Optimising Training for Service Delivery

Ilankaikone Senthooran
Data Science & AI, Monash University, Clayton, Australia

Pierre Le Bodic
Data Science & AI, Monash University, Clayton, Australia

Peter J. Stuckey
Data Science & AI, Monash University, Clayton, Australia

Abstract

We study the problem of training a roster of engineers, who are scheduled to respond to service calls that require a set of skills, and where engineers and calls have different locations. Both training an engineer in a skill and sending an engineer to respond a non-local service call incur a cost. Alternatively, a local contractor can be hired. The problem consists in training engineers in skills so that the quality of service (i.e. response time) is maximised and costs are minimised. The problem is hard to solve in practice partly because (1) the value of training an engineer in one skill depends on other training decisions, (2) evaluating training decisions means evaluating the schedules that are now made possible by the new skills, and (3) these schedules must be computed over a long time horizon, otherwise training may not pay off. We show that a monolithic approach to this problem is not practical. Instead, we decompose it into three subproblems, modelled with MiniZinc. This allows us to pick the approach that works best for each subproblem (MIP or CP) and provide good solutions to the problem. Data is provided by a multinational company.

2012 ACM Subject Classification Theory of computation → Integer programming; Theory of computation → Constraint and logic programming

Keywords and phrases Scheduling, Task Allocation, Training Optimisation

Digital Object Identifier 10.4230/LIPIcs.CP.2021.48

Acknowledgements We are grateful for our industry partner for this opportunity to work on a challenging real-life workforce planning problem and for the many discussions that have allowed us to conduct this work.

1 Introduction

Large and/or complex machinery and equipment needs regular servicing, so a significant role for companies who maintain such equipment is to schedule engineers to visit customers who have such equipment. Apart from regular servicing, equipment can break down so (emergency) repair visits by engineers also need to be scheduled.

In this work we consider a company that provides services for a wide variety of equipment. Each piece of equipment is complex, and engineers need to be explicitly trained on how to service each piece of equipment. In such circumstances the scheduling problem becomes hard: each engineer is trained to provide services for many, but still a limited subset of, types of machinery. Assigning an engineer to a service call is only possible if they possess all skills (typically few) required by that service call.

In this setting, given an existing roster of engineers and a forecast of service calls over a long horizon, the problem we are tackling consists in strategically deciding which engineers should be trained, and in what skill(s), in order to minimise costs and ensure a short response time. This generates a complex multi-layer decision problem:

- How many engineers do we need for each skill in order to cover demand?

- Which engineers should be trained with which new skills in order to meet demands?
- And finally, how do we schedule engineers to ensure customers needs are met in a timely manner?

The focus of this paper is on the first two parts of this problem. Of course the efficacy of the first two parts of the problem are only realised if we consider generating actual schedules, in order to see that we are meeting customer demand. The ability to simulate "what-if" scenarios, such as upcoming training opportunities and/or preparing for an anticipated change in demand, at a regular interval (e.g. once a year) allows the service delivery provider to make the best decisions regarding "whom to train" and "on what skill".

Another aspect of our problem is geographic. The partner we work with is international and has engineers available in different states (in this case, different geographical parts of Australia). Assigning service calls to engineers outside of their home state incurs travel times and costs that can be reduced by training the right engineers in the right skills in the right locations. States have different numbers and types of service activities, each one therefore needs a (possibly empty) tailor-made roster of engineers.

In this paper we give a mathematical model for optimising training for service delivery. We illustrate the model on real-world data provided by our industry partner. The model is broken into three parts to answer the three questions in turn: which skills are required? who should be trained with them? and how does this affect service task scheduling? We show that this separation is able to provide much more scalable, and indeed better solutions, than a monolithic model attempting to answer all three questions at once. We show that the training optimisation can deliver significant savings in comparison to the current assignment processes (with some caveats).

2 The Problem: Training for Service Delivery

From an operational point of view, service calls arise on the fly and engineers must be assigned to these jobs so that the calls can be answered as early as possible. An engineer assigned to a call must possess all the skills required by that job. An engineer is assigned to a job for the duration of the job, and to at most one job at a time. Assigning an engineer to a job outside of their home state incurs an extra fixed travel cost and a per-day living cost. A service call within Australia should not be assigned to an overseas engineer. However, an overseas job can be allocated to any engineer, including overseas engineers.

While we have just described the operational problem as an online scheduling problem, in this paper we consider it to be offline, as what we actually want to determine is how engineers should be trained, which is a strategic decision. In other words, we suppose that all jobs are known in advance. This is a simplification we make 1) because our industrial partner is mostly interested in how to train its workforce, rather than than how to schedule its jobs, and 2) because the actual online method used to assign jobs to engineers used by our industry partner is too complex to be replicated, for instance by simulation, and needs to be abstracted to make strategic decisions.

We also consider that training takes no additional time, because for our industrial partner, training a new skill takes a few days and is usually only available at a few times during the year. Hence if from a strategic point of view, we determine that it is valuable for an engineer to learn a given skill, it would outweigh any operational consideration. However, we take into consideration the training cost, which varies based on skill.

3 Modelling and Solving the Problem by Decomposition

Our industrial partner provided historical scheduling data to be used as a projection for future demand. While a straightforward formalisation of this problem into a single model was clearly a possibility, it proved not to be a good one. Indeed, while it is possible to create a model that trains engineers, assigns them to service calls and schedules them into a single monolithic problem that is solved using an off-the-shelf solver, be it a Constraint Programming (CP) or a Mixed-Integer Programming (MIP) solver, this approach would hit two major brick walls. First, as the reader might expect, and as our experiments will show, this is completely impractical due to the combined complexity of the intertwined subproblems. Second, if the model could be solved to optimality, the training decisions that would be taken would overfit the historical data by using the fact that future service requests are completely visible at training time. In reality, future demands are a "fog of war" that cannot be captured by a single scenario. Instead, modern optimisation modelling practice dictates that we would need to obtain or generate more scenarios. It is standard to model uncertainty with chance constraints that need to be satisfied with a certain probability, usually 90 or 95%. See [3] for instance. However, our experiments with a single scenario already fail to produce good solutions in a reasonable time.

