Journal of Scheduling manuscript No.
(will be inserted by the editor)

A Two-Stage Stochastic Program for Multi-shift, Multi-analyst, Workforce Optimization with Multiple On Call Options*

Douglas S. Altner · Anthony C. Rojas · Leslie D. Servi

Received: September 29, 2016 / Revised: October 2017 / Accepted: date

Abstract Motivated by a cybersecurity workforce optimization problem, this paper investigates optimizing staffing and shift scheduling decisions given unknown demand and multiple on call staffing options at a 24/7 firm with three shifts per day, three analyst types, and several staffing and scheduling constraints. We model this problem as a two-stage stochastic program and solve it with a column-generation-based heuristic. Our computational study shows this method only needs three minutes to produce solutions within 6% of a true lower bound of the optimal for 99% of over 150 test cases.

1 Introduction

One of the distinguishing challenges of shift scheduling in a cybersecurity context is highly variable workloads. Unlike staffing retail stores or hospitals, the volume of incoming alerts is difficult to forecast and can vary greatly. Staffing too many analysts can be expensive. Staffing too few poses a security risk and could lead to employee burnout.

The challenges of cybersecurity operation center (CSOC) staffing vary from operation to operation. Some CSOCs have multiple locations in multiple time zones, but the workforce at each serves one shift and can be scheduled independently of other locations. Other CSOCs possess multiple, possibly overlapping shifts at the same location, leading to a more complex scheduling problem.

* **Approved for Public Release; Distribution Unlimited. Case Number 16-0631.**

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

In this paper, we study a workforce optimization problem in a 24/7 setting with three non-overlapping shifts, three analyst types, unpredictable, highly variable workloads, several staffing and scheduling constraints, and multiple on call staffing options. Our research is targeted at CSOCs and the model described throughout the paper is in cybersecurity terms. However, this methodology could be adapted to other staffing and scheduling projects that face similar challenges.

Ganesan et al. 2016 [16], in collaboration with the Army Research Laboratory, introduced a CSOC workforce optimization problem. In several ways, our research builds upon but differs from their work. First, our model contains three 8-hour shifts instead of two 12-hour shifts, and includes additional constraints such as shift-specific seniority ratios and limits on red eye shifts. These features make the problem much more complex in terms of the number of variables and constraints. However, several firms prefer an 8-hour model. Likewise, the shift-specific staffing ratio constraints afford the planner more control over the seniority ratios in the workplace during every single shift, which can facilitate better supervision and mentoring opportunities.

Another difference is our model is a stochastic programming model whereas Ganesan et al. formulate a deterministic model. A deterministic model implicitly assumes a point estimate of future demand is sufficient for planning purposes. In contrast, a stochastic programming model explicitly plans for a wide range of possible demand scenarios, and thereby produces solutions that are more resilient to uncertainty. Our stochastic model allows the planner to hedge against uncertainty by considering two different types of on call staffing options. One at a lower rate that requires an upfront reservation cost, and another at a higher rate. The upfront reservation costs add an extra touch of realism, as many workplaces compensate employees for being on call, even if they are not actually asked to call in.

A third difference is that our method combines the staffing and scheduling decisions allowing for simultaneous optimization. Ganesan et al, in contrast, first optimize the staffing levels and then optimize the schedules given fixed staffing levels.

Lastly, our method employs a column-generation-based integer programming heuristic, which could be extended to an exact method via branch-and-price. Ganesan et al. employ a genetic algorithm to heuristically optimize staffing levels and use a "days off" scheduling algorithm to choose the shift schedules. Genetic algorithms, while they have their merits in certain contexts, cannot offer the same optimality guarantees that integer programming solutions can offer.

The main contributions of this paper are as follows. First, we offer a stochastic programming model for workforce optimization problems with a two-week time horizon, three non-overlapping shifts, three analyst types, unknown workloads, several scheduling and staffing constraints, and two on call staffing options. Our main focus is aiding cybersecurity firms but our techniques can be extended to other workforce management problems with similar features. Second, we offer a column-generation-based heuristic that we show quickly computes good quality solutions for a wide range of instances, and can be extended to an exact branch-and-price algorithm.

Our paper is organized as follows. The rest of this section presents a brief literature review of related problems and research. Section 2 presents a detailed, verbal description of the problem. Section 3 introduces our two-stage stochastic programming model. Section 4 describes our column generation method for heuristically solving the problem. Section 5 presents computational results. Section 6 suggests future work and discusses additional challenges of cybersecurity workforce optimization. This paper also includes an Appendix, which presents a few details about our column generation pricing subproblem.

1.1 Literature Review

The literature on workforce scheduling optimization is vast; see Ernst et al. 2004 [15] for a relatively recent survey. To list a few examples, complex shift scheduling problems have been studied for staffing nurses [7], call centers [12], railroad crews [29], airline crews [13] [18], and postal workers [5]. Researchers have created solutions for developing sophisticated schedules with multiple varying breaks within a single day [3] as well as schedules spanning two-week pay periods [6]. Papers have been published exploring branch-and-price options for shift scheduling [2] [27] as well as using grammars for generating complex shift patterns and work rules [14].

The contributions of Ganesan et al. 2016 [16] introduce an operations research model for workforce optimization at CSOCs. They propose a deterministic approach that splits staffing and scheduling into two separate problems, heuristically optimizing the former with genetic algorithms and the latter with a days off scheduling method. In a subsequent paper [17] that was developed concurrently with our work, Ganesan et al. incorporate a third problem that uses stochastic dynamic programming to determine an optimal policy for exercising a finite number of on call options on a day-to-day basis after the full-time schedule is fixed. Although this method considers future uncertainty, there are some key differences between our model and their model. First, our model assumes on call staffing options are tied to specific shifts. In contrast, their model assumes on call staff can be used over the course of the pay period subject to pay-period-wide constraints, which motivates a deriving an optimal policy. Second, our model considers a wide range of future demand scenarios while creating the schedule whereas Ganesan et al. use point estimates for future demand—essentially one average scenario—while creating the schedule. Combining elements from both models is potential future research, but beyond the scope of this paper.

There is also a growing number of papers on stochastic programming approaches to staffing and shift scheduling in general. Kao and Queyranne 1985 [21] introduced a two-stage problem that determines regular staffing hours in the first stage and overtime hours in the second stage. Bard, Morton, and Wang 2007 [6] study a two-stage stochastic shift scheduling problem in which the number of employees to have on payroll is determined in the first stage and particular work shifts are determined in the second stage. Zhu and Sherali 2009 [30] and Bodur and Luedtke 2016 [11] study workforce optimization models in which the second-stage decisions entail assigning a specific workload to specific employees. Robbins and Harrison 2010 [28] 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. Gurvich, Luedtke and Tezcan 2010 [19] 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. Kim and Mehrotra 2015 [23] 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. Altner, Mason, and Servi 2017 [1] address a two-stage stochastic programming model for multi-skilled analysts that also includes options for training analysts to perform new tasks.

Our paper complements the existing two-stage stochastic programming literature on employee scheduling by presenting a method tailored for tackling a different combination of challenges that is not covered by existing models—three non-overlapping shifts, several worker types, per-shift staffing ratio constraints, scheduling constraints, two-stage decision framework with future demand uncertainty, and on call staffing options with upfront reservation costs.

2 Problem Statement

This section provides a detailed, verbal description of the problem.

We are modeling determining staffing needs and scheduling shifts at a 24/7 cybersecurity operations center with two-week pay periods, on call staffing options, several scheduling and staffing constraints, and a wide range of possible future demand scenarios.

This is a workforce planning model, not a real-time execution tool. The intent of our model is to determine optimal staffing plans and shift schedules for a *typical* two-week pay period. The solution to our model can then be used to set target staffing levels, and for defining a core schedule that is repeated each pay period with small adjustments. The inputs for the model can be improved over time as lessons learned from previous periods can be incorporated into future runs of the model to revise future demand forecasts and scheduling-related business rules.

The main decisions the model recommends are employee staffing levels, employee shift schedules, and the number of employees who should be reserved for potential on call work for each time slot. Each of these decisions are made before the exact volume of work is known. No schedule for on call employees is locked in stone, even though the model will consider potential uses of on call staff under various demand scenarios as a means to calculate the expected costs of on call labor. This model can also be modified to assume a fixed workforce roster, or to assume bounds on the change in the total number of employees of each type.

The objective of the model is to minimize the expected staffing costs of meeting all demand in light of various constraints and business requirements.

