A Two-Stage Stochastic Shift Scheduling Model for Cybersecurity Workforce Optimization with On Call Options

Doug Altner*, Les Servi
Operations Research Department
The MITRE Corporation
McLean, Virginia 20111
{ daltner, lservi } @mitre.org

April 10, 2016

Abstract

This paper proposes a model and algorithm for optimizing staffing and shift scheduling decisions at a 24/7 cybersecurity operations center. We study a two-stage stochastic shift scheduling problem over a 14-day pay period with three shifts per day, several staffing and scheduling constraints, uncertain workloads and on call staffing options. We solve this problem with a heuristic simplification of branch-and-price whereby we only generate columns at the root node in the branch-and-bound tree. Our computational study shows this method quickly finds good quality solutions on instances with 10 independent workload scenarios for each shift and each day.

1 Introduction

One of the distinguishing challenges of shift scheduling in a cybersecurity operations context is highly variable workloads. Unlike manning retail stores or hospitals, the volume of incoming alerts is difficult to forecast and can vary greatly. Staffing too many analysts is expensive, as these are highly paid professionals. Staffing too few poses a security risk.

In this paper, we study optimizing staffing and shift scheduling decisions in a setting with unpredictable, highly variable workloads and on call staffing options. In addition to these features, our model considers three shifts per day, three analyst types, multiple per-shift staffing ratio constraints, overall staffing ratio constraints, and constraints limiting days of consecutive work, weekend shifts and red eye shifts. We model this problem as a two-stage stochastic integer program with integer recourse and solve it with a heuristic simplification of branch-and-price whereby we only generate columns at the root node in the branch-and-bound tree.

The main contributions of this paper are as follows. First, we offer a model for workforce optimization with unpredictable, highly variable workloads and on call staffing options. Our main focus is workforce management at 24/7 cybersecurity firms but our model can be extended

*Corresponding Author

to other manpower management problems with similar features. Second, we offer a column generation based heuristic that quickly computes good quality solutions, and can be used as the basis for a full branch-and-price implementation. Third, we show the clear advantages of a stochastic programming approach to this problem compared to a natural deterministic approximation, and discuss how these advantages stem from the fact that a stochastic programming approach spreads out excess manpower more evenly compared to deterministic approaches.

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 presents our two-stage stochastic programming model for the problem. Section 4 describes our methods for solving the problem. Section 5 presents our computational results, which shows the effectiveness of our heuristic as well as the value added of a stochastic programming approach over a deterministic approach. Section 6 suggests future work and discusses additional challenges of cybersecurity workforce optimization. Section 7 is the appendix, which discusses some details about our column generation pricing subproblem.

1.1 Literature Review

The literature on workforce scheduling and optimization is vast; see Ernst et al. 2004 [12] for a relatively recent survey. Complex shift scheduling problems have been studied for staffing nurses [5], call centers [9], railroad crews [18], airline crews [10], [14], and postal workers [3]. Researchers have explored developing sophisticated schedules with multiple varying breaks within a single day [2] as well as schedules covering two-week pay periods [4]. Papers have been published exploring branch-and-price options for shift scheduling (E.g., [1] and [23]) as well as using grammars for generating complex shift patterns and work rules (E.g., [11].)

Ganesan et al. 2016 [13] are the first to propose and study a staffing and shift scheduling problem for cybersecurity operations centers. They propose a deterministic model over a two-week pay period with two 12-hr shifts per day, three analyst types, staffing ratio constraints and constraints that limit consecutive days of work and weekend work. They split staffing and scheduling into two separate problems, heuristically approaching the former with genetic algorithms and the latter with a days off scheduling method.

There is also a growing number of papers on stochastic programming approaches to shift scheduling in general. Kao and Queyranne 1985 [19] 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 [4] 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 [25] and Bodur and Luedtke 2014 [8] study workforce optimization models in which the second-stage decisions entail assigning a specific workload to specific employees. Gurvich, Luedtke and Tezcan 2010 [15] offer a chance-constrained, stochastic programming approach to staffing a call center. Kim and Mehrotra 2015 [20] 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.

Our paper builds upon the work of Ganesan et al. [13] by extending their model to a two-stage stochastic variant with variable demand and on-call staffing options, integrating staffing and shift scheduling decisions into a single model, considering three shifts instead of two, and incorporating additional staffing and scheduling constraints such as shift-specific seniority ratios

as well as limits on red eye shifts. We also solve our model with a column generation based heuristic rather than genetic algorithms, and our method has the potential to be extended to an exact algorithm with a full branch-and-price implementation.

Our paper also complements the existing two-stage stochastic programming literature on staffing and shift scheduling by offering a different problem variant—one in which the first stage consists of earlier staffing and shift scheduling decisions, and the second stage consists of mitigating shortfalls in manpower with on call staffing options. We also demonstrate the practicality of such a two-stage model with multiple analyst types, three shifts per day, and a variety of staffing and scheduling constraints.

2 Problem Statement

In this section, we give a detailed, verbal description of the problem we are addressing.

We are modeling determining staffing needs and scheduling shifts at a 24/7 cybersecurity operations center. We are interested in identifying an optimal staffing plan for a single pay period, which is two weeks or fourteen days. Each day contains three possible shifts: 12am-8am (red eye), 8am-4pm (day shift), and 4pm-12am (night shift). We assume the volume of incoming security alerts varies according to day and time.

