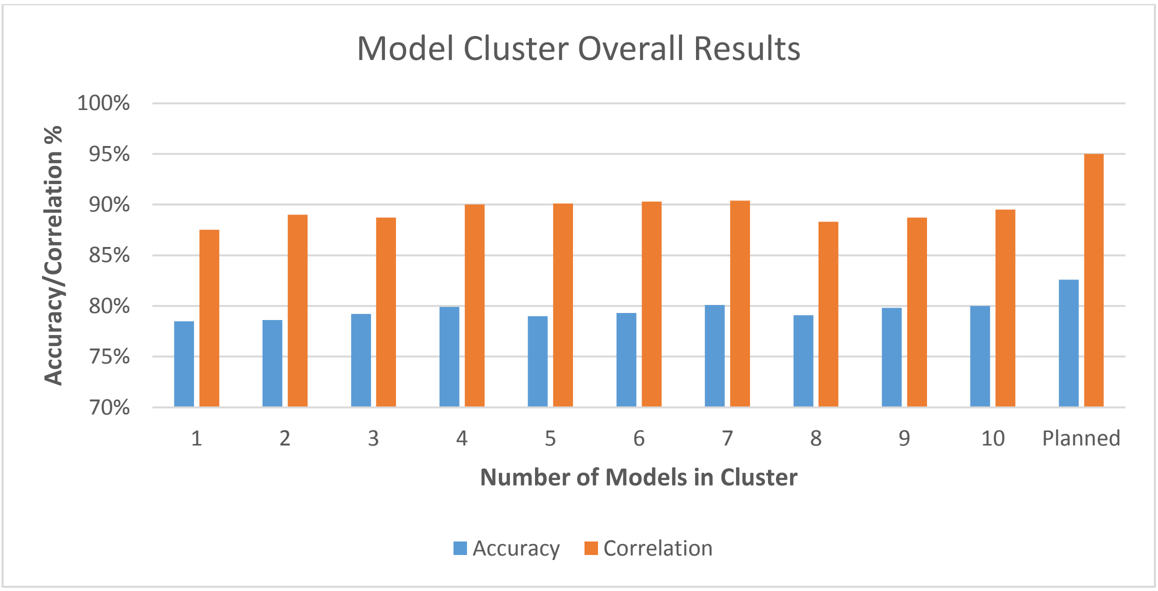

Fig. 18 Model Cluster Overall Experiment Results

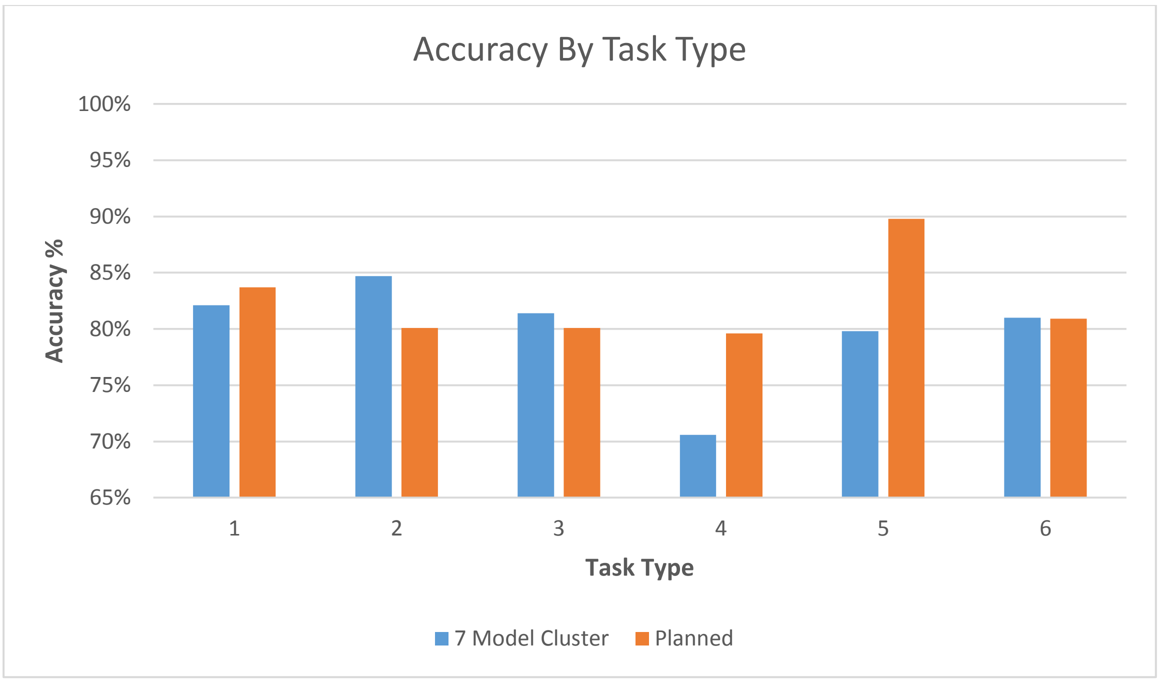

Fig. 19   7 Models in Cluster Accuracy by Task Type

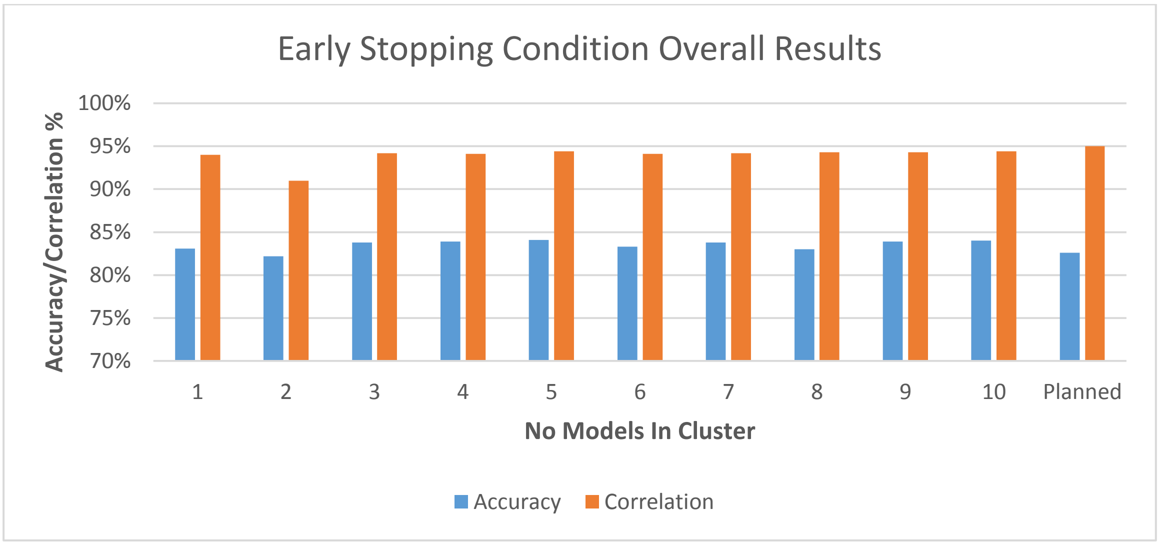

Fig. 20   Early Stopping Training Condition Overall Experiment Results

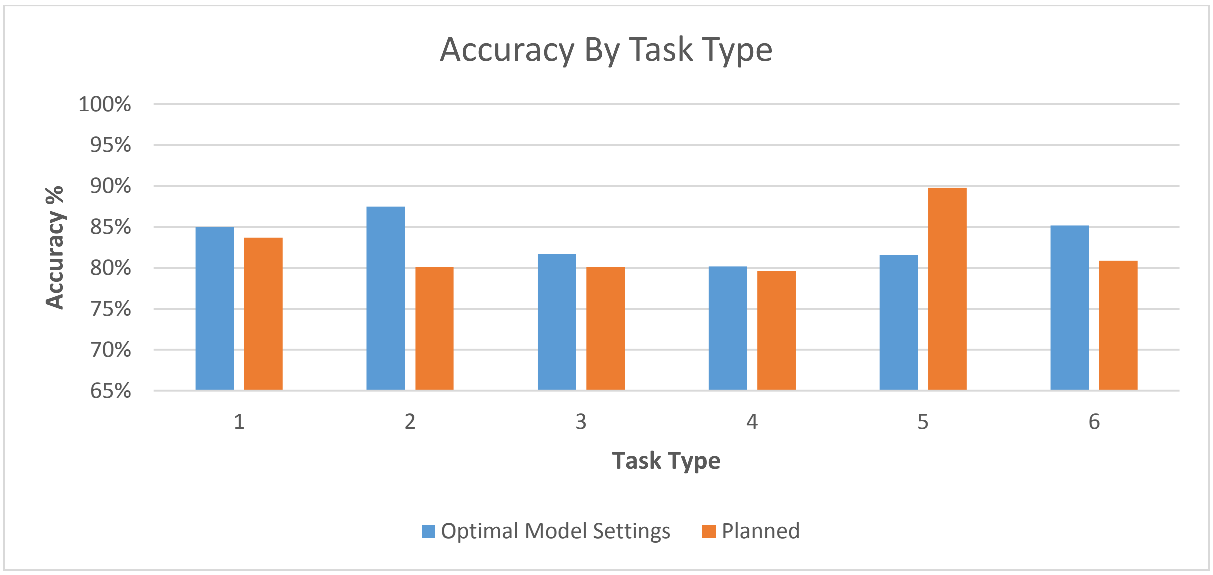

Fig. 21 Optimal Model Settings Accuracy by Task Type

6.6.3 Prediction Quality

In Fig. 22, 23 and 24 we look more in depth at how the optimally setup model is predicting the weekly trend. These graphs show the total actual completions per day across the four areas, against the total predicted by the model and the total predicted by the planners. In Fig 22. We pick out a typical example of a task type where the model is producing slightly more accurate results than the planner, as in task types 1 and 3. From this graph we can see that although the prediction is slightly off the planned, the planners were further off for most of the week. The predicted line does tend to follow the trend of the actual across the whole week also.

In Fig. 23 we can see an example where the predicted accuracy is significantly superior to the planner, as seen in task types 2 and 6. The predicted line follows the actual almost exactly for the first half of the week, although there is some discrepancy on Thursday. In general however the trend does follow that of the actual values, hitting near the exact for most of the week.

In Fig. 24 we show the breakdown for the problem task types, that of 4 and 5. Here we see that the prediction is significantly off on Saturday and doesn’t tend to follow the trend for the remainder of the week, merely taking the average point. Interestingly, although we achieve greater accuracy than the planners in task type 4 we can also see this Saturday discrepancy. In this case however the average value through the week is better than what the planners were plotting and thus we achieved a greater accuracy. The difference in these cases would suggest that the problem arises from the fact that unlike the other 4 task types the values achieved on a Saturday are significantly different to the rest of the week. Thus these two may be better served with a Weekday and Weekend model split. However this problem should not be an issue when working with larger training sets with enough data to train individual day of the week models. This can be seen in the initial high accuracy for the models on the Friday only dataset.

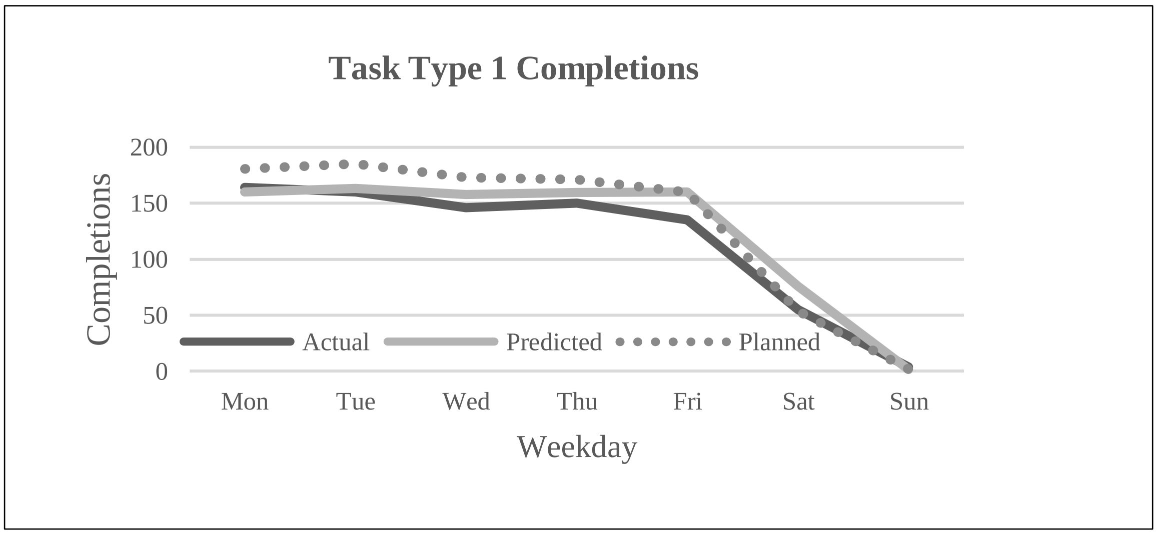

Fig. 22 Typical task type where the model is slightly more accurate than human planners