This model considers two types of short-term staffing options: *reserved on call staff* and *unreserved on call staff*. The former requires an upfront reservation cost. The latter does not require an upfront reservation cost but is even more expensive to exercise. The on call employees are presently treated as if they come from a different workforce pool than the fulltime employees, but this assumption can be relaxed with additional constraints.

This model considers three non-overlapping shifts: 12am-8am (red eye), 8am-4pm (day shift), and 4pm-12am (night shift). Three seniority levels of full-time analysts can be hired: junior, senior, and principal. In general, more senior analysts work faster than more junior analysts, but are also more expensive. There are also constraints to ensure a certain mix of each seniority level is employed, partly for best practice reasons so the more senior analysts can coach and oversee the more junior analysts. For simplicity, we assume on call staff are senior employees but this assumption can be relaxed.

Our model assumes the volume of incoming security alerts can be reasonably modeled with a probability mass function. The parameters of the probability mass function for the number of alerts varies with the day and shift. The model presently assumes that the workload for each week-day-shift triplet is independent of the workload for every other such triplet. This assumption can be relaxed for use cases where this is not a realistic assumption, but relaxing this assumption will likely require taking additional measures to handle the large number of scenarios.

The costs of a staffing and scheduling plan include the salary rates of full-time analysts, the expected wages paid to on call analysts, and bonuses given to employees for working red eye shifts. Since on call staff are brought in on short notice, they have a much higher hourly rate than their full-time counterparts who have identical productivity rates.

There are four broad categories of constraints: 1.) ensure the staff can analyze all incoming alerts, 2.) ensure the firm has an acceptable mix of analysts of each seniority level at every particular shift and in general, 3.) ensure each shift schedule for each analyst satisfies a number of work rules, and 4.) ensure the number of reserved on call options exercised is limited by the number of reservations actually paid for.

For the demand satisfaction constraints, we assume all alerts must be completed during the shifts in which they arrive. Any work that cannot be completed with the scheduled full-time analysts must be completed by on call analysts. For best practice reasons and to guarantee a certain level of readiness, we also assume the firm wants the full-time staff scheduled for each shift to be able to analyze all alerts without using any on call staff in at least 50% of demand scenarios. This is referred to as the *best practice readiness constraint.*

For the staffing mix constraints, we assume that at least 25% of the analysts on staff must be junior analysts, 25% must be senior analysts, and 20% must be principal analysts. Similarly, we assume each shift must have either one senior analyst for every three junior analysts, or one principal analyst for every six junior analysts, and that no shift is ever entirely staffed by junior analysts. Likewise, we assume there must be one principal analyst for every five senior analysts on each shift when the number of senior analysts is rounded down to the nearest group of 5. Thus, a shift that is staffed by four or fewer senior analysts need not have a principal analyst. These staffing ratio constraints enable the more senior analysts to mentor and train the more junior analysts. These constraints only apply to full-time staff.

For the work rule constraints, we require that each full-time analyst works 5 8-hour shifts per week, so 10 such shifts per pay period. We also require that each full-time analysts works at most 1 shift per 24-hr period, at most five days in a row, and at most one weekend per pay period. Furthermore, we require that analysts work at most two red eye shifts per week and at most three red eye shifts per pay period.

3 Mathematical Formulation

In this section, we present a stochastic integer programming formulation of this problem.

Following Kall and Wallace [20], we use standard stochastic programming terminology. For our particular model, the number of incoming alerts for each shift is modeled as an independent discrete random variable. For each shift, each possible value that the random variable can take is referred to as a *scenario.* The variables corresponding to decisions that must be made before the particular scenario is realized are *first-stage decision variables.* The variables corresponding to decisions made after the scenario is realized are *second-stage decision variables.*

To simplify the notation, throughout this paper we refer to a week-day pair as a simply a *day* so we treat a pay period as consisting of fourteen days.

In this problem, we also assume each shift for each day has its own set of scenarios which occur independently of the scenarios of every other shift. In addition, we assume each shift has a *base scenario*, which occurs with probability .5. The base scenario is formed without loss of generality by combining a set of scenarios to implicitly enforce the best practice readiness constraint.

To make it easier to distinguish variables from coefficient data in the notation, we denote variables with individual letters and data while truncated words.

Sets:

- $D := \{Su1, Mo1, \ldots, Fr2, Sa2\}$, the set of days in the two-week period.
- $S := \{s_1(day), s_2(night), s_3(redeye)\}$, the set of possible shifts an analyst could work each day.
- $L := \{\ell_1(junior), \ell_2(senior), \ell_3(principal)\}$, the set of possible seniority levels for analysts.
- $\mathcal{P}$, the set of feasible two-week shift schedules that an analyst could work.
- $K_{d,s}$, the set of possible demand scenarios for day d and shift s.

Data:

- $cost^{full}_{\ell,p}$, the per pay period cost of having a full-time, level ℓ analyst work shift schedule p, including bonuses paid for red eye shifts.
- $cost^{book}_{s}$, the cost of reserving an analyst for an on call option for a shift of type s.
- $cost^{res}_{s}$, the cost of exercising a *reserved* on call option for a shift of type s.
- $cost^{unres}_{s}$, the cost of exercising an *unreserved* on call option for a shift of type s.
- $alerts^{k}_{d,s}$, the number of incoming alerts on day d during shift s under scenario k.

- $alerts^0_{d,s}$, the base scenario for shift s on day d.
- $prob^k_{d,s}$, the probability of scenario k for shift s on day d.
- $rate_\ell$, the number of alerts that one level ℓ analyst can analyze during a shift.
- $\lambda^p_{s,d}$, takes the value 1 if shift schedule p includes the analyst working shift s on day d and 0 otherwise.

Parameters:

- $minper_\ell$, the minimum percentage of level ℓ analysts needed overall, for each $\ell \in L$.

Decision Variables:

1. $x_{\ell,p}$:= the number of level ℓ analysts working shift schedule p, for each $p \in \mathcal{P}$ and each $\ell \in L$.
2. $u_{d,s}$:= the number of on call analysts who should be reserved upfront for shift s on day d, for each $s \in S$ and $d \in D$.
3. $v^k_{d,s}$:= the number of *reserved* on call options that should be exercised for shift s on day d under scenario k, for each $s \in S$, $d \in D$, and $k \in K_{d,s}$.
4. $w^k_{d,s}$:= the number of *unreserved* on call options that should be exercised for shift s on day d under scenario k, for each $s \in S$, $d \in D$, and $k \in K_{d,s}$.

Problem Formulation:

Minimize

$$\sum_{\ell\in L}\sum_{p\in\mathcal{P}} cost^{full}_{\ell,p}\ x_{\ell,p} + \sum_{d\in D, s\in S} cost^{book}_{s}\ u_{d,s} \tag{1}$$

$$+ \sum_{d\in D, s\in S}\ \sum_{k\in K_{d,s}} prob^{k}_{d,s}\ \left(cost^{res}_{s}\ v^{k}_{d,s} + cost^{unres}_{s}\ w^{k}_{d,s}\right) \tag{2}$$

Subject to:

$$rate_{\ell_2}\ \left(v^{k}_{d,s} + w^{k}_{d,s}\right) + \sum_{p\in\mathcal{P}}\sum_{\ell\in L}\lambda^{p}_{s,d}\ rate_{\ell}\ x_{\ell,p} \geq alerts^{k}_{d,s} \qquad \forall\ k \in K_{d,s}\backslash\{k^{d,s}_{0}\},\ \forall\ s\in S, \forall\ d\in D \tag{3}$$

$$\sum_{p\in\mathcal{P}}\sum_{\ell\in L}\lambda^{p}_{s,d}\ rate_{\ell}\ x_{\ell,p} \geq alerts^{0}_{d,s} \qquad \forall\ s\in S, \forall\ d\in D \tag{4}$$

$$\sum_{p\in\mathcal{P}} x_{\ell,p} \geq minper_{\ell}\Big(\sum_{p\in\mathcal{P}}\sum_{\ell\in L} x_{\ell,p}\Big) \qquad \forall\ \ell\in L \tag{5}$$

$$\sum_{p\in\mathcal{P}}\lambda^{p}_{s,d}\ x_{\ell_1,p} \leq \sum_{p\in\mathcal{P}}\lambda^{p}_{s,d}\big(3x_{\ell_2,p} + 6x_{\ell_3,p}\big) \qquad \forall\ s\in S, \forall\ d\in D \tag{6}$$