To avoid both poor and overfitted solutions, we decompose the problem into three subproblems to reduce the complexity, increase the solving speed, and hide the scheduling decisions from the training decisions. The basic idea is to first forecast future demand using past data, typically over a longer period of time, and determine what skills are in short supply. This is the *capacity planning* subproblem. It is actually a combinatorial problem, not "just" data analysis, as for each skill there may be enough capacity to cover the jobs that require it, but no feasible assignment when taking all skills into account. This serves as an input to the second subproblem, which allocates these new skills to specific engineers, the *skill allocation* subproblem. The second subproblem embeds as a guide a relaxation of the third subproblem, *job scheduling*, which assigns engineers to jobs and schedules them, using their newly-acquired skills, if any. The third subproblem does not decide on training, but it allows us to measure the quality of the solution of the first two subproblems with respect to service quality (i.e. delays to answer calls) and travel costs, which we optimise over in the third subproblem.

To summarise, we solve in sequence the three following subproblems:

- **Capacity planning** identifying skills that are shortage in each state to serve local service calls with the local engineers based on the historical data.
- **Skill allocation** finding whom to train and on what skill to train from the identified set of skills that are in shortage based on the future service calls, allowing travel.
- **Job scheduling** assigning service calls to engineers and scheduling them so as to minimise the overall cost and/or response time.

The subproblems are discussed in detail in the following subsections.

3.1 Capacity Planning

Capacity planning is done for every state individually to identify the skills that are required to serve the local jobs with the local engineers. We look at the past data to identify what skills were in shortage for each state. This is done by matching the available skills with the requirement in a way that minimises the cost to train engineers to meet the shortage. Suppose a state has two engineers, one, say E1, with skills A and B, and other, say E2 with C. There are two jobs: one that requires skill A, and another that requires skill B. Suppose

that both require 5 days. Technically, in this case, both jobs can be performed by the local engineer E1 with skills A and B. However, one job must wait until the other is completed. To minimise the response time, engineer E2 should to be trained on either A or B. This decision depends on the training cost of the skills. If A is more expensive than B, E1 will perform the job that requires skill A and E2 will perform the other given training on skill B.

Now, lets say, we have an option to use local contractors to perform the job and the new objective is to minimise the total cost, both training and contractor charges. For the above example, if the cost of the contractor is cheaper than training an engineer, the tool will not suggest any skill training.

In the event where a state has no local engineers or not enough to cater for the demand within the period in consideration, no new skills will be suggested as there are no local engineers to be trained. Naturally, in this case, service calls are made using engineers from other states or by local contractors.

Next, we show the formulation of the capacity planning problem for a state.

3.1.1 Input and Derived Data

Input Index Sets.

- $\mathcal{J} = \{1, \ldots, N^{\mathcal{J}}\}$: set of jobs (service calls) that need to be assigned engineers and scheduled.
- $\mathcal{E} = \{1, \ldots, N^{\mathcal{E}}\}$: set of engineers / technicians.
- $\mathcal{SK} = \{1, \ldots, N^{\mathcal{SK}}\}$: set of skills.
- $\mathcal{S} = \{1, \ldots, N^{\mathcal{S}}\}$: set of all states both where jobs need to be served and engineers are located.
- $\mathcal{SK}_e^{eng} \subseteq \mathcal{SK}$: subset of skills that engineer $e \in \mathcal{E}$ has training on
- $\mathcal{E}_s \subseteq \mathcal{E}$: subset of engineers who are located in state $s \in \mathcal{S}$
- $\mathcal{J}_s^{sk} \subseteq \mathcal{J}$: subset of jobs from state $s \in \mathcal{S}$ that require skill $sk \in \mathcal{SK}$

Input Data.

- h^{start}: planning horizon start date
- h^{end}: planning horizon end date
- $wdays$: number of working days within the planning horizon $h^{start} - h^{end}$
- $nsdays$: number of days that becomes available upon training a new skill
- $split \in \mathbb{Z}^+$: number of periods that the planning horizon is split into
- loc_e^{eng}: location (state) of engineer $e \in \mathcal{S}$
- loc_j^{job}: location (state) of job $j \in \mathcal{J}$ needs to be performed
- d_j: duration of job $j \in \mathcal{J}$
- C^{cont}: per day cost to contract a job
- C_{sk}^{train}: cost of training on skill $sk \in \mathcal{SK}$

3.1.2 Decision Variables

- $skshort_{sk}^s \in \mathbb{Z}^+$: number of days that skill $sk \in \mathcal{SK}$ is in shortage in state $s \in \mathcal{S}$ during the planning horizon.
- $sksupp_{sk}^s \in \mathbb{Z}^+$: number of days that skill $sk \in \mathcal{SK}$ is supplied/allocated in state $s \in \mathcal{S}$ during the planning horizon.
- $skreq_{sk}^s \in \{0, 1\}$: 1 if skill $sk \in \mathcal{SK}$ is identified as required skill in state $s \in \mathcal{S}$ during the planning horizon, otherwise 0.

