# Tactical Plan Optimisation for Large Multi-Skilled Workforces using a Bi-Level Model

Russell Ainslie
School of Computing Science and Digital Media
Robert Gordon University
Aberdeen, UK
russell.ainslie@bt.com

John McCall
School of Computing Science and Digital Media
Robert Gordon University
Aberdeen, UK
j.mccall@rgu.ac.uk

Sid Shakya
EBTIC
Khalifa University,
Abu Dhabi
sid.shakya@kustar.ac.ae

Gilbert Owusu
Business Modelling and Operational Transformation Practice
British Telecom
Adastral Park, UK
gilbert.owusu@bt.com

***Abstract*—The service chain planning process is a critical component in the operations of companies in the service industry, such as logistics, telecoms or utilities. This process involves looking ahead over various timescales to ensure that available capacity matches the required demand whilst maximizing revenues and minimizing costs. This problem is particularly complex for companies with large, multi-skilled workforces as matching these resources to the required demand can be done in a vast number of combinations. The vastness of the problem space combined with the criticality to the business is leading to an increasing move towards automation of the process in recent years. In this paper we focus on the tactical plan where planning is occurring daily for the coming weeks, matching the available capacity to demand, using capacity levers to flex capacity to keep backlogs within target levels whilst maintaining target levels for provision of new revenues. First we describe the tactical planning problem before defining a bi-level model to search for optimal solutions to it. We show, by comparing the model results to actual planners on real world examples, that the bi-level model produces good results that replicate the planners' process whilst keeping the backlogs closer to target levels, thus providing a strong case for its use in the automation of the tactical planning process.**

***Keywords—Bi-level, tactical planning, optimisation, genetic algorithm, GA, linear programming***

## I. INTRODUCTION

In recent decades, with the advent of privatization 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 [1]. Recent research has shown that this is increasingly involving the automation of planning processes [2].

One of the key areas for development lies within the field of service chain planning. Planning for a service industry involves the matching of resources' available time to the jobs requiring completion. This matching occurs along the dimensions of time periods, area and skill. Each job requires a skill to complete, in a certain geographical area, within a certain period of the plan. The planner attempts to match this with the skills and areas a resource can cover within each period, by using some simple rules [3]. The key objective of planning is to have the right resource available at the right time in the right place to fulfil the customer demand. Advanced planning of resources helps firms to maximize the utilization and minimize the wastage, and by doing so it helps to fulfil customer demand and maximize revenue, at the same time minimizing cost. Planning can be categorized into three groups [4].

1. Strategic planning - a long term plan looking 3-5 years ahead deciding company strategy for the next few years. [5]
2. Operational planning - a medium term plan looking at 12 to 18 months ahead taking operational decisions such as demand prioritization and resource recruitment. [6]
3. Tactical planning– a short term plan up to 90 days ahead balancing daily capacity against daily workstack [7].

We will be focusing on tactical planning in the rest of the paper. Tactical planning is typically done on a daily basis up to 90 days in advance. However much of the input from the planner is required for the first 7 to 14 days, i.e. near the execution date, where the unforeseen fluctuation in available resource capacity as well as change in demand has to be taken into account to produce an actionable plan to match demand to supply. Depending upon the type of task to be undertaken, the planning process can be different. For example, tasks that are related to faults and require repair are predicted in advance due to their repetitive nature and past patterns of fault volumes. This means there are a steady stream of fault tasks requiring completion within the plan that add up into a backlog that the planner aims to keep at a manageable level. In contrast there are also tasks where the planner has some control over the volumes such as installation tasks; here the planner sets the number of appointments to open up for selling to the customer. For example, in the case of a telecoms company this could be appointments to supply a new customer with a fibre connection. The goal of the planning process is to ensure there is enough capacity available to keep the fault backlogs at a level where service targets are being met whilst also opening up enough installation tasks to meet selling targets to maximise new revenue. To meet this goal, capacity can be flexed by applying factors such as overtime, contractors or moving resources between areas. This, in essence, is the tactical planning problem which becomes very large and complex when dealing with a large multi-skilled workforce. A large number of resources, each capable of performing multiple different skills, requiring matching up to a vast number of tasks, with different levers available to flex the capacity, leads to a vast number of possible solutions. In this paper we propose a bi-level model [8] [9] to solve this large problem and compare the solutions provided with those reached by some actual planners using some real world data.

The paper is organized as below. Section 2 describes the tactical planning problem and illustrates its size providing a justification for breaking the problem down into two sections to create a bi-level model. Section 3 defines this bi-level model, describing the leader and follower models used to solve the problem. Section 4 reports the results of the experiments comparing the model outputs with actual planners on real world data. We conclude the paper in section 5.