$$\sum_{p\in\mathcal{P}}\lambda^{p}_{s,d}\ x_{\ell_2,p} \leq 4 + \sum_{p\in\mathcal{P}} 5\lambda^{p}_{s,d}x_{\ell_3,p} \qquad \forall\ s\in S, \forall\ d\in D \tag{7}$$

$$\sum_{p\in\mathcal{P}}\sum_{d\in D}\lambda^{p}_{s_3,d}x_{\ell_3,p} = 0 \tag{8}$$

$$v^{k}_{d,s} \leq u_{d,s} \qquad \forall\ k \in K_{d,s}\backslash\{k^{d,s}_{0}\},\ \forall\ s\in S, \forall\ d\in D \tag{9}$$

$$x_{\ell,p} \in \{0\}\cup\mathbb{Z}^{+} \qquad \forall\ \ell\in L, \forall\ p\in\mathcal{P}$$

$$u_{d,s} \in \{0\}\cup\mathbb{Z}^{+} \qquad \forall\ d\in D, \forall\ s\in S$$

$$v^{k}_{d,s} \in \{0\}\cup\mathbb{Z}^{+} \qquad \forall\ k\ \in K_{d,s},\ \forall\ d\in D, \forall\ s\in S$$

$$w^{k}_{d,s} \in \{0\}\cup\mathbb{Z}^{+} \qquad \forall\ k\ \in K_{d,s},\ \forall\ d\in D, \forall\ s\in S$$

The objective function is to minimize the total expected payroll cost. The first part of the objective function (1) captures the costs of the full-time analysts, including bonuses paid for working red eye shifts, plus the upfront costs for reserving on call analysts. The second line (2) is a discretization of the expected costs of exercising on call staffing options. This consists of the expected costs of exercising reserved on call options plus the expected costs of exercising unreserved options.

The constraints (3) require that the firm have enough full-time and on call staff to analyze all incoming alerts under every scenario that is not the base scenario, for each shift. The constraints (4) ensure all alerts in each base scenario for each shift can be analyzed using only full-time staff. The constraints (5) guarantees the required minimum percentage of each type of analyst on the overall roster. The constraints (6) enforces two conditions. First, that for each shift, we have at least one senior or principal analyst (since each shift needs at least one full-time analyst.) Second, that there is one full-time senior analyst for every three full-time junior analysts and/or one full-time principal analyst for every six full-time junior analysts. Constraint set (7) ensures there is one full-time principal analyst for every five full-time senior analysts, for each shift that has five or more senior analysts. Set (8) prevents principal analysts from working red eye shifts. Set (9) limits the number of exercised on call options at the reserved rate to the number of such options that were paid for upfront. The remaining sets of constraints define the variables as non-negative integers.

All of the scheduling constraints are implicitly enforced in the set $\mathcal{P}$. The next section includes a mathematical characterization of $\mathcal{P}$.

As previously mentioned, we are assuming that scenarios for each shift are independent of scenarios for each other shift. If these scenarios should instead be correlated, we can reformulate this model with system-level scenarios such that each scenario specifies a level of incoming alerts for each shift on each day. This approach would surely make it computationally infeasible to explicitly add decision variables for every single scenario to the master problem formulation. Instead, the problem would likely need additional measures to handle the large number of scenarios such as a sampling-type approach (e.g., [26] and [24]), a constraint generation approach (e.g., [4] and [25]) and/or some kind of scenario-reduction approach (e.g., [22]).

The best practice readiness constraint can be enforced via the base scenario without loss of generality because if each of the scenarios subsumed by the base scenario was written as a separate constraint then the constraint corresponding to the most demanding of the 50% less demanding scenario would dominate all of the others. Hence, all of the dominated constraints can be dropped, and the dominating constraint can be regarded as a base scenario that occurs with probability .5.

3.1 Tightening the Master Problem Formulation

In this subsection, we describe a simple rounding technique to tighten the aforementioned formulation.

The first two sets of constraints—(3) and (4)—in formulation are the *demand satisfaction constraints.* Let $rate_{gcd}$ be defined as the greatest common denominator for the set $\{rate_\ell : \ell \in L\}$. Then, the following is a tightening of the demand satisfaction constraints:

$$\begin{aligned}
&\tfrac{rate_{\ell_2}}{rate_{gcd}}\left(v^k_{d,s} + w^k_{d,s}\right) \\
&+ \textstyle\sum_{p\in\mathcal{P}} \sum_{\ell\in L} \lambda^p_{s,d}\, \tfrac{rate_\ell}{rate_{gcd}}\, x_{\ell,p} \geq \left\lceil \tfrac{alerts^k_{d,s}}{rate_{gcd}} \right\rceil \ \forall\ k \in K_{d,s}\backslash\{k_0^{d,s}\}, \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad \forall\ s \in S, \forall\ d \in D \qquad (10) \\
&\textstyle\sum_{p\in\mathcal{P}} \sum_{\ell\in L} \lambda^p_{s,d}\, \tfrac{rate_\ell}{rate_{gcd}}\, x_{\ell,p} \geq \left\lceil \tfrac{alerts^0_{d,s}}{rate_{gcd}} \right\rceil \quad \forall\ s \in S, \forall\ d \in D
\end{aligned}$$

This is a tightening because the left hand side of the inequalities will still always be integral by the fact that $rate_{gcd}$ is the greatest common denominator of each of the analyst rates, meaning that it will divide evenly into a coefficient of each of the terms on the left hand side. However, the right hand side could potentially be fractional. Therefore, if we round up the right hand side to the nearest integer, these constraints serve as valid inequalities to the original master problem, which in turn means they can replace the original demand satisfaction constraints.

4 Solution Techniques

In this section, we introduce methods for solving our workforce optimization problem.

This problem can be solved exactly with branch-and-price. Branch-and-price is an exact method for solving highly structured integer programs with a large number of decision variables that combines column generation with branch-and-bound [8].

Instead of using branch-and-price, we solve this problem using a similar, column-generation-based heuristic whereby we only generate columns at the root node of the branch-and-bound tree. This heuristic simplification is easier to implement, runs faster, and quickly produces good quality solutions. Our pricing framework can also be extended to a full branch-and-price implementation.

Following Bertsimas and Tsitsiklis [9], we use conventional terminology for column generation and branch-and-price methods. The complete, main integer program that we are solving is the *master problem.* The subproblem formed on a subset of the variables is the *restricted master problem.* Decision variables are alternately referred to as a *column* and each $x_{\ell,p}$ decision variable corresponds to a feasible shift schedule. The subproblem that we solve to determine which new columns to add, if any, to the restricted master problem is the *pricing subproblem.*

In the next subsection, we discuss the pricing subproblem for generating new columns to add to the restricted master program. In the second subsection, we discuss how we generate an initial set of columns. In the third subsection, we describe our heuristic simplification to the branch-and-price process.

4.1 Pricing Subproblem

In this subsection, we describe our integer program for generating new columns (which correspond to promising new shift schedules) to add to the restricted master problem.

We list the objective coefficients for the decision variables of the pricing subproblem as $\pi_{s,d}$. These are derived from the shadow prices of the constraints of the linear programming relaxation of the restricted master problem, when it is solved to optimality. The appendix contains a detailed discussion of how these coefficients are derived as well as how the objective value of a feasible solution to the pricing subproblem can be used to compute the reduced cost of the corresponding column for the master problem.

Decision Variables:

1. $y_{s,d} := 1$ if this shift schedule entails an analyst working shift s on day d, and 0 otherwise, for each $s \in S$ and each $d \in D$.
2. $z := 1$ if this shift schedule entails the analyst having $Sa1, Su2$ as his weekend off, and 0 if he will have $Su1, Sa2$ off.

Maximize

$$\sum_{s \in S} \sum_{d \in D} \pi_{s,d} \; y_{s,d} \tag{11}$$

Subject to:

$$\sum_{\bar{s}=s}^{s+2} y_{\bar{s},d} \leq 1 \qquad \forall \; s \in S, \forall \; d \in D \tag{12}$$

$$\sum_{d \in W_i} \sum_{s \in S} y_{s,d} = 5 \qquad \forall \; i \in \{1, 2\} \tag{13}$$

$$\sum_{\bar{d}=d}^{d+5} \sum_{s \in S} y_{s,\bar{d}} \leq 5 \qquad \forall \; d \in D \tag{14}$$

$$\sum_{s \in S} y_{s,d} \leq 1 - z \qquad \forall \; d \in \{Sa1, Su2\} \tag{15}$$

$$\sum_{s \in S} y_{s,d} \leq z \qquad \forall \; d \in \{Su1, Sa2\} \tag{16}$$

$$\sum_{d \in W_i} y_{s_3,d} \leq 2 \qquad \forall \; i \in \{1, 2\} \tag{17}$$

$$\sum_{d \in D} y_{s_3,d} \leq 3 \tag{18}$$

$$y_{s,d} \in \{0, 1\} \qquad \forall \; s \in S, \forall \; d \in D$$

$$z \in \{0, 1\}$$

As is detailed in the appendix, the objective function is mathematically equivalent to determining the column with the best reduced cost.