3.1.3 Constraints and Objective Function

Upper bound. on the skill supplied and in shortage is limited to the required amount.

$$skshort^s_{sk} \leq \sum_{j \in \mathcal{J}^{sk}_s} d_j, \quad sk \in \mathcal{SK}, s \in \mathcal{S}. \tag{1}$$

$$sksupp^s_{sk} \leq \sum_{j \in \mathcal{J}^{sk}_s} d_j, \quad sk \in \mathcal{SK}, s \in \mathcal{S}. \tag{2}$$

The amount of skill supplied cannot exceed the number of working days.

$$sksupp^s_{sk} \leq wdays, \quad sk \in \mathcal{SK}, s \in \mathcal{S}. \tag{3}$$

New skill suggestion. Skills that are not required by the jobs within the planning horizon are not suggested.

$$\sum_{j \in \mathcal{J}^{sk}_s} d_j = 0 \rightarrow skreq^s_{sk} = 0, \quad sk \in \mathcal{SK}, s \in \mathcal{S}. \tag{4}$$

Supply and demand. Demand for skill is met by the supply or by contractor (shortage) or by training.

$$\sum_{j \in \mathcal{J}^{sk}_s} d_j \leq sksupp^s_{sk} + skshort^s_{sk} + nsdays \times skreq^s_{sk}, \quad sk \in \mathcal{SK}, s \in \mathcal{S}. \tag{5}$$

Objective function. sum of the cost to contract the jobs that are shortage in skill and to conduct training on suggested new skills.

$$\min \leftarrow obj = C^{cont} \times \textstyle\sum_{sk \in \mathcal{SK}, s \in \mathcal{S}} skshort^s_{sk} + \sum_{sk \in \mathcal{SK}, s \in \mathcal{S}} C^{train}_{sk} \times skreq^s_{sk}. \tag{6}$$

3.2 Skill Allocation

From the previous step (capacity planning), we know what are the set of skills a state needs to handle the future jobs. So deciding whom to train and on what skill to train them is crucial since the wrong decision might lead to longer response time and higher cost. This step looks at some jobs in hand, preferably for a shorter period, and decides whom to train in the skills found to be in short supply in the previous step for all states simultaneously. Here we allow a job to be allocated to an engineer from a different state than the job's location (state). The objective here is to reduce the cost of training someone and the cost of travel involved in attending jobs that are assigned to an engineer from a different state. The travel includes the return flight cost and the accommodation cost equivalent to the length of the job duration. If a state needs a skill and that state has excess workforce, training someone locally is cheaper than the alternatives, and the algorithm will recommend training a local engineer on that skill. In this case, the local engineer can perform the job that requires the recommended skill.

Next, we present the formulation of the skill allocation problem of a state.

3.2.1 Input and Derived Data

Input Index Sets.

- $\mathcal{SK}^{job}_j \subseteq \mathcal{SK}$: subset of skills that are required to perform job $j \in \mathcal{J}$.
- $\mathcal{E}^{sk} \subseteq \mathcal{E}$: subset of engineers that have training on $sk \in \mathcal{SK}$

Input Data.
- $C^{flight}_{a,b} \in \mathbb{Z}^+, a \neq b$: flight cost of travelling from state $a \in \mathcal{S}$ to state $b \in \mathcal{S}$.
- $C^{acco}_{a} \in \mathbb{Z}^+$: per day accommodation cost in state $s \in \mathcal{S}$.
- cap^{train}: maximum number of new skills an engineer is allowed to acquire.
- cap^{job}: maximum number of jobs an engineer is allowed to undertake.

Functions. The constraints in our model use the following functions.
- $\mathsf{isOverseasJob}(j)$: returns true if job $j \in \mathcal{J}$ needs to be performed overseas, otherwise false.
- $\mathsf{isOverseasEngineer}(e)$: returns true if engineer $e \in \mathcal{E}$ is from overseas, otherwise false.

3.2.2 Decision Variables

- $newskills_e \in \mathcal{SK}^{eng}_e$: list of skills suggested/allocated to engineer $e \in \mathcal{E}$.
- $alloc_j \in \mathcal{E}^{sk}$, $sk \in \mathcal{SK}^{job}_j$: engineer that job $j \in \mathcal{J}$ is assigned to.
- $cost^{travel}_j$: travel cost to perform job $j \in \mathcal{J}$.
- $cost^{train}_{sk}$: cost of training an engineer in skill $sk \in \mathcal{SK}$.

3.2.3 Constraints and Objective Function

Available skills for training are restricted to those identified by the previous solution

$$newskills_e \subseteq \{sk \mid sk \in \mathcal{SK}, skreq^s_{sk} = 1\}, \quad s \in \mathcal{S}, e \in \mathcal{E}_s \tag{7}$$

Limit. on the number of new skill recommendations for an engineer.

$$|newskills_e| \leq cap^{train}, \quad e \in \mathcal{E} \tag{8}$$

A limited number of jobs are assigned to an engineer when deciding the skill recommendation.

$$\sum_{j \in \mathcal{J}} (alloc_j = e) \leq cap^{job}, \quad e \in \mathcal{E} \tag{9}$$

Travel cost. If a job is allocated to an engineer from a different state, apply flight and accommodation cost otherwise set the cost to zero.

$$loc^{eng}_{alloc_j} \neq loc^{job}_j \rightarrow cost^{travel}_j = C^{flight}_{loc^{eng}_{alloc_j}, loc^{job}_j} + C^{acco}_{loc^{job}_j} \times d_j, \quad j \in \mathcal{J} \tag{10}$$

$$loc^{eng}_{alloc_j} = loc^{job}_j \rightarrow cost^{travel}_j = 0, \quad j \in \mathcal{J} \tag{11}$$

Location. An engineer should only be assigned to a job if their existing skill set and the newly recommended skill matches the job skill requirement

$$\mathcal{SK}^{job}_j \subseteq \mathcal{SK}^{eng}_{alloc_j} \cup newskills_{alloc_j}, \quad j \in \mathcal{J} \tag{12}$$

An overseas engineer cannot perform jobs in Australia.

$$\neg\mathsf{isOverseasJob}(j) \rightarrow \neg\mathsf{isOverseasEngineer}(alloc_j), \quad j \in \mathcal{J} \tag{13}$$

Dominance. To increase the solving efficiency by eliminating the symmetries, we apply a dominance constraint – one that favours skill addition to an engineer with a subset of skills when compared to another engineer's skills.

$$\mathcal{SK}^{eng}_{e1} \subseteq \mathcal{SK}^{eng}_{e2} \rightarrow |newskills_{e1}| \geq |newskills_{e2}|, \quad e1, e2 \in \mathcal{E}, e1 \neq e2. \tag{14}$$

When the maximum number of new skills that an engineer can acquire within the planning period is restricted to one, as is often the case with our partner company, the above constraint is a dominance constraint, it does not remove any optimal solutions. However, when a new engineer can be trained in two or more skills we have no proof that this is a dominance constraint, and instead use it as a rule of thumb to improve solving efficiency.

