TBD manuscript No.
(will be inserted by the editor)

Two-stage stochastic days-off scheduling of multi-skilled analysts with training options*

Douglas S. Altner · Erica K. Mason · Leslie D. Servi

Received: November 2017 / Revised: October 2018 / Accepted: date

Abstract Motivated by a cybersecurity application, this paper studies a two-stage, stochastic days-off scheduling problem with 1) many types of jobs that require specialized training, 2) many multi-skilled analysts, 3) the ability to shape analyst skill sets through training decisions, and 4) a large number of possible future demand scenarios. We provide a integer linear program for this problem and show it can be solved with a direct feed into Gurobi with as many as 50 employees, 6 job types, and 50 demand scenarios per day without any decomposition techniques. In addition, we develop a matheuristic—that is, an integer-programming-based local search heuristic—for instances that are too large for a straightforward feed into a commercial solver. Computational results show our matheuristic can, on average, produce solutions within 4-7% of an upper bound of the optimal objective value.

Keywords days-off scheduling · multi-skilled workforce · stochastic integer programming · matheuristics · cybersecurity operations · training

1 Introduction

Scheduling analysts at a cybersecurity operations center (CSOC) raises a number of interesting logistical challenges. First, future demand is unknown and

* **Approved for Public Release; Distribution Unlimited. Case Number 17-1132.**

Douglas S. Altner, corresponding author · Leslie D. Servi
Operations Research Department
The MITRE Corporation
McLean, Virginia 20111
{ daltner, lservi } @mitre.org

Erica K. Mason
ThirdLove
San Francisco, CA 94107
emason@thirdlove.com

highly variable. Workforce planners must decide how to schedule professionals without knowing the future volume and mix of jobs that analysts will need to complete. Second, depending on the setting, there could be a mix of jobs that require specialized knowledge and training to perform well. For example, analysts could be monitoring several different types of systems for signs of potential unauthorized access or malicious activity, and quickly discerning whether these signs are actual threats requires specialized knowledge of the system and how it is used. Third, any analyst could in principle be trained to monitor alerts from a particular system; but analysts cannot effectively protect a system without specialized knowledge of that system.

These interwoven challenges motivate the study of workforce optimization under uncertainty in a many-job-type, multi-skilled analyst context. Scheduling too few analysts, or not scheduling analysts with the right mix of skill sets, could be a security risk. At the same time, scheduling too many analysts could be inefficient and costly. Likewise, if jobs require specialized training but analysts could be trained to perform any job, how should the workforce be shaped to create a more desirable baseline capability?

To help support these types of decisions, we study a stochastic days-off scheduling problem in an environment in which 1) there is a wide range of possible future demand scenarios, 2) there are several distinct job types, 3) employees are *multi-skilled*—meaning they are capable of performing more than one type of job depending on their training, and 4) planners can train individual employees to increase the set of jobs they can complete. The structure of our problem is that a planner already has an existing multi-skilled workforce, knows that a wide range of future demand scenarios are possible, and wants to decide 1) a future, typical two-week work schedule for that workforce and 2) which employees should be trained in which skills to improve the CSOC's overall ability to respond to alerts. The objective of our model is to maximize the expected number of alerts that can be analyzed during the days at which they arrive. These objective must be met while adhering to a number of scheduling constraints and budget limitations on training.

This type of problem naturally lends itself to two-stage stochastic integer programming. One potential difficulty with such models is that they can quickly become intractable when there are many decisions and a wide range of future scenarios to hedge against. For this reason, we focus on demonstrating the *computational tractability* of such a model—which, in this context we mean the ability to quickly construct near optimal solutions on a standard work station—for this type of project.

Our experimental results show that even instances with 6 alert (i.e., job) types, 50 distinct employees, and 50 demand scenarios per day can be solved by a direct feed of the model into Gurobi without any decomposition techniques. For larger instances, we develop a *matheuristic*—that is, an integer-programming-

based local improvement heuristic—so we can solve cases that are too large for a straight forward feed into a commercial integer programming solver either because the system would run out of memory or because it would just take too long to obtain a near optimal solution. Computational results show our matheuristic can, on average, produce solutions within 4-7% of an upper bound of the optimal objective value for larger sized test cases. This includes instances with as many as 20 distinct alert types or as many as 1500 demand scenarios per day.

The main contributions of this work are as follows. First, we offer a two-stage stochastic integer programming model for a days-off scheduling problem with a unique combination of challenges: wide range of possible demand scenarios, many job types, many multi-skilled analysts, and the ability to increase individual employee skill sets with training; and we show this model can be solved without any decomposition techniques for reasonably sized instances. Second, we show how solutions can be computed for larger-scale instances using local improvement techniques that are simple to understand and simple to implement.

Our paper is organized as follows. Section 2 presents a literature review of related work and discusses the contributions of this paper vis-a-vis the contributions of previous investigations. Section 3 presents a detailed verbal description of the problem. Section 4 presents our solution approach. This includes our integer programming formulation and our local improvement solution for large-scale instances. Section 5 details our computational study, showing how these methods produce solutions on small and large instances. Section 6 draws conclusions and offers directions for future work.

2 Literature Review

Workforce optimization is an immensely studied topic and particular workforce optimization problems can take many forms. They can include *staffing optimization*—determining how many employees of a particular type should be employed; *shift scheduling*—deciding specific shifts for employees; *days-off scheduling*—deciding which days specific employees or employee types work and which days they do not work; *rostering*—assigning individual employees to individual schedules; and *crew assignment*—assigning entire crews to pre-determined work schedules, routes, or tours of duty. Complex workforce scheduling problems have been studied for a variety of settings including hospitals [31], airlines [6], railroads [28], convenience stores [2], fashion retailers [25], and post offices [4]. Van Den Bergh et al. [35] provide a recent survey that reviews nearly 300 articles related to personnel scheduling alone.

The literature on stochastic programming approaches to workforce scheduling in particular is growing. To list a few, Bard et al. [5] study a two-stage stochastic shift scheduling problem in which the number of post office workers to have on payroll is determined in the first stage and particular shift schedules are decided in the second stage. Zhu and Sherali [37] and Bodur and Luedtke [8] research workforce optimization models that schedule employees in the first stage and assign specific workloads to specific employees in the second stage. Kim and Mehrotra [21] develop a model for creating an initial schedule in the first stage and then revising it in the second stage, once more information is known about incoming demand. Parisio and Jones [25] propose a stochastic programming model for scheduling employees for a Swiss retail outlet where the first stage is to create a schedule and the second stage is to make overtime decisions. Restrepo et al. [26] decide days-off and shifts in the first stage, and decide break times and job assignments in the second stage, assuming all employees have equal skill sets.