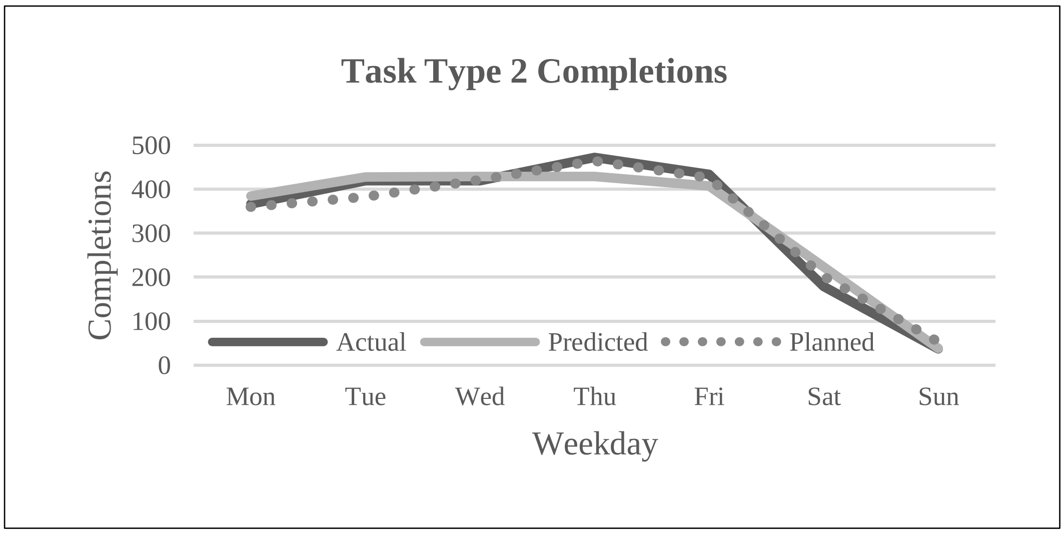

Fig. 23 Typical task type where the model is significantly more accurate than human planners

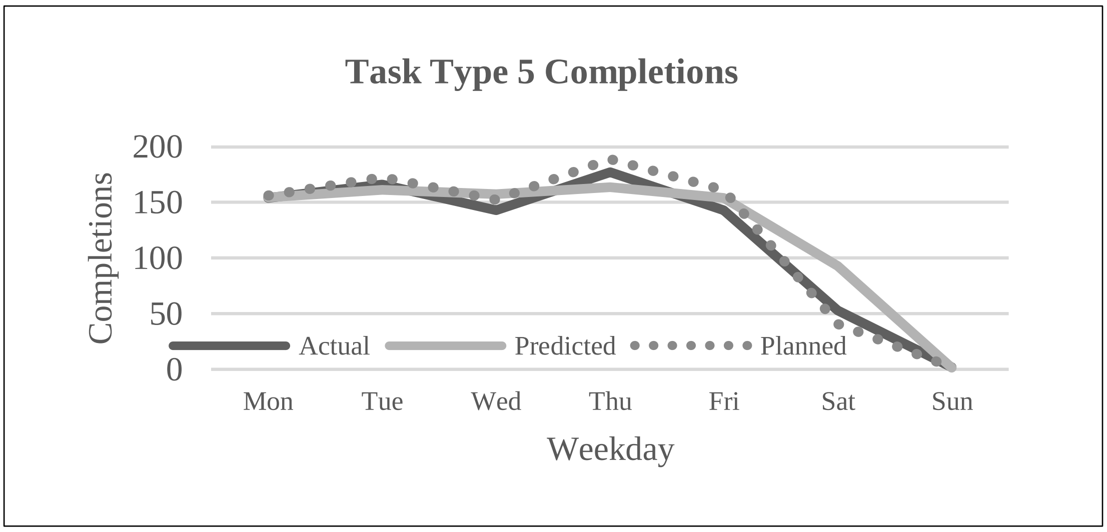

Fig. 24 Problem task type example

6.7 Conclusions

In this chapter we investigated methods to accurately predict completions in the real world situations where there is incomplete planning data. We defined the predictive planning problem and identified some possible methods that could be used to solve it. From these methods we chose the NN to achieve our goals. We describe several scenarios we used to develop and refine the model through rigorous experimentation, aiming to achieve the goal of creating a system that is at least as accurate as the current manual planning process but would complete the task in far less time and could be used as part of an automated planning algorithm when there is incomplete data.

The initial general model developed in this chapter produced very acceptable results. We managed to produce a very high accuracy whilst also ruling out the need to utilize any techniques, such as correlation filtering, which would add an extra calculation burden to the training process and also complicate the storage and use of these trained models within the final application. We also ruled out the need to use past completion data, which would begin to utilize predicted data points as the model is run on subsequent days.

Moving onto the specific scenario where there are less inputs available to the model, we have managed to further improve the initial model to the point where it is producing a better overall accuracy than the current planning process despite having a limited dataset. Improving upon the current accuracy has met the goal of this chapter as it will allow the use of this model to perform demand allocation in the real world scenarios where not all data is available.

Chapter 7

Solution Evaluation with Incomplete Data

7.1 Introduction

In order to introduce automation to the tactical planning process there needs to be a way to accurately evaluate different plan solutions. The cost of resources used forms part of this evaluation, and is relatively simple to calculate, the remainder coming from the cost associated with which tasks are being completed and when. In previous chapters we have just been assigning a cost to missing a task based on the priority for that type, however a key component of this cost, in real world scenarios, is derived from the service level agreement (SLA) (Verma, 2004) of these tasks. In general the service level agreement states that a certain percentage of those tasks must be completed successfully within a specified time period. Different tasks can fall under different service level agreements. Failing to meet these agreed levels results in financial penalties for the company.

A formula for this is formulated in section 3.4.3.2 however using this to evaluate a plan solution is complicated by the fact that planning usually occurs at the aggregate level, as described in the chapter 6. The problem this introduces to the evaluation of a solution is that, after grouping tasks into the workstacks based on the skill required to complete them, the individual tasks within each workstack may have different service level agreements. Thus available inputs to the problem do not contain data at a high enough level of detail to allow grouping by service level agreement.

One solution is to further subdivide each workstack by the service level agreement of the tasks contained within. However this level of detail is not always easy to achieve, particularly in the case study in this thesis, where it will result in some aggregate groupings having very low volumes and thus be difficult to predict accurately. The intake predictions for the plan are also not something being explored in this thesis, instead focusing on solving the planning problem itself.

An alternate solution therefore, and the focus of this chapter, is to attempt to predict the service levels achieved for each service level agreement using the currently available plan inputs. This prediction, if accurate enough, could be used to evaluate plan solutions. A further advantage of this approach is that it could also be used as a standalone solution to assist the current planning process, allowing senior planners to identify where action is required to avoid failing to meet service level agreements.

For this purpose we present a neural network (NN) (Anderson, 1995) (Kourentzes & Crone, 2010) model to predict the upcoming service levels based on the current status of the plan. A NN model was chosen for this purpose as it had already been shown to perform well when predicting the completions using real world planning data in the previous chapter. We investigate the accuracy this model achieves using some anonymized real world data, since service level results are commercially sensitive information, from our case study. The model predicts the service levels daily. It does so in such a way that aggregate predictions can also be produced allowing a weekly prediction that can be presented to management to be useful in the current processes of the case study business. As such the actual volume of tasks successfully completed are predicted and then used to calculate the service level percent, rather than predicting the percentage directly.

In this chapter we first define this service level prediction problem in section 7.2, describing the general problem and introduce the specific real world example. We then outline our NN model used to solve the problem in section 7.3. Section 7.4 contains the results achieved by this solution, looking at both the daily accuracy and the aggregated weekly prediction accuracy, before we conclude in section 7.5.

7.2 The Service Level Prediction Problem

In this section we define the service level prediction problem. First we define the general service level prediction problem before describing the dataset used in this chapter. We then perform some analysis on the available inputs in the data to further refine the problem definition for the real world example we are solving.

7.2.1 General Problem Definition

The service level prediction problem is that of accurately predicting the percentage of tasks, *Ri*, by service level agreement, *i*, which will be successfully completed on time given the current state of the plan. The time allowed varies depending on which service level agreement the task in question is covered by.

The current state of the plan is defined by the current planned completions, *Cj* by skill, *j*, the start of day workstack levels, *Sj*, by skill, *j*, the intakes *Ij*(the number of new tasks entering the plan daily) by skill, *j*, and the available capacity, *Vk,* by resource types, *k*. Additional features of the plan, for example overtime decisions, generally serve to modify the values already listed. Thus those inputs are an adequate representation of the plan state that is relevant to the service level outcomes and can be used to define the problem.

$$R_i = f_i(C, S, I, V) \qquad (37)$$

Here *C*, *S* , *I* and *V* define the input set of all completions, workstacks, intakes and capacities respectively, f_i is the function of these inputs to produce the service level of service level agreement *i*.

In addition, given the nature of a service level agreement, that it is a commitment to complete tasks of that type successfully within a certain time frame, historic values of the inputs also need to be considered. For example, if the service level agreement for a particular task is to complete a certain number successfully within two days then the intake from two days ago would contain some tasks that required completion by today. Further to that, the historic number of jobs being completed and resources available, which might affect what types of task are being completed, would influence which specific types are still left requiring completion by today. We further define C_{jt} as the planned completions by skill *j t* days before the prediction date, similarly for S_{jt} and I_{jt} as the workstack and intake for skill *j t* days prior respectively and V_{kt} as the capacity for resource type *k t* days prior. For example the input C_{j2} would be the planned completion for skill *j* two days before the prediction date. Defining *T* as the maximum number of days and the sets C_T, S_T, I_T and V_T as the sets of all completions, workstacks, intakes and capacities respectively, where 0 <= t <= T gives the updated problem definition.

$$R_i = f_i(C_T, S_T, I_T, V_T) \quad (38)$$

7.2.2 Problem Example

The data used in this chapter consisted of 203 data points, each data point representing a day. For each data point (or day) there was the plan state (*C, S, I, V*) and resulting service levels, *Ri*, for 56 separate areas. The plan state included the capacity levels for 3 resource types and the completions, intakes and workstacks for the 6 fault skills giving 21 input values for each plan day. Resource capacity levels varied by day of the week and type from only low single figures on the lowest volume resource type, through from 0 on Sundays to 10-20 for the mid volume resource type with the final largest ranging from 10-20 on Sundays up to around 200 during the week. The completions, intake and workstack levels similar to those from the data set used in chapter 5. Namely workstacks of around 50 to 200 on average per skill along with intakes from 10 to 100 and completions of a similar level. The number of failed tasks by type was very low volume in the single figures daily with success being close to the completion levels. Following the work in chapter 6 the installation skills are not included as the real world data does not track intakes and workstacks across the whole plan. The service levels were for tasks on two different service level agreements.

To allow the outputs to be aggregated the number of tasks successfully completed, Y_i, and the number failed, F_i, require predicting separately for each service level, *i.* This way they can be summed to calculate the percentage of tasks successfully completed at any reporting level required. For example we can produce a weekly service level report or an aggregated value for multiple areas. The resulting service level for any aggregated level being calculated using the total success and failure predictions of the individual predictions as follows.

$$R_i = {}^{Y_i}\!/_{(Y_i + F_i)} \qquad (39)$$

Here Y_i and F_i are still a function of the current plan state for the prediction date and the plan state for the past *T* days giving similar equations to (38).

$$Y_i = g_i(C_T, S_T, I_T, V_T) \quad (40)$$

$$F_i = h_i(C_T, S_T, I_T, V_T) \quad (41)$$

Another option would have been to predict the number of successfully completed tasks and the total number of tasks required of that type on that day. The problem with that approach is that with the prediction models being separate it may have been possible that on a given day the predictions state that more jobs were successful than the total number of tasks required of that type on that day.

7.2.3 Input Analysis

As mentioned in 7.2.1, historical plan states are likely to affect the service level outcomes for today due to the time factor in the service level agreement. As such, part of the problem definition process is deciding how many previous plan states, *T*, are required as additional inputs. To inform that decision a correlation analysis is performed between the outputs (Y_i and F_i) and the plan state for that day along with the plan state for each day going back a further 6 days across all areas in the data set (C_6, S_6, I_6, V_6). For the analysis the data is clustered into sets by the day of the week to also investigate if the current weekday has any effect on which past plan states are more correlated to the current service level outcomes.

Some examples of the results achieved are presented in tables 5 and 6. On each table the rows are the days of the week the cluster is for and the columns indicate *t*, the days prior to the current day the entry is for. For example the 1 column for the Mon row indicates the average correlation across all 56 areas between the output for that table on the Monday and the input for that table from one day before, in this case the Sunday.

The analysis covers all 4 outputs, success and failure for each of the two service levels, versus each of the 21 inputs. Analysis of all the results, focusing on which number of days prior to the current day produce the highest correlation, show that two main patterns emerged.

Table 5 shows an example of the first pattern. With the highest and second highest correlated values highlighted a distinct diagonal line can be seen across the days of the week, excluding the weekend. Most weekdays are correlated highest to the Monday or Tuesday value for that week in this pattern. This can mostly be seen in the capacity input, although appears in a few of the clears inputs also. This suggests that the amount of the backlog that has built up over the weekend which gets cleared early in the week has an effect on the resulting service levels for the entire working week.

Table 5 V_3 average correlation with Y_1

Weekday	Lag Days, *t*						
	0	1	2	3	4	5	6
Mon	0.64	0.43	0.46	0.42	0.36	0.37	0.45
Tue	0.51	0.43	0.34	0.36	0.36	0.33	0.35
Wed	0.39	0.41	0.46	0.34	0.34	0.36	0.33
Thu	0.32	0.35	0.37	0.33	0.31	0.32	0.33
Fri	0.32	0.35	0.37	0.37	0.34	0.32	0.33
Sat	0.38	0.31	0.32	0.33	0.33	0.31	0.32
Sun	0.70	0.50	0.43	0.41	0.36	0.39	0.37

An example of the second pattern predominant in the data is shown in Table 6. In this pattern we see that the highest correlated values tend to occur for a fixed *t* value. This pattern is seen most often in the workstack and intake inputs but also, as in the example shown, occurs in a few of the clears. This fixed *t* seen in this pattern is likely caused by the time allowance portion of the service agreements.