Objective function. sum of training and travel costs.

$$\min \leftarrow obj = \sum_{e \in \mathcal{E}} \sum_{ns \in newskills_e} cost^{train}_{ns} + \sum_{j \in \mathcal{J}} cost^{travel}_{j}. \tag{15}$$

3.3 Job Scheduling

From the previous step (skill allocation), we know which engineer to train on what skill in order to perform given future jobs. The next step is to assign jobs to engineers while scheduling them. In this step, we assume the engineers have already acquired the training on the recommended skills and have added those skills to their existing skill set. The overall objective is to minimise the total travel cost.

Next, we show the formulation of the job scheduling problem of a state.

3.3.1 Input and Derived Data

Input Index Sets.

- $\mathcal{J}^{sk} \subseteq \mathcal{J}$: subset of jobs that require skill $sk \in \mathcal{SK}$
- $\mathcal{C}$: set of contractors

Input Data.

- $arrivalDate_j$: arrival date of job $j \in \mathcal{J}$
- $wait_j$: preferred wait time of job $j \in \mathcal{J}$
- $maxWait$: maximum time a job can wait until its been attended since the allowed start date, i.e. $arrivalDate_j + wait_j, j \in \mathcal{J}$

3.3.2 Decision Variables

- $startDate_j$: start date of job $j \in \mathcal{J}$
- $cost^{cont}_j$: cost to contract job $j \in \mathcal{J}$.

3.3.3 Constraints and Objective Function

All the constraints listed in step 2, except (a) the constraint on the number of new skills an engineer is allowed to acquire, (b) the dominance constraint and (c) the constraint on the set of possible engineers who can perform a job in the event contractors are permitted, are applied in this step. In addition to these, the following constraints are also enforced:

Non-overlapping. Two jobs that are allocated to the same engineer cannot overlap in time. We use the $\texttt{disjunctive}(s, d)$ constraint which forces tasks with start times given by array s and durations given by array d not to overlap. We apply this to each engineer e by replacing the duration of tasks that the engineer is not allocated by 0.

$$\texttt{disjunctive}(startDate, [\text{if } alloc_j = e \text{ then } d_j \text{ else } 0 | j \in \mathcal{J}]), \quad e \in \mathcal{E} \tag{16}$$

Limit on overlapping jobs. When contractors are not used, we enforce a redundant constraint to apply a cap on the number of jobs that can overlap, which is equal to the available engineers with the required skill. We use the global $cumulative(s, d, r, b)$ constraint which forces at each point in time, the total number of tasks (with start times given by array s, durations given by array d and resources required to perform the task given by array r) that overlap that point, does not exceed the limit given by b. We apply this to each skill sk.

$$\texttt{cumulative}([startDate_j | j \in \mathcal{J}^{sk}], [d_j | j \in \mathcal{J}^{sk}], [1 | j \in \mathcal{J}^{sk}], card(\mathcal{E}^{sk})), \quad sk \in \mathcal{SK} \tag{17}$$

Scheduling window. A job must only be scheduled after a set period since its arrival date. Each job depending on its type can have a different wait period.

$$startDate_j \geq arrivalDate_j + wait_j, \quad j \in \mathcal{J} \tag{18}$$

A job must be served within a set period after the wait time.

$$startDate_j \leq arrivalDate_j + wait_j + maxWait, \quad j \in \mathcal{J} \tag{19}$$

Possible engineers for a job. When contractors are permitted, we allow assigning an engineer with the required skill or a contractor for a job. Here we assume the contractor has the necessary skill and available in every state.

$$alloc_j \in \mathcal{E}^{sk} \cup \mathcal{C}, sk \in \mathcal{SK}_j^{job}, sk \in \mathcal{SK}, \quad j \in \mathcal{J} \tag{20}$$

Contractor Cost. If a job is allocated to a contractor, apply the associated cost otherwise set the cost to zero.

$$alloc_j \in \mathcal{C} \rightarrow cost_j^{cont} = C^{cont} \times d_j, \quad j \in \mathcal{J} \tag{21}$$

$$alloc_j \notin \mathcal{C} \rightarrow cost_j^{cont} = 0, \quad j \in \mathcal{J} \tag{22}$$

Objective function. sum of travel and contractor costs.

$$\min \leftarrow obj = \textstyle\sum_{j \in \mathcal{J}} cost_j^{travel} + cost_j^{cont}. \tag{23}$$

3.4 Monolithic Model

The single monolithic problem is essentially modelled by combining all constraints in step 2 and step 3, where the $newskills_e$ are kept as variables so that deciding whom to train on what, allocating jobs to engineers and scheduling are performed together. Equation (7) is omitted so there is no limit on the possible set of new skills that can be trained. The objective is to reduce the sum of training and travelling costs, i.e. Equation (15).

4 Experiment

We have evaluated our algorithm by executing it as a single-thread process on an Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz on actual data provided by an industry partner. The data has approximately 8800 jobs over two years requiring 113 different skills. We are given 53 engineers with a varying skill sets to perform the jobs at hand, and be trained further. The engineers are located in 7 states, including one overseas, while service calls occur in 22 states, including 14 overseas from where 10% of the service calls originate. We split the data into two, the first year (Y1) data is used for capacity planning and the second year (Y2) data is used to perform skill allocation and job scheduling.