## II. The Tactical Planning Problem

This section of the paper first defines the tactical planning problem, briefly explaining the planning process, before formalising the portion of the plan we are looking to optimise. We then give a couple of examples of possible solution approaches along with the justification for investigating the bi-level approach which is followed in the rest of this paper.

### A. *Problem Definition*

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 day 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.

The demand portion of the plan consists of the jobs that are to be completed. These jobs 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 can set the number to be completed. The former type usually involves repairing of faults where the intake 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 paper the former shall be referred to as faults and the latter as installation.

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 for the purposes of the model built in this paper is a configurable percentage of the backlog at the start of that day in the plan.

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. For the purpose of the model built in this paper these workstacks contain a target number of jobs to aim for on each day of the plan.

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; the second part is the attempt to bridge the gap between the capacity and the demand through the use of various levers.

The demand lever as mentioned earlier is to decide the number of installation jobs to make available for booking. The remaining levers all relate to the capacity. These include things such as overtime, resource loans (loaning resource from one area to another), shrinkage (amount of leave and sickness), contractors and productivities (the amount of time a resource takes to complete a job of a given skill). For the purpose of the problem explored in this paper we are taking the productivity and shrinkage levers as fixed, leaving the remaining overtime, contractors and area loan levers available to attempt to shape the capacity to fit the demand.

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.

### B. *Solution Methods*

The tactical planning problem, particularly for organisations with large multi-skilled workforces, quickly becomes very large. Two of the initial solution methods considered were that of a linear programming model [10] or use of a meta-heuristic [11] [12]such as a genetic algorithm (GA) [13] [14] or particle swarm optimisation (PSO) [15] [16].

In both cases, the large size of the problem was highlighted as a bottleneck. Even taking a reduced example, where the overall problem is split geographically into a number of separate areas, to solve the planning problem for and solving for a single week only still remains large.

Let’s take a small example where we have 50 different resources that, on average, can perform 7 skills each out of a total of 13 skills. This gives approx. 350 decision variables for each day of the plan and 2450 across the week, simply for allocating their time to each of their skills. Adding in overtime decisions, 50 for each day of the plan for 350 in total, contractor decisions, one for each 13 skills, on each day of the plan for a total of 91 decision variables, and area loans, same number as overtime for another 350, the problem is already very large at

3241 variables before even adding the demand decision variables.

These decisions also come with a large number of constraints which further complicates the linear modelling approach. The heuristic does not fare much better as, to model all of the possible decision variables, a large chromosome with many possible values for each allele is required, leading to a very large search space. It may be possible to effectively search this space using some advanced techniques, the solution explored in this paper is that of breaking the problem down into smaller, inter-connected, problems to allow the use of different techniques for solving each sub-problem. The solution was a bi-level model with the leader model, pulling the capacity levers, to set the capacity constraints that is used by follower model. This is outlined in the next section.

## III. Bi-Level Model

In this section of the paper, the bi-level model used to solve the tactical planning problem is specified. 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.

### A. *General Structure*

The general structure of a bi-level model is given by the equation [17]:

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

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 paper, 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 the rest of this section.

### B. *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. In this paper a genetic algorithm was used to model this level of the problem. The following sub-sections of this paper describe the chromosome structure used for this GA, followed by the constraints applied to the problem, the evolution methods and finally the fitness function.

#### 1) *Chromosome Structure*

Fig. 1 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. 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 hour) 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.

#### 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. Let's define the set of decision variables for the follower model as $Y$ and those for the leader as $c_{nt}$, $o_{mt}$ and $r_{mt}$ which are the values in period $t$ for contractors applied to skill $n$, and overtime and reductions applied to resource $m$ respectively. If we further define the variable $\tau_{mt}$ 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:

$$min: F(c, o, r, y) \text{ for } c\in C, o\in O, r\in R, y\in Y \tag{2}$$

*subject to*:

1. $\sum_{nt} c_{nt} \le C_{budget}$
2. $\sum_{mt} o_{mt} \le O_{budget}$
3. $\sum_{m} r_{mt} \le R_{budget} \ \forall\ t \in T$
4. $o_{mt} \le O_{max} \ \forall\ \text{m} \in M, t \in T$
5. $o_{mt} + \tau_{mt} \le \tau_{max} \ \forall\ \text{m}\in M, t\in T$

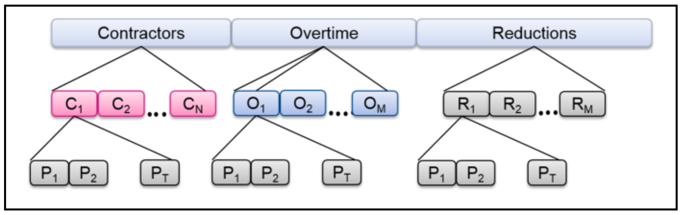

Fig. 1. Leader model solution chromosome

The first three constraints for (2) are budget constraints which are simply the maximum allowed contractors, $C_{budget}$, overtime, $O_{budget}$, and reductions, $R_{budget}$, allowed within the plan. For contractors and overtime, this just set 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 can't 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, $O_{max}$, and the maximum total time a resource could have in a period, $A_{max}$. 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.