Table 6 C_2 average correlation with Y_2

Weekday	Lag Days, *t*						
	0	1	2	3	4	5	6
Mon	0.38	0.37	0.41	0.36	0.34	0.33	0.35
Tue	0.39	0.42	0.62	0.40	0.35	0.33	0.36
Wed	0.35	0.47	0.62	0.37	0.37	0.36	0.35
Thu	0.39	0.37	0.51	0.42	0.36	0.37	0.34
Fri	0.34	0.34	0.37	0.33	0.35	0.32	0.30
Sat	0.32	0.34	0.37	0.35	0.34	0.35	0.35
Sun	0.40	0.33	0.33	0.31	0.31	0.34	0.32

The main conclusion drawn from the correlation analysis, with regards to this problem formulation, is that allowing a value of *t* = 4 previous data points is required to capture the best correlated values for each input. This is enough to ensure that when looking at the output on a Friday then the Monday inputs are still being considered. Going further to a value of 5 or 6 for *t* is not required as the correlation analysis shows that they rarely contain any of the top correlated values.

The data analysed shows the average correlations across the 56 areas, with a lot of variation seen within each. Thus it will not give the best accuracy to simply use this average correlation analysis to pick the precise inputs for the final model. As such we keep all possible lag values, up to the selected cap of 4, for each input, reducing the number of data points to 199 but increasing the number of inputs to 105, and instead filter these dynamically during the model creation for each output in each area.

7.3 Neural Network Solution Method

This section describes the neural network model that we use to solve the service level prediction problem. We use some dynamic model construction to create a different model to solve equations (40) and (41) in each area. First we define the NN model used to solve the problem in 7.3.1. This we follow with a brief description of a filtering technique that is used to dynamically reduce the number of inputs used by each model in 7.3.2. Finally in 7.3.3 we cover the training techniques that are used to train the model.

7.3.1 Model Construction

A feedforward multi-layered perceptron (Gardner & Dorling, 1998) neural network is chosen as they are widely used for forecasting (Zhang, et al., 1998). A single hidden layer, containing 12 nodes, is used to predict each of the outputs, Y_i and F_i, individually. As such each has a single output node. 12 nodes was selected as an increase in the number of inputs was found to require additional nodes from the model created in chapter 6. Some initial tuning experiments found 12 nodes to provide a good accuracy in this problem. . The number of input nodes varies for each output as we dynamically select the inputs to use from the plan state sets of C_4, S_4, I_4 and V_4. Each layer uses the sigmoid activation function (Sibi, et al., 2013). The network is implemented using the Encog (Heaton, 2015) library in Java 1.7. As with previous chapters this was implement on a windows 8.1 Lenovo ThinkPad P50 laptop running an Intel core i7 6820HQ processor with 40GB of RAM

7.3.2 Input Filtering

With such a large number of inputs there is a risk of noise impacting the accuracy of the trained models. As such we employ the dynamic filtering technique developed in chapter 6 to use each input's non-linear correlation with the output that the current model is being built for to select which inputs to use. Some initial quick tests show that choosing a value of 0.5 proves to be a good cut-off point. Thus, during the construction of the model to predict each output in each area only inputs with a correlation value of 0.5 or greater are used.

7.3.3 Training Techniques

The models are trained using the resilient backpropagation algorithm (Kişi & Uncuoğlu, 2005) with 25 random restarts, chosen as it was shown to be effective in the previous chapter. The training data is split into two sets, the training set and the validation set. The models being trained using the training set and having their final accuracies evaluated on the validation set.

Two techniques are investigated to avoid overfitting, the first being the early stopping strategy (Prechelt, 1994) that proved effective in 6.6.2 and the second using dropout (Srivastava, et al., 2014) to investigate a potential alternative. In the early stopping strategy, after each training iteration, the current accuracy of the model is tested using the validation set. Once the accuracy on the validation set stops improving and starts to decrease then overfitting has started to occur on the training set so the training process is halted.

Dropout is another technique currently in use to avoid overfitting, in this case at each training generation a node has a defined probability of being excluded, or dropped out, at each training generation. The weights applied to each node are then multiplied by this probability when the final trained model is used. This has the result, in essence, of training multiple thinned networks that are used as an ensemble when performing predictions without the large computing performance required to train and use a large number of networks. Training in the dropout case is set to end when stagnation is detected, in this case when the trained accuracy has not improved by 0.2% over 50 iterations as this is a reasonable indicator of stagnation.

In both cases the training process is run 25 different times using different random seeds to set the initial weights with the models that produce the best accuracy on the validation set chosen for each output to give a reasonable range of results.

7.4 Results

In this section we first outline the method we use in these experiments in 7.4.1. Analysing these results we cover, first when looking at daily prediction values in 7.4.2 then by looking at the aggregated weekly results in 7.4.3.

7.4.1 Experimental Method

The dataset outlined in section 7.2 is used to test our model accuracies. For each area we split the data into 178 training points and 21 for testing. This simulates predicting for 3 weeks in a plan performed on the first date in the testing set. This data is further clustered by day of the week giving 25 to 26 training points and 3 testing points per cluster. For the training process the training portion is split further, removing 5 points for validation leaving 20-21 points for training.

We then use this data to create and train a model for each of the 4 outputs (Y_1, Y_2, F_1 and F_2) for each of the 56 areas for each day of the week. Each model is trained using the training data points and then tested on the test data. The accuracies of the predictions against the testing data are presented for each experiment.

The experiment is run for all areas using the early stopping strategy and dropout for comparison. Results displayed show the average accuracies across all 56 areas

7.4.2 Daily Prediction Results

First we analyse the results of the daily predictions, the results displayed are grouped by the day of the week. For example the Monday entry shows the average accuracies for predictions made using the Monday model created for each of the 56 areas. The accuracy is shown for the success prediction model, Y_i, and failure prediction model, F_i, for both of the service levels i, along with a column showing the resulting accuracy when the models are combined to give the service level prediction R_i.

Table 7 shows the accuracies achieved using the early stopping strategy. The accuracy calculation used is identical to that described in 6.4.1 for the experiments in chapter 6, being that of the inverse of the weighted MAPE. The accuracy for the failure model is fairly low, being about 52.4% on average for the two service levels. The number of failures per day tends to be fairly low however, thus the low volume drives this seemingly low accuracy. Being one away from the correct value when the total is five is a much higher percentage error than being one off when the total is fifty. This also explains the general lower accuracy seen for Sunday predictions as these tend to be days with low total number of jobs completed. Even taking this into account however the overall accuracy of the individual success and failure models is too low to be used in practice.

Table 7 Average accuracy of predictions using the early stopping strategy over all areas by day of the week

Day Of Week	Service Level 1			Service Level 2		
	Y_1	F_1	R_1	Y_2	F_2	R_2
Sat	80.3%	56.4%	89.7%	87.1%	59.4%	92.9%
Sun	61.0%	4.2%	86.1%	52.3%	-4.0%	63.8%
Mon	86.2%	55.0%	93.9%	87.8%	67.3%	93.8%
Tue	84.6%	58.7%	95.3%	74.6%	48.4%	83.2%
Wed	87.8%	62.8%	95.9%	86.7%	64.5%	93.2%
Thu	87.6%	61.8%	93.0%	85.9%	63.3%	93.3%
Fri	89.0%	66.1%	95.0%	88.8%	70.0%	93.0%
Average	**82.4%**	**52.2%**	**92.7%**	**80.5%**	**52.7%**	**87.6%**

This conclusion changes when we combine the success and failure model outputs to generate predictions of the resulting service level (percentage of successful tasks) instead. The service level column, R_i, shows the average accuracy across the areas, by day of the week, for this combined prediction. The average accuracy across all days and both service levels in this case is about 90.2% even when including the lower accuracy produced by the reduced volumes on a Sunday. Considering the goal of the problem is to predict the service level this is a much better evaluator and as such we can see this model is useable in practice.

The same experiment is run using dropout instead of the early stopping strategy, the results for this shown in table 8. We see the same general trends as using the early stopping strategy, lower accuracies for the low volume failure and Sunday models. Much improved accuracy when they are combined to produce the final service level predictions. Further to that, we can see that compared to using the early stopping strategy as an overfitting avoidance measure, using dropout produces some accuracy improvements across the board. Most relevant being the increase in the average accuracy of the service level prediction from 90.2% to 90.5%.

Table 8 Average accuracy of predictions using dropout over all areas by day of the week

Day Of Week	Service Level 1			Service Level 2		
	Y_1	F_1	R_1	Y_2	F_2	R_2
Sat	81.7%	60.7%	90.8%	87.5%	61.7%	93.1%
Sun	62.8%	12.0%	85.1%	54.1%	-3.3%	66.3%
Mon	86.9%	58.9%	94.4%	88.8%	68.2%	94.1%
Tue	83.9%	59.2%	95.4%	75.7%	47.4%	83.1%
Wed	87.3%	64.1%	96.1%	87.1%	65.6%	93.5%
Thu	87.3%	63.1%	93.1%	86.9%	64.0%	93.8%
Fri	88.3%	67.3%	95.2%	89.0%	70.5%	93.1%
Average	**82.6%**	**55.0%**	**92.9%**	**81.3%**	**53.4%**	**88.1%**

7.4.3 Weekly Prediction Results

In order to also evaluate the usefulness of this model to the current planning procedures, whereby senior planners take a weekly view of each areas workstacks to decide where might need additional resourcing, we also analyse the accuracy of the predictions when aggregated to the weekly level. We sum the success and failure models for each of the three weeks and apply equation (39) to calculate the resulting service level for that week. We count the number of times the error is less than a certain percentage in each area, to evaluate the confidence level of the predictions, and also calculate the average accuracy across all 56 areas for each week.

Table 9 shows the weekly results achieved by the early stopping strategy solution. The accuracy is recorded as well as the number of predictions that fell within 8, 5 and 2% of the actual value in each case. The significance of these bands is that in the real world scenario there is a 5% range between green, meaning the service looks safe, and red where immediate action is required to improve service levels. This 5% gap is given as amber and indicates uncertainty over whether service will be good or bad. Thus to be able to have some confidence in the prediction when presented within these bands the prediction would have to fall within 5% of the actual the majority of the time. The weekly level prediction shows an expected improvement over the daily accuracy, producing an average of 96.7% across all areas across all weeks. We can also see that around 78.6% of the time the error of the prediction in an individual area is less than 5% increasing to 92.9% when extending the range to 8%. This shows that decisions can be taken using this model with reasonable certainty, particularly as some of the higher error values are caused by areas with lower volumes. Something the senior planner would have knowledge of.

Table 9 Weekly accuracy across all areas using early stopping strategy

Week	Service Level 1				Service Level 2			
	< 8%	< 5%	< 2%	Avg Acc	< 8%	< 5%	< 2%	Avg Acc
1	56	49	26	97.4%	46	41	22	96.4%
2	54	49	23	97.2%	52	42	21	96.4%
3	52	44	24	96.8%	52	39	18	96.3%

The situation improves again when using the dropout method, as shown in table 10. The average accuracy now increases to 96.9% with 80.7% of the time predictions fall within 5% error. Both methods produce results that greatly improve upon the current planning methods and provide a strong tool to assist the planning process. We can also see from these results that dropout produces some slight improvements over using the early stopping strategy to avoid overfitting.

Table 10 Weekly accuracy across all areas using dropout

Week	Service Level 1				Service Level 2			
	< 8%	< 5%	< 2%	Avg Acc	< 8%	< 5%	< 2%	Avg Acc
1	55	49	28	97.6%	47	43	23	96.4%
2	54	49	24	97.3%	49	44	22	96.5%
3	54	46	23	96.9%	52	40	21	96.4%

7.5 Conclusion

In this chapter we identified the requirement for a service level predictor to assist the evaluation of plan solutions for the tactical plan when dealing with incomplete data. This can also be used by automated planning methods as well as provide the opportunity to improve the current planning process. We define this service level prediction problem, stating the general problem in equation (38) and introducing a specific real world example. Initial analysis of the data uncovered the requirement to include 4 days plan states as part of the inputs and we further split the problem into predicting the success (equation 40) and failure (equation 41) separately to allow their combination to produce a service level prediction at any level of aggregation.

We then described the neural network model used to solve these problems, including a method to dynamically filter the larger number of inputs created by the addition of the past plan states. The network setup was described as well as introducing two methods to avoid overfitting to the data set that we investigated in this chapter, the early stopping strategy and dropout.

The resulting models were then evaluated on the real world data, comparing the two solutions to avoiding overfitting whilst evaluating the model performance. Dropout was found to produce slightly better results than the early stopping strategy. Additionally we showed that we were producing good daily accuracy to give a reasonable evaluation of plan solutions as well as give current planners indication of days where problems are occurring. As well as that we showed that the accuracy at the weekly level, where it would be used within the current planning process, were very good providing a good tool to assist decisions relating to release of additional resources to specific areas.

This has successfully provided a means to solve the problem of properly evaluating plan solutions in the usual real world situation where there is not data available to the level of detail required to calculate these directly. This model can therefore be used as part of the solution evaluation for the overall tactical planning problem.