For more complex models of demand uncertainty, researchers have crafted solutions that combined elements from stochastic programming and simulation. For example, Robbins and Harrison 2010 [27] present a stochastic programming model for call center staffing and scheduling in light of arrival rate uncertainty whereby they estimate the service level using a convex linear approximation of output from a queuing model. Likewise, Gurvich et al. 2010 [16] study a multi-skilled, call center staffing problem with chance constraints, and solve it by first developing a metamodel that approximates optimal staffing levels with known arrival rates, and then solves the metamodel for a finite number of possible arrival rates.

There have also been a few papers on workforce scheduling optimization and related challenges for cybersecurity operations centers. These include deterministic optimization approaches to staffing and shift scheduling [12], stochastic programming approaches to staffing and shift scheduling [1], determining an optimal policy for exercising on call options on a day-to-day basis [13], assigning sensors to analysts [30], and dynamically reallocating sensors to analysts [29].

Numerous researchers have studied the challenges, benefits, and trade-offs associated with scheduling a multi-skilled workforce (e.g., [3], [7], [10], [15], [16], [31]). Several others have researched the challenges of deciding how to optimally multi-skill or cross-train a workforce—meaning, increasing the number of job types that specific employees can perform via training decisions or job routing policies—while also considering scheduling constraints and challenges (e.g., [14], [17], [20], [32]). In the intersection of multi-skilling and optimization under uncertainty, Bodur and Luedtke study a two-stage multi-skilled employee scheduling problem with test cases with up to six job types and six employee types, each of which allow employees to complete two different types of jobs. Henao et al. [18] take on the challenges of simultaneously optimizing

training decisions with hours assignment decisions in a robust optimization context—meaning they assume future demand is unknown, does not follow a known distribution, but is confined to known intervals.

Our work builds upon this great body of literature by tackling a unique combination of challenges: solving a two-stage stochastic days-off scheduling model with many job types, multi-skilled employees, and the ability to increase the job types that any individual employee can complete through first-stage training decisions. To provide a point of comparison, Bodur and Luedtke 2016 [8] who push the envelope with stochastic multi-skilled workforce optimization, consider instances with up to six job types and six employee types. We consider instances with over twelve job types and one class of instances with twenty job types, and fifty to one hundred employees.

3 Problem Statement

We offer a model for determining scheduling and training decisions at a cybersecurity operations center (CSOC) with many distinct alert types (i.e., job types), many multi-skilled analysts, the ability to increase employee skills with training, and a large number of possible future demand scenarios. Although our problem statement and model is expressed in cybersecurity terms, the ideas in this paper can be extended to any multi-skilled workforce setting where training and scheduling decisions must be made under uncertainty.

The purpose of the model is to help a planner with a given multi-skilled workforce—in this case, a set of individual employees each of whom is trained to complete one or more job types—decide 1) which days each employee should work during a future pay period, which we assume is two weeks or fourteen days; and 2) whether any employees should undergo training to increase the set of job types they can complete. These decisions must be made for future pay periods before the specific volume and mix of security alerts for each day of the coming pay period is known. The objective of our model is to maximize the expected number of covered alerts—that is, alerts that cannot be completed during the day they arrive—for a typical pay period—that is, a pay period subject to the given set of possible future demand scenarios.

There are three broad categories of constraints: 1.) limits on employee abilities to analyze alerts, 2.) scheduling constraints, and 3.) training-related constraints.

For the employee capacity constraints, we assume each analyst can work up to 8 hours during the days they are scheduled to work, and that they can only work on alert types they are trained for. At present, we do not carry over uncovered alerts to subsequent days (in part because these could be addressed

with overtime) and we leave addressing this issue as a topic for future research. For the scheduling constraints, we require that each full-time analyst works five 8-hour days per week, so ten such days per pay period. We also require that each full-time analysts works at most five days in a row and at most one weekend per pay period. For the training-related constraints, we assume there may be constraints on the maximum number of skills that an analyst can possess, and on the amount of new training that an analyst can receive. We also include a budget constraint for training.

As with any modeling effort of anything complicated, our model makes a number of assumptions about its target applications. First, we assume a bijective mapping between *skills* and *alert types*. Each alert type requires employees to possess a specific skill. Each skill enables employees to analyze alerts of exactly one type.

Second, our model assumes future demand can be modeled with a probability mass function specifying a finite number of demand scenarios for each work day. For this study, we assume the volume of alerts of any type on one day is independent of the volume of alerts of any type on another day. In a cybersecurity context, this can reflect the volatility and unpredictability of day-to-day workloads. We also assume the volume of alerts of one type on any one day is independent of the volume of alerts other types on that day, but our model can be easily adapted to assume correlations between within-day volumes.

Third, our model assumes employees possess the same productivity rates for analyzing alerts regardless of whether they are assigned only alerts of one type, or assigned several different types of alerts. We leave factoring in a potential loss of productivity as a topic for future research. Fourth, our model assumes all alerts arrive at the beginning of the day, and does not factor in the dynamics of assigning alerts to analysts while other alerts are still coming in. The tactical issues of dynamically assigning alerts to analysts and/or to dynamically assign/re-assign analysts to monitoring stations is beyond the scope of this study, and has been explored in [30]. Fifth, our model assumes any alerts that are not analyzed that day are covered by exogenous staffing resources (e.g., overtime, on-call analysts) and are not carried over to the next day. How to scheduling analysts in light of uncertainty and on-call staffing options is a topic that has been addressed in a previous paper [1], although only in a context with a single alert type.

4 Solution Approach

This section details our solution approach. First, we present our integer programming formulation of the problem. Then we discuss measures we took to break symmetry within the solution space. Then we present our matheuristic

framework for solving large-scale instances.

4.1 Integer Programming Formulation

In this paper, we use standard stochastic programming terminology following, for example, Kall and Wallace [19]. A *demand scenario* for a given day is a vector specifying the number of alerts of each type that arrive at the beginning of the day. The variables corresponding to decisions that must be made before the particular demand scenarios are realized are *first-stage decision variables*. The variables corresponding to decisions made afterwards are *second-stage decision variables*.