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, and there are constraints to ensure a certain mix of each seniority level is employed.

Our model is a two-stage stochastic programming problem. The first stage decision is to decide how many full-time analysts to hire for each possible shift schedule subject to a variety of staffing and scheduling constraints. The number and mix of analysts as well as their particular shift schedules must be decided before the volume of security alerts is known (although we assume a probability model for incoming security alerts is known.)

For each shift, the second stage decision is how many additional senior analysts to hire "on the spot," if any, to meet any shortfalls in manpower. For these decisions, the per shift pay rate for an on call senior analyst is higher than the amortized per shift rate of a full-time senior analyst.

We assume the range of possible volumes of security alerts for each shift can be adequately modeled with a finite discrete probability distribution, which we assume is independent from shift to shift. The distribution should contain enough variance to reflect the reality that the volume of incoming alerts is unknown and difficult to forecast.

The objective of this problem is to minimize the expected payroll costs. This includes the salary rates of full-time analysts, wages paid to on-call analysts, and bonuses given to employees for working red eye shifts.

There are three 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, and 3.) ensure each shift schedule for each analyst satisfies a number of work rules.

For the demand satisfaction constraints, we assume the firm must meet any shortfall in full-time manpower with on call analysts. For best practice reasons, we also assume the firm wants to be able to analyze all alerts without using any on call staff at least 50% of the time, for each shift.

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. These staffing ratio constraints enable the more senior analysts to mentor and train the more junior analysts[1] These constraints only apply to full-time analysts, not on call 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 discretized stochastic integer programming formulation of this problem.

As mentioned in the previous section, we are assuming we can adequately model the number of incoming alerts at each shift with a finite discrete probability distribution that is independent from the distribution for other shifts. Therefore, we can model this problem as a two-stage stochastic programming problem with a finite number of scenarios, and each shift has a set of scenarios which are independent from the scenarios of all other shifts.

In this paper, 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* or the *recourse variables.* For an introductory tutorial in stochastic programming, we recommend Higle 2005 [17].

For this problem in particular, we assume each shift has a *base scenario*, which occurs more frequently than all other scenarios, and that the firm wants to field all alerts in each base scenario without using any on call staff.

[1]We are describing a specific problem with specific parameters. However, this specificity is merely meant to make the problem statement easier to digest and any of these parameters can easily be modified. Furthermore, we are not providing and cannot provide parameters which reflect actual staffing conditions at an actual cybersecurity operations center.

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:

- $cost_\ell$, the per pay period cost of having a full-time, level ℓ analyst work shift schedule p, including bonuses paid for red eye shifts.
- $spotcost_s$, the cost of hiring a senior analysts on call for one shift of type s.
- $alerts^k_{d,s}$, the number of incoming alerts on day d during shift s under scenario k.
- $prob^k_{d,s}$, 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.
- $k_0^{d,s}$, the base scenario for shift s on day d.

Parameters:

- $minpercent_\ell$, 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. $v^k_{d,s}$:= the number of senior analysts who should be hired on an on call basis 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:

$$\begin{aligned} Minimize \quad & \textstyle\sum_{\ell \in L} \sum_{p \in \mathcal{P}} cost_{\ell,p}\ x_{\ell,p} \\ & + \textstyle\sum_{d \in D, s \in S} \sum_{k \in K_{d,s}} prob^k_{d,s}\ spotcost_s\ v^k_{d,s} \end{aligned} \tag{1}$$

Subject to :

$$
\begin{aligned}
& rate_{\ell_2}\ v^k_{d,s} + \textstyle\sum_{p\in\mathcal{P}} \sum_{\ell\in L} \lambda^p_{s,d}\ rate_\ell\ x_{\ell,p} \geq alerts^k_{d,s} && \forall\ k \in K_{d,s}\backslash\{k_0^{d,s}\},\ \forall\ s \in S, \forall\ d \in D \\
& \textstyle\sum_{p\in\mathcal{P}} \sum_{\ell\in L} \lambda^p_{s,d}\ rate_\ell\ x_{\ell,p} \geq alerts^{k_0^{d,s}}_{d,s} && \forall\ s \in S, \forall\ d \in D \\
& \textstyle\sum_{p\in\mathcal{P}} x_{\ell,p} \geq minpercent_\ell(\sum_{p\in\mathcal{P}} \sum_{\ell\in L} x_{\ell,p}) && \forall\ \ell \in L \\
& \textstyle\sum_{p\in\mathcal{P}} \lambda^p_{s,d}\ x_{\ell_1,p} \leq \sum_{p\in\mathcal{P}} \lambda^p_{s,d}\big(3x_{\ell_2,p} + 6x_{\ell_3,p}\big) && \forall\ s \in S, \forall\ d \in D \\
& \textstyle\sum_{p\in\mathcal{P}} \lambda^p_{s,d}\ x_{\ell_2,p} \leq 4 + \sum_{p\in\mathcal{P}} 5\lambda^p_{s,d} x_{\ell_3,p} && \forall\ s \in S, \forall\ d \in D \\
& \textstyle\sum_{p\in\mathcal{P}} \sum_{d\in D} \lambda^p_{s_1,d} x_{\ell_p,p} = 0 && \\
& x_{\ell,p} \in \{0\} \cup \mathbb{Z}^+ && \forall\ \ell \in L, \forall\ p \in \mathcal{P} \\
& y^k_{d,s} \in \{0\} \cup \mathbb{Z}^+ && \forall\ k\ \in K_{d,s},\ \forall\ d \in D, \forall\ s \in S
\end{aligned}
\tag{2}
$$

The objective function is to minimize the total expected payroll cost. The first part of the objective function captures the costs of the full-time analysts, including bonuses paid for working red eye shifts. The second part of the objective function is a discretization of the expected costs of hiring on call analysts.