Chapter 8

A Surrogate Follower model for the Bi-level tactical planning problem

8.1 Introduction

In chapter 5 a bi-level framework was introduced to solve the capacity and demand decisions in the planning problem as separate leader and follower models. The models used, a GA leader and linear follower, were found to effectively solve the planning problem in a static planning situation where updates are required only a couple of times per day. However the time to reach a solution was found to still be a bit too long for use in a dynamic real world planning application where the model would need re-solved each time someone made a tweak to a plan or new information became available. Further to this, the model required assumptions on the level of detailed data available as it still involved distributing individual resources' time, albeit clustered into groups, to the range of skills they are capable of performing. It also did not contain a full evaluation of the plan quality, which requires a proper calculation of the number of jobs completed on time, as there was no knowledge of the precise due dates of tasks in the model, only a rough target for percentage of available jobs to complete each day.

In the subsequent two chapters these problems, often arising in real world situations, with making planning decisions with incomplete data were highlighted and explored. First a solution to the demand allocation problem was developed in chapter 6. This was followed by chapter 7 where a method to properly evaluate plans was developed. This was the creation of a model to predict the number of jobs likely to be completed on time, which is a requirement for calculating whether the service level targets are being met. These two models were developed using neural networks, meaning the bulk of the computational work can be done in advance during the training process, then the trained networks can be stored and fetched later for use when solving a specific planning problem.

These two features, handling incomplete data and less computation required during run time, have the potential to be combined to perform as a surrogate for the follower model within the bi-level framework. We hypothesise that this will both allow the bi-level model to now perform on more real world data sets where a lot of future information is unknown and also may bring the time to reach a good solution down to levels that would make it useable within a real world interactive planning application, where users (or data updates) are expected to be making frequent changes that requires re-computation of the plan.

This chapter explores this hypothesis, first the planning problem from chapter 5 is re-defined to deal with more realistic data availability as per a real world scenario in section 8.2. In section 8.3 the model that will be used to solve this problem is defined, introducing a combination of the demand allocation and solution evaluation models from chapters 6 and 7 into a surrogate follower model for use in the bi-level tactical planning model. Some experiments are performed on some real world data in section 8.4 to evaluate the model performance before concluding in section 8.5.

8.2 Real world planning problem with incomplete data

In chapter 5 a representative sub-problem of the overall tactical planning problem was defined. A selection of capacity levers to allow modelling of different decision types were chosen. Overtime adds capacity to a resource, contractors bring additional capacity directly applied to a skill and reduction moves resource capacity, e.g. for an area move or to be held in contingency to respond to emergency situations. Along with those the installation selling lever and the capacity-demand matching required to allow proper evaluation of these decisions. The evaluation in this case was done using a target for the percentage of fault jobs to be completed each day along with target installation selling levels.

Some assumptions inherent in this model are not true when working in a large number of real world scenarios however. For example, the problem involved the allocation of resources to skills based on the known skillsets of each resource in the plan for each day. In a real world scenario there has to be some allowance for resource absences, some of which are already known. However for events, such as meetings or training, not yet booked in the calendar, or something uncontrollable like illness, it is impossible to know which individual resources will be unavailable. This necessitates the modelling of resource at an aggregated level where the total volume of resource time rather than the individual resource time available is considered. This loses the ability to match individual resources to their available skills. There is a similar level of uncertainty within the demand, in that a large portion is forecast in the tactical planning phase and thus it is not possible to know precisely where or when a fault will occur or even what precise task type it will be. To ensure some accuracy the tasks tend to be aggregated into workstacks by the skill required to complete them within more general geographic areas. This makes it not possible to directly calculate service levels as it can't be known what the target completion date was for the tasks that don't yet exist to calculate if they will be completed on time within the current plan.

To account for this the tactical planning sub-problem from chapter 5 is redefined to account for the real world incomplete data issue.

8.2.1 Tactical planning sub-problem with incomplete data

In the tactical planning sub-problem with incomplete data, the resources (or resource groups once they have been grouped by the skill set they can perform) are further aggregated into resource types. These types can be decided based on various features, e.g. they may be high, medium and low skilled or split by mobile (more likely to work in more than one area) and fixed (usually works in their home area) or any number of other groupings that makes sense for that data-set. In the case study data-set there are 3 types, with the groupings decided based on mobile, fixed and new hires/apprentices. Aggregating to this level loses the knowledge of exact skill makeup but is required when working with real world data where it is not possible to know which exact resources may be absent on any future periods of the plan, among other factors.

For the sub-problem explored in this chapter we again select the three representative capacity levers, overtime, contractors and reductions. These cover standard capacity modification moves of adding capacity to a resource, applying external capacity and moving resource capacity. These levers modify the capacity available for the capacity-demand matching process and have usage costs and budgets (max constraints) associated with them. Rather than apply these changes directly to resources however they instead are altering the aggregated buckets of capacity available to each resource type.

Similarly to resources being aggregated by type, tasks are aggregated by the skill required to complete them and the general area they fall within into workstacks. This is due to the increasing inaccuracy the more fine grained the attempted fault prediction is. This aggregation occurs for both fault and installation skills, since in installation although the demand is not forecast there is no way of knowing precisely where the new appointments opened up for booking will be picked up. This also involves factors such as time to complete the task being averaged.

The capacity available for each of these resource types is then allocated to tasks to allow evaluation of the capacity decisions. This allocation phase also includes an additional lever which alters how many installation slots to open up for selling. Available capacity is allocated to cover appointments first, those being installation slots that have been sold already or fault tasks that have been appointed. This is followed by allocating capacity to cover expected (forecast) extra faults. Any remaining capacity is then used to open those additional installation slots.

Given the forecast nature of a majority of the plan the tasks that are having capacity allocated are not being actually scheduled. That occurs in the operational planning stage (see fig 1.). Instead the planning here is an attempt to estimate what will be completed each day in order to effectively evaluate the outcome given the current capacity and installation selling lever decisions.

The final part of the problem is how a solution is evaluated. A good planning solution would be one that maximises service levels and installation selling while minimising capacity use. Evaluating the portion applied to lever use, capacity decisions and installation selling levels, just involves a direct costing of those values. Judging expected service levels is more complicated however as they cannot be directly calculated on the aggregated data. This is because the due date for each individual aggregated task cannot be known so it is not possible to know which are being done on time. In practice to evaluate their plans a planner will keep track of the daily workstack level trying to keep it at what they judge to be a healthy level. Too low and there may not be enough tasks available to keep resource utilisation high, too low and jobs will be failed. This is not an entirely accurate method of plan evaluation however as the aggregated nature of the workstacks conceals some information. A certain workstack level on one day might result in no failed tasks however on another day there may be more urgent tasks sitting concealed in the workstack requiring completion. These extra urgent tasks will result in some failures if the workstack is maintained at the same level as previously. For the purposes of this problem we will instead be predicting the expected success rates based on the current plan data to provide a more accurate evaluator of solution evaluation.

8.3 Bi-Level Model with NN based Follower Surrogate

In this section the work to develop the bi-level model in chapter 5 is combined with the neural network models developed in chapters 6 and 7 to predict the task completions and service level outcomes respectively. These can replace the previous linear capacity-demand allocation follower model to allocate the capacity to tasks and evaluate the resulting solution. The models utilising the parameters chosen for them during those chapters. This follower can both still produce solutions on the incomplete data and also, we hypothesise, require less computation to do so. The GA leader model from 5.2.2 is re-used with the linear capacity-demand matching follower model replaced with the surrogate defined in the rest of this section. In 8.3.1 we define the inputs for the surrogate model. The demand allocation part of the surrogate is defined in 8.3.2 making use of the task completion NN model. Finally the method to evaluation the surrogate solutions is defined in 8.3.3 utilising the service level prediction NN model.

8.3.1 Surrogate Follower Model Inputs

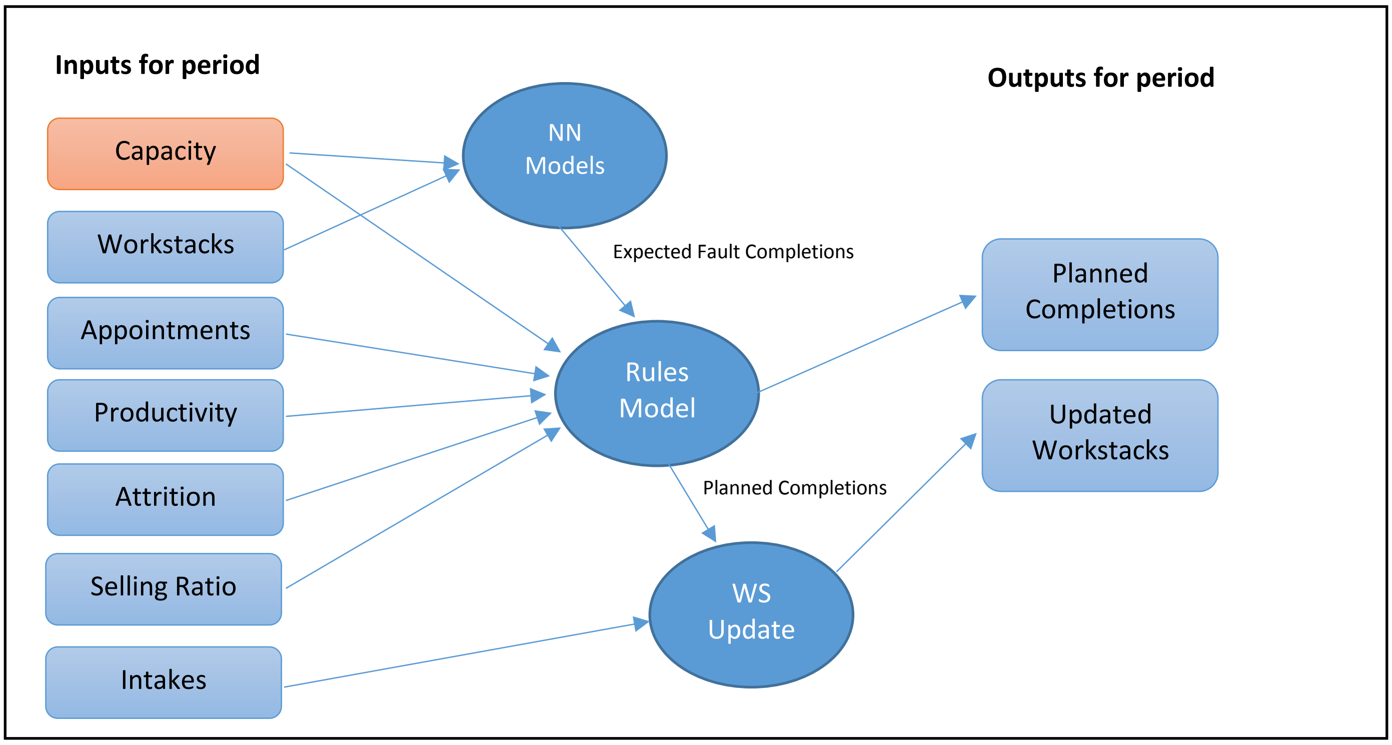

Fig. 25 Single period of Follower Model

The left hand side of fig. 25 shows the inputs used by the surrogate model. Some of these have been seen in previous chapters, however some are new and a consequence of working with real world data. These are:

- Capacity - The current capacity in the plan, modified by the capacity lever decisions taken by the leader
- Workstacks - The tasks making up the demand are aggregated together by the skill required to complete them into workstacks. Each workstack has an initial value, which is the current backlog of tasks for that skill at the start of the plan.
- Appointments – This is the minimum tasks that have to be completed for each day. The value varies by day and skill depending on the number of installation appointments booked or fault tasks that require site visits at a specific time/day.

- Productivity – Each resource type has a productivity for each skill which indicates how much of their available time it takes to complete one task requiring that skill. This links the capacity, which is in available time, to demand which is in task completions.
- Attrition – This comes in two types, “on the day” and “before the day” and is estimated for each day of the plan. The values indicate a proportion of tasks that are removed from the plan on each day as they are not expected to be completed.
    - “Before the day” - jobs expecting to disappear from the plan before that future date becomes today, e.g. through customer cancellations.
    - “On the day” – models tasks where it is discovered on the actual day of implementation that they cannot be completed for various reasons, such as no one is there to provide access or the resource discovers they cannot complete the task.
- Selling Ratio – The ratio at which additional capacity should be distributed to installation skills. This is used to ensure realistic spread of appointment slots are made available for each installation skill.
- Intakes - For the fault skills they have a forecast intake indicating new tasks added to their workstacks in each period of the plan. The installation skills have no new intake, instead new tasks are added through the installation selling lever within the model

8.3.2 Surrogate Model Demand Allocation

Capacity-demand allocation in this model is performed using a combination of the completions prediction NN model from chapter 6 and a number of capacity allocation rules. The rules are required for two purposes, to fill in the skills not covered by the model (as it only predicts the fault jobs, does not set the installation selling levels) and to ensure the output is a valid plan that meets all constraints. Fig 25. Shows how the NN and the rules combine to create the single period model. The capacity, modified by the leader model, and current workstack value are used as input to the NNs which produce the expected fault job completions used, along with the capacity, appointments, productivities, attrition, and selling ratio, as inputs for the rules. These then allocate the completions to all the skills for that period. The final step is to update the workstacks for all skills by adding on the intake for that period and subtracting the planned completions.