The constraint set (12) ensures the constructed shift schedule contains at most one shift per 24-hour period. The set (13) enforces that the constructed shift schedule contains exactly five shifts per week. For these constraints, W_i is the set of days corresponding to the ith week in the pay period. The set (14) ensures the corresponding shift schedule contains shifts in at most five consecutive days. Constraint sets (15) and (16) collectively ensure the shift schedule will include at least one weekend off. Constraint sets (17) and (18) respectively enforce that analysts are assigned at most two red eye shifts per week, and at most three red eye shifts in a single pay period. The last constraint sets define the decision variables as binary variables.

Note that the set of all feasible solutions to the pricing subproblem corresponds to the set $\mathcal{P}$ described in the previous section.

4.2 Constructing the Initial Columns

This section discusses how we generated an initial set of shift schedules, which we used to generate an initial set of decision variables for the restricted master problem.

We obtain an initial set of columns by greedily constructing a feasible solution to a related covering problem, and then using the solution to that covering problem to determine the starting set of columns. The covering problem is to cover the demand in each shift by feasible shift schedules. Each shift schedule in this covering problem is given a *weight* to indicate how much demand it will cover in each shift that is included in that shift schedule.

To be more precise, let $u^i_{s,d}$ denote the number of uncovered alerts for shift s on day d during iteration i of the greedy covering algorithm and let $\mathcal{P}^i$ denote the set of feasible shift schedules generated so far. We initialize the algorithm by setting $\mathcal{P}^0 = \emptyset$ and $u^0_{s,d} = E[a_{s,d}]$, the expected demand for shift s on day d.

To determine the next shift schedule to add to $\mathcal{P}^{i+1}$, we determine the shift schedule that maximizes $\sum_{s\in S,d\in D} u^i_{s,d}\ y_{s,d}$ subject to $y \in \mathcal{P}$, where $y_{s,d}$ is the binary variable that takes value 1 if the schedule entails working shift s on day d and 0 otherwise. Let y^* denote the optimal solution to the subproblem for this iteration. The next step is to determine the weight to give to the schedule denoted by y^*, to determine how many alerts this shift schedule will be used to cover. This weight is obtained by computing $\omega = min_{s\in S,d\in D}\big\{u^i_{s,d}\ y^*_{s,d} : u^i_{s,d}\ y^*_{s,d} > 0\big\}$, and we subtract ω from the number of uncovered alerts for each shift covered by y^* that still have uncovered alerts. We continue the algorithm until all alerts are covered.

Algorithm 1: Greedy algorithm for constructing a solution to the related covering problem

```
begin
    Set: $i \leftarrow 0$
    Set: $\mathcal{P}^i \leftarrow \emptyset$
    Set: $u^i_{s,d} \leftarrow E[a_{s,d}]\ \forall s\ \in S,\ d\ \in D$
    while $\exists (s,d) \in S \times D : u^i_{s,d} > 0$ do
        Set: $y^* \leftarrow argmax_{y\in\mathcal{P}}\big\{\sum_{s\in S,d\in D} u^i_{s,d}\ y_{s,d}\big\}$
        Set: $\omega \leftarrow min_{s\in S,d\in D}\big\{u^i_{s,d}\ y^*_{s,d} : u^i_{s,d}\ y^*_{s,d} > 0\big\}$
        Set: $u^{i+1}_{s,d} \leftarrow u^i_{s,d} - \omega\ y^*_{s,d}$
        Set: $\mathcal{P}^{i+1} \leftarrow \mathcal{P}^i \cup \{y^*\}$
        Set: $i \leftarrow i + 1$
    end
end
```

		Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week 1	Red Eye	0	0	96	15	64	108	127
	Day	0	29	0	0	0	78	189
	Night	193	0	0	6	49	49	205
		Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Week 2	Red Eye	102	67	28	28	4	76	79
	Day	102	6	12	0	87	6	319
	Night	196	0	0	0	124	0	292

Table 1 Illustrating how the greedy algorithm constructs a shift schedule given an array of uncovered alerts. The selected shifts for this iteration are shaded.

Algorithm 1 contains pseudocode for our greedy algorithm. This algorithm produces a set of feasible shift schedules $\mathcal{P}^j \subseteq \mathcal{P}$.

Table 1 provides a concrete example of how the greedy algorithm works for a single iteration. The table contains an array of outstanding, uncovered alerts. The shaded cells correspond to the shifts selected by the greedy algorithm in its current iteration. These shifts correspond to a feasible shift schedule that maximizes the sum of the uncovered alerts. It might appear that the solution can be improved by switching some night shifts to red eye shifts, but this would violate the constraint that requires that a shift schedule includes at most three red eye shifts. For the example in Table 1, the weight ω would be 15 because the number of uncovered alerts on Wednesday of the first week is the lowest for all selected shifts with a nonzero number of uncovered alerts.

To initialize the set of columns for the restricted master problem, we create three columns for each shift schedule in $\mathcal{P}^j$, one for each analyst level. A basic feasible solution would always be possible on this set of columns were it not for the constraint preventing principal analysts from working red eye shifts. To see this, just consider satisfying all demand with junior analysts and only adding senior and principle analysts as necessary to satisfy the staffing ratio constraints. In principle, this approach must be modified if an infeasible situation is created by the prohibition on principal analysts being scheduled for red eye shifts. However, in practice, we never needed to devise such a modification.

4.3 Column-Generation-Based Heuristic

This subsection describes our overall solution method, which is delineated in Figure 1 as a process flow diagram. At a high-level, this method amounts to first solving the linear programming (LP) relaxation of the complete master problem using column generation (Phase I in Figure 1). Then, using all of the columns that were generated during the solution of the LP relaxation, solve an integer version of the restricted master problem defined only on the columns generated (Phase II). In essence, this is a heuristic simplification of branch-and-price where columns are only generated at the root node of the

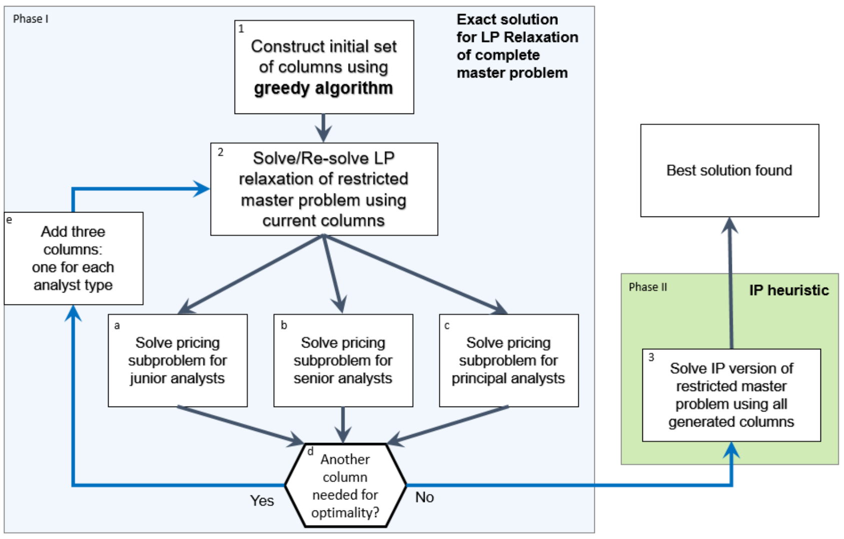
Phase I
Exact solution for LP Relaxation of complete master problem
1 Construct initial set of columns using **greedy algorithm**
2 Solve/Re-solve LP relaxation of restricted master problem using current columns
e Add three columns: one for each analyst type
a Solve pricing subproblem for junior analysts
b Solve pricing subproblem for senior analysts
c Solve pricing subproblem for principal analysts
d Another column needed for optimality?
Yes
No
Phase II
IP heuristic
3 Solve IP version of restricted master problem using all generated columns
Best solution found

Fig. 1 Process flow diagram for the column-generation-based heuristic.

branch-and-bound tree.