To emphasize a point, the second stage decision variables only exist as a means of calculating the expected number of covered alerts for a particular scheduling/training plan. These variables (and the structure of our problem) allows this expectation to be calculated without having to perform a Monte Carlo simulation. They do not actually represent decisions that would be recommended to the system user.

Sets:

- $D := \{Mo1, Tu1, \ldots, Sa2, Su2\}$, the set of days of the two week pay period.
- $K_d :=$ the set of possible demand scenarios for day d. Each demand scenario is a vector of workloads for each alert type.
- $E :=$ the set of employees on staff.
- $T :=$ the set of possible alert types.
- $T(e) :=$ the set of alert types that employee e can analyze.
- $D(i) :=$ the set of days during week i of the pay period.
- $D5(d) :=$ the set of the next five consecutive days in the pay period after day d, wrapping around. For example, $D5(Th2) = \{Fr2, Sa2, Su2, Mo1, Tu1\}$.

Data:

- $alerts_d^{t,k}$, the number of incoming alerts of type t that occur on scenario k for day d.
- $prob_d^k$, the probability of scenario k for day d.
- $rate_{e,t}$, the number of type t alerts that employee e can analyze during one hour.
- $cost_t$, the cost of training an employee to analyze alerts of type t
- $budget$, the budget available for training employees
- $maxnewskills$, the maximum number of new skills that an employee can be trained in

Decision Variables:

1. $x_{e,t} := 1$ if employee e is trained to analyze additional alert type t (for $t \notin T(e)$) and 0 otherwise

2. $y_d^e := 1$ if employee e works on day d and 0 otherwise
3. $w^e := 1$ if employee e does not work on the weekend $Sa1$, $Su1$, and 0 if e does not work on the weekend $Sa2$, $Su2$
4. $h_{e,t,d}^k :=$ the number of hours employee e spends on type t alerts under scenario k on day d

Problem Formulation:
Maximize

$$\sum_{d,k} prob_d^k \sum_{e,t} rate_{e,t}\ h_{e,t,d}^k \tag{1}$$

Subject to:

$$\sum_{e\in E} rate_{e,t}\ h_{e,t,d}^k \le alerts_d^{t,k} \qquad \forall\ k \in K_d,\ \forall\ t \in T,\ \forall\ d \in D \tag{2}$$

$$\sum_{t\in T} h_{e,t,d}^k \le 8 y_d^e \qquad \forall\ e \in E,\ \forall\ k \in K_d,\ \forall\ d \in D \tag{3}$$

$$\sum_{k\in K_d, d\in D} h_{e,t,d}^k \le 80 \left(\sum_{d\in D} |K_d|\right) x_{e,t} \qquad \forall\ t \notin T(e),\ \forall\ e \in E \tag{4}$$

$$\sum_{e\in E, t\notin T(e)} cost_t\ x_{e,t} \le budget \tag{5}$$

$$\sum_{t\notin T(e)} x_{e,t} \le maxnewskills \qquad \forall\ e \in E \tag{6}$$

$$\sum_{d\in D(i)} y_d^e = 5 \qquad \forall\ e \in E,\ \forall\ i \in \{1,2\} \tag{7}$$

$$y_d^e + \sum_{\bar{d}\in D5(d)} y_{\bar{d}}^e \le 5 \qquad \forall\ e \in E,\ \forall\ d \in D \tag{8}$$

$$y_d^e \le 1 - w^e \qquad \forall\ e \in E,\ \forall\ d \in \{Sa1, Su1\} \tag{9}$$

$$y_d^e \le w^e \qquad \forall\ e \in E,\ \forall\ d \in \{Sa2, Su2\} \tag{10}$$

$$x_{e,t} \in \{0,1\} \qquad \forall\ e \in E,\ \forall\ t \notin T(e)$$

$$y_d^e \in \{0,1\} \qquad \forall\ e \in E,\ \forall\ d \in D$$

$$w^e \in \{0,1\} \qquad \forall\ e \in E$$

$$h_{e,t,d}^k \ge 0 \qquad \forall\ t \in T,\ \forall\ e \in E,\ \forall\ d \in D,\ \forall\ k \in K_d$$

The objective function maximizes the expected number of alerts that employees can analyze. This expectation has been discretized to the sum of the alerts that employees can analyze under each scenario multiplied by the probability of that scenario.

Constraints (2)-(4) concern the allocation of employee hours. The set (2) limits the number of alerts employees can analyze by the actual number of alerts

that arrive under that scenario, for each day and alert type. The set (3) enforces that each employee only has at most 8 hours to spend on the different alert types, for each day and scenario. Constraint set (4) is a set of if-then constraints that enforces that employees cannot be assigned alerts of type t for any alert type outside of their initial skill set $T(e)$ unless if the employee has been trained to handle alerts of type t. The right-hand side coefficient is a sufficiently large constant $\left(80 \sum_{d \in D} |K_d|\right)$ to ensure this constraint will not be restrictive for alert type t if employee e is indeed trained to handle alerts of type t. The constraints in (4) are, without loss of fidelity, aggregated across all demand scenarios and days to reduce the number of rows in the integer program. This allows us to enforce the same if-then condition but to solve the problem with less computation time.

Constraints (5) and (6) refer to training. Set (5) is the budget constraint on additional training. Set (6) enforces that each employee can be trained in a finite number of additional skills.

The next few constraint sets (7)-(10) implement scheduling constraints. Set (7) requires that each employee works exactly five days per week. Set (8) guarantees each employee works at most five consecutive days. These constraints specify that for every employee e, for every six consecutive days, that employee e can work during at most five of those six consecutive days. Sets (9) and (10) ensure each employee gets at least one weekend off.

The final constraint sets declare the variables. All variables are binary except for the $h^k_{e,t,d}$ variables, which we allow to be fractional.

4.2 Symmetry Breaking

This section describes a class of symmetry breaking cuts that we derived and included in our model. These cuts modestly improved our model's execution time.

Two feasible solutions are *symmetric* if there exists a permutation on the decision variables to transform one solution into the other without impacting the feasibility or objective value. Symmetry presents a computational challenge because symmetric solutions may be indistinguishable from the user's perspective, but an integer programming solver may see them as a combinatorial number of solutions that must be cut away during the branch-and-bound process. E.g., see [23] and [24].

