PCAM: A Data-Driven Probabilistic Cyber-Alert Management Framework

HAIPENG CHEN, Harvard University, USA
ANDREW DUNCKLEE, Clark University, USA
SUSHIL JAJODIA, George Mason University, USA
RUI LIU, Dartmouth College, USA
SEAN MCNAMARA, Dartmouth College, USA
V.S. SUBRAHMANIAN, Northwestern University, USA

We propose PCAM, a Probabilistic Cyber-Alert Management framework, that enables chief information security officers (CISOs) to better manage cyber-alerts. Workers in Cyber Security Operation Centers (CSOCs) usually work in 8 or 12 hour shifts. Before a shift, PCAM analyzes data about all past alerts and true alerts during the shift time-frame in order to schedule a given set of analysts in accordance with workplace constraints so that the expected number of "uncovered" true alerts (i.e. true alerts not shown to an analyst) is minimized. PCAM achieves this by formulating the problem as a bi-level non-linear optimization problem and then shows how to linearize and solve this complex problem. We have tested PCAM extensively. Using statistics derived from 44 days of real world alert data, we are able to minimize the expected number of true alerts that are not manually examined by a team consisting of junior, senior, and principal analysts. We are also able to identify the optimal mix of junior, senior, and principal analysts needed during both day and night shifts given a budget, outperforming some reasonable baselines