We modelled our problem using MiniZinc [15], which allowed us to try different off-the-shelf solvers (CP and MIP) on the same model before deciding on the most suitable one. MiniZinc translates constraints into forms suitable for the chosen solver. For example, in MiniZinc, sets are native. For both CP and MIP solvers, the set constraints, given in Equation (7), are mapped to zero/one representations [2]. To choose an appropriate solver for each step in the decomposed approach and the monolithic model, we ran the preprocessing for each experiment using several solvers (CP and MIP) - Chuffed, Gecode, CBC, and Gurobi. The results were consistent on all occasions and we chose a MIP solver (Gurobi) for steps 1 and 2, and a CP solver (Chuffed) for step 3. For the monolithic model, a CP solver (Chuffed) worked best.

For step 3 and the monolithic model we use a programmed search strategy to find solutions quickly. During scheduling we choose the unscheduled job with the earliest start time and the then fix its start time to this earliest possible time and assign an engineer for that job first trying to schedule an engineer who resides in the same location as the job, thus favouring solutions with lower travel costs. We experimented with a number of search strategies and found this to be overall the most robust.

In all the tables provided here, steps 1, 2 and 3 refer to capacity planning, skill allocation and job scheduling stages of the decomposed approach, respectively. Tables 1 shows the results of experiments conducted to compare the performance of the decomposed approach against the monolithic approach for two settings: skill allocation and job scheduling over one-month and two-month periods. For this comparison, we performed the capacity planning using the first-year (Y1) data, and the skill allocation and job scheduling on the second-year (Y2) data. To have a fair comparison, we used the same Y2 data used in the monolithic model for the skill allocation step of the decomposed approach for each period. All the runs had a cutoff time set at 360 seconds except for the monolithic model, which had it set at 3600 seconds. The values shown in the tables indicate the values at the timeout.

The first line of Table 1 shows that determining the skills in short supply is straightforward, we can find an optimal solution in 10 seconds. The results in Table 1 show that while the monolithic model can generate better solutions within its much longer time limit for some of the one month long instances, it scales very poorly, unable to find solutions to one one month problem and any two month problems. Clearly decomposing the problem into 3 parts does restrict the resulting solutions, since an “optimal solution” for the decomposed problem can be bettered by the monolithic model. However, this was when we were looking at a shorter period to decide the skill allocation. From the experimental results presented next, we can see that when skill allocation is performed over an extended period, that is, when looking at the larger problem, the decomposed approach outperforms the monolithic model in terms of solution quality and solving speed. This is true for the all months, including the

cases that were previously bettered by the monolithic model. Overall it is clear that the monolithic model is not practical, even given far more resources and solving a restriction of the problem it cannot compete.

The second experiment, shown in Table 2, tackles a much more practical version of the problem, and is the default problem used in the tool delivered to the industrial partner. Here the decisions of what skills are in short supply and which engineers should be trained on which skills (steps 1 and 2) are both performed over 1 year of data. The table compares three scenarios: when no new skills can be trained, when we can train at most one new skill per engineer, and when we can train at most two new skills per engineer. Clearly given more flexibility of training skills allows us to train more engineers. The second part of the table shows the monthly solutions over the year under the three different scenarios. It considers two different maximum waiting times for jobs. It gives the time to compute the solution, the total wait time over all jobs, the number of job serviced by interstate engineers and the total travel cost, for each month of jobs. For each different wait time it also shows the year sums of the statistics. The headline result is that (perhaps unsurprisingly) enabling new skill training can significantly improve upon the overall costs of providing the service calls. Indeed new skill training is required when we restrict the waiting time, Y2M7 and Y2M8 have no solution with the available engineers and skills. The savings of targeted training are significant reducing overall costs by 30%. Interestingly the more flexible scenario where we can train two skills per person does not always lead to a better overall cost solution. Recall that the training decisions are made on the Y1 data and hence may not be completely reflected in the Y2 data that we actually schedule, and indeed the total number of new skills assigned to engineers only marginally increases. The more flexible scenario does lead to significantly less waiting time for customers though.

Given that in some months we find no viable schedule without using contractors, we extend the step 3 model to allow contractors. This requires a more complex search strategy to obtain good results, but ensures that we find a viable schedule for each month. We applied a search strategy that is similar to the one used in the step 3 model before the extension, which chooses the unscheduled job with the earliest start time and then fix its start time to the earliest possible time and assign an engineer for that job first trying to schedule an engineer who is not a contractor and resides in the same location as the job. The results shown in Table 3 are very similar to those in Table 2 since we try to minimize the use of contractors. The results here do not change the previous conclusions.

5 Related Work

There is extensive literature on the workforce allocation problem [10, 17, 1], including with CP models [12, 14]. A significant part of the literature deals with the problem of assigning *crossed-trained* workers, i.e. trained in multiple skills, to jobs that require a single skill. Although not necessary when each task requires a single skill, cross-training allows the reduction of service delivery delays and increase the utilization of the workforce. A number of papers, among which [4, 5, 6, 13, 18], study the effect of cross-training of the entire workforce as a single varying parameter, but not with decision variables that describe the specific skills that each staff member must learn. This approach is most appropriate when the workforce is large and many staff members have the same skill profile.

Table 1 Performance comparison between decomposed and monolithic approaches. Steps 1,2,3 represents capacity planning, skill allocation, and job scheduling stages, respectively. MM refers to monolithic model. A † indicates a greedy optimal solution for steps 2 and 3 was discovered. The best overall cost solution found of the two approaches is in bold. A – indicates no solution was found within the time limit.