Explaining the heuristic further, step 1 corresponds to Algorithm 1, which is how we generate the initial set of columns for this method. Step 2 entails solving the linear programming (LP) relaxation of the restricted master problem on the current working set of columns. To see if the solution of the LP relaxation of the restricted master problem is an optimal solution of the LP relaxation of the complete master problem, three pricing subproblems must be checked—one for creating a new feasible schedule for each analyst type (steps a, b, and c). If any of these pricing subproblems indicate that a new column (corresponding to a new feasible schedule p) should be generated, then we create three new columns corresponding to schedule p, one for each analyst level (step e). This is the equivalent of adding variables $x_{\ell_1,p}, x_{\ell_2,p}$ and $x_{\ell_3,p}$ to the restricted master problem. (The Appendix contains a more detailed presentation of how we compute reduced costs, and how these are used to derive the objective function for the pricing subproblems.) Finally, step 3 is to solve the integer version of the restricted master problem (using Gurobi) on the set of columns generated during the prior stages of this algorithm.

The fact that our method is tantamount to only generating columns at the root node of the branch-and-bound tree means it is theoretically possible that a better solution could be found if columns are also generated at some of the sub-nodes of the tree. We leave it to future research to explore the advantages and trade-offs of a full branch-and-price implementation.

	Salary per period	**Alert processing rate per 8 hours**	**Required min. percent on staff**
Junior	\$3,000	40	25%
Senior	\$4,000	60	25%
Principal	\$6,000	80	20%

Table 2 Analyst attributes

5 Computational Experimentation

In this section, we describe our computational results. The first subsection discusses how we generated our test instances. The second subsection presents our results in terms of solution quality, showing that our heuristic only needed a few minutes to find solutions within at most 7% of a lower bound of the true optimal solution. The third subsection discusses how solution quality varies over the time if the integer programming solver is allowed to run longer. The fourth subsection illustrates how our stochastic program outperforms a natural deterministic approximation to the same problem, sometimes cutting expected staffing costs by nearly 20%. This section also offers intuition for the superiority of the stochastic programming approach, discussing how it is more effective at evenly allocating spare capacity.

5.1 Generating Instances

These experiments employ one hundred sixty randomly generated test cases. In all of our test instances, we assume that junior, senior and principal analysts can process 40, 60, and 80 alerts per 8-hour shift respectively. Junior analysts are paid \$3,000 per pay period; senior analysts are paid \$4,000 per pay period; and principal analysts are paid \$6,000 per pay period. All analysts (including on-call) receive a 5% bonus for each red eye shift they work. Table 2 summarizes the analyst attributes.

On call analysts reserved in advance can also process 60 alerts per 8-hour shift and cost \$480 per shift (so they would cost \$4,800 if they were to work the standard 10 8-hour shifts in a pay period.) The reservation costs requires that 10% be paid up front and the remaining 90% is paid only if that analyst is called in for that shift. Unreserved on call analysts can be brought in by the hour and are assumed to process 10 alerts per hour. To ensure unreserved on call labor is prohibitively expensive for most planning scenarios, we set the cost to \$225 per hour (equivalent to \$18,000 per 10 8-hour shifts in a pay period.)

For day d and shift s, $a_{d,s}$ is a discrete random variable with k scenarios. The base scenario occurs 50% of the time, and entails X alerts arriving. To generate these values for our instance, we draw the base demand X from the shift

	Sunday	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**
Day	300 to 600	600 to 900	600 to 900	600 to 900	600 to 900	450 to 750	300 to 600
Night	200 to 400	400 to 600	400 to 600	400 to 600	400 to 600	300 to 500	200 to 400
Red eye	66 to 133	133 to 200	133 to 200	133 to 200	133 to 200	100 to 167	66 to 133

Table 3 Uniform distributions for generating the base levels of alerts, by day and shift

specific uniform distribution listed in Table 3. These ranges are chosen so that the highest volume of alerts will typically be seen during standard business hours. (X is a data point for each stochastic programming instance, and is not a random variable in the model.)

The remaining scenarios $i \in \{2, 3, \ldots, k\}$ take the values of the $\frac{1}{2} + \frac{i-1}{2(k-1)}$ quantile of a finite discrete probability distribution. To ensure a broad range of potential cases are considered, we vary the type of demand distribution, the standard deviation of the distribution, and the number of scenarios k. Half of the cases considered use a discrete uniform distribution, and the other half use a truncated, discretized normal random variable. Both distributions have a shift-specific mean of X, and a standard deviation taking values in $\{75, 100, 150, 200\}$. The number of scenarios k takes values in $\{5, 10, 15, 20\}$. For each combination of the parameters outlined above, we solve 5 instances, randomly generating the base demand X.

5.2 Measuring Solution Quality

To measure the quality of our solutions, we compare the objective value of the best solution found at the time of termination, with the optimal objective value of the linear programming relaxation of the master problem. The latter serves as a lower bound to the optimal objective value to the integer version of the problem. Whether this is a relatively tight bound is an open question for future research.

We ran our experiments on a virtual machine with 8 2.60 Ghz Intel processors and 16.00 GB of RAM. Our model is implemented in Python 3.5 and use Gurobi 7.0.0 to solve all mathematical programs. The maximum amount of time the integer program is allowed to run is three minutes and the relative gap tolerance is set to 1%. We restrict the run time to three minutes to see how well this method performs if it were part of a broader, optimization-based decision support system in which users want good quality answers in a few minutes.

Figure 2 illustrates the overall results of these experiments. The true gap is measured as the difference between the best known solution to the restricted master program—that is the integer program solved during step 3 of Algorithm 2, which is a restriction of the true master program described in Section

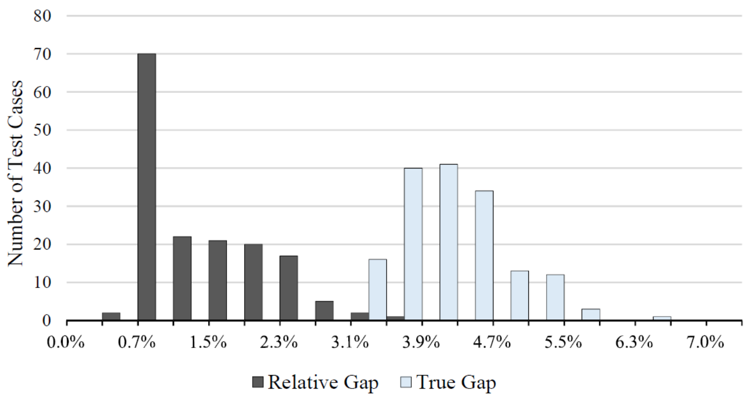

Fig. 2 Histogram of solution gaps for 160 test instances, after 3 minutes of solver run time. The relative gap indicates how close our algorithm came to solving the heuristically restricted version of the true problem. The true gap indicates how close our algorithm came to a bound of the optimal objective value of the true, unrestricted problem.

4—and the solution to the linear programming relaxation of the true master problem. The relative gap in this context is the relative gap for the restricted master problem that was outputted by Gurobi after three minutes of runtime. The relative gap indicates how close our algorithm came to solving the heuristically restricted version of the true problem. The true gap indicates how close our algorithm came to a bound of the optimal objective value of the true, unrestricted problem.

In all cases, our method found solutions to each of the test instances within at most 7% of the true lower bound in three minutes. On average, the solutions are within 4.2% of the lower bound, and 88% of the test instances are within 5% of this bound. Since the benchmark is a lower bound, it is an open question whether this gap largely exists because there are better solutions that our heuristic overlooked but a full branch-and-price implementation would have found, or because the lower bound could be improved. Likewise, the integrality gap for the restricted master problem (shown as the relative gap in Fig. 2) is always under 4%, and within 2% for 75% of the cases, upon termination. So the restricted master problems are all almost solved exactly even after a few minutes.