The first set of constraints ensures that, for each shift, for every scenario for that shift that is not the base scenario, the number of alerts that the staff can handle is greater than or equal to the number of alerts that will arrive under this scenario. The second set of constraints ensures that all alerts in each base scenario for each shift can be analyzed without using any on call staff. The third set of constraints guarantees a minimum percentage of each type of analyst on staff. The fourth set of constraints 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. The fifth set ensures there is one full-time principal analyst for every five full-time senior analysts, for each shift. The sixth set prevents principal analysts from working red eye shifts. The last two sets of constraints define the variables as non-negative integers.

All of the work rule 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 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 likely make it computationally infeasible to explicitly add decision variables for every single scenario to the master problem formulation. Instead, the problem would likely need to be addressed with a sampling approximation approach. (E.g., see [22] and [21].)

3.1 Tightening the Master Problem Formulation

In this subsection, we describe a simple rounding technique to tighten the aforementioned formulation that does not require any significant computational effort.

Let the first two sets of constraints in formulation 2 be 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:

$$\begin{aligned} &\tfrac{rate_{\ell_2}}{rate_{gcd}}\ v^k_{d,s} + \textstyle\sum_{p\in\mathcal{P}}\sum_{\ell\in L}\lambda^p_{s,d}\ \tfrac{rate_\ell}{rate_{gcd}}\ x_{\ell,p} \geq \left\lceil \tfrac{alerts^k_{d,s}}{rate_{gcd}} \right\rceil && \forall\ k \in K_{d,s}\backslash\{k_0^{d,s}\}, \\ &&& \qquad \forall\ s \in S, \forall\ d \in D \\ &\textstyle\sum_{p\in\mathcal{P}}\sum_{\ell\in L}\lambda^p_{s,d}\ \tfrac{rate_\ell}{rate_{gcd}}\ x_{\ell,p} \geq \left\lceil \tfrac{alerts^{k_0^{d,s}}_{d,s}}{rate_{gcd}} \right\rceil && \forall\ s \in S, \forall\ d \in D \end{aligned} \tag{3}$$

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 cybersecurity 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. For an overview of branch-and-price, we recommend the instant classic paper by Barnhart et al. 1998 [6].

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. We believe this is a useful heuristic simplification that is easier to implement, likely runs faster and quickly produces good quality solutions. Our pricing framework can also be used as the basis for a full branch-and-price implementation, if needed.

We use the 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 alternatively be 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.* For background material on column generation, we recommend the textbook by Winston and Venkataramanan [24].

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 generated 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. This pricing subproblem can also be used for a full implementation of an exact branch-and-price algorithm for this 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.

$$Maximize \quad \sum_{s\in S} \sum_{d\in D} \pi_{s,d}\, y_{s,d} \tag{4}$$

$$\begin{aligned} Subject\ to: \\ & \sum_{\bar{s}=s}^{s+2} y_{\bar{s},d} \leq 1 && \forall\ s \in S, \forall\ d \in D \\ & \sum_{d\in W_i} \sum_{s\in S} y_{s,d} = 5 && \forall\ i \in \{1,2\} \\ & \sum_{\bar{d}=d}^{d+5} \sum_{s\in S} y_{s,\bar{d}} \leq 5 && \forall\ d \in D \\ & \sum_{s\in S} y_{s,d} \leq z && \forall\ d \in \{Sa1, Su2\} \\ & \sum_{s\in S} y_{s,d} \leq 1 - z && \forall\ d \in \{Su1, Sa2\} \\ & \sum_{d\in W_i} y_{s_1,d} \leq 2 && \forall\ i \in \{1,2\} \\ & \sum_{d\in D} y_{s_1,d} \leq 3 \\ & y_{s,d} \in \{0,1\} && \forall\ s \in S, \forall\ d \in D \\ & z \in \{0,1\} \end{aligned} \tag{5}$$

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

The first set of constraints ensures the constructed shift schedule contains at most one shift per 24-hour period. The second set 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 third set ensures the corresponding shift schedule contains shifts in

		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.

at most five consecutive days. The fourth and fifth constraints collectively ensure the shift schedule will include at least one weekend off. The next two constraint sets 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 set defines the decision variables as binary variables.

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.

To construct an initial solution, we greedily choose shift schedules that cover the maximum amount of uncovered alerts until all alerts are covered. 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 algorithm and let $\mathcal{P}$ be the set of all feasible shift schedules described by the constraints of the pricing subproblem. We iteratively find the shift schedule described by the same $y_{s,d}$ variables that maximize $\sum_{s\in S,d\in D} u^i_{s,d}\ y_{s,d}$ subject to $y \in \mathcal{P}$. We then choose a step size, iteratively update the uncovered alerts, and repeat the process until all alerts are covered.

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.