Our scheduling problem at hand suffers from symmetry to the extent the input contains *symmetric employees*—meaning employees with the same training and productivity rates. Since our model assumes employees can have one of

three productivity rates, our model contains a number of symmetric employees. However, this would not be an issue if each employee is assumed to have a unique productivity rate.

To decrease the solving time, we included what we hereby refer to as *basic symmetry breaking constraints* to our model. Given a set of symmetric employees $E_s = \{e_1, e_2, \ldots, e_k\}$ with $k \geq 2$, an arbitrary ordering and an arbitrary day $d \in D$, we express the basic symmetry breaking constraints as follows:

$$y_d^{e_i} \geq \sum_{\bar{i}=i+1}^{k} y_d^{e_{\bar{i}}} \qquad \forall\, i \in \{1, 2, \ldots, k-1\} \tag{11}$$

Thus, for example, if we have a set of four symmetric employees: e_1, e_2, e_3, e_4 and we arbitrarily pick $Mo1$ as the day, the basic symmetry breaking constraints for this employee set are the following:

$$\begin{array}{l} y_{Mo1}^{e_1} \geq y_{Mo1}^{e_2} + y_{Mo1}^{e_3} + y_{Mo1}^{e_4} \\ y_{Mo1}^{e_2} \geq y_{Mo1}^{e_3} + y_{Mo1}^{e_4} \\ y_{Mo1}^{e_3} \geq y_{Mo1}^{e_4} \end{array} \tag{12}$$

These constraints indicate that if one of these employees is to be chosen to work on Monday of week 1, then it should be e_1. If two employees are to work on that day, then they should be e_1 and e_2. If three employees, then e_1, e_2, and e_3.

We also derived and tested more elaborate *conditional symmetry breaking constraints* that broke the symmetry for cases when symmetric employees are either all scheduled to work or all scheduled to not work on the day selected for their basic symmetry breaking constraint. However, these constraints tended to slow rather than accelerate the runtime.

4.3 Local Search Framework for Large Instances

Once the problem instances get sufficiently large, this problem can no longer be solved by a straightforward feed of the integer programming model into a commercial solver like Gurobi. For example, Gurobi 6.0 cannot find a solution within 40% of the best upper bound for problem instances with 50 employees, 6 alert types and 500 demand scenarios per day, even if we allow it to run overnight on an 8-processor virtual machine. For instances with 50 employees, 6 alert types, and 1500 demand scenarios per day, Gurobi struggles to produce an integer feasible solution even if it runs overnight on the same virtual machine.

This section presents a *matheuristic*—that is, an integer-programming-based local search heuristic—for solving large-scale problem instances[1]. Our matheuristic iteratively solves smaller integer programs induced by freeing the decision variables for a small number of employees, and fixing all other decision variables to the values they assume in the current best known solution. This heuristic is simple to understand and simple to implement, and is also arguably easier to implement than more exact decomposition methods such as the L-shaped method [36]. We also note that other authors have employed similar local search approaches to deterministic workforce scheduling problems, including Della Croce and Salassa [11] and Smet et al. [33] and for a stochastic workforce planning in a production context [34].

Algorithm 1 Local search method for large-scale instances

procedure LocalSearchHeuristic()
1. Construct an initial solution: (x_0, y_0, w_0, h_0)
2. $i \leftarrow 1$
repeat
a. Arbitrarily partition the employees into k mutually exclusive groups: $G_1, G_2, \ldots, G_k$
b. $(x_{i.0}, y_{i.0}, w_{i.0}, h_{i.0}) \leftarrow (x_{i-1}, y_{i-1}, w_{i-1}, h_{i-1})$
for j in $\{1, 2, \ldots, k\}$ **do**
I. Unfix the h variables for all employees
II. Unfix the x, y, and w variables for each employee in G_j
III. Re-solve the MIP, keeping all other x, y, and w variables fixed
Let $(x_{i.j}, y_{i.j}, w_{i.j}, h_{i.j})$ denote the optimal solution from II
if $objval(x_{i.j}, y_{i.j}, w_{i.j}, h_{i.j}) < objval(x_{i.j-1}, y_{i.j-1}, w_{i.j-1}, h_{i.j-1})$ **then**
IV.1 Refix the x, y, w variables for employees in G_j to the values in $(x_{i.j}, y_{i.j}, w_{i.j})$
else
IV.2 Refix the x, y, w variables to the values in $(x_{i.j-1}, y_{i.j-1}, w_{i.j-1})$
end if
end for
c. $(x_i, y_i, w_i, h_i) \leftarrow (x_{i.k}, y_{i.k}, w_{i.k}, h_{i.k})$
d. $i \leftarrow i + 1$
until $objval(x_{i+1}, y_{i+1}, w_{i+1}, h_{i+1}) \geq objval(x_i, y_i, w_i, h_i)$
end procedure

Algorithm 1 contains pseudocode for our heuristic method. At a high level, this method consists of iteratively fixing all x, y and w variables in the mixed integer program for all but a small subset of employees, and then solving the induced mixed integer program on the unfixed variables. The induced mixed integer program amounts to optimally revising the training and day scheduling decisions for the subset of employees, while keeping those for all other employees fixed, and simultaneously determining the optimal assignment of jobs to all employees under each scenario.

[1] *Matheuristic* is a portmanteau combining *mathematical programming* and *heuristics* that is becoming an increasingly popular term for this technique [9].

To construct an initial solution for step 1 of our algorithm, we first solved the program with an alternate objective function, shown in (13), to find feasible values for the w, x, and y variables that were not all zero. The first term in (13) maximizes the training for employees with high productivity rates in alert types with high demand. In this term, $\hat{\mu}_t$ is the expected workload of alerts of type t given the demand scenarios. The second term ensures each weekend is assigned staff in the initial solution. To get initial values for the h variables, the binary variables were fixed to the values obtained in the previous step, and the problem was re-solved as a LP with the original objective function (1).

$$Maximize \ \sum_{e \in E, t \notin T(e)} \hat{\mu}_t \ rate_{e,t} \ x_{e,t} + \sum_{e \in E, d \in Sa1, Sa2} \ y_d^e \tag{13}$$

In step 2.a of our algorithm, we randomly partitioned the employees into subsets of size 5 each, which generally found better results compared to other partitioning strategies. Future research can explore the potential benefits and run time costs of defining neighborhoods by strategically choosing subsets of employees based on how they interact with respect to their current schedules and training vectors.