#### 3) Evolution Methods

The GA used the standard roulette wheel selection [18] to select the parents for subsequent generations. Mutation chance was configured separately for each of the 3 sections of the chromosome, with mutation causing a 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 standard deviation of 0.333. For the reductions section, a mutation just increased or decreased the reduction amount for that resource in that period by 1. All of the mutations were restricted to only allow solutions that would meet the model constraints. If a solution was mutated then a check and fix process was performed that would modify the solution to ensure it remained within the valid solution space if required.

Crossover was performed using a standard crossover process with a configurable number of crossover points and probability of occurring. Crossover was allowed to occur within sections as well as between sections, however crossover that split a section would got through a *check and fix* stage afterwards to ensure that the max constraint for that section was not broken.

#### 4) Fitness Function

The fitness of a solution was 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 $\omega_c$, $\omega_o$ and $\omega_r$ 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:

$$F(c, o, r, z) = \omega_c \Sigma_{nt} c_{nt} + \omega_o \Sigma_{mt} o_{mt} + \omega_r \Sigma_{mt} r_{mt} + f(z) \quad (3)$$

### C. 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. For the purposes of this paper, the model was built using linear programming as previous work had shown linear programming to be good at solving this sub-problem [6]. Here we first describe the linear formulation of this problem before expanding on the methods used to optimise towards the target workstack levels.

#### 1) Linear Model Formulation

Further defining the members of *Y*, the follower decision variables are defined 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 jobs for fault skill *s* in period *t*,

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

$d''_{st}$ is completion of installation jobs beyond the target 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:

$$min: f(a, d, d', d'', L, E) = \sum_{t \in T} \left[ \begin{array}{c} \sum_{m \in M} \left[ \sum_{s \in S_r} \omega_m a_{mst} \right] + \\ \sum_{s \in S_{fault}} [\omega_{lt} L_{st} + \emptyset E_{st} - \omega_f d_{st}] - \sum_{s \in S_{prov}} [\omega_p d'_{st} + \omega_{ap} d''_{st}] \end{array} \right] \quad (4)$$

subject to:

1. $\Sigma_s \, a_{mst} \leq o_{mt} + \tau_{mt} \; \forall \; m, t$
2. $\Sigma_m \, p_{mst} a_{mst} - d_{st} = 0 \; \forall t, s_{fault}$
3. $\Sigma_m \, p_{mst} a_{mst} - d'_{st} - d''_{st} = 0 \; \forall t, s_{prov}$
4. $d_{st} + L_{st} + E_{st} - L_{s(t-1)} - E_{s(t-1)} = I_{s(t-1)} \; \forall t, s_{fault}$
5. $E_{st} + (\rho_{st} - 1) \, E_{s(t-1)} \leq (1 - \rho_{st}) \, I_{s(t-1)} \; \forall t, s_{fault}$
6. $L_{st} - L_{s(t-1)} - \rho_{st} \, E_{s(t-1)} \leq \rho_{st} I_{s(t-1)} \; \forall t, s_{fault}$
7. $d_{st} - \sigma_{st} \, L_{s(t-1)} - \sigma_{st} \, E_{s(t-1)} \leq \sigma_{st} \, I_{s(t-1)} \; \forall t, s_{fault}$
8. $d'_{st} \leq \rho_{st} \; \forall t, s_{prov}$
9. $(d'_{st} + d''_{st}), d_{st} \geq d^{min}_{st} \; \forall t, s$
10. $a_{mst}, L_{st}, d'_{st}, d''_{st}, E_{st} \geq 0 \; \forall t, s, m$

The weight variables $\omega_m$, $\omega_{lt}$, $\omega_f$, $\omega_p$ and $\omega_{ap}$ are the costs and benefits for using resource time, rolling over target fault completions, completing faults jobs, completing installation tasks up to the installation target and completing additional installation tasks respectively. The final cost variable $\emptyset$ is a zero cost that is attached to rolling over of additional fault jobs beyond the target.