Period	Step	#Jobs	Time	Total Shortage	Total Wait	#New Skills	#Interstate Jobs	Training Cost (x100)	Travel Cost (x100)	Total Cost (x100)
Y1	1	4301	10	759						
1-month, $maxWait = 15$										
Y2M1	2,3	364	8+7	222	8	18	237	1184	†1421	
	MM	364	14400	287	8	18	237	1049	**1286**	
Y2M2	2,3	400	8+21	873	7	16	198	1262	†1460	
	MM	400	14400	723	7	15	201	1112	**1313**	
Y2M3	2,3	429	10+360	896	6	28	165	1632	**1797**	
	MM	429	14400	690	4	39	107	1883	1990	
Y2M4	2,3	393	8+10	414	5	18	140	1389	†1529	
	MM	393	14400	522	6	18	140	1127	**1267**	
Y2M5	2,3	449	10+91	816	6	29	165	2017	†2182	
	MM	449	14400	750	6	29	165	1825	**1990**	
Y2M6	2,3	380	7+360	863	3	19	84	1889	**1973**	
	MM	380	14400	595	0	31	0	2053	2053	
Y2M7	2,3	459	10+360	1353	5	31	140	2480	**2620**	
	MM	459	14400	1202	0	64	0	2878	2878	
Y2M8	2,3	370	8+15	638	6	17	171	1487	†**1658**	
	MM		14400	—	—	—	—	—	—	
Y2M9	2,3	371	7+10	703	3	15	84	1007	†1091	
	MM	371	14400	657	4	15	104	921	**1025**	
Y2M10	2,3	386	8+15	581	6	18	168	1104	†**1272**	
	MM	386	14400	574	7	18	188	1131	1319	
Y2M11	2,3	637	14+360	975	8	33	224	1999	**2223**	
	MM	637	14400	1156	0	70	0	2897	2897	
Y2M12	2,3	299	6+7	332	4	16	115	1154	†1269	
	MM	299	14400	267	4	16	115	1131	**1246**	
2-month, $maxWait = 15$										
Y2M1-2	2,3	812	14+360	2346	9	53	254	3352	**3606**	
	MM		14400	—	—	—	—	—	—	
Y2M2-3	2,3	810	21+360	1731	8	43	221	3325	**3546**	
	MM		14400	—	—	—	—	—	—	
Y2M3-4	2,3	818	19+360	1475	10	38	274	3373	**3647**	
	MM	818	14400	2429	0	101	0	4534	4534	
Y2M4-5	2,3	814	15+360	2015	8	47	218	4631	4849	
	MM	814	14400	1651	0	83	0	4815	**4815**	
Y2M5-6	2,3	829	23+360	3079	7	78	196	5471	**5667**	
	MM		14400	—	—	—	—	—	—	
Y2M6-7	2,3	814	15+360	3089	9	62	255	4646	**4901**	
	MM		14400	—	—	—	—	—	—	
Y2M7-8	2,3	741	16+360	1852	7	41	199	3228	**3427**	
	MM		14400	—	—	—	—	—	—	
Y2M8-9	2,3	741	17+360	1572	6	32	168	2030	**2198**	
	MM	741	14400	1628	0	71	0	2832	2832	
Y2M9-10	2,3	996	26+360	1755	9	56	260	3507	**3767**	
	MM		14400	—	—	—	—	—	—	
Y2M10-11	2,3	637	15+360	975	8	33	224	2010	**2234**	
	MM	637	14400	1138	0	70	0	2897	2897	
Y2M11-12	2,3	573	14+33	591	8	22	224	2162	†2386	
	MM	573	14400	559	6	27	171	2080	**2251**	

Table 2 Comparison – Effect of allowing 0,1 or 2 skills per engineer on the total cost and changing $maxWait$ on the solving time – time out per run 360s. A "—" indicates the problem is unsatisfiable.

			No New Skills				One-Skill Per Person				Two-Skill Per Person			
Period	**Step**	**#Jobs**	**Time**		**#New Skills**	**Training Cost (x100)**	**Time**		**#New Skills**	**Training Cost (x100)**	**Time**		**#New Skills (# Engs)**	**Training Cost (x100)**
Y1	1,2	4301	0		0	0	9+216		19	540	9+467		20 (16)	565
				Total Wait	**#Inter-state Jobs**	**Travel Cost (x100)**		**Total Wait**	**#Inter-state Jobs**	**Travel Cost (x100)**		**Total Wait**	**#Inter-state Jobs**	**Travel Cost (x100)**
$maxWait = 30$														
Y2M1	3	364	10	322	43	1740	5	235	13	1047	4	249	12	1038
Y2M2	3	400	21	604	37	1805	10	670	12	1173	12	616	12	1173
Y2M3	3	429	61	1071	49	2229	20	921	19	1305	23	932	19	1305
Y2M4	3	393	28	618	42	1776	9	648	13	1241	11	488	13	1241
Y2M5	3	449	80	881	55	2488	35	933	20	1826	35	833	20	1826
Y2M6	3	380	360	880	30	2146	360	803	14	1753	360	826	14	1753
Y2M7	3	459	360	1716	63	2969	53	1272	21	2015	54	1225	21	2015
Y2M8	3	370	18	826	33	2156	7	627	13	1519	8	568	13	1519
Y2M9	3	371	15	727	26	1181	8	544	10	973	12	6	10	973
Y2M10	3	386	15	634	43	1817	10	418	13	1132	9	427	13	1132
Y2M11	3	637	360	1482	68	2919	294	1295	29	1794	303	1271	26	1695
Y2M12	3	299	7	308	28	1697	3	225	12	1150	3	221	11	1141
Total				**10069**	**517**	**24923**		**8591**	**189**	**17468**		**8256**	**184**	**17376**
$maxWait = 15$														
Y2M1	3	364	19	278	43	1750	6	235	13	1047	4	249	12	1038
Y2M2	3	400	18	546	37	1805	15	577	12	1173	14	559	12	1173
Y2M3	3	429	360	731	49	2370	76	681	20	1434	89	772	20	1434
Y2M4	3	393	36	573	42	1776	10	576	13	1241	11	467	13	1241
Y2M5	3	449	75	805	55	2488	41	777	20	1826	40	735	20	1826
Y2M6	3	380	360	565	31	2226	360	761	16	1925	360	755	16	1925
Y2M7	3	459	—	—	—	—	188	1184	21	2026	360	1119	23	2171
Y2M8	3	370	—	—	—	—	14	578	13	1519	13	545	15	1539
Y2M9	3	371	32	740	26	1191	19	648	10	983	32	662	10	983
Y2M10	3	386	85	514	44	1838	9	416	14	1153	8	394	14	1153
Y2M11	3	637	360	1093	71	3115	360	944	31	1938	360	925	29	1844
Y2M12	3	299	5	225	28	1697	3	220	12	1150	4	192	12	1223
Total				**6070**	**426**	**20256**		**7597**	**195**	**17955**		**7374**	**196**	**18115**

Table 3 Comparison (contractors allowed) – Effect of allowing 0, 1 or 2 skills per engineer on the total cost and changing $maxWait$ on the solving time – time out per run 360s.