The input for this surrogate model has values for all inputs for all periods, except for the required workstack value as it cannot be known what the workstack will be on a given period unless the completions running up to that period have already been set. For this reason the model is repeated for each period, by starting with the initial workstack levels for the first period then for

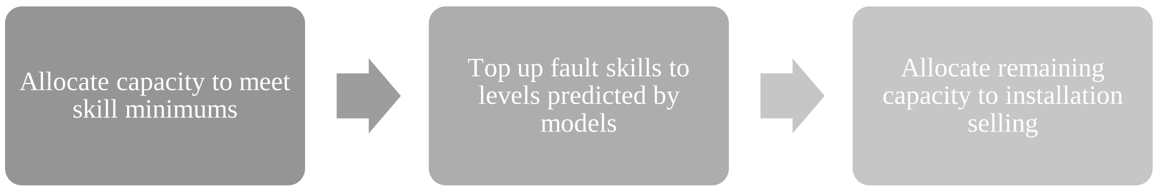
Allocate capacity to meet skill minimums
Top up fault skills to levels predicted by models
Allocate remaining capacity to installation selling

Fig. 26 Follower Model Rules Flow

every subsequent period the updated workstacks from the previous period are used. This process continues, running for one period at a time, till the end of the plan is reached and completions and workstack levels for all periods have been generated.

The simple demand allocation rules model is seen in fig 26. These were generated based on the real world planning process. First capacity is allocated to skills to ensure the minimum for each skill is met, e.g. that any appointed tasks are covered. Then, if any capacity is remaining, the rules will allocate capacity to top up the fault skill completions to the levels predicted by the NN completions model. This is done to ensure enough capacity is ring-fenced to cover the expected fault completions requirements. Finally, any remaining capacity is used for additional installation job selling. The remaining capacity is allocated to installation jobs using the selling ratio. This is often derived from the past average selling levels for each skill. This means the lever deciding the installation selling levels, seen in the linear follower model, is no longer modelled as a lever, instead is just calculated based on the excess capacity after appointments and faults are covered. However, it could be argued this means it is indirectly set by the leader decisions as any excess capacity that it creates will be applied to installation selling. If at any step of the chain there is not enough capacity to cover what is required then the completions required at that stage are reduced proportionally to their volume till the requirements fit within the capacity and then the process exits.

8.3.3 Surrogate Model Solution Evaluation

The solution reached by the follower model is evaluated in two parts. The benefits from installation selling can be directly calculated by giving a weight to the selling for each skill. The penalty associated with insufficient fault skill completions cannot be directly calculated as discussed. For this we will instead use the service level prediction NN to predict the expected fault task success rate based on the current state of the plan variables.

The demand allocation process generates the required completions and workstack inputs for the service level prediction model. The required intakes and capacities that make up the final two inputs are simply those that are part of the overall problem inputs. This produces the expected success and failure rates by service level for each period of the plan. These are used to calculate the percentage of required tasks successfully completed each day which then, along with the configured percentage targets, give a penalty cost to fault tasks not meeting the requirements. This is added to the benefit associated with the installation selling to give the overall fitness for the follower models current solution.

8.4 Results

In this section of the chapter the bi-level model with the new surrogate follower defined in 8.3 is tested on the real world case study data used in chapter 5 along with the historical datasets from chapters 6 and 7 to be used to train the neural networks. First the data used is outlined in 8.4.1 along with highlighting some restrictions imposed to properly simulate a real world planning activity. This is followed by the experimental technique used in 8.4.2 and then the results and their analysis in 8.4.3.

8.4.1 Experiment Data

The same real world planning dataset for a week in October 2017 from chapter 5 was re-used for these experiments but with the added restrictions on knowledge of available capacity that would have been available when planning for that week at the time. Namely that when the plan was created, on the first day of the week the data is from, it could not be known for any future days what precise resources may have been unavailable. Thus an accurate skill breakdown, as discussed in previous chapters, is not available for the resources. Instead the data used here is the resources aggregated into three resource types with no skill sets allocated as no direct allocation is possible.

Additionally, 6 months of historical data, for the previous 6 months in 2017, was used to train the demand allocation neural networks and the service level prediction networks. This included workstack, intake, and completions data for the 13 skills, resource capacity for the three resource types and the success and failure volumes for tasks on the two most common service levels. This data had similar ranges as the data sets used in the previous chapters described in 6.3 and 7.2.2.

Ten anonymised areas were again selected from the dataset to give the 10 different problem instances. They were selected to provide a mix of predicted service level outlooks to test how the model behaves in different scenarios. For each area the decisions taken by the planners were also available to provide a comparator. It should be noted that this data cannot be used for accuracy comparisons, since the different decisions taken for capacity levels would naturally have led to different actual outcomes on those days. This is the reason that the results will be compared with the planners' decisions rather than with the actual results. This allows evaluation of whether the model is taking good decisions, e.g. if in the planned data we can see that the service level predictions were too low then we'd expect the model to be providing more capacity than was there in the actual plans for those days.

8.4.2 Experimental Method

The model was run on each of the 10 datasets with the evolution set to stop after 10 generations of stagnation, the same setting used in chapter 5 as this allows for stopping early when convergence is quick or for running longer if a lot of exploration is required. The mutation probability for all 3 sections of the chromosome were set to 0.05 with the crossover probability of 0.85, again the same as those used for the GA leader in chapter 5. The overtime budget was set to 500 and the contractors to 1800 which were both slightly higher than the highest amount used by any of the planners in the problem instances. This allows the algorithm the leeway to decide to use a bit more than was actually used whilst still keeping the values realistic. The maximum amount of reductions allowed per day was set to 20, the same value from the previous experiments. After the models had reached their stopping point the resulting completions and capacity lever values were recorded. These were then input into the service level prediction model to be compared with the expected service levels from the planners' decisions in those instances.

As with the previous chapters the model was implemented using java 1.7 with the watchmaker framework 0.71 used to implement the leader GA and the NN based follower implemented within the fitness function using Encog with the models tuned to the same parameters as those used in chapters 6 and 7. This was then all run on a windows 8.1 Lenovo ThinkPad P50 laptop running an Intel core i7 6820HQ processor with 40GB of RAM.

8.4.3 Results Analysis

Given the sensitivity of using real world data we cannot directly report the service level percentages achieved here. For the purposes of evaluating this model we will analyse the service levels using a RAG (red, amber, green) system. Red will indicate a service level result that is below target, amber a result that is between target and 5% above target and thus is in danger of failing. Finally green indicates a safe service level that is more than 5% above target. In this case the target is the percentage of jobs successfully completed on time. The target percentage being the levels at which additional penalty payments will start to be charged. In order to reduce the risk of dipping below this level then the above amber buffer is used to indicate to planners and management which areas may be at risk of dipping below that level.

In Table 11 we can see the decisions and results from the algorithm compared with those actually planned. We can see the algorithm has improved the expected service level outcomes across the board, taking the initial estimate of 2 red, 5 amber and 3 green areas and improving it to 8 green and 2 amber. Of the amber areas the historical data showed that amber was the best achieved within the historic data thus it wasn't possible for the algorithm to get a green prediction. This may highlight one potential problem with the approach as the algorithm has applied a lot of overtime and contractors to both these areas in an attempt to get them to reach green. In reality the amber may have been accepted in this case without so much extra resource applied in a failed attempt to improve the situation to green since amber is only danger of failing not an indication of actual failure. This could be mitigated however by more careful configuration of the overtime and contractor constraints to prevent too much being applied. Across the 10 areas we can see the algorithm applying an average of 13.2 additional overtime to achieve this service level uplift, however at the same time it reduces contractor use by 2.48. This is likely due to service levels only being attached to fault skills whereas within this dataset contractors could only be applied to installation skills.

Table 11 Planners vs. Algorithm Comparison

Area	**Service Level RAG**		**Average Per Day**		
	Planners	***Algorithm***	***Overtime***	***Contractors***	***Resources***
Area 1	Amber	Green	3.4	36.4	-4.1
Area 2	Red	Amber	24.1	50.2	-0.5

Area	Service Level RAG		Average Per Day		
	Planners	***Algorithm***	***Overtime***	***Contractors***	***Resources***
Area 3	Green	Green	6.1	-41.1	-2.6
Area 4	Green	Green	11.2	-31.4	-2.9
Area 5	Green	Green	-6.1	11.6	-1.1
Area 6	Red	Green	34.1	-59.1	-2.4
Area 7	Amber	Green	17.9	-31.2	-2.6
Area 8	Amber	Amber	15.1	73.1	-0.9
Area 9	Amber	Green	6.1	-2.1	-1.2
Area 10	Amber	Green	20.1	-31.2	-2.6
Overall	2R 5A 3G	2A 8G	13.2	-2.48	-2.09

8.4.4 Solution run time

The other element we record in these experiments is the solution run time. This varied across the areas with the average ranging from 31 minutes up to 2 hours 40 minutes. The average across all instances being 1 hour 13 minutes. This is an improvement over the linear model however still not quite into the realms of the performance required for a dynamic tactical planning application. This was run on a windows 8.1 laptop however. With the potential to parallelise the solution evaluations plus the application of more powerful hardware it should be possible to improve the solution times further.

8.5 Conclusions

From the experiment utilising real world data performed in this chapter we can see that we can effectively utilise a surrogate consisting of the completions and service level prediction neural network models within the bi-level model to solve the tactical planning problem in a real world scenario. The use of the surrogates effectively allows for an automated solution to be found in the scenario where some details in the data required by the optimisation models are not available. We can see it improves the predicted service level outlook when compared against the planner in almost all cases. However a limitation is uncovered in that due to the nature of the neural networks that are used it cannot reach a situation that does not exist within the historical data.

This is observed in two of the cases where it capped out at an amber service level prediction whilst applying a lot of additional capacity. This would require some manual intervention when used as an automated system to ensure these capacity levers were capped a bit lower in these cases.

The run time was observed to be an improvement with the addition of the surrogates, though not quite at the level required of a dynamic planning model as yet. However this may be improved if run on more powerful hardware. This work does show the concept works and that surrogates can be used to handle the real world planning problem to produce effective automated plans.

Further work from this could be undertaken to further improve the model performance. Some potential areas to explore could be parallel processing, an improved leader model, or further refinements to the follower surrogate.

Chapter 9

Conclusions and Further Work

In this thesis we investigated an under-represented complex real world planning problem that has great impact. We updated the definition of the tactical planning problem to more closely match the real world application within large service industries. This planning layer is often overlooked however in this thesis we aimed to show that effectively solving this problem could have real world benefits for a member of the service industry. During the thesis we developed methods to effectively solve this problem even under real world restrictions to data availability. This has resulted in some practical benefit as algorithms developed in these chapters have been applied within real world applications already. These have had tangible benefits where deployed and this is explored in the impact section (9.2) of this chapter. The models developed are not without the limitations however and these will also be explored in the further work in 9.3. First we will go back to the research objectives defined in 1.4.2 to evaluate how these have been covered in this thesis.

9.1. Research Objectives Revisited

9.1.1 Develop an initial formulation for the tactical planning problem to capture the key features (O1)

This objective was covered by chapter 3 where a linear model formulation of the tactical planning problem was created. The model created effectively captured all the key features of the tactical planning problem and served as an initial point for the remainder of the modelling work within the thesis. It was also shown that this full model formulation was too complex to solve within any reasonable time frame as the number of decision variables increased rapidly as initial areas, skills, resources, etc. were added. This helped to further motivate the work undertaken in the rest of the thesis to investigate breaking down the problem into linked sub-problems.

9.1.2 Determine if this could be modelled as a combination of linked sub-problems to improve solution time (O2)

The literature review undertaken in chapter 2 completed this objective effectively. It was identified that a portion of the tactical planning problem defined in chapter 1 and then formulated in chapter 3 had the elements of a common problem in the literature, that of the generalised assignment problem. When this problem was removed it left the capacity levers from the tactical plan as the other sub-problem. The interactions between these problems were seen to match that of a bi-level model described in the literature. The capacity lever leader settings the constraints on the allocation problem follower fit well with the nested bi-level solution approach. Although it is described as potentially being computationally expensive the utilisation of the leader-follower framework to allow separate modelling of each portion did still suggest a potential to find a way to improve the solution time over the monolithic problem model approach. Chapter 5 eventually showed this to be true when the GA leader and linear follower model solved all the problems in under 8 hours where the CPLEX solver had not been able to solve the full problem within the 8 hours it was run for in an earlier test on the full model. Finally chapter 8 further improved on the solution time after introducing a surrogate which further showed the potential of the bi-level framework approach for reducing the time take to solve the tactical planning problem.

9.1.3 Investigate suitable algorithms to solve these sub-problems (O3)

The literature review in 2.1 and 2.2 picked out solution methods previously used for the allocation sub-problem and the overall bi-level framework respectively. In chapter 4 we tested two methods found in both reviews, the linear solver and genetic algorithm. We tested this against a greedy hill climber from a real world planning application. The investigation showed that for the allocation sub-problem the linear solver was the most suitable as it was faster and also reached optimal in all cases. However the GA performed very well also and identified itself as a candidate to apply to the leader model in the overall bi-level framework.

9.1.4 Develop a suitable framework to optimise the linked sub-problems (O4)

Following on from the bi-level suitability identified in the literature review it was chosen as the framework to develop. In chapter 4 we had identified the linear solver as best to use for the capacity-demand allocation follower. Out of the same chapter the performance of the GA had motivated the selection of it to use as the leader for the initial testing of the bi-level framework in chapter 5. In this chapter we were able to show that the bi-level framework could effectively solve the linked sub-problems to provide a good solution for the tactical planning problem that improved upon that of the real world planners. The solution time was seen to be too long to use in a dynamic planning scenario where the model would be run multiple times after data changes. It was however capable of solving it within a time frame of 1 to 7 hours meaning it could be used in a static planning scenario where the model needed re-solved less frequently.

9.1.5 Investigate potential solutions to deal with the real world situations where not all data-points are available (O5)

This objective was covered in both chapters 6 and 7, where neural network models were developed to predict completions and service levels respectively. These models are capable of providing solutions in real world scenarios where aggregated rather than detailed data is used. The completions model can provide the capacity allocation for the capacity-demand allocation model. The service level prediction model was shown to be an effective way to evaluate the likely plan outcome when success rate for tasks cannot be directly calculated. Both models were found to have good accuracy with the completions model providing better results than those given by a manual planner.