5 Computational Experimentation

In this section, we describe our computational results. All experiments were implemented on a virtual machine with 8 2.60 Ghz Intel processors with 8 GB of RAM. The methods were coded using Python 2.7 and Gurobi 6.0.4 was used to solve all mathematical programs. We changed Gurobi's settings to solve the root relaxation with primal simplex, as this generally yielded the fastest results. All other parameters were kept at their default values unless otherwise stated.

The first subsection discusses how we generated our test instances. The second subsection presents the results on smaller test cases, showing near optimal results were found in one minute to four hours. The third subsection describes how our local search method consistently produces solutions within 9% of the best bound on instances that are generally too large for a direct feed into a commercial IP solver.

5.1 Generating Test Instances

We randomly generated 54 test cases for these experiments. This consists of 6 *classes* of small instances, 5 classes of large-A instances, and 5 classes of large-B instances. For each class of small instances, we generated 4 test cases, and

Size	Class	Emp.	Alert Types	Demand Scenarios	Bin. Vars.	Max Init. Skills
small 4 per class	**a**	35	6	50	665	3
	b	50	6	35	950	3
	c	35	10	35	805	3
	d	50	6	50	950	3
	e	35	6	35	665	3
	f	35	6	35	700	1
large-A 3 per class	**g**	50	6	500	950	3
	h	50	6	1500	950	3
	i	50	10	500	1150	3
	j	85	6	500	1615	3
	k	100	20	50	3300	3
large-B 3 per class	**l**	50	12	50	1275	2
	m	50	12	85	1250	3
	n	85	12	50	2125	3
	o	85	12	50	2167	2
	p	50	12	85	1275	2

Table 1 Parameters for test cases

for each class of large instances we generated 3 test cases. The small instances are used to test a straightforward feed of our integer program into Gurobi, and the large instances are used to test our matheuristic. As is discussed later, the large-B instances differ from the large-A instances primarily in that they allow for substantially more training decisions.

Each class of instances specifies different values to a few key dimensions to the test cases: the number of employees, the number of demand scenarios per day, and the number of alert types. How these values vary with the class of instances is summarized in Table 1. Within a class, the instances differ by the composition of the employee productivity levels and initial skill sets as well as the volume and mix of alert types specified by each demand scenario for each day.

The first four columns of Table 1 specify the instance class identifier, the number of employees in that class, the number of alert types for that class, and the number of demand scenarios per day for that class. The **Bin. Vars.** column specifies the approximate number of binary variables for each instance in this class, which is a rough and imperfect indicator of the difficulty of the test case. This is an approximate number because this can vary with the number of skills that employees start with as we do not create variables to teach employees skills they already have.

The **Max Init. Skills** column specifies the maximum number of initial skills each employee could possesses—which is equivalent to the number of different types of alerts the employee can analyze. If an instance contains a maximum initial skills value of, say, 3 then this implies employees can start with 1, 2, or 3 skills. The number of skills that each employee starts with is selected

uniformly at random, and the initial skills the employee starts with are also selected uniformly at random (without replacement).

To generate demand scenarios for each test case, we modeled the volume of each type of alert for each day as a log-normal random variable. The mean of each log-normal distribution for each alert type for each day ranges from 100 to 400, and the standard deviation ranges from 50 to 200. Employees are individually assigned a random productivity rates of either 32, 48, or 56 alerts per 8-hour day. This productivity rate applies to all alert types that the employee is trained for. We selected these numbers with the general goal of getting the aggregate number of average alerts to roughly equal to the aggregate number of alerts that employees could analyze, so the test instances pose the challenge choosing the right skills and the right mix of analysts for the right days.

The training budget for all test cases is fixed at 12000 units. The cost of training employees in each skill ranges from 600 to 1200 units, meaning the budget affords the planner to train 10 to 20 employees in one new skill. The only exception is for class **p** of the large-B instances, in which we made the budget 20000 units to allow for more training opportunities.

In small and large-A instances, the additional training constraints dictate that employees can be trained in at most one additional skill and that no employee can be trained in more than 3 skills. For large-B instances, these constraints are dropped and employees can be trained in an unlimited number of skills.

5.2 Demonstrating the Viability of the Straightforward Integer Programming Approach

Although many stochastic integer programs can be reformulated into deterministic integer programs by discretizing the expected value function, this often leads to formulations that are too large for commercial solvers to solve in a reasonable amount of time. In this section, we demonstrate how our integer programming formulation can be used to solve reasonably sized instances with a straightforward feed into Gurobi.

This test was conducted on the *small* instances. All test instances were solved to a relative gap tolerance of 2% and there was no time limit.

Figure 1 plots the results. Each bar corresponds to a randomly generated instance. Each instance is given a name where the first letter corresponds to the class from Table 1, the first number denotes to the instance number within the class, and the next three numbers specify the number of employees, number of alert types, and number of demand scenarios. The height of the bars represent the time in minutes that it took for the various models to reach the

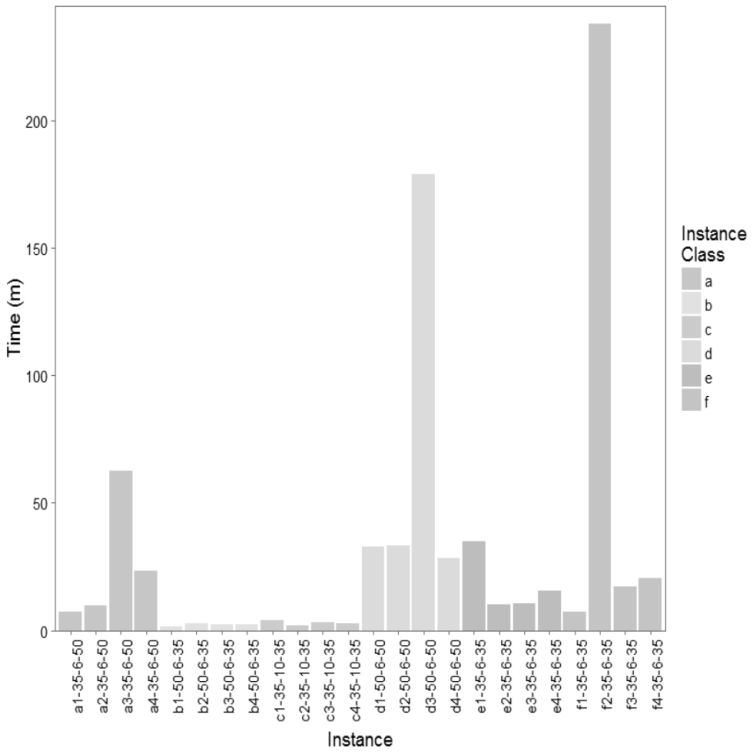

Fig. 1 Measuring solution time for small test cases

required gap tolerance. As the figure shows, most of these instances solved in well under one hour, and two cases took over two hours. Why a small number of test cases take considerably longer to solve is not obvious and is a topic for future investigation.