Figure 3 shows how the proportion of hours worked by each analyst type changes as a result of differing assumptions on the distribution of incoming alerts. The four top bars are based on instances using a discrete uniform distribution, where the standard deviation takes values of 75, 100, 150, and 200. The bottom four bars use the same values for standard deviation, but instead assume the alerts follow a discretized truncated normal distribution. The hours for the on call analysts are totaled over the two week period, assuming the scenario with the highest volume of alerts occurs for every shift. This is to

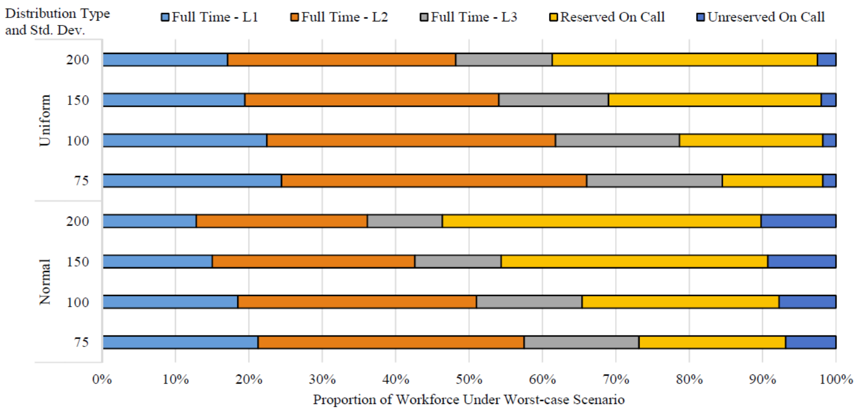

Fig. 3 An illustration of the analyst type distribution in the scenario with the highest alert volume, and how this mix is affected by differing assumptions on the demand distribution.

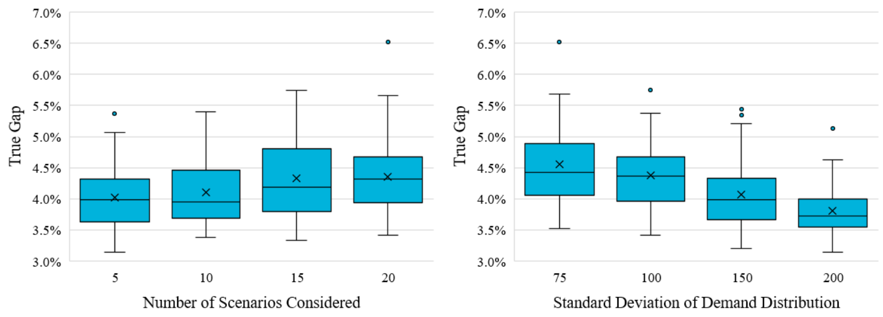

Fig. 4 Boxplots showing good integrality gaps are obtained across a range of inputs. To the left, we vary the number of scenarios per shift-day-week. To the right, we vary the standard deviation of the underlying distribution.

showcase the response of the on-call options to increases in uncertainty. Each of the 8 bars above represent the average behavior over 20 test instances.

Intuitively, as the uncertainty in demand increases, so does the requirement for on call staff. Due to the long tails of the normal distribution, the scenario with the highest alert volume is much more severe in the normal instances than their uniform counterparts.

The eight boxplots in Figure 4 show how our heuristic obtains good quality solutions across a range of demand distributions. The four boxplots on the left show how solution quality remains strong even after increasing the number of workload scenarios. As the number of scenarios increases, this introduces more variables and constraints into the integer program, and would intuitively lead to a more difficult problem to solve. However, increasing the number of scenarios from 5 to 20 did not make a significant difference in solution quality.

The four boxplots on the right of Figure 4 show how the true integrality gap improves as the standard deviation of the underlying demand distribution increases from 75 to 200. This might be counterintuitive for some. But this could be explained by the fact that the instances with a high standard deviation have much worse, worst-case scenarios. For these instances, the choice of when to rely on unreserved on call and reserved on call options might be more obvious.

The formulations of all of these instances are tightened using the technique described in subsection 3.1. Tightening the constraints in this fashion allows us to reduce the true gap by a little over 1% on average.

5.3 Solution Quality with Time

In the numerical results presented in subsection 5.2, the algorithm is terminated after three minutes. To gauge how these solutions improve with time, we generate 12 test instances with a mix of model parameters, and allow the solver to run for an hour to find a solution. Over this period, the incumbent solution, relative integrality gap, and time are logged.

For these 12 instances, the solution progress is summarized in Table 4. Note that on average, 60 minutes of solver run time only brings the solution 0.58% closer to the lower bound than the solution after 3 minutes. An alternate view is given in Figure 5, which shows the worst case behavior of these 12 instances over the hour. The dotted curve (furthest from the origin) represents the maximum of the solution gaps for our 12 instances at each point in time. To better understand how the majority of instances progress, the dashed and solid curves illustrate the maximum gap disregarding the worst case, and two worst cases, respectively.

Figure 5 shows that given enough time, the relative integrality gap for the restricted master problem will approach zero, which implies the true integrality gap will approach a static lower bound. Thus, much of the true integrality gap that exists after three minutes is either due to the looseness of the lower bound or to the fact that a full branch-and-price implementation would have added more columns at a sub-node of the branch-and-price tree. In other words, our heuristic is likely not overlooking substantially better solutions due to the fact that the restricted master problem is terminated early.

Inst	1 min	2 mins	3 mins	5 mins	10 mins	15 mins	30 mins	60 mins
Inst 1	6.82%	6.80%	4.70%	4.70%	4.70%	4.70%	4.70%	4.70%
Inst 2	6.57%	6.54%	6.38%	6.34%	4.85%	4.85%	4.83%	4.44%
Inst 3	4.05%	4.05%	4.05%	4.05%	3.84%	3.84%	3.84%	3.84%
Inst 4	5.39%	3.84%	3.84%	3.84%	3.84%	3.84%	3.84%	3.84%
Inst 5	6.79%	5.34%	4.80%	4.79%	4.76%	4.76%	4.76%	4.69%
Inst 6	6.69%	6.69%	6.69%	6.51%	5.59%	4.82%	4.67%	4.63%
Inst 7	6.60%	5.96%	4.33%	4.32%	4.32%	4.32%	4.26%	4.22%
Inst 8	4.06%	4.06%	4.02%	4.02%	4.02%	4.02%	4.02%	3.85%
Inst 9	6.39%	4.78%	4.78%	4.69%	4.69%	4.65%	4.46%	4.46%
Inst 10	4.57%	4.57%	4.51%	4.51%	4.27%	4.27%	4.27%	4.17%
Inst 11	3.98%	3.95%	3.95%	3.90%	3.90%	3.90%	3.82%	3.82%
Inst 12	5.21%	5.21%	5.20%	4.91%	3.64%	3.62%	3.59%	3.59%
Average	5.59%	5.15%	4.77%	4.72%	4.37%	4.30%	4.26%	4.19%

Table 4 Behavior of the solution gap, versus the time limit imposed on the solver.

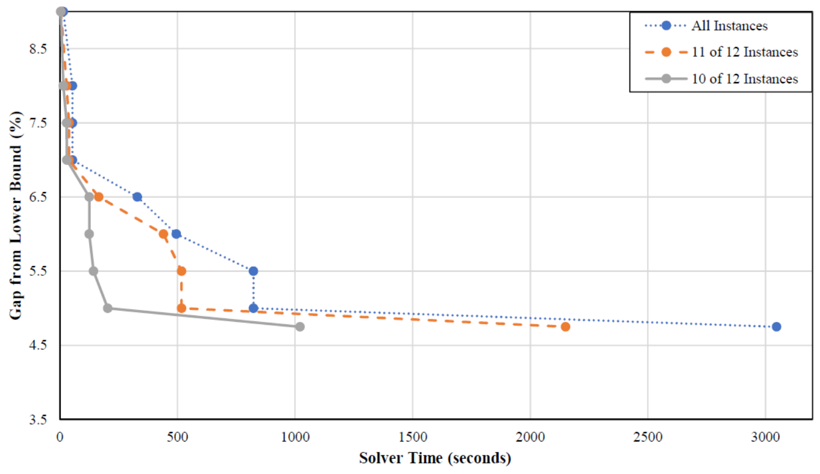

Fig. 5 Illustrating the worst-case behavior of the solution quality, as the solver run time is increased.

5.4 Measuring the Value Added by Stochastic Programming

Stochastic programming requires more effort to implement than deterministic programming, and the additional effort is not always worth the additional increase in solution quality. Following an example set by Birge 1995 [10], we conduct an experiment to measure the value added by stochastic programming over a natural deterministic approximation approach. This subsection describes this experiment and its results.