			No New Skills				One-Skill Per Person				Two-Skill Per Person			
Period	**Step**	**#Jobs**	**Time**		**#New Skills**	**Training Cost (x100)**	**Time**		**#New Skills**	**Training Cost (x100)**	**Time**		**#New Skills (# Engs)**	**Training Cost (x100)**
Y1	1,2	4301	0		0	0	7+523		31	850	7+3297		23 (20)	643
				Total Wait	**#Inter-state, Contracted Jobs**	**Total Cost (x100)**		**Total Wait**	**#Inter-state, Contracted Jobs**	**Total Cost (x100)**		**Total Wait**	**#Inter-state, Contracted Jobs**	**Total Cost (x100)**
$maxWait = 30$														
Y2M1	3	364	25	314	36,15	1651	12	177	12,12	1149	360	2059	27,16	2505
Y2M2	3	400	47	592	30,17	1696	29	560	8,14	1080	360	3072	28,20	2407
Y2M3	3	429	92	1054	40,21	2099	57	753	13,17	1194	340	2076	24,23	2973
Y2M4	3	393	44	604	36,14	1683	28	401	8,13	1154	360	2791	33,5	3841
Y2M5	3	449	129	837	47,16	2394	69	791	17,11	1762	360	3895	30,16	4010
Y2M6	3	380	360	861	25,14	2050	360	724	12,10	1675	360	4414	27,24	3667
Y2M7	3	459	360	1635	51,23	2813	152	1518	13,20	1897	202	1629	8,38	2132
Y2M8	3	370	38	754	24,23	2018	24	518	5,22	1387	131	1526	13,23	1814
Y2M9	3	371	30	732	18,17	1079	23	511	3,16	877	360	2669	35,10	2933
Y2M10	3	386	40	623	35,12	1750	17	385	10,6	1096	199	1321	18,15	1595
Y2M11	3	637	360	1487	63,12	2870	200	1117	20,15	1562	30	443	11,13	1916
Y2M12	3	299	14	311	21,8	1650	8	196	8,5	1120	30	443	11,13	1916
Total				**9804**	**426,192**	**23753**		**7651**	**129,171**	**15953**		**26338**	**265,216**	**31709**
$maxWait = 15$														
Y2M1	3	364	51	258	36,15	1661	14	177	12,12	1149	8	215	8,12	966
Y2M2	3	400	37	534	30,17	1696	29	523	8,14	1080	23	589	8,14	1080
Y2M3	3	429	360	692	45,13	2341	360	613	14,16	1276	360	722	15,15	1347
Y2M4	3	393	48	579	36,14	1683	39	392	8,13	1154	26	558	8,13	1154
Y2M5	3	449	98	787	47,16	2394	70	714	17,11	1762	67	807	17,11	1762
Y2M6	3	380	360	609	29,7	2179	360	765	15,5	1795	360	718	16,5	1897
Y2M7	3	459	360	1068	53,18	2946	262	1306	13,20	1908	360	1222	21,18	2135
Y2M8	3	370	56	535	23,24	2106	31	506	5,22	1387	76	514	6,22	1469
Y2M9	3	371	44	721	18,18	1084	47	638	3,17	882	27	670	3,17	882
Y2M10	3	386	287	502	36,12	1771	15	320	11,6	1117	13	361	11,6	1117
Y2M11	3	637	360	1140	66,10	3100	360	1028	26,6	1832	360	1042	22,11	1848
Y2M12	3	299	11	237	21,8	1650	7	196	8	1120	6,5	218	7,5	1111
Total				**7662**	**440,172**	**24611**		**7178**	**140,150**	**16462**		**7636**	**142,149**	**16768**

We focus on papers that propose methods to decide how to both train, allocate jobs and schedule them. De Bruecker et al. [8] specifically review the workforce planning literature that takes skills into account, and in particular (in Section 3.3.3) the papers that allow the workforce to be trained. One important dichotomy is whether the skills are *hierarchical* or *categorical*. With hierarchical skills, there are only skill levels, where a worker at a given skill level can perform all jobs at this or a lower skill level. With categorical skills, there is no comparison between skills.

Huang et al. [11] optimise the service level delivered by a consultancy-type business where projects require certain skills. Since the problem is solved via a discrete event simulator, this allows them to model many different aspects of the problem, such as employees deciding to leave. Within the simulation, the decision to assign staff to projects is made by a Linear Program, hence fractional (simultaneous) assignments are possible. The future horizon is divided into planning periods, during which staff can be trained to ensure that enough capacity of each skill is available for the projects in that period. In this model, each staff can only possess one skill, and training that staff is a "transfer", i.e. the staff loses the previous skill. Other papers consider that staff can be re-trained, losing their original skills, or transferred between departments [16].

De Bruecker et al. [7] propose a three-step approach to training and scheduling aircraft maintenance teams from one season to the next. Each stage solves a Mixed-Integer Program and feeds into the next step. The first stage consists in scheduling maintenance shifts to ensure the flights can operate on schedule, and assigning workers to the shifts. The second stage refines the set of skills needed by each team of workers in order to reduce the amount of training needed. The third stage attempts to schedule the training needed in the training season.