To choose a step size for the greedy algorithm, let $y^*_{s,d}$ be the optimal solution to the subproblem for this iteration, and let $\Delta = min_{s\in S,d\in D}\{u^i_{s,d}\ y^*_{s,d} : u^i_{s,d}\ y^*_{s,d} > 0\}$. Δ is our step size, and we subtract Δ from the number of uncovered alerts for each shift during each iteration. We continue the algorithm until we reach an iteration i such that $u^i_{s,d} = 0$ for each $s \in S$ and $d \in D$. For the example in Table 1, the step size 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.

Algorithm 1 contains pseudocode for our greedy algorithm, which produces an initial list of shift schedules but doesn't specify the number of analysts of each level needed to construct an

Algorithm 1: Greedy algorithm for constructing initial columns

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

initial feasible solution.

To start our column generation based heuristic, we create three columns in our initial restricted master problem, one for each analyst level. There are several ways to construct an initial basic feasible solution from this initial set of columns. One approach is to choose a sufficient number of junior analysts to cover the alerts, and then iteratively add senior and principal analysts until all analyst mix constraints are satisfied. In principle, this approach must be modified if the workload for analysts working red eye constraints creates an infeasible situation when combined with the constraints that forbid principal analysts from taking red eye shifts. However, in practice, we never needed to devise such a modification.

4.3 Heuristic Simplification of Branch-and-Price

To quickly and heuristically produce good quality solutions, we only generated columns at the root node of our branch-and-bound tree. This means that it is theoretically possible that a better solution could be found if columns were also generated at some of the sub-nodes of the branch-and-bound tree. But this also means the sub-nodes would be slower to explore, meaning that fewer would be processed if the user needs an answer within a few minutes, which in turn means the best solution found at the time of termination may not be as good. Our heuristic should be of value to developers who need good solutions quickly, but a full branch-and-price may be preferable to developers who are not constrained by running time or development time.

Algorithm 2 summarizes our heuristic for solving our stochastic shift scheduling problem. When we generated columns for the restricted master problem for step 2, we had three pricing subproblems that we checked whenever we considered adding new columns—one for each analyst level. Whenever a new shift schedule column was added during this stage, we also generated the identical shift schedule columns for the other two analyst levels.

Note that even if the integer program solved in step 3 of Algorithm 2 is solved exactly, this still might not be the optimal solution to the original problem because the full set of decision variables was not considered; we only used the decision variables (i.e., columns) generated by steps 1 & 2 of the algorithm.

Algorithm 2: Heuristic method for stochastic shift scheduling problem

begin

1. Generate an initial set of columns.

2. Starting with the initial set of columns, solve the LP relaxation of the restricted master problem using column generation.

3. Resolve a restricted master problem as an integer program, using only the columns generated in steps 1 & 2.

end

	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 5% of a lower bound of the true optimal solution. The third section illustrates how our stochastic programming approach clearly outperforms a natural deterministic approximation to the same problem, sometimes cutting expected staffing costs by nearly 20%. This section also offers a causal explanation for the superiority of the stochastic programming approach, discussing how it is more effective at evenly allocating spare manpower.

5.1 Generating Instances

We randomly generated forty test instances for these experiments. In all of our test instances, we assumed 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. On call analysts can also process 60 alerts per 8-hour shift and cost \$800 per shift (so they would cost \$8,000 if they were to work the standard 10 8-hour shifts in a pay period.) All analysts receive a 5% bonus for each red eye shift they work. Table 2 summarizes the analyst attributes.

We tested two classes of instances. For the first class of instances, the number of alerts for day d and shift s, $alerts_{d,s}$, is a discrete random variable with the following 10 scenarios: Scenario 1, the base scenario, occurs 50% of the time and entails X alerts arriving. Scenarios 2-10 involve $X+30, X+60, \ldots, X+270$ alerts arriving respectfully, and each occurs with probability $\frac{1}{18}$. To generate these scenarios for our instance, we drew X from the shift-specific uniform distribution listed in Table 3. (X is a data point for each stochastic programming instance, and is not a random variable in the model.)

For the second class of instances, we modeled the number of alerts as a truncated, discretized

	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

normal random variable. To generate these instances, we randomly selected the mean of the discretized normal distribution from the corresponding shift-specific uniform distribution listed in Table 3. We always selected a standard deviation of 100 alerts. The volume of alerts for the first scenario (i.e., the base scenario) is the mean of the distributed before truncation. The volume of alerts for the ith scenario, for each i in $\{2, 3, \ldots, 10\}$, is the number such that the probability of receiving fewer than this number of alerts with a normal distribution with the same mean and standard deviation is $.5 + \frac{i}{9}$. So, for example, if the mean demand for a shift is 499 alerts, then the 10 possible demand scenarios are 499, 513, 527, 542, 558, 575, 596, 621, 658, and 808 alerts. The probability of scenario 1 occurring is $\frac{1}{2}$ and scenarios 2-10 each occur with probability $\frac{1}{18}$.

For the sake of brevity, we refer to the first class of instances as the *uniform* instances and the second class of instances as the *normal* instances.

5.2 Measuring Solution Quality

To get a sense of the quality of our solutions, we compared 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 or not this is a relatively tight bound is an open question for future research.

We ran our experiments on machine with 2 2.93 Ghz Intel processors and 2.00 GB of RAM. We implemented our model using Python 3.5 and used Gurobi 6.0.4 to solve all mathematical programs. The maximum amount of time the integer program was allowed to run was three minutes and the relative gap tolerance was set to 1%. We restricted the run time to three minutes to see how well this method would perform if it were part of a broader, optimization-based decision support system in which users want good quality answers in a few minutes.