9.1.6 Determine if these can be used within the linked optimisation framework to produce feasible solutions to the planning problem in real world scenarios (O6)

This final objective was met in chapter 8 where a surrogate was developed from a combination of the neural network completions and service level prediction models. This surrogate, when slotted into the follower slot in the bi-level framework, was able to provide a tactical planning solution in real world scenarios. The surrogate was able to provide plan outputs in situations where only aggregated data was available. The resulting plans were shown to improve on the expected service level outcomes achieved by a manual planner on the same data. The solution time for this model was also shown to be improved over using the linear solved follower, however it still wasn't quite into the solution times required to be useable in a dynamic planning scenario. The tests were run on a low powered laptop however so there is still scope to bring the solution times down further through parallelisation and deployment on a more powerful server.

9.2 Research Impact

The work in this thesis has provided great impact already. Four publications have been generated from this work, from the contents of each of the chapters from 4 to 7. There is also an intention to write up the overall work and submit it as a journal paper once the thesis is completed.

There has also been a real world impact from this research as the models and algorithms developed during this thesis have been developed into practical applications that are already deployed in a large telecoms company. The most impactful contribution has been from the surrogate model that was produced for chapter 8. This has been deployed into a planning application that is being used across the whole of the UK. Here it automates the capacity-demand matching portion of the planning process while the planner takes on the leader model role and focuses on the capacity decisions. Analysis performed by the end user to investigate the impact of this shift showed significant savings were generated due to the increased accuracy of the plan as well as the time saved during the planning process. Fig. 27, 28 and 29 show the anonymised results of this analysis. The values have been normalised to be a proportion of the initial April payment amount but to give an idea of scale the initial April payments were all in the hundreds of thousands. Fig 27. Shows appointment availability payments is a penalty not mentioned thus far in this thesis. It is applied if there is not an available installation appointment available within a set amount of days. If say the target is 5 working days then the planners must ensure to have new appointments available for booking on each of the installation skills from the 6th working day onwards.

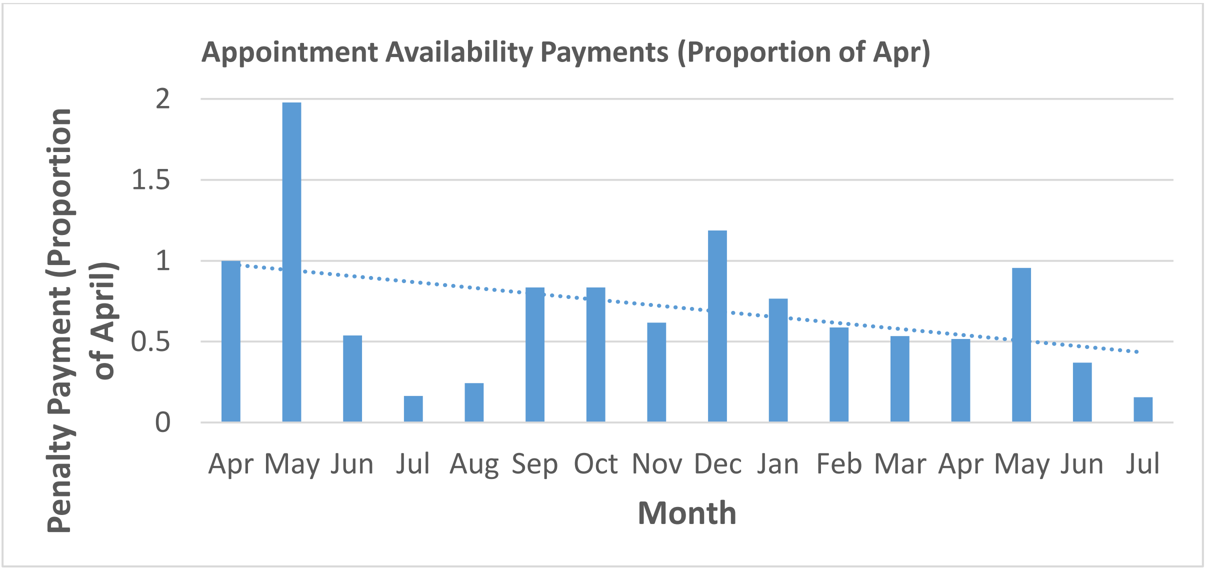

Fig. 27 Appointment availability penalty payments trend

Missed appointment penalties shown in fig 28 are more self-explanatory as these are penalty payments applied if an engineer does not complete an appointed job on the appointed day. The final one, faults missed commitments in fig 29, is the penalty related to failing to meet target service levels.

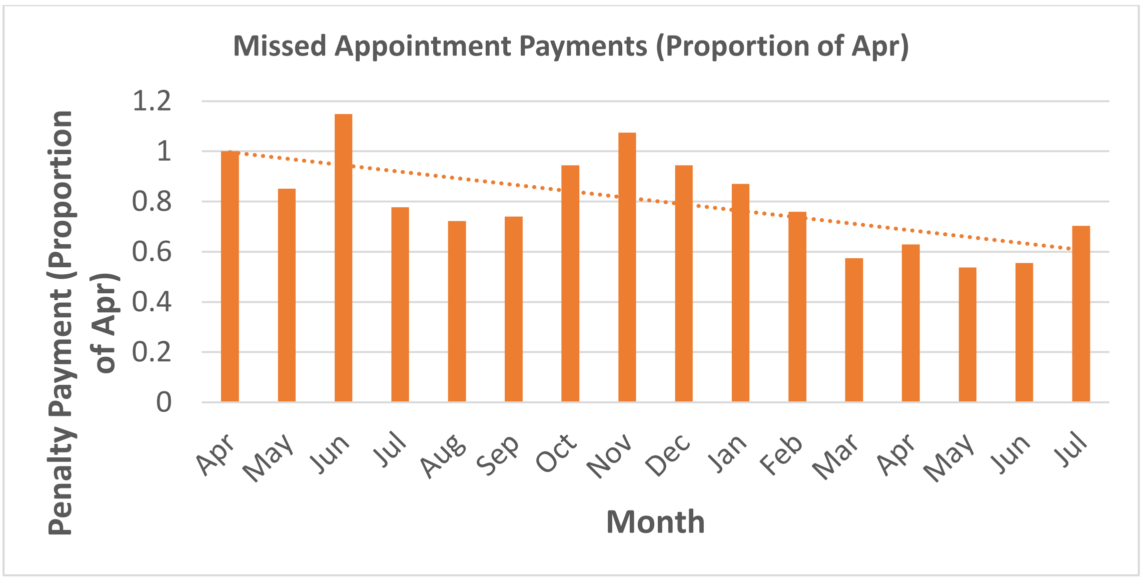

Fig. 28 Missed appointments penalty payments trend

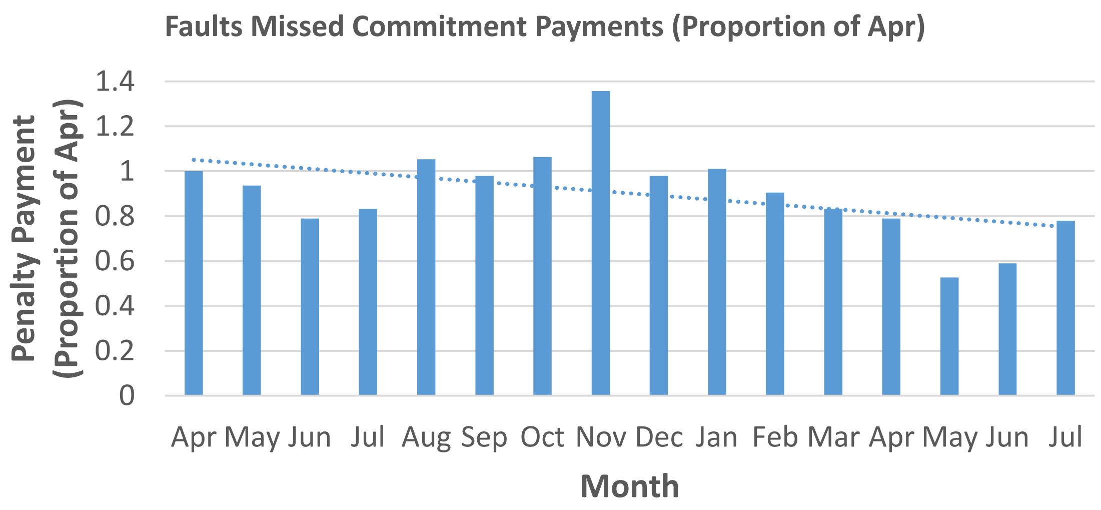

Fig. 29 Faults missed commitments penalty payments trend

The analysis showed there was a reduction of about 50% in monthly appointment availability payments, about 35% in monthly missed appointment payments and 27% in faults missed commitments payments. Given that in this large telecoms company each of these payments were in six figures monthly that is a very large saving showing a significant impact from this research. The reason for the reductions brought out in the analysis performed was that for the faults missed commitments and the missed appointments the reduction was due to the improved plan accuracy. This ensured that enough capacity was there to cover faults and not too many installation jobs were being sold. The appointments availability payments improvement however was due to the removal from human bias from the process. The algorithm was accurately showing when there wasn't enough capacity to produce appointments within the correct time scale which allowed correct identification of which areas required additional resourcing. When a human planner was performing the process however they were found to often "tweak" their plans to make them look better by adding installation selling within the targets even when it wasn't available. This was obscuring the areas that were needing assistance.

9.3 Further Work

This final section will outline areas of further improvement that could be undertaken on this work, as well as some future avenues for research.

Although this research met all the research objectives it still fell slightly short on reducing the solution times to levels that would make the whole model usable in a dynamic planning application. As such only the surrogate has been deployed in practice thus far. This means there are still avenues for research around reducing the computational complexity of this problem. This could be through development of alternate leader or follower models to fit into the bi-level framework, or through the introduction of parallelism into the computation process. A more customised heuristic to the capacity lever setting problem may perform better than the more general GA metaheuristic approach for example.

Further areas of research around the tactical planning problem definition could also focus on developing methods to deal with the uncertain outcomes from decisions. Some levers, such as the overtime or installation selling levers, have an uncertain outcome on the plan. The decision may be taken to use 10 hours of overtime tomorrow or open 20 installation appointments next week, however there is no guarantee that these will actually happen. The overtime is predicated on enough members of the workforce agreeing to pick up some overtime, similarly the installation appointments being opened up requires a customer to actually decide to book them. This was not covered in this thesis but may be an interesting avenue for future research, possibly involving some sort of learning process by the algorithm to learn what values of specific levers are likely to actually happen if set on different days of the plan.

A further future avenue could focus on a different kind of uncertainty, that on the forecast values in the plan. Currently the forecast for new intake jobs is a single value per day as this is what a human planner can reason with. However the reality is that the forecasting models produce more of a probability distribution of likely outcomes with the mean just being entered in for use in the plan. With the move to the use of automation in the planning process, algorithms could be produced that reason with this distribution instead of the point value. Thus it could create a plan that is more robust and can cope with the potential fluctuations in the actual fault levels that occur.

Bibliography

Ahuja, R. K. et al., 2004. A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science,* 50(6), pp. 749-760.

Anandalingam, G. & White, D., 1990. A solution method for the linear static Stackelberg problem using penalty functions. *IEEE Transactions on Automatic Control,* 35(10), pp. 1170-1173.

Anderson, J., 1995. *An Introduction to Neural Networks.* Cambridge: MIT Press.

Angelo, J. S. & Barbosa, H. J., 2015. A study on the use of heuristics to solve a bilevel programming problem. *International Transactions in Operational Research,* 22(5), pp. 861-882.

Angelo, J. S., Krempser, E. & Barbosa, H. J., 2013. Differential evolution for bilevel programming. In: *2013 IEEE Congress on Evolutionary Computation.* Cancun: IEEE, pp. 470-477.

Angelo, J. S., Krempser, E. & Barbosa, H. J., 2014. Differential evolution assisted by a surrogate model for bilevel programming problems. In: *2014 IEEE Congress on Evolutionary Computation.* Beijing: IEEE, pp. 1784-1791.

Barbas, J. & Marin, A., 2004. Maximal covering code multiplexing access telecommunication networks. *European Journal of Operational Research,* 159(1), pp. 219-238.

Barcia, P. & Jörnsten, K., 1990. Improved Lagrangean decomposition: An application to the generalized assignment problem. *European Journal of Operational Research,* 46(1), pp. 84-92.

Bard, J. F., 2013. *Practical bilevel optimization: algorithms and applications.* Berlin: Springer Science & Business Media.

Benders, J. & Van Nunen, J., 1983. A property of assignment type mixed integer linear programming problems. *Operations Research Letters,* 2(2), pp. 47-52.

Benjaafar, S., ElHafsi, M. & Vericourt, F., 2004. Demand allocation in multiple-product, multiple-facility, make-to-stock systems. *Management Science,* 50(10), pp. 1431-1448.

Bennett, K. P., Kunapuli, G., Hu, J. & Pang, J.-S., 2008. Bilevel optimization and machine learning. In: *IEEE world congress on computational intelligence.* Hong Kong: Springer, pp. 25-47.

Bhosekar, A. & Ierapetritou, M., 2018. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering,* Volume 108, pp. 250-267.