5.3 Demonstrating the Effectiveness of the Matheuristic

In this subsection, we demonstrate the effectiveness of our matheuristic on instances that are too large for a straightforward feed into a commercial integer programming (IP) solver.

To bound the quality of our solutions, we compare the objective value of the best solution found with the optimal objective value of the linear programming (LP) relaxation. We found that disaggregating the third constraint set (4) provided a tighter LP bound than that found when the model was relaxed in its original form. To disaggregate the constraints, we replace the third constraint

Instance	Obj. Val.	Upper Bound	Num. Iters	Time	Gap
g1-50-6-500	3099.8	3203.6	4	28.2 min	3.3%
g2-50-6-500	2948.1	2049.5	6	41.7 min	3.4%
g3-50-6-500	3190.9	3304.4	3	22.4 min	3.6%
h1-50-6-1500	2966.8	3078.4	4	145.0 min	3.8%
h2-50-6-1500	3083.4	3177.4	4	160.1 min	3.1%
h3-50-6-1500	3001.9	3109.8	2	165.6 min	3.6%
i1-50-10-500	3029.5	3156.3	2	140.6 min	4.2%
i2-50-10-500	3062.4	3167.7	4	237.7 min	3.4%
i3-50-10-500	3141.1	3262.4	7	368.4 min	3.9%
j1-85-6-500	4330.3	4474.1	4	79.4 min	3.3%
j2-85-6-500	4487.2	4624.9	6	110.3 min	3.1%
j3-85-6-500	4369.6	4507.3	4	74.8 min	3.2%
k1-100-20-50	5325.3	5600.1	13	106.4 min	5.2%
k2-100-20-50	5337.4	5644.1	4	34.0 min	5.7%
k3-100-20-50	5004.4	5296.3	4	39.3 min	5.8%
Avg:					**3.9%**

Table 2 Computational results on large-A instances

set with a modified constraint set that comprised one constraint for each h variable as shown in (14). In all of the test cases, each integer programming sub-problem is solved, per step 3.a.II. of Algorithm 1, with a 1% tolerance for the relative integrality gap.

$$h^k_{e,t,d} \leq 8\ x_{e,t}\ \forall\ t \notin T(e),\ \forall\ e \in E,\ \forall\ k \in K_d,\ \forall\ d \in D \tag{14}$$

Table 2 presents the computational results for the large-A instances. The first column contains the instance name. The second column contains the final objective value found by Algorithm 1. The third column contains upper bound from the LP relaxation. The fourth column contains the number of iterations the local search repeated—that is, repartitioned the employees—until no better solution was found. The fifth column contains the total time that it took to run the entire algorithm. The sixth column contains the percent difference between LP relaxation objective value and the final problem objective value.

Our local search heuristic was able to produce solutions that were, on average, within 3.9% of the optimal solution. Even better, it produced these solutions in a few hours while a straightforward feed of some of these instances into a commercial IP solver might not even produce a feasible solution in a few hours.

We also test our local search heuristic on the large-B instances, which do not limit the number of skills that employees can be trained in. Similar to the previous sections, we label the instances *class instance − num employees − num alert types − num scenarios*.

Figure 2 displays two examples of what the composition of employees looks like in terms of the number of alert types they can analyze initially, and as a result of the training decisions. The first example is of instance *n2-85-12-50* in which

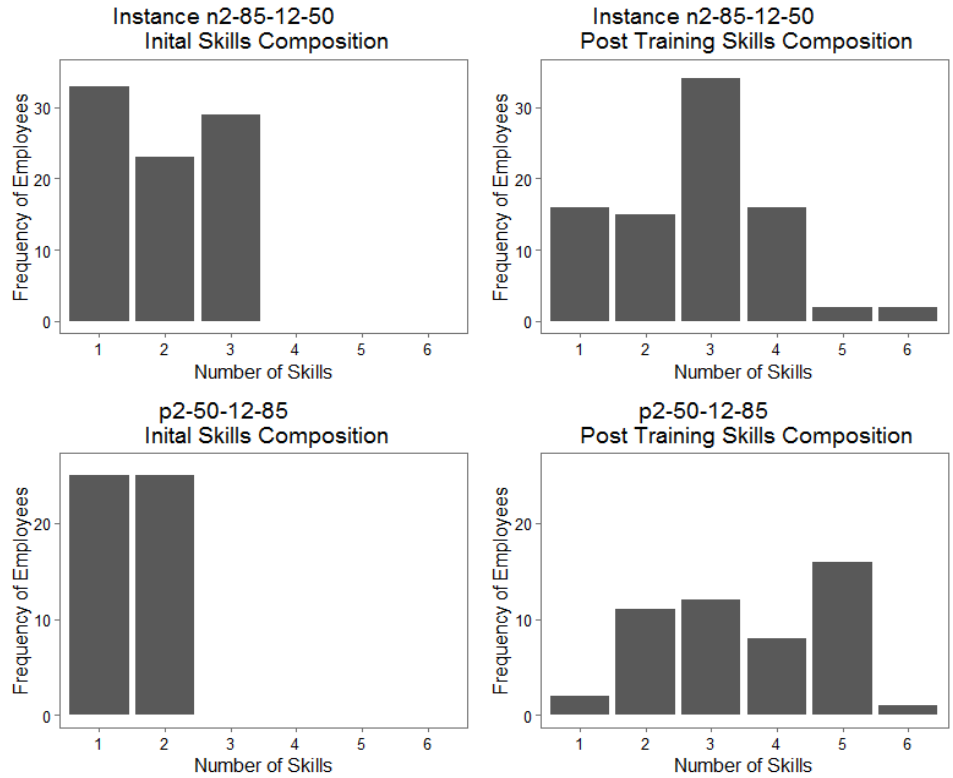

Fig. 2 Example of initial and post training skills density

initially, every employee had either 1, 2, or 3 initial skills. After the training decisions are made, the number of alert type skills per employee predictably shifts to the right. Interestingly the solution increases the number of skills up to six for two of the employees, while leaving sixteen employees with only one skill. Looking at the solution in more detail shows the employees who are not prescribed training are more likely to have initial skills for high demand alert types. The second row of charts in Figure 2 shows a similar example for *p2-50-12-85*. In this case there are fewer employees with only one skill after the training decisions are made because this test case has a larger training budget (20000 instead of 12000 units), but again the model favors giving one employee six alert types while leaving two employees with only one.