Table 4 illustrates the results of this experiment on the uniform instances. Each row corresponds to a randomly generated instance. The first column contains the name of the instance. The second column contains the objective value of the best solution found within the three minutes of allowed run time. The third column contains the relative integrality gap of the restricted master program that we solved—that is, the integer program solved during step 3 of Algorithm 2, which is a restriction of the true master program described in Section 3. The fourth column presents the optimal objective value of the linear programming relaxation of true master problem, which serves as a true lower bound. The fifth column contains the true gap between the objective value of the best solution we found within three minutes and the true lower bound.

In all cases, our method found solutions to each of the uniform instances within at most 5% of the true lower bound in three minutes. Since the benchmark is a lower bound, it is an open

Inst.	Stoch. Obj. Val.	Res. Mas. Gap.	LB	TrueGap
Uniform01	$175,477.77	1.24%	$170,407.25	2.88%
Uniform02	$179,506.66	2.09%	$171,752.79	4.31%
Uniform03	$176,456.66	2.44%	$168,071.31	4.75%
Uniform04	$170,894.44	1.48%	$164,471.93	3.75%
Uniform05	$175,994.44	1.42%	$170,147.54	3.32%
Uniform06	$182,521.11	1.22%	$177,431.63	2.78%
Uniform07	$170,947.77	1.72%	$164,539.23	3.74%
Uniform08	$174,904.44	1.19%	$169,238.76	3.23%
Uniform09	$176,962.22	1.63%	$170,248.46	3.79%
Uniform10	$181,421.11	0.94%	$176,625.71	2.64%
Uniform11	$177,218.88	1.96%	$170,061.39	4.03%
Uniform12	$175,225.55	1.94%	$167,834.27	4.21%
Uniform13	$178,940.00	1.83%	$171,626.86	4.08%
Uniform14	$178,772.22	1.68%	$171,882.36	3.85%
Uniform15	$177,482.22	1.97%	$170,128.06	4.14%
Uniform16	$175,295.55	1.93%	$167,922.53	4.20%
Uniform17	$170,941.11	1.32%	$165,226.46	3.34%
Uniform18	$174,021.11	1.94%	$167,156.20	3.94%
Uniform19	$180,656.66	1.79%	$173,844.06	3.77%
Uniform20	$178,693.33	1.42%	$172,281.10	3.58%
Average:	$176,616.66	1.66%	$170,044.90	3.72%

Table 4: Measuring solution quality for the uniform instances

question whether this gap largely exists because there are better solutions that our heuristic did not yet find, or because the bound could be improved. Likewise, the relative integrality gap for the restricted master problem was always under 2.5% upon termination. So the restricted master problems were all almost solved exactly.

Table 5 shows the results of this experiment on the normal instances and the headers are the same as in Table 4. Comparable to the uniform instances, our heuristic method found solutions to each of the normal instances within 6% of the true optimal solution in three minutes. In addition, the relative integrality gap of the restricted master problem was roughly the same.

The formulations of all of these instances were tightened using the technique described in subsection 3.1. Tightening the constraints in this fashion allowed us to reduce the true gap from 6.1% on average to 4.7% on average.

5.3 Measuring Value Added by Stochastic Programming

In this subsection, we describe our experiments in which we compared our stochastic programming approach to a natural deterministic approximation approach. For the deterministic approximation, we replaced the random variable modeling the volume of alerts for each shift, with a single point estimate. To choose the point estimate for each shift, we chose the target demand that minimizes the expected staffing costs for that shift, assuming that shift was

Inst.	Stoch. Obj. Val.	Res. Mas. Gap.	LB	TrueGap
Normal01	$162,307.77	1.17%	$154,935.19	4.54%
Normal02	$167,512.22	1.96%	$159,375.46	4.85%
Normal03	$168,085.55	1.63%	$160,067.31	4.77%
Normal04	$168,194.44	1.40%	$161,186.69	4.16%
Normal05	$169,231.11	1.64%	$160,915.98	4.91%
Normal06	$168,611.11	2.72%	$158,492.83	6.00%
Normal07	$173,132.22	0.97%	$166,831.81	3.63%
Normal08	$169,911.11	1.73%	$161,695.24	4.83%
Normal09	$173,041.11	1.13%	$166,020.59	4.05%
Normal10	$170,876.66	1.54%	$162,720.92	4.77%
Normal11	$172,902.22	1.57%	$165,219.73	4.44%
Normal12	$166,141.11	1.30%	$158,892.27	4.36%
Normal13	$170,006.66	1.88%	$161,746.46	4.85%
Normal14	$169,440.00	1.38%	$162,211.87	4.26%
Normal15	$166,107.77	2.31%	$157,444.01	5.21%
Normal16	$166,344.44	1.42%	$158,426.01	4.76%
Normal17	$172,381.11	1.48%	$164,061.04	4.82%
Normal18	$168,047.77	2.16%	$158,663.27	5.58%
Normal19	$167,473.33	1.76%	$159,129.31	4.98%
Normal20	$172,612.22	1.27%	$166,221.66	3.70%
Average:	$169,118.00	1.62%	$161,212.88	4.67%

Table 5: Measuring solution quality for the normal instances

staffed independently of all other shifts[2].