Bliek, C., Bonami, P. & Lodi, A., 2014. Solving mixed-integer quadratic programming problems with IBM-CPLEX: a progress report. In: *Proceedings of the twenty-sixth RAMP symposium.* Tokyo: RAMP, pp. 16-17.

Bostian, M. et al., 1-12. Valuing water quality tradeoffs at different spatial scales: An integrated approach using bilevel optimization. *Water Resources and Economics,* Volume 11, p. 2015.

Bouajaja, S. & Dridi, N., 2017. A survey on human resource allocation problem and its applications. *Operational Research International Journal,* Volume 17, pp. 339-369.

Bressoud, T. C., Rastogi, R. & Smith, M. A., 2003. Optimal configuration for BGP route selection. In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No. 03CH37428).* San Francisco: IEEE, pp. 916-926.

Brown, G. et al., 2005. A two-sided optimization for theater ballistic missile defense. *Operations research,* 53(5), pp. 745-763.

Calvete, H. I., Gale, C. & Iranzo, J. A., 2013. An efficient evolutionary algorithm for the ring star problem. *European Journal of Operational Research,* 231(1), pp. 22-33.

Camacho-Vallejo, J. F., Muñoz-Sánchez, R. & González-Velarde, J. L., 2015. A heuristic algorithm for a supply chain′ s production-distribution planning. *Computers & Operations Research,* Volume 61, pp. 110-121.

Camacho-Vallejo, J.-F., Mar-Ortiz, J., Lopez-Ramos, F. & Rodriguez, R. P., 2015. A genetic algorithm for the bi-level topological design of local area networks. *PloS one,* 10(6), p. e0128067.

Campbell, G. M., 1999. Cross-utilization of workers whose capabilities differ. *Management Science,* 45(5), pp. 722-732.

Campbell, G. M., 2011. A two-stage stochastic program for scheduling and allocating cross-trained workers. *Journal of the Operational Research Society,* 62(6), pp. 1038-1047.

Campbell, G. M. & Diaby, M., 2002. Development and evaluation of an assignment heuristic for allocating cross-trained workers. *European Journal of Operational Research,* 138(1), pp. 9-20.

Caramia, M. & Mari, R., 2016. A decomposition approach to solve a bilevel capacitated facility location problem with equity constraints. *Optimization Letters,* 10(5), pp. 997-1019.

Caron, G., Hansen, P. & Jaumard, B., 1999. The assignment problem with seniority and job priority constraints. *Operations Research,* 47(1), pp. 449-453.

Cattrysse, D. G. & Van Wassenhove, L. N., 1992. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research,* 60(3), pp. 260-272.

Cattrysse, D., Salomon, M. & Van Wassenhove, L. N., 1994. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research,* 72(1), pp. 167-174.

Cecchini, M., Ecker, J., Kupferschmid, M. & Leitch, R., 2013. Solving nonlinear principal-agent problems using bilevel programming. *European Journal of Operational Research,* 230(2), pp. 364-373.

Ceylan, H. & Bell, M. G., 2004. Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing. *Transportation Research Part B: Methodological,* 38(4), pp. 329-342.

Chang, G. J. & Ho, P.-H., 1998. The β-assignment problems. *European Journal of Operational Research,* 104(3), pp. 593-600.

Chauvin, Y. & Rumelhart, D. E., 1995. *Backpropagation: theory, architectures, and applications.* Hillsdale: Psychology Press.

Chen, A., Kim, J., Lee, S. & Kim, Y., 2010. Stochastic multi-objective models for network design problem. *Expert Systems with Applications,* 37(2), pp. 1608-1619.

Cheng, C. H., Goh, C.-H. & Lee, A., 1996. Solving the generalized machine assignment problem in group technology. *Journal of the Operational Research Society,* 47(6), pp. 794-802.

Christiansen, S., Patriksson, M. & Wynter, L., 2001. Stochastic bilevel programming in structural optimization. *Structural and multidisciplinary optimization,* 21(5), pp. 361-371.

Chu, P. & Beasley, J., 1997. A genetic algorithm for the generalized assignment problem. *Computers and Operations Research,* 24(1), p. 17–23.

Chu, Y., You, F., Wassick, J. M. & Agarwal, A., 2015. Integrated planning and scheduling under production uncertainties: Bi-level model formulation and hybrid solution method. *Computers & Chemical Engineering,* Volume 72, pp. 255-272.

Clarke, S. M., Griebsch, J. H. & Simpson, T. W., 2005. Analysis of support vector regression for approximation of complex engineering analyse. *Journal of Mechanical Design, Transactions of the ASME,* Volume 127, pp. 1077 - 1087.

Colson, B., Marcotte, P. & Savard, G., 2005. A trust-region method for nonlinear bilevel programming: algorithm and computational experience. *Computational Optimization and Applications,* 30(3), pp. 211-227.

Cromley, R. G. & Hanink, D. M., 1999. Coupling land use allocation models with raster GIS. *Journal of geographical systems,* 1(2), pp. 137-153.

Dell'Amico, M. & Martello, S., 1997. The k-cardinality assignment problem. *Discrete Applied Mathematics,* 76(1-3), pp. 103-121.

Dempe, S., 2002. *Foundations of bilevel programming.* Berlin: Springer Science & Business Media.

Dempe, S., Dutta, J. & Mordukhovich, B., 2007. New necessary optimality conditions in optimistic bilevel programming. *Optimization,* 56(5-6), pp. 577-604.

Dıaz, J. A. & Fernández, E., 2001. A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research,* 132(1), pp. 22-38.

Dobson, G. & Nambimadom, R. S., 2001. The batch loading and scheduling problem. *Operations research,* 49(1), pp. 52-65.

Drexl, A., 1991. Scheduling of project networks by job assignment. *Management Science,* 37(12), pp. 1590-1602.

Dyer, D., 2010. *Watchmaker Framework for Evolutionary Computation.* [Online] Available at: http://watchmaker.uncommons.org/. [Accessed 25 May 2015].

Fisher, M., 1981. The Lagrangian relaxation method for solving integer programming problems. *Management Science,* 27(1), pp. 1-18.

Fisher, M., 2004. The Lagrangian relaxation method for solving integer programming problems. *Management Science,* 50(12), pp. 1861-1871.

Fisher, M. L. & Jaikumar, R., 1981. A generalized assignment heuristic for vehicle routing. *Networks,* 11(2), pp. 109-124.

Fisher, M. L., Jaikumar, R. & Van Wassenhove, L. N., 1986. A multiplier adjustment method for the generalized assignment problem. *Management Science,* 32(9), pp. 1095-1103.

Fleischer, L., Goemans, M. X., Mirrokni, V. S. & Sviridenko, M., 2006. *Tight approximation algorithms for maximum general assignment problems.* Miami, ACM-SIAM, pp. 611-620.

Freling, R., Romeijn, H. E., Morales, D. R. & Wagelmans, A. P., 2003. A branch-and-price algorithm for the multiperiod single-sourcing problem. *Operations research,* 51(6), pp. 922-939.

Fyfe, C., 2005. Artificial neural networks. In: *Do smart adaptive systems exist?.* Berlin: Springer, pp. 57-79.

Gardner, M. W. & Dorling, S. R., 1998. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment,* 32(14), pp. 2627-2636.

Goldberg, D. E. & Deb, K., 1991. A comparative analysis of selection schemes used in genetic algorithms. In: *Foundations of genetic algorithms 1.* Burlington: Morgan Kaufmann Publishers, Inc., pp. 69-93.

Golden, B., Assad, A., Levy, L. & Gheysens, F., 1984. The fleet size and mix vehicle routing problem. *Computers & Operations Research,* 11(1), pp. 49-66.

Gouveia, L. & Saldanha-da-Gama, F., 2006. On the capacitated concentrator location problem: a reformulation by discretization. *Computers & operations research,* 33(5), pp. 1242-1258.

Grygiel, G., 1981. Algebraic∑ k-assignment problems. *Control and Cybernetics,* 10(3-4), pp. 155-165.

Haddadi, S., 1999. Lagrangian decomposition based heuristic for the generalized assignment problem. *INFOR: Information Systems and Operational Research,* 37(4), pp. 392-402.

Haddadi, S. & Ouzia, H., 2004. Effective algorithm and heuristic for the generalized assignment problem. *European Journal of Operational Research,* 153(1), pp. 184-190.

Haga, M., Demuth, H., Beale, M. & De Jesús, O., 1996. *Neural Network Design (Vol 20).* Boston: PWS publishing company.

Halter, W. & Mostaghim, S., 2006. Bilevel optimization of multi-component chemical systems using particle swarm optimization. In: *2006 IEEE International Conference on Evolutionary Computation.* Vancouver: IEEE, pp. 1240-1247.

Hansen, P., Jaumard, B. & Savard, G., 1992. New branch-and-bound rules for linear bilevel programming. *SIAM Journal on scientific and Statistical Computing,* 13(5), pp. 1194-1217.

Harvey, N. J. A., Ladner, R. E., Lovász, L. & Tamir, T., 2006. Semi-matchings for bipartite graphs and load balancing. *Journal of Algorithms,* 59(1), pp. 53-78.

Heaton, J., 2015. Encog: Library of Interchangeable Machine Learning Models for Java and C#. *J. Mach. Learn. Res.,* 16(1), pp. 1243-1247.

Higgins, A. J., 1999. Optimizing cane supply decisions within a sugar mill region. *Journal of Scheduling,* 2(5), pp. 229-244.

Hosder, S. et al., 2001. Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. *Optimization and Engineering,* 2(4), pp. 431-452.

Hu, Z. & Mahadevan, S., 2016. A single-loop kriging surrogate modeling for time-dependent reliability analysis. *Journal of Mechanical Design,* 138(6).

Ishizuka, Y. & Aiyoshi, E., 1992. Double penalty method for bilevel optimization problems. *Annals of Operations Research,* 34(1), pp. 73-88.

Islam, M. M., Singh, H. K. & Ray, T., 2017. A surrogate assisted approach for single-objective bilevel optimization. *IEEE Transactions on Evolutionary Computation,* 21(5), pp. 681-696.

Jörnsten, K. & Näsberg, M., 1986. A new Lagrangian relaxation approach to the generalized assignment problem. *European Journal of Operational Research,* 27(3), pp. 313-323.

Kalashnikov, V. V. a. M. R. C. H., Camacho-Vallejo, J.-F. & Kalashnykova, N. I., 2016. A heuristic algorithm solving bilevel toll optimization problems. *The International Journal of Logistics Management,* 27(1), pp. 31-51.

Kassem, S., Hagras, H., Owusu, G. & Shakya, S., 2012. A type2 fuzzy logic system for workforce management in the telecommunications domain. In: *2012 IEEE International Conference on Fuzzy Systems.* Brisbane: IEEE, pp. 1-8.

Kern, M., Anim-Ansah, G., Owusu, G. & Voudouris, C., 2006. FieldPlan: Tactical Field Force Planning in BT. In: *IFIP International Conference on Artificial Intelligence in Theory and Practice.* Boston: Springer, pp. 345-354.

Kern, M. & Owusu, G., 2008. Tactical Resource Planning and Deployment. In: *Service Chain Management.* Berlin: Springer, pp. 65-77.

Kern, M., Shakya, S. & Owusu, G., 2009. Integrated resource planning for diverse workforces. In: *2009 International Conference on Computers & Industrial Engineering.* Troyes: IEEE, pp. 1169-1173.

Kişi, Ö. & Uncuoğlu, E., 2005. Comparison of three back-propagation training algorithms for two case studies. *Indian Journal of Engineering and Materials Sciences,* 12(5), pp. 434-442.

Kourentzes, N. & Crone, S., 2010. *Advances in forecasting with artificial neural networks,* Lancaster: The Department of Management Science.

Küçükaydin, H., Aras, N. & Altinel, I. K., 2011. Competitive facility location problem with attractiveness adjustment of the follower: A bilevel programming model and its solution. *European Journal of Operational Research,* 208(3), pp. 206-220.

Kuhn, H. W., 1955. The Hungarian Method for the Assignment Problem. *Naval Research Logistics Quarterly,* 2(1-2), pp. 83-97.

Kutner, M., 1996. *Applied linear statistical models.* Chicago: Irwin.

Lee, D. & Kim, Y.-D., 1998. A multi-period order selection problem in flexible manufacturing systems. *Journal of the Operational Research Society,* 49(3), pp. 278-286.

Lee, M.-K., 1992. A storage assignment policy in a man-on-board automated storage/retrieval system. *The International Journal of Production Research,* 30(10), pp. 2281-2292.

Li, H. & Wang, Y., 2007. A hybrid genetic algorithm for solving nonlinear bilevel programming problems based on the simplex method. *Third International Conference on Natural Computation (ICNC 2007),* Volume 4, pp. 91-95.

Liret, A. & Dorne, R., 2008. Work Allocation and Scheduling. In: *Service Chain Management.* Berlin: Springer, pp. 139-152.

Liu, Y. Y. & Wang, S., 2015. A scalable parallel genetic algorithm for the Generalized Assignment Problem. *Parallel Computing,* Volume 46, pp. 98-119.

Li, X., Tian, P. & Min, X., 2006. *A hierarchical particle swarm optimization for solving bilevel programming problems.* Berlin, Springer, pp. 1169-1178.

Lorena, L. A., Narciso, M. G. & Beasley, J. E., 1999. A constructive genetic algorithm for the generalized assignment problem. *Evolutionary Optimization.*

Lorena, L. N. & Narciso, M., 1996. Relaxation heuristics for a generalized assignment problem. *European Journal of Operational Research,* 91(3), pp. 600-610.

Lourenço, H. R. & Serra, D., 1998. Adaptive approach heuristics for the generalized assignment problem. *DEE-UPF, Economic Working paper Series,* Volume 288.