Table 3 shows the results for the large-B instances. The neighborhood search heuristic did not perform as well for these instance, even though it still produced solutions within 6.1% to 8.3% of the best bound. The ratio between the training budget and the number of employees had a noticeable impact on the time it took to solve the problem. For the **l** and **m** classes the budget-per-

Instance	Obj. Val	Upper Bound	Num. Iters	Time	Gap
l1-50-12-50	8335.6	8928.9	4	30.1 min	7.1%
l2-50-12-50	8663.2	9303.8	8	111.3 min	7.4%
l3-50-12-50	8816.3	9412.9	5	35.3 min	6.8%
m1-50-12-85	9078.0	9664.9	3	102.5 min	6.5%
m2-50-12-85	8865.6	9413.9	3	129.0 min	6.2%
m3-50-12-85	9290.4	9906.9	6	6.5 hr	6.7%
n1-85-12-50	13277.1	14137.1	5	18.5 min	6.5%
n2-85-12-50	13938.1	14888.4	2	9.3 min	6.8%
n3-85-12-50	13726.8	14564.4	7	24.9 min	6.1%
o1-85-12-50	12799.6	13671.2	8	31.0 min	6.8%
o2-85-12-50	13218.1	14122.1	3	13.4 min	6.8%
o3-85-12-50	13017.1	13926.1	5	19.4 min	6.5%
p1-50-12-85	9293.3	9965.6	5	17 hr	7.2%
p2-50-12-85	9334.9	10029.1	8	13 hr	7.4%
p3-50-12-85	9161.2	9922.8	3	25 hr	8.3%
Avg:					**6.9**%

Table 3 Computational results on large-B instances

		Straightforward feed		matheuristic	
	Instance	**Best Gap**	**Time**	**Best Gap**	**Time**
large-A	**g1-50-6-500**	41.9%	15 hr	3.3%	28.2 min
large-A	**h4-50-6-1500**	41.4%	49 hr	3.1%	165 min
large-B	**o1-85-12-50**	55.8%	7 hr	6.8%	31 min
large-B	**p1-50-12-85**	30.7%	16 hr	7.2%	17 hr

Table 4 Comparing run times of straightforward feed vs. matheuristic

employee was $\frac{12000}{50} = 240$ whereas in classes **n** and **o** the budget-per-employee was $\frac{12000}{85} = 141$, and the budget-per-employee for class **p** was $\frac{20000}{50} = 400$. This last class yielded the poorest results in terms of average integrality gap, but still achieved a single digit gap. Most likely, this is because the LP relaxation solutions trains employees in many fractions of skills, which does not resemble an integer optimal solution.

Table 4 provides a few examples of how our matheuristic performed against a straightforward feed into Gurobi if both methods are allowed to run for the same amount of time. For each technique, we provide the best gap obtained and the time required by that method to obtain it. In each of these cases, the matheuristic discovered remarkably better solutions compared to the straightforward feed, which never achieves a gap better than 30%. We note the second example is not an instance used in the above study, but is still comparable in scale to the previously cited **h** instances.

6 Conclusions and Future Work

We studied optimizing days-off scheduling decisions in an environment with unknown, highly variable workloads, many multi-skilled analysts, many job

types, the ability to increase employee skill sets with training, and a variety of staffing and scheduling constraints. We formulated this problem as a two-stage stochastic integer program and demonstrated this formulation can solve instances with as many as 50 employees, 6 alert types, and 50 demand scenarios per day by directly feeding the model into a commercial integer programming solver, without use of decomposition methods. Furthermore, for instances that are too large for a straightforward feed into a commercial solver, we demonstrate that an integer-programming-based local improvement technique—that is, a matheuristic—can be used to produce solutions that are, on average, within 4-7% of a bound of the optimal objective value. Our matheuristic also has the advantage of being easy to understand and relatively simple to implement.

There are a few of opportunities to build upon this work. First, the model could be extended to a problem where the volume of security alerts on each day is correlated with the volume of alerts on surrounding days. In terms of the cybersecurity application, cybersecurity attacks can sometimes occur on multiple fronts over several days, and so it may be advantageous to extend this study to a framework that considers such correlations. One approach to address this is to re-define scenarios in terms of system-level realizations of demand across the whole pay period rather than have each day possess its own set of demand scenarios. This model would likely require a large number of demand scenarios, and may need to be solved with a sampling approximation approach [22], or perhaps a modeling technique that strategically chooses scenarios that are sufficiently spread out across the demand space.

Second, our model presently makes days-off scheduling decisions for a typical two-week pay period in the first stage, and considers how jobs might be assigned to employees for any particular day in the second stage. Future efforts can also look into incorporating more recourse-related considerations into the demand/work scenarios and the second stage variables. These include, but are not limited to planned and unplanned employee absenteeism, overtime options, and on-call staffing options.

Third, there might be opportunities to make the local search more guided. At present, the algorithm randomly partitions employees and defines neighborhoods based on those employees subsets. However, there might be heuristics for strategically choosing subsets of employees for a neighborhood based on mismatches between supply and demand in the best known solution.

Fourth, more work can be done to model employee productivity. Our model implicitly assumes that insufficient data is available to estimate employee productivity rates on an individual employee-by-employee basis. However, such data may actually be available in this age of Big Data. This would allow for individual employee productivity rates to be estimated, which could serve as a natural alternative to breaking the symmetry between employees with iden-

tical skill sets and productivity rates.

Furthermore, our model assumes there is no loss in productivity if employees are assigned three different types of alerts versus one type of alert. This assumption can be relaxed, and a more realistic set of assumptions for a cybersecurity operations center can be identified. Perhaps employees lose some efficiency by switching between different types of alerts. But perhaps that loss of efficiency from context-changing could be offset by some subtle productivity gains from allowing employees to switch between tasks when they prefer to work on a different type of job.