As with our stochastic programming model, we implemented our deterministic approximation integer program in Python 3.5 and used Gurobi 6.0.4 to solve it. To ensure that both methods were compared by the same objective, we plugged the optimal staffing plan from the point estimate approach into the stochastic programming objective function. Both methods were allowed to run for up to three minutes for each instance and the relative gap tolerance was set to 1%.

Table 6 shows the results of this experiment on the uniform instances. Each row corresponds to a randomly generated instance. The first column contains the name of the instance. The second column contains the stochastic programming objective value of the best solution found within the three minutes by our stochastic programming heuristic. The third column contains the stochastic programming objective value of the best solution found within three minutes by the deterministic approximation approach. The fourth column calculates the percentage of savings, in terms of expected payroll cost, if the stochastic programming derived solution is used over the deterministic approximation derived solution.

In all cases, the stochastic programming method led to noticeable savings over the alternative. The average savings was around 12% and in some cases the stochastic programming method led to savings of nearly 20%. The benefits of a stochastic programming approach are clear.

[2]Shifts are clearly not staffed independently, but we assumed they were for the purposes of deriving a simple heuristic.

Inst.	Stoch. Obj. Val.	Det. Obj. Val.	Savings
Uniform01	$175,477.77	$216,382.22	18.90%
Uniform02	$179,506.66	$189,901.11	5.40%
Uniform03	$176,456.66	$192,815.55	8.40%
Uniform04	$170,894.44	$192,344.44	11.10%
Uniform05	$175,994.44	$203,411.11	13.40%
Uniform06	$182,521.11	$226,028.88	19.20%
Uniform07	$170,947.77	$188,886.66	9.40%
Uniform08	$174,904.44	$211,324.44	17.20%
Uniform09	$176,962.22	$193,495.55	8.50%
Uniform10	$181,421.11	$227,049.99	20.00%
Uniform11	$177,218.88	$201,001.11	11.80%
Uniform12	$175,225.55	$188,827.77	7.20%
Uniform13	$178,940.00	$194,393.33	7.90%
Uniform14	$178,772.22	$201,714.44	11.30%
Uniform15	$177,482.22	$199,618.88	11.00%
Uniform16	$175,295.55	$194,033.33	9.60%
Uniform17	$170,941.11	$201,266.66	15.00%
Uniform18	$174,021.11	$200,750.00	13.30%
Uniform19	$180,656.66	$204,383.33	11.60%
Uniform20	$178,693.33	$209,338.88	14.60%
Average:	$176,616.66	$201,848.38	12.24%

Table 6: Comparing stochastic programming to a deterministic approximation on the uniform instances

Inst.	Stoch. Obj. Val.	Det. Obj. Val.	Savings
Normal01	$162,307.77	$171,278.88	5.20%
Normal02	$167,512.22	$189,904.44	11.70%
Normal03	$168,085.55	$189,233.33	11.10%
Normal04	$168,194.44	$189,732.22	11.30%
Normal05	$169,231.11	$177,728.88	4.70%
Normal06	$168,611.11	$172,837.77	2.40%
Normal07	$173,132.22	$203,625.55	14.90%
Normal08	$169,911.11	$182,290.00	6.70%
Normal09	$173,041.11	$200,455.55	13.60%
Normal10	$170,876.66	$179,975.55	5.00%
Normal11	$172,902.22	$197,327.77	12.30%
Normal12	$166,141.11	$184,660.00	10.00%
Normal13	$170,006.66	$181,468.88	6.30%
Normal14	$169,440.00	$186,146.66	8.90%
Normal15	$166,107.77	$179,411.11	7.40%
Normal16	$166,344.44	$176,862.22	5.90%
Normal17	$172,381.11	$180,423.33	4.40%
Normal18	$168,047.77	$184,815.55	9.00%
Normal19	$167,473.33	$184,646.66	9.30%
Normal20	$172,612.22	$212,658.88	18.80%
Average:	$169,118.00	$186,274.16	8.95%

Table 7: Comparing stochastic programming to a deterministic approximation on the normal instances

Table 7 contains the computational results on the normal instances and the headers are the same as in Table 6. The savings for these instances are just as pronounced as with the uniform instances. The average savings was around 9% and in some cases the stochastic programming method led to savings of nearly 19%.

One of the reasons why the stochastic programming approach outperformed the deterministic approximation is because the former method is better at more evenly distributing excess manpower—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 manpower evenly, and so some shifts might have a large excess capacity relative to others. In contrast, the objective function of the stochastic program drives the solution to spread out spare manpower more evenly, to prepare for scenarios with unusually high workloads.

Figure 1 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 manpower. The red line corresponds to the excess manpower for the staffing plan generated by our stochastic programming heuristic. The blue line corresponds to the excess manpower 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 manpower that was not strategically allocated in anticipation of above average workload scenarios.