Lu, J., Han, J., Hu, Y. & Zhang, G., 2016. Multilevel decision-making: A survey. *Information Sciences,* Volume 346–347, pp. 463-487.

Lukač, Z., Šorić, K. & Rosenzweig, V. V., 2008. Production planning problem with sequence dependent setups as a bilevel programming problem. *European Journal of Operational Research,* 187(3), pp. 1504-1512.

Lv, Y., Hu, T., Wang, G. & Wan, Z., 2007. A penalty function method based on Kuhn--Tucker condition for solving linear bilevel programming. *Applied mathematics and computation,* 188(1), pp. 808-813.

Martello, S., Pulleyblank, W. R., Toth, P. & De Werra, D., 1984. Balanced optimization problems. *Operations Research Letters,* 3(5), pp. 275-278.

Martello, S. & Toth, P., 1981. *An algorithm for the generalized assignment problem.* Hamburg, North-Holland.

Mathieu, R., Pittard, L. & Anandalingam, G., 1994. Genetic algorithm based approach to bi-level linear programming. *Operations Research,* 28(1), p. 1–21.

Mesbah, M., Sarvi, M. & Currie, G., 2011. Optimization of transit priority in the transportation network using a genetic algorithm. *IEEE Transactions on Intelligent Transportation Systems,* 12(3), pp. 908-919.

Mirjalili, S., 2019. Genetic algorithm. In: *Evolutionary algorithms and neural networks.* Berlin: Springer, pp. 43-55.

Mohamed, A., Hagras, H., Shakya, S. & Owusu, G., 2012. Tactical resource planner for workforce allocation in telecommunications. In: *International Conference on Autonomous and Intelligent Systems.* Aveiro: Springer, pp. 87-94.

Mohamed, A., Hagras, H., Shakya, S. & Owusu, G., 2013. A fuzzy-genetic tactical resource planner for workforce allocation. In: *2013 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS).* Singapore: IEEE, pp. 98-105.

Nowakovski, J., Schwarzler, W. & Triesch, E., 1999. Using the generalized assignment problem in scheduling the ROSAT space telescope. *European Journal of Operational Research,* 112(3), pp. 531-541.

O'Hanley, J. R. & Church, R. L., 2011. Designing robust coverage networks to hedge against worst-case facility losses. *European Journal of Operational Research,* 209(1), pp. 23-36.

Oduguwa, V. & Roy, R., 2002. Bi-level optimisation using genetic algorithm. In: *Proceedings 2002 IEEE International Conference on Artificial Intelligence Systems (ICAIS 2002).* Divnomorskoe: IEEE, pp. 322-327.

Önal, H., 1993. A modified simplex approach for solving bilevel linear programming problems. *European Journal of Operational Research,* 67(1), pp. 126-135.

Osman, I. H., 1995. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *Operations-Research-Spektrum,* 17(4), pp. 211-225.

Owusu, G., Anim-Ansah, G & Kern, M., 2008. Strategic Resource Planning. In: *Service Chain Management.* Berlin: Springer, pp. 35-49.

Owusu, G. & O'Brien, P., 2013. Transforming Field and Service Operations with Automation. In: *Transforming Field And Service Operations.* Berlin: Springer, pp. 15-28.

Owusu, G. et al., 2006. On optimising resource planning in BT plc with FOS. In: *2006 International Conference on Service Systems and Service Management.* Troyes: IEEE, pp. 541-546.

Ozcanan, S. & Atahan, A., 2021. Minimization of Accident Severity Index in concrete barrier designs using an ensemble of radial basis function metamodel-based optimization. *Optimization and Engineering,* Volume 22, p. 485–519.

Panin, A. A., Pashchenko, M. G. & Plyasunov, A. V., 2014. Bilevel competitive facility location and pricing problems. *Automation and Remote Control,* 75(4), pp. 715-727.

Pentico, D. W., 2007. Assignment problems: A golden anniversary survey. *European Journal of Operational Research,* 176(2), pp. 774-793.

Pigatti, A., De Aragao, M. P. & Uchoa, E., 2005. Stabilized branch-and-cut-and-price for the generalized assignment problem. *Electronic Notes in Discrete Mathematics,* 36(C), pp. 389-395.

Pirkul, H., 1986. An integer programming model for the allocation of databases in a distributed computer system. *European Journal of Operational Research,* 26(3), pp. 401-411.

Planatscher, H. & Schober, M., 2015. *SCPSolver - an easy to use Java Linear Programming Interface.* [Online]
Available at: http://scpsolver.org/
[Accessed 1 January 2015].

Prechelt, L., 1994. *PROBEN1 - A Set of Neural Network Benchmark Problems and Benchmarking Rules,* Karlsruhe: Technical Report 21/94.

Racer, M. & Amini, M., 1994. A robust heuristic for the generalized assignment problem. *Annals of Operations,* Volume 50, p. 487–503.

Rangarajan, L., 2010. Bi-level dimensionality reduction methods using feature selection and feature extraction. *International Journal of Computer Applications,* 4(2), pp. 33-38.

Ravindran, A. & Ramaswami, V., 1977. On the bottleneck assignment problem. *Journal of Optimization Theory and Applications,* 21(4), pp. 451-458.

Ross, E., 2017. *Cross-trained workforce planning models,* Lancaster: Lancaster University.

Ross, G. T. & Soland, R. M., 1975. A branch and bound algorithm for the generalized assignment problem. *Mathematical programming,* 8(1), pp. 91-103.

Ross, G. T. & Soland, R. M., 1977. Modeling facility location problems as generalized assignment problems. *Management Science,* 24(3), pp. 345-357.

Ross, G. T. & Soland, R. M., 1980. A multicriteria approach to the location of public facilities. *European journal of operational research,* Volume 4, pp. 307-321.

Ruland, K. S., 1999. A model for aeromedical routing and scheduling. *International Transactions in Operational Research,* 6(1), pp. 57-73.

Sahni, S. & Gonzalez, T., 1976. P-complete approximation problems. *Journal of the ACM (JACM),* 23(3), pp. 555-565.

Schmidt, G. & Wilhelm, W. E., 2000. Strategic, tactical and operational decisions in multi-national logistics networks: A review and discussion of modelling issues. *International Journal of Production Research,* 38(7), pp. 1501-1523.

Shakya, S. et al., 2013. Enhancing Field Service Operations via Fuzzy Automation of Tactical Supply Plan. In: *Transforming Field and Service Operations.* Berlin: Springer, pp. 101-114.

Shi, C., Lu, J. & Zhang, G., 2005. An extended Kuhn--Tucker approach for linear bilevel programming. *Applied Mathematics and Computation,* 162(1), pp. 51-63.

Shi, C., Lu, J., Zhang, G. & Zhou, H., 2006. An extended branch and bound algorithm for linear bilevel programming. *Applied Mathematics and Computation,* 180(2), pp. 529-537.

Shmoys, D. B. & Tardos, É., 1993. An approximation algorithm for the generalized assignment problem. *Mathematical programming,* 62(1), pp. 461-474.

Sibi, P., Jones, S. & Siddarth, P., 2013. Analysis of different activation functions using back propagation neural networks. *Journal of Theoretical and Applied Information Technology ,* 47(3), pp. 1264-1268.

Sinha, A., Lu, Z., Deb, K. & Malo, P., 2020. Bilevel optimization based on iterative approximation of multiple mappings. *Journal of Heuristics,* 26(2), pp. 151-185.

Sinha, A., Malo, P. & Deb, K., 2013. Efficient evolutionary algorithm for single-objective bilevel optimization. *arXiv preprint arXiv:1303.3901.*

Sinha, A., Malo, P. & Deb, K., 2015. Transportation policy formulation as a multi-objective bilevel optimization problem. In: *2015 IEEE Congress on Evolutionary Computation (CEC).* Sendai: IEEE, pp. 1651-1658.

Sinha, A., Malo, P. & Deb, K., 2017. Evolutionary algorithm for bilevel optimization using approximations of the lower level optimal solution mapping. *European Journal of Operational Research,* 257(2), pp. 395-411.

Sinha, A., Malo, P., Frantsev, A. & Deb, K., 2013. Multi-objective stackelberg game between a regulating authority and a mining company: A case study in environmental economics. In: *2013 IEEE congress on evolutionary computation.* Cancun: IEEE, pp. 478-485.

Sinha, A., Malo, P., Frantsev, A. & Deb, K., 2014. Finding optimal strategies in a multi-period multi-leader–follower Stackelberg game using an evolutionary algorithm. *Computers & Operations Research,* Volume 41, pp. 374-385.

Sinha, A., Malo, P., Xu, P. & Deb, K., 2014. A bilevel optimization approach to automated parameter tuning. In: *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation.* Vancouver: GECCO, pp. 847-854.

Spears, W. M. & De Jong, K. D., 1995. *On the virtues of parameterized uniform crossover,* Washington DC: Naval Research Lab.

Srivastava, N. et al., 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research,* 15(1), pp. 1929-1958.

Suryan, V., Sinha, A., Malo, P. & Deb, K., 2016. Handling inverse optimal control problems using evolutionary bilevel optimization. In: *2016 IEEE Congress on Evolutionary Computation (CEC).* Vancouver: IEEE, pp. 1893-1900.

Trick, M. A., 1992. A linear relaxation heuristic for the generalized assignment problem. *Naval Research Logistics,* 39(2), pp. 137-151.

Tuy, H., Migdalas, A. & Värbrand, P., 1993. A global optimization approach for the linear two-level program. *Journal of Global Optimization,* 3(1), pp. 1-23.

Van Ackere, A., 1993. The principal/agent paradigm: Its relevance to various functional fields. *European Journal of Operational Research,* 70(1), pp. 83-103.

Verma, D. C., 2004. Service level agreements on IP networks. *Proceedings of the IEEE,* 92(9), pp. 1382-1388.

Vicente, L., Savard, G. & Júdice, J., 1994. Descent approaches for quadratic bilevel programming. *Journal of Optimization theory and applications,* 81(2), pp. 379-399.

Voudouris, C., 2008. Defining and Understanding Service Chain Management. In: *Service Chain Management.* Berlin: Springer, pp. 1-17.

Voudouris, C., Owusu, G., Dorne, R. & Lesaint, D., 2008. Forecasting and Demand Planning. In: *Service Chain Management.* Berlin: Springer, pp. 51-64.

V, V., 2009. Decomposition Techniques for MILP: Lagrangian Relaxation. In: *Encyclopedia of Optimization.* Boston: Springer, pp. 632-638.

Wang, G. G. & Shan, S., 2006. Review of metamodeling techniques in support of engineering design optimization. *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference,* Volume 4255, pp. 415-426.

Wang, G., Wan, Z., Wang, X. & Lv, Y., 2008. Genetic algorithm based on simplex method for solving linear-quadratic bilevel programming problem. *Computers & Mathematics with Applications,* 56(10), p. 2550–2555.

Wang, J. Y., Ehrgott, M., Dirks, K. N. & Gupta, A., 2014. A bilevel multi-objective road pricing model for economic, environmental and health sustainability. *Transportation Research Procedia,* Volume 3, pp. 393-402.

Wan, Z., Wang, G. & Sun, B., 2013. A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems. *Swarm and Evolutionary Computation,* 8(1), p. 26–32.

Whittaker, G. et al., 2017. Spatial targeting of agri-environmental policy using bilevel evolutionary optimization. *Omega,* Volume 66, pp. 15-27.

Wiesemann, W., Tsoukalas, A., Kleniati, P.-M. & Rustem, B., 2013. Pessimistic bilevel optimization. *SIAM Journal on Optimization,* 23(1), pp. 353-380.

Wilson, J., 1997. A simple dual algorithm for the generalized assignment problem. *Journal of Heuristics,* 2(4), pp. 303-311.

Yagiura, M., Ibaraki, T. & Glover, F., 2004. An Ejection Chain Approach for the Generalized Assignment Problem. *INFORMS Journal on computing,* 16(2), pp. 133-151.

Yagiura, M., Ibaraki, T. & Glover, F., 2006. A path relinking approach with ejection chains for the generalized assignment problem. *European Journal of Operational Research,* 169(2), pp. 548-569.

Yagiura, M., Yamaguchi, T. & Ibaraki, T., 1999. A variable depth search algorithm for the generalized assignment problem. In: *Metaheuristics: advances and trends in local search paradigms for optimization.* Boston: Kluwer Academic Publisher, p. 459–471.

Yamada, T., Russ, B. F., Castro, J. & Taniguchi, E., 2009. Designing multimodal freight transport networks: A heuristic approach and applications. *Transportation science,* 43(2), pp. 129-143.

Ye, J. J. & Zhu, D., 2010. New necessary optimality conditions for bilevel programs by combining the MPEC and value function approaches. *SIAM Journal on Optimization,* 20(4), pp. 1885-1905.

Yin, Y., 2000. Genetic-algorithms-based approach for bilevel programming models. *Journal of transportation engineering,* 126(2), pp. 115-120.

Yu, Y. & Prasanna, V. K., 2003. Resource allocation for independent real-time tasks in heterogeneous systems for energy minimization. *J. Inf. Sci. Eng.,* 19(3), pp. 433-449.

Zhang, G., Eddy Patuwo, B. & Hu, M. Y., 1998. Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting,* 14(1), pp. 35-62.

Zhu, X., Yu, Q. & Wang, X., 2006. A hybrid differential evolution algorithm for solving nonlinear bilevel programming with linear constraints. In: *2006 5th IEEE International Conference on Cognitive Informatics.* Beijing: IEEE, pp. 126-131.