7 Acknowledgments

We would like to thank our MITRE co-worker Kael Stilp for his comments on an earlier draft of this manuscript, which have helped us improve the exposition and content of the paper. We would also like to thank comments from our anonymous reviewers, who we think helped improved the clarity and exposition of this paper and brought a few interesting studies to our attention.

References

1. Altner DS, Rojas AC, Servi LD (2018) A two-stage stochastic program for multi-shift, multi-analyst workforce optimization with multiple on-call options. J Sched 21:517-531
2. Al-Yakoob SM, Sherali HD (2007) A column generation approach for an employee scheduling problem with multiple shifts and work locations. J Oper Res Soc 59:34-43
3. Avramidis AN, Chan W, Gendreau M, L'Ecuyer P, Pisacane O (2010) Optimizing daily agent scheduling in a multiskill call center. Euro J Oper Res 200:822-832
4. Bard JF, Binici C, de Silva AH (2003) Staff scheduling at the United States Postal Service. Comp & Oper Res 30:745-771.
5. Bard JF, Morton DP, Wang YM (2007) Workforce planning at USPS mail processing and distribution centers using stochastic optimization. Ann Oper Res 155:51-78
6. Barnhart C, Cohn AM, Johnson EL, Klabjan D, Nemhauser GL, Vance PH (2003) Airline crew scheduling. In: Hall RW (eds) Handbook of Transportation Science. Springer
7. Bhulai S, Koole G, Pot A (2008) Simple methods for shift scheduling in multiskill call centers. Manu & Serv Oper Man 10:411-420
8. Bodur M, Luedtke, JR (2016) Mixed-integer rounding enhanced Benders' decomposition for multiclass service system staffing and scheduling with arrival rate uncertainty. Man Sci 63:2073-2091
9. Boschetti MA, Maniezzo V, Roffilli M, Röhler AB (2009) Matheuristics: optimization, simulation and control. In: Hybrid Metaheuristics, Springer, pp 171177
10. Cuevas R, Ferrer J-C, Klapp, M, Muñoz J-C (2016) A mixed integer programming approach to multi-skilled workforce scheduling. J Sched 19:91-106
11. Della Croce F, Salassa F (2014) A variable neighborhood search based matheuristic for nurse rostering problems. Ann Oper Res 218:185-199
12. Ganesan R, Jajodia S, Cam H (2016) Optimal scheduling of cybersecurity analysts for minimizing risk. to appear in ACM Trans Int Sys Tech
13. Ganesan R, Jajodia S, Shah A, Cam H (2016) Dynamic scheduling of cybersecurity analysts for minimizing risk using reinforcement learning. ACM Trans Int Sys Tech 8

14. Gnanlet A, Gilland WG (2014) Impact of productivity on cross-training configurations and optimal staffing decisions in hospitals. Euro J Oper Res 238:254-269
15. Gomar JE, Haas CT, Morton DP (2002) Assignment and allocation optimization of partially multiskilled workforce. J Con Eng Man 128:103-109
16. Gurvich I, Luedtke JR, Tezcan T (2010) Staffing call centers with uncertain demand forecasts: a chance-constrained optimization approach. Man Sci 56:1093-1115
17. Henao C-A, Ferrer J-C, Muñoz J-C (2015) The impact of multi-skilling on personnel scheduling in the service sector: a retail industry case. J Oper Res Soc 66:1949-1959
18. Henao C-A, Ferrer J-C, Muñoz J-C, Vera J (2016) Multiskilling with closed chains in a service industry: A robust optimization approach. Int J Prod Econ 179:166-178
19. Kall P, Wallace SW (1994) Stochastic programming. John Wiley & Sons.
20. Kilincli G, Zhang X (2017) Mathematical models and solution approach for cross-training staffscheduling at call centers. Comp & Oper Res 87:258-269
21. Kim K, Mehrotra S (2015) A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. Oper Res 63:1431-1451
22. Mak W-K, Morton DP, Wood RK (1999) Monte Carlo bounding techniques for determining solution quality in stochastic programs. Oper Res Let 24:47-56
23. Margot F (2009) Symmetry in integer linear programming. In: 50 Years of Integer Programming: 1958 - 2008, Springer, 647-688
24. Ostrowski J, Anjos M, Vannelli A (2010) Symmetry in scheduling problems. Optimization Online accessed Sept. 2016.
25. Parisio A, Jones CN (2015) A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand. Omega 53:97-103
26. Restrepo MI, Gendron B, Rousseau L-M (2017) A two-stage stochastic programming approach for multi-activity tour scheduling. Euro J Oper Res 262:620-635
27. Robbins TR, Harrison TP (2010) A stochastic programming model for scheduling call centers with global service level agreements. Euro J Oper Res 207:1608-1619
28. Roth B, Balakrishnan A, Dewan P, Kuo A, Mallampati D, Morales J-C (2018) Crew Decision Assist: system for optimizing crew assignments at BNSF Railway. Interf to appear
29. Shah A, Ganesan R, Jajodia S, Cam H (2018) Adaptive reallocation of cybersecurity analysts to sensors for balancing risk between sensors. Serv Ori Comp App 12:123-135
30. Shah A, Ganesan R, Jajodia S, Cam H (2018) Optimal assignment of sensors to analysts in a cybersecurity operations center. IEEE Sys J to appear
31. Sir MY, Nestler D, Hellmich T, Das D, Laughlin MJ, Dohlman MC, Pasupathy K (2017) Optimization of multidisciplinary staffing improves patient experiences at the Mayo Clinic. Interf 47:425-441
32. Slomp J, Bokhorst JAC, Molleman E (2005) Cross-training in a cellular manufacturing environment. Comp & Ind Eng 48:609-624
33. Smet P, Wauters T, Mihaylow M, Vanden Berghe G (2014) The shift minimisation personnel task scheduling problem: a new hybrid approach and computational insights. Omega 46:64-73
34. Valeva S, Hewitt M, Thomas BW (2017) A matheuristic for workforce planning with employee learning and stochastic demand. Int J Prod Res 55:7380-7397
35. Van Den Bergh J, Beliën J, Brucker PD, Demeulemeester E, De Boeck L (2013) Personnel scheduling: a literature review. Euro J Oper Res 226:367-385
36. Van Slyke RM, Wets R, (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J App Math 17:638-663
37. Zhu X, Sherali HD (2009) Two-stage workforce planning under demand fluctuations and uncertainty. J Oper Res Soc 60:94-103