This experiment focuses on a previous version of the problem that contains only one on call staffing option that does not require an upfront reservation cost. For the deterministic approximation, we replace the random variable modeling the volume of alerts for each shift, with a single point estimate. To

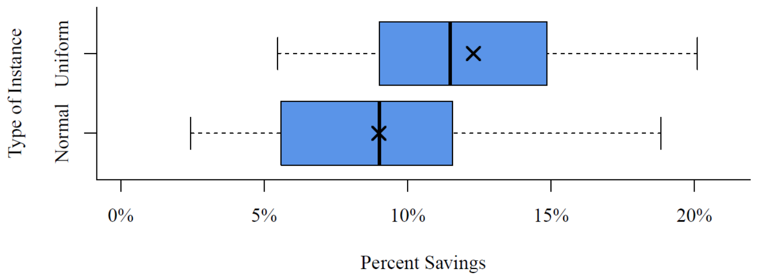

Fig. 6 Boxplots showing the percent savings obtained by stochastic programming relative to the deterministic approximation over a set of 40 test instances. Half of these instances were generated using a normal demand distribution, and the other half used a uniform distribution. The black "x" indicates the average savings over the 20 specific test instances.

choose the point estimate for each shift, we choose the target demand that minimizes the expected staffing costs for that shift, assuming that shift is staffed independently of all other shifts[1].

We implement both models in Python 3.5 and use Gurobi 6.0.4 to solve each. To ensure both methods are compared by the same objective, we plug the optimal staffing plan from the point estimate approach into the stochastic programming objective function. Both methods are allowed to run for up to three minutes for each instance and the relative gap tolerance was set to 1%.

This experiment is performed on 20 instances based on a discrete uniform distribution, and on 20 instances based on a discrete normal distribution. To ensure we are evaluating the two methods according to the same standard, we compute the stochastic programming objective value that would be obtained by the deterministic approximation if its solution was used to fix the first stage variables of the stochastic programming formulation, and we compare that value to the actual objective value obtained by the stochastic programming method. Based on these two values, we compute the percent savings in terms of expected staffing costs and present them in a boxplot in Figure 6.

In all cases, the stochastic programming method leads to noticeable savings over the alternative. The average savings is over 10% and in a few cases the stochastic programming method leads to savings of nearly 20%. The benefits of a stochastic programming approach are clear.

One of the reasons why the stochastic programming approach outperforms the deterministic approximation is because the former method is better at more

[1] Shifts are clearly not staffed independently, but we assume they are here for the purposes of deriving a simple heuristic.

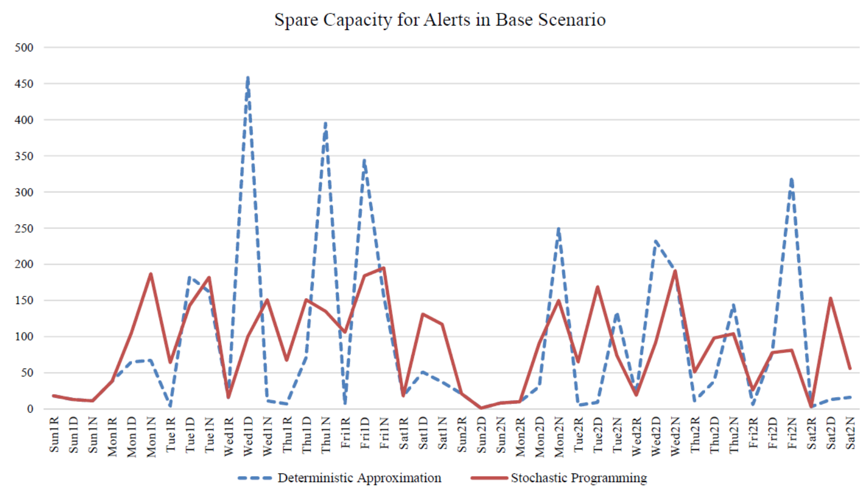

Fig. 7 Illustrating how the stochastic programming approach produces solutions that more evenly distribute excess capacity.

evenly distributing excess capacity—that is, the total number of alerts that the available staff can analyze minus the number of alerts in the base scenario. The objective function in the deterministic approach gives no incentive to distribute excess capacity evenly, and so some shifts may have a large excess capacity relative to others. In contrast, the objective function of the stochastic program drives the solution to spread out spare capacity more evenly, to prepare for scenarios with unusually high workloads.

Figure 7 illustrates this phenomenon for a single instance. The horizontal axis lists each shift and the vertical axis is the volume of alerts that can be handled by the analysts. The solid line corresponds to the excess capacity for the staffing plan generated by our stochastic programming heuristic. The dashed line corresponds to the excess capacity for the staffing plan generated by the deterministic approximation. Note that the stochastic programming solution tends to be smoother in contrast to the other solution, which has much greater spikes. These spikes correspond to excess capacity that was not strategically allocated in anticipation of above average workload scenarios.

6 Conclusions and Future Work

We presented a two-stage stochastic integer programming model for optimizing staffing and shift scheduling decisions in light of unpredictable, highly variable workloads and on call staffing options. Our main focus is workforce optimization decisions for cybersecurity operations centers, but our methods can be extended to other applications with similar features. Furthermore, we presented a column-generation-based heuristic for solving these problems. We demonstrated that our method quickly produces solutions that are close to a lower bound of the true optimal solution. We also demonstrate that our stochastic programming approach is superior to a deterministic approach to the same problem, and can cut expected staffing costs by nearly 20%.

The general approach of this paper could be extended in a number of directions. First, it can be extended to a problem with multiple alert types. Some alerts can be analyzed by any analyst. Others might require an analyst with a specialized skill set. Thus, to be adequately prepared, a cybersecurity operations center may need to have a good mix of skill sets on staff, in addition to a desirable mix of seniority levels. If the staffing for each alert type cannot be decoupled from one another, then this creates a more difficult workforce optimization problem. This is addressed in a concurrent paper [1].

This work can also be extended to incorporate many other considerations. These include the possibility of overlapping shifts to aid smoother work transitions, options to allow work to be completed in subsequent shifts, options for part-time employees, and additional constraints on the availability of on call options. This work can also be extended to consider a much longer time horizon rather than a two week pay period where it is important to get regular, fairly predictable schedules for employees.

Lastly, future work could explore combining the ideas in this paper with those for dynamically bringing in on call analysts from the Ganesan et al. [17] paper. That is, a two-stage model that considers a wide range of possible demand scenarios while planning the schedule for full-time employees, but also considers an optimal policy for deploying analysts during the execution of the schedule. This would require a different problem structure—one that gives the planner a choice on the shifts in which on call options can be exercised.

7 Appendix

7.1 Reduced Costs for Pricing Subproblem:

To calculate the reduced costs for the pricing subproblem, we need to define some notation. The following symbols denote dual variables for the constraints of an LP relaxation of the restricted master problem.

Dual Variables:

1. $\pi^{demand}_{s,d,k}$:= the dual variable associated with the demand constraint for scenario k for shift s on day d
2. $\pi^{min\%}_{\ell}$:= the dual variable associated with the minimum overall full-time analyst percent constraint for analysts of level ℓ

3. $\pi_{s,d}^{junratio}$:= the dual variable associated with the junior-to-more-senior analyst ratio constraint for shift s on day d
4. $\pi_{s,d,k}^{senratio}$:= the dual variable associated with the senior-to-principal analyst ratio constraint for shift s on day d
5. $\pi^{noredeye}$:= the dual variable associated with the constraint that enforces that principal analysts are not assigned red eye shifts

Let $y_{s,d}^*$ denote an optimal solution to a pricing subproblem. Using these dual variables, the reduced cost for a new column corresponding to shift schedule p and junior level analyst ℓ_1 can be expressed as follows:

$$\bar{c}_{\ell_1,p} = cost_{\ell_1}^{full} - \pi_{s,d,k}^{demand}\ rate_{\ell_1}\ y_{s,d}^* - \pi_{\ell_1}^{min\%} + \sum_{\ell in L} \pi_\ell^{min\%}\ minpercent_\ell - \pi_{s,d}^{junratio}\ y_{s,d}^* \tag{19}$$

Similarly, the reduced cost for a new column corresponding to shift schedule p and senior level analyst ℓ_2 is:

$$\begin{aligned}\bar{c}_{\ell_2,p} = cost_{\ell_2}^{full} - \pi_{s,d,k}^{demand}\ rate_{\ell_2}\ y_{s,d}^* - \pi_{\ell_2}^{min\%} \\ + \sum_{\ell\in L} \pi_\ell^{min\%}\ minpercent_\ell + 3\ \pi_{s,d}^{junratio}\ y_{s,d}^* - \pi_{s,d}^{senratio}\ y_{s,d}^*\end{aligned} \tag{20}$$