6 Conclusion

We have demonstrated the potential value of additional training in reducing overall costs for our service delivery problem. Together with the substantial cost reduction, our solutions also provides auxiliary benefits, such as a more highly trained workforce, and considerably less travel, thereby improving staff wellbeing and reducing the company's carbon footprint.

Furthermore, we have shown that using a single monolithic model to solve the entire problem was not practical, as the solutions it can find in a time similar to our decomposition approach are much worse.

One limitation is our current inability to solve the third subproblem, which schedules jobs and evaluate the quality of our training decisions, for a one-year horizon. We believe that the cost we obtain for the month-by-month approach are a reasonable approximation of the cost that would be obtained for the entire year, but we have not been able to experimentally verify this hypothesis.

Another potential limitation of the current approach is that our third subproblem is set as an offline job scheduling problem. While this integrates well with the first two subproblems, offline scheduling might give a biased measure of the efficiency with which the workforce can deal with jobs that in practice would need to be allocated on the fly. Hence future work includes solving the job scheduling subproblem online, which, since recently, can be modelled in Minizinc using the techniques presented in [9].

References

1 Mark Antunes, Vincent Armant, Kenneth N. Brown, Daniel A. Desmond, Guillaume Escamocher, Anne-Marie George, Diarmuid Grimes, Mike O'Keeffe, Yiqing Lin, Barry O'Sullivan, Cemalettin Ozturk, Luis Quesada, Mohamed Siala, Helmut Simonis, and Nic Wilson. Assigning and scheduling service visits in a mixed urban/rural setting. *International Journal on Artificial Intelligence Tools*, 29(03n04):2060007:1–2060007:31, 2020. doi:10.1142/S0218213020600076.

2 Gleb Belov, Peter J. Stuckey, Guido Tack, and Mark Wallace. Improved linearization of constraint programming models. In Michel Rueher, editor, *Principles and Practice of Constraint Programming*, Lecture Notes in Computer Science, pages 49–65. Springer, 2016. International Conference on Principles and Practice of Constraint Programming 2016, CP 2016 ; Conference date: 05-09-2016 Through 09-09-2016. doi:10.1007/978-3-319-44953-1_4.

3 John R. Birge and François Louveaux. *Introduction to Stochastic programming (2nd edition)*. Springer, New York, NY, 2011.

4 Michael J. Brusco. An exact algorithm for a workforce allocation problem with application to an analysis of cross-training policies. *IIE Transactions*, 40(5):495–508, 2008. doi:10.1080/07408170701598124.

5 G M Campbell. A two-stage stochastic program for scheduling and allocating cross-trained workers. *Journal of the Operational Research Society*, 62(6):1038–1047, 2011. doi:10.1057/jors.2010.16.

6 Gerard M. Campbell. Cross-utilization of workers whose capabilities differ. *Management Science*, 45(5):722–732, 1999. doi:10.1287/mnsc.45.5.722.

7 Philippe De Bruecker, Jeroen Beliën, Jorne Van den Bergh, and Erik Demeulemeester. A three-stage mixed integer programming approach for optimizing the skill mix and training schedules for aircraft maintenance. *European Journal of Operational Research*, 267(2):439–452, 2018. doi:10.1016/j.ejor.2017.11.047.

8 Philippe De Bruecker, Jorne Van den Bergh, Jeroen Beliën, and Erik Demeulemeester. Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243(1):1–16, 2015. doi:10.1016/j.ejor.2014.10.038.

9 Alexander Ek, Maria Garcia de la Banda, Andreas Schutt, Peter J. Stuckey, and Guido Tack. Modelling and solving online optimisation problems. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1477–1485, April 2020. doi:10.1609/aaai.v34i02.5506.

10 A.T Ernst, H Jiang, M Krishnamoorthy, and D Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, 2004. Timetabling and Rostering. doi:10.1016/S0377-2217(03)00095-X.

11 Huei-Chuen Huang, Loo-Hay Lee, Haiqing Song, and Brian Thomas Eck. Simman—a simulation model for workforce capacity planning. *Computers & Operations Research*, 36(8):2490–2497, 2009. Constraint Programming. doi:10.1016/j.cor.2008.10.003.

12 Serdar Kadioglu, Mike Colena, Steven Huberman, and Claire Bagley. Optimizing the cloud service experience using constraint programming. In Gilles Pesant, editor, *Principles and Practice of Constraint Programming*, pages 627–637, Cham, 2015. Springer International Publishing.

13 Haitao Li and Keith Womer. Scheduling projects with multi-skilled personnel by a hybrid milp/cp benders decomposition algorithm. *J. Scheduling*, 12:281–298, June 2009. doi:10.1007/s10951-008-0079-3.

14 Y. Naveh, Y. Richter, Y. Altshuler, D. L. Gresh, and D. P. Connors. Workforce optimization: Identification and assignment of professional workers using constraint programming. *IBM Journal of Research and Development*, 51(3.4):263–279, 2007. doi:10.1147/rd.513.0263.

15 N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack. Minizinc: Towards a standard CP modelling language. In C. Bessiere, editor, *Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming*, volume 4741 of *LNCS*, pages 529–543. Springer-Verlag, 2007.

16 Haiqing Song and Huei-Chuen Huang. A successive convex approximation method for multistage workforce capacity planning problem with turnover. *European Journal of Operational Research*, 188(1):29–48, 2008. doi:10.1016/j.ejor.2007.04.018.

17 Jorne Van den Bergh, Jeroen Beliën, Philippe De Bruecker, Erik Demeulemeester, and Liesje De Boeck. Personnel scheduling: A literature review. *European Journal of Operational Research*, 226(3):367–385, 2013. doi:10.1016/j.ejor.2012.11.029.

18 Kum-Khiong Yang, Scott Webster, and Robert A. Ruben. An evaluation of worker cross training and flexible workdays in job shops. *IIE Transactions*, 39(7):735–746, 2007. doi:10.1080/07408170701244687.