The constraints for (4) 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, $\tau_{mt}$, 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 or $d'_{st} + d''_{st}$ in the case of an installation skill. 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 $\rho_{st}$. 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 $\sigma_{st}$. The 8th constraint also uses the $\rho_{st}$ 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.

### 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, $\rho_{st}$, and second is the remainder. Constraints 5 and 6 of (4) encode this behaviour into the linear model.

Fig. 2 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 $L^{max}_{s1}$ is the target number of completions for skill $s$ in the first period, defined as:

$$L^{max}_{s1} = \rho_{s1} * Backlog \tag{5}$$

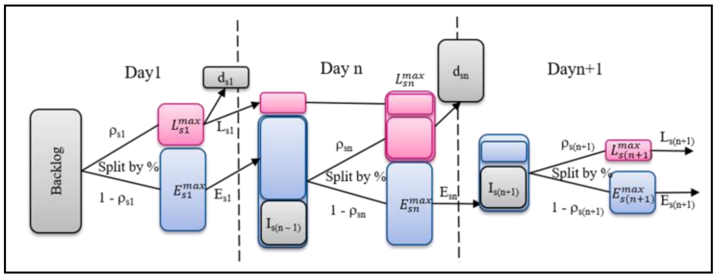

Fig. 2. Fault workstack propagation in the linear model

The remainder of the backlog is represented by $E^{max}_{s1}$. 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:

$$L^{max}_{sn} = \rho_{sn} \left(I_{s(n-1)} + E_{s(n-1)}\right) + L_{s(n-1)} \tag{6}$$

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

$$E^{max}_{sn} = (1 - \rho_{sn}) \left(I_{s(n-1)} + E_{s(n-1)}\right) \tag{7}$$

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.

## D. *Model Configuration*

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

The first fix for the model weights was 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 was 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 $\omega_m / p_{mst}$. 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 we 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, $\omega_m$ was set to 1.0, with the benefit for completing a fault job, $\omega_f$, set to -1.0 and the penalty for rolling over a target fault job to the next period, $\omega_{lt}$, 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, $\omega_p$, was set to -1.05 and the benefit for any additional jobs, $\omega_{ap}$, 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, $\omega_o$, was set to 0.05 and contractors, $\omega_c$, 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.

## IV. Results

In this section of the paper 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.

### A. 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.

The target percentages for the fault workstacks were set to the same as those used by the planners during that week with similar target installation levels set. 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.

### B. Experimental Method

The model was run on each of the 10 datasets with the evolution set to stop after 10 generations of stagnation. The mutation probability for all 3 sections of the chromosome were set to 0.05 with the crossover probability of 0.85. 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. The maximum amount of reductions allowed per day was set to 20. 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.

### C. Comparison to Planners

Table 1 shows the results of the comparison experiments, that 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.

TABLE I. Results comparing the model vs real planners

| **Area** | **Average Skill Per Day** | | **Average Per Day** | | |
|---|---|---|---|---|---|
| | ***Fault Off-Target*** | ***Installation Sold*** | ***Overtime*** | ***Contractors*** | ***Resources*** |
| 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 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. The data shows that on average the model used 11.1 more hours of overtime daily but used 4 less 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 mode, thus in that sense the model has produced a more optimal solution than the actual planners across all the problem instances. 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.

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. 3. 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 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. 4 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.

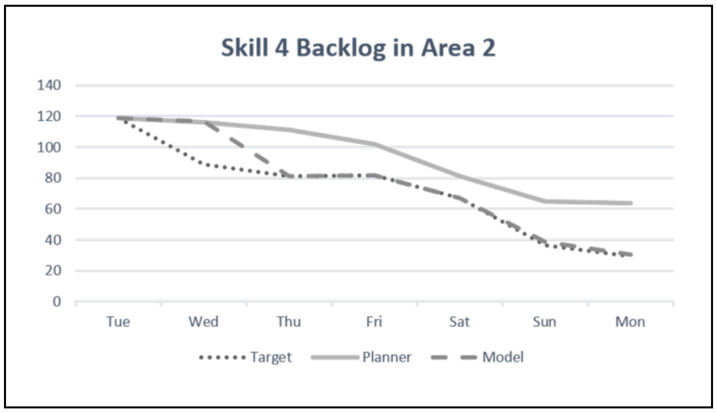

Fig. 4. Target backlog level for skill 4 in area 2 vs. that achieved by the planner and the bi-level model