Likewise, the reduced cost for a new column corresponding to shift schedule p and principal analyst ℓ_3 is:

$$\begin{aligned}\bar{c}_{\ell_3,p} = cost_{\ell_3}^{full} - \pi_{s,d,k}^{demand}\ rate_{\ell_3}\ y_{s,d}^* - \pi_{\ell_3}^{min\%} \\ + \sum_{\ell\in L} \pi_\ell^{min\%}\ minpercent_\ell + 6\ \pi_{s,d}^{junratio}\ y_{s,d}^* + 5\ \pi_{s,d}^{senratio}\ y_{s,d}^* - \pi^{noredeye}\ y_{s,d}^*\end{aligned} \tag{21}$$

7.2 Objective Coefficients for Pricing Subproblem:

We now discuss how these reduced costs can be used to compute the objective coefficients for the pricing subproblem. Recall from linear programming theory that a non-basic column will want to enter the basis of the current basic feasible solution to the LP relaxation of the restricted master problem (which is a minimization problem) if and only if it has a negative reduced cost [9]. We can rewrite the reduced cost equation for $\bar{c}_{\ell_1,p}$ to:

$$\bar{c}_{\ell_1,p} = cost_{\ell_1}^{full} - \pi_{\ell_1}^{min\%} + \sum_{\ell\in L} \pi_\ell^{min\%}\ minpercent_\ell - y_{s,d}\left(\pi_{s,d,k}^{demand}\ rate_{\ell_1} + \pi_{s,d}^{junratio}\right) \tag{22}$$

Hence, finding the non-basic column that minimizes $\bar{c}_{\ell_1,p}$ is equivalent to solving the pricing subproblem with an objective function of:

$$\sum_{s\in S}\sum_{d\in D} y_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}\right) \tag{23}$$

Thus, using our previous notation for the pricing subproblem objective coefficients from Section 4.1, $\pi_{s,d} = \pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}$.
That being stated, a non-basic column corresponding to a junior analyst can enter the basis if:

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}\right) > cost^{full}_{\ell_1} - \pi^{min\%}_{\ell_1} + \sum_{\ell\in L}\pi^{min\%}_{\ell}\ minpercent_{\ell} \tag{24}$$

Similarly, it can be shown that a non-basic column corresponding to a senior analyst can enter the basis if:

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_2} + \pi^{senratio}_{s,d} - 3\ \pi^{junratio}_{s,d}\right) > cost^{full}_{\ell_2} - \pi^{min\%}_{\ell_2} + \sum_{\ell\in L}\pi^{min\%}_{\ell}\ minpercent_{\ell}, \tag{25}$$

and that a non-basic column corresponding to a principal analyst can enter the basis if:

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_3} + \pi^{noredeye} - 6\ \pi^{junratio}_{s,d} - 5\ \pi^{senratio}_{s,d}\right) > cost^{full}_{\ell_3} - \pi^{min\%}_{\ell_3} + \sum_{\ell\in L}\pi^{min\%}_{\ell}\ minpercent_{\ell} \tag{26}$$

□

References

1. Altner, D.S., Mason, E., & Servi, L.D., (2017) Scheduling and training multi-skilled analysts under uncertainty, *Submitted.*
2. Al-Yakoob, S.M., & Sherali, H.D., (2007). A column generation approach for an employee scheduling problem with multiple shifts and work locations, *Journal of the Operational Research Society*, 59(1), 34-43.
3. Aykin, T., (1996). Optimal shift scheduling with multiple break windows, *Management Science*, 42(4), 591-602.
4. Baker, S., Palekar, U., Gupta, G., Kale, L., Langer, A., Surina, M., & Venkataraman, R., (2012). Parallel computing for DoD airlift allocation, MITRE Corporation Technical Report.
5. Bard, J.F., Binici, C., & de Silva, A.H., (2003). Staff scheduling at the United States Postal Service, *Computers & Operations Research*, 30(5), 745-771.
6. Bard, J.F., Morton, D.P., & Wang, Y.M., (2007). Workforce planning at USPS mail processing and distribution centers using stochastic optimization, *Annals of Operations Research*, 155(1), 51-78.
7. Bard, J.F., & Purnomo, H.W., (2005). Preference scheduling for nurses using column generation, *European Journal of Operational Research*, 164(2), 510-534.

8. Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., & Vance, P.H., (1998). Branch-and-price: column generation for solving huge integer programs, *Operations Research*, 46(3), 316-329.
9. Bertsimas, D., & Tsitsiklis, J.N., (1997). *Introduction to Linear Optimization*, Athena Scientific.
10. Birge, J.R., (1995). Models and model value in stochastic programming, *Annals of Operations Research*, 59(1), 1-18.
11. Bodur, M., & Luedtke, J.R., (2016). Mixed-integer rounding enhanced Benders' decomposition for multiclass service system staffing and scheduling with arrival rate uncertainty, *Management Science*, 63(7), 2073-2091.
12. Çezik, T., Günlük, O., & Luss, H., (2001). An integer programming model for the weekly tour scheduling problem, *Naval Research Logistics*, 48(7), 607-624.
13. Chu, H.D., Gelman, E., & Johnson, E.L., (1997). Solving large scale crew scheduling problems, *European Journal of Operational Research*, 97, 260-268.
14. Côté, M.-C., Gendron, B., & Rousseau, L.-M., (2011). Grammar-based integer programming models for multiactivity shift scheduling, *Management Science*, 57(1), 151-163.
15. Ernst, A.T., Jiang, H., Krishnamoorthy, M., & Sier, D., (2004). Staff scheduling and rostering: a review of applications, methods and models, *European Journal of Operational Research*, 153, 3-27.
16. Ganesan, R., Jajodia, S., & Cam, H., (2016). Optimal scheduling of cybersecurity analysts for minimizing risk, *ACM Transactions on Intelligent Systems and Technology*,to appear.
17. Ganesan, R., Jajodia, S., Shah, A., & Cam, H., (2016). Dynamic scheduling of cybersecurity analysts for minimizing risk using reinforcement learning, *ACM Transactions on Intelligent Systems and Technology*, 8(1).
18. Gopalakrishnan, B., & Johnson, E.L., (2005). Airline crew scheduling: state of the art, *Annals of Operations Research*, 140(1), 305-337.
19. Gurvich, I., Luedtke, J.R., & Tezcan, T., (2010). Staffing call centers with uncertain demand forecasts: a chance-constrained optimization approach, *Management Science*, 56(7), 1093-1115.
20. Kall, P., & Wallace, S.W., (1994). *Stochastic Programming*, John Wiley & Sons.
21. Kao, E.P.C., & Queyranne, M., (1985). Budgeting costs of nursing in a hospital, *Management Science*, 31(5), 608-621.
22. Karuppiah, R., Martin, M., & Grossmann, I.E., (2010). A simple heuristic for reducing the number of scenarios in two-stage stochastic programming, *Computers & Chemical Engineering*, 34(8), 1246-1255.
23. Kim, K., & Mehrotra, S., (2015). A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management, *Operations Research*, 63(6), 1431-1451.
24. Linderoth, J., Shapiro, A., & Wright, S., (2006). The empirical behavior of sampling methods for stochastic programming, *Annals of Operations Research*, 142(1), 215-241.
25. Linderoth, J., & Wright, S., (2003). Decomposition algorithms for stochastic programming on a computational grid, *Computational Optimization and Applications*, 24(2), 207-250.
26. Mak, W.-K., Morton, D.P., & Wood, R.K., (1999). Monte Carlo bounding techniques for determining solution quality in stochastic programs, *Operations Research Letters*, 24(1), 47-56.
27. Mehrotra, A., Murphy, K.E., & Trick, M.A., (2000). Optimal shift scheduling: a branch and price approach, *Naval Research Logistics*, 47(3), 185-200.
28. Robbins, T.R., & Harrison, T.P., (2010). A stochastic programming model for scheduling call centers with global service level agreements, *European Journal of Operational Research*, 207(3), 1608-1619.
29. Vaidyanathan, B., Jha, K.C., & Ahuja, R.K., (2007). Multicommodity network flow approach to the railroad crew scheduling problem, *IBM Journal of Research and Development - Business Optimization*, 51(3), 325-244.
30. Zhu, X., & Sherali, H.D., (2009). Two-stage workforce planning under demand fluctuations and uncertainty, *Journal of the Operational Research Society*, 60(1), 94-103.