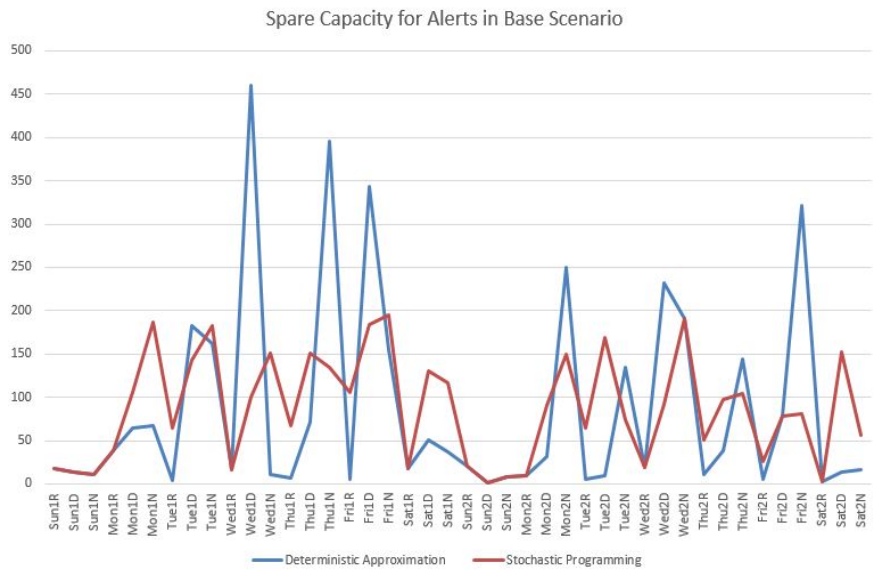

Figure 1: Illustrating how the stochastic programming approach produces solutions that more evenly distribute excess manpower.

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 manpower management 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. In reality, the security alerts that analysts receive can vary greatly. Some might be suspicious emails that are likely innocuous, even though there is a small chance that they pose a serious threat. Others might be indicative of a dire security breach. Likewise, alerts can vary in that they might require particular specialized skills to analyze. Some alerts can be analyzed by any analysts. Others might require an analyst with a particular 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.

The white paper on managing and growing cybersecurity analysts by Hewlett Packard [16] contains a few scheduling concerns which this work can be extended to include. Many overlapping shifts can be incorporated to ensure smooth work transitions when analysts arrive and

leave the center. This work can also be extended to suggest manpower management plans that guarantee regular, predictable schedules from pay period to pay period, options for part-time analysts versus full-time analysts, options and limitations for scheduling full-time analysts for overtime.

While all these issues require extensions to our model, our model can be naturally built upon to address them.

7 Acknowledgements

We'd like to thank our employer, the MITRE Corporation, for supporting this research with internal funding. We'd also like to thank MITRE's cybersecurity innovation area leader for suggesting the area workforce optimization of cybersecurity analysts as a topic to study further. Finally, we'd like to thank our colleague Matt Olson for providing feedback on an earlier draft of this manuscript.

8 Appendix

8.1 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. $\pi_{s,d,k}^{demand}$:= the dual variable associated with the demand constraint for scenario k for shift s on day d
2. $\pi_{\ell}^{min\%}$:= 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:

$$\bar{c}_{\ell_1,p} = cost_{\ell_1} - \pi_{s,d,k}^{demand}\ rate_{\ell_1}\ y_{s,d}^* - \pi_{\ell_1}^{min\%} + \sum_{\ell in L} \pi_{\ell}^{min\%}\ minpercent_{\ell} - \pi_{s,d}^{junratio}\ y_{s,d}^* \quad (6)$$

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

$$\bar{c}_{\ell_2,p} = cost_{\ell_2} - \pi^{demand}_{s,d,k}\ rate_{\ell_2}\ y^*_{s,d} - \pi^{min\%}_{\ell_2} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell} + 3\ \pi^{junratio}_{s,d}\ y^*_{s,d} - \pi^{senratio}_{s,d}\ y^*_{s,d} \quad (7)$$

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

$$\bar{c}_{\ell_3,p} = cost_{\ell_3} - \pi^{demand}_{s,d,k}\ rate_{\ell_3}\ y^*_{s,d} - \pi^{min\%}_{\ell_3} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell} + 6\ \pi^{junratio}_{s,d}\ y^*_{s,d} + 5\ \pi^{senratio}_{s,d}\ y^*_{s,d} - \pi^{noredeye}\ y^*_{s,d} \quad (8)$$

8.2 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 [7]. We can rewrite the reduced cost equation for $\bar{c}_{\ell_1,p}$ to:

$$\bar{c}_{\ell_1,p} = cost_{\ell_1} - \pi^{min\%}_{\ell_1} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell} - y_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}\right) \quad (9)$$

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:

$$\sum_{s\in S}\sum_{d\in D} y_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}\right) \quad (10)$$

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:

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_1} + \pi^{junratio}_{s,d}\right) > cost_{\ell_1} - \pi^{min\%}_{\ell_1} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell} \quad (11)$$

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

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_2} + \pi^{senratio}_{s,d} - 3\ \pi^{junratio}_{s,d}\right) > cost_{\ell_2} - \pi^{min\%}_{\ell_2} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell}, \quad (12)$$

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

$$y^*_{s,d}\left(\pi^{demand}_{s,d,k}\ rate_{\ell_3} + \pi^{noredeye} - 6\ \pi^{junratio}_{s,d} - 5\ \pi^{senratio}_{s,d}\right) > cost_{\ell_3} - \pi^{min\%}_{\ell_3} + \sum_{\ell in L} \pi^{min\%}_{\ell}\ minpercent_{\ell} \quad (13)$$

□

References

[1] Al-Yakoob, S.M., and Sherali, H.D., "A Column Generation Approach for an Employee Scheduling Problem with Multiple Shifts and Work Locations," *Journal of the Operational Research Society*, Vol. 79, p.34-43, 2008.