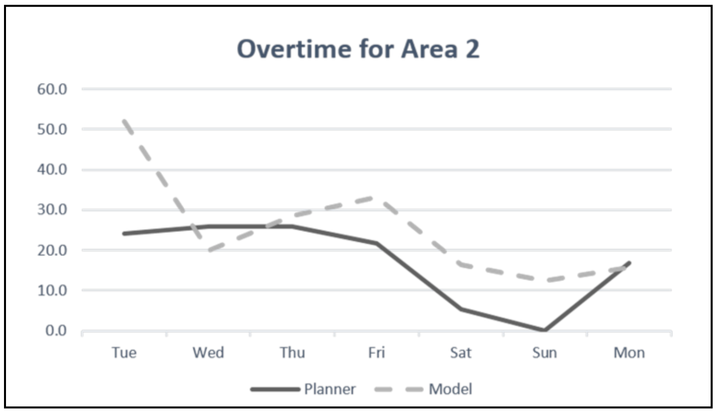

Fig. 3. Overtime applied by the planner and the bi-level model in area 2

## V. CONCLUSION

In this paper we defined the tactical planning problem as a large and complex problem when dealing with a large multi-skilled workforce. We described a bi-level model for solving this problem, using a GA as the leader model that set the capacity constraints for the follower model, which was formulated as a linear programming 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. We also gave the model 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.

We showed, using real world data that this model produces solutions that 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 capacity plan in the future.

Future work in this area would be to investigate methods to improve the run-time of the model, such as by taking advantage of the bi-level structure to replace the follower model with a surrogate requiring cheaper computation effort.

## REFERENCES

[1] C. Voudouris, "Defining and Understanding Service Chain Management," in *Service Chain Management*, Springer, 2008, pp. 1-17.

[2] G. Owusu, P. O'Brien, J. McCall and N. F. Doherty, Transforming Field and Service Operations, Springer, 2013.

[3] S. Shakya, S. Kassem, A. Mohamed, H. Hagras and G. Owusu, "Enhancing Field Service Operations via Fuzzy Automation of Tactical

Supply Plan," in *Transforming Field and Service Operations*, Springer, 2013, pp. 101-114.

[4] C. Voudouris, G. Owusu, R. Dorne and D. Lesaint, Service Chain Management, Springer, 2008.

[5] G. Owusu, Anim-Ansah, G and M. Kern, "Strategic Resource Planning," in *Service Chain Management*, Springer, 2008, pp. 35-49.

[6] R. Ainslie, S. Shakya, J. McCall and G. Owusu, "Optimising Skill Matching in the Service Industry for Large Multi-Skilled Workforces," in *Research and Development in Intelligent Systems XXXII*, Springer International Publishing, 2015, pp. 231-243.

[7] M. Kern and G. Owusu, "Tactical Resource Planning and Deployment," in *Service Chain Management*, Springer, 2008, pp. 65-77.

[8] A. Sinha, P. Malo and K. Deb, "Evoltuionary Bi-Level Optimization: An Introduction and Recent Advances," in *Adaption, Learning and Optimization 20*, Springer, 2017, pp. 71-105.

[9] J. Lu, J. Han, Y. Hu and G. Zhang, "Multilevel decision-making: A survey," *Information Sciences*, pp. 463-487, 2016.

[10] C. A. Floudas and X. Lin, "Mixed Integer Linear Programming in Process Scheduling: Modeling, Algorithms, and Applications," in *Annals of Operations Research*, Springer, 2005, pp. 131-162.

[11] M. Dorigo and T. Stutzle, Ant Colony Optimization, MIT Press, 2004.

[12] S. Shakya and R. Santana, "A Review of Estimation of Distribution Algorithms and Markov Networks," in *Markov Networks in Evolutionary Algorithms. Adaptation, Learning and Optimization. series Vol. 14*, Springer, 2012, pp. 21-37.

[13] D. Goldberg, Genetic Algorithms in Search, Optimization amd Machine Learning, Addison-Wesley, 1989.

[14] R. L. Haupt and S. Haupt, Practical Genetic Algorithms, 2nd Edition, Wiley, 2004.

[15] Y. Zhang, S. Wang and G. Ji, "A Comprehensive Survey on Particle Swam Optimization Algorithm and Its Applications," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[16] S. Kiranyaz, T. Ince and M. Gabbouj, Multidimensional Particle Swarm Optimization for Machine Learning and Pattern Recognition, Springer, 2014.

[17] A. Sinha, P. Malo and K. Deb, "A review on Bilevel Optimization: From Classical to Evolutionary Approaches and Applications," *IEEE Transactions on Evolutionary Computations*, vol. PP, no. 99, 2017.

[18] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," in *Foundations of genetic algorithms 1*, Morgan Kaufmann Publishers, Inc., 1991, pp. 69-93.