[2] Aykin, T., "Optimal Shift Scheduling with Multiple Break Windows," *Management Science*, Vol. 42(4), p. 591-602, 1996.

[3] Bard, J.F., Binici, C., and de Silva, A.H., "Staff Scheduling at the United States Postal Service," *Computers and Operations Research*, Vol. 30, p. 145-175, 2003.

[4] Bard, J.F., Morton, D.P., and Wang, Y.M., "Workforce Planning at USPS Mail Processing and Distribution Centers Using Stochastic Optimization," *Annals of Operations Research*, Vol. 155, p. 51-78, 2007.

[5] Bard, J.F., and Purnomo, H.W., "Preference Scheduling for Nurses Using Column Generation," *European Journal of Operational Research*, Vol. 164(2), p. 510-534, 2005.

[6] Barnhart, C., Johnson, E.L., Nemhauser, G.L., Savelsbergh, M.W.P., and Vance, P.H., "Branch-And-Price: Column Generation for Solving Huge Integer Programs," *Operations Research*, Vol. 46(3), p. 316-329, 1998.

[7] Bertsimas, D., and Tsitsiklis, J.N., *Introduction to Linear Optimization*, Athena Scientific, 1997.

[8] Bodur, M., and Luedtke, J.R., "Mixed-Integer Rounding Enhanced Benders Decomposition for Multiclass Service System Staffing and Scheduling with Arrival Rate Uncertainty," *Management Science*, to appear.

[9] Çezik, T., Günlük, O., and Luss, H., "An Integer Programming Model for the Weekly Tour Scheduling Problem," *Naval Research Logistics*, Vol. 48(7), p. 607-624, 2001.

[10] Chu, H.D., Gelman, E., and Johnson, E.L., "Solving Large Scale Crew Scheduling Problems," *European Journal of Operational Research*, Vol. 97, p. 260-268, 1997.

[11] Côté, M.-C., Gendron, B., and Rousseau, L.-M., "Grammar-Based Integer Programming Models for Multiactivity Shift Scheduling," *Management Science*, Vol. 57(1), p. 151-163, 2011.

[12] Ernst, A.T., Jiang, H., Krishnamoorthy, M., and Sier, D., "Staff Scheduling and Rostering: A Review of Applications, Methods and Models," *European Journal of Operational Research*, Vol. 153, p. 3-27, 2004.

[13] Ganesan, R., Jajodia, S., Shah, A., and Cam, H., "Optimal Scheduling of Cybersecurity Analysts for Minimizing Risk," *ACM Transactions on Intelligent Systems and Technology*, to appear. Slides available at `http://esorics2015.sba-research.org/wp-content/uploads/2014/11/ESORICS-Keynote-Sushil-Jajodia.pdf`. Accessed January 2016.

[14] Gopalakrishnan, B., and Johnson, E.L., "Airline Crew Scheduling: State of the Art," *Annals of Operations Research*, Vol. 140(1), p. 305-337, 2005.

[15] Gurvich, I., Luedtke, J.R., and Tezcan, T., "Staffing Call Centers With Uncertain Demand Forecasts: A Chance-Constrained Optimization Approach," *Management Science*, Vol. 56(7), p. 1093-1115, 2010.

[16] Hewlett Packard, "Growing the Security Analyst: Hiring, Training and Retention," Company white paper, `http://www8.hp.com/h20195/v2/getpdf.aspx/4AA5-3982ENN.pdf`, Accessed December 2015.

[17] Higle, J.L., "Stochastic Programming: Optimization When Uncertainty Matters," INFORMS TutORials in Operations Research, 2005.

[18] Huismann, D., "A Column Generation Approach for the Rail Crew Re-Scheduling Problem," *European Journal of Operational Research*, Vol. 180, p. 167-177, 2007.

[19] Kao, E.P.C., and Queyranne, M., "Budgeting Costs of Nursing in a Hospital," *Management Science*, Vol. 31(5), p. 608-621, 1985.

[20] Kim, K., and Mehrotra, S., "A Two-Stage Stochastic Integer Programming Approach to Integrated Staffing and Scheduling with Application to Nurse Management," *Operations Research*, 2015.

[21] Kleywegt, A.J., Shapiro, A., and Homem-De-Mello, T., "The Sample Average Approximation Method for Stochastic Discrete Optimization," *SIAM Journal on Optimization*, Vol. 12(2), p. 479-502, 2001.

[22] Mak, W.-K., Morton, D.P., and Wood, R.K., "Monte Carlo Bounding Techniques for Determining Solution Quality in Stochastic Programs," *Operations Research Letters*, Vol. 24, p. 47-56, 1999.

[23] Mehrotra, A., Murphy, K.E., and Trick, M.A., "Optimal Shift Scheduling: a Branch and Price Approach," *Naval Research Logistics*, Vol. 47(3), p. 185-200, 2000.

[24] Winston, W.L., and Venkataramanan, M., *Introduction to Mathematical Programming*, Duxbury Press, 2002.

[25] Zhu, X., and Sherali, H.D., "Two-Stage Workforce Planning Under Demand Fluctuations and Uncertainty," *Journal of the Operational Research Society*, Vol. 60(1), p. 94-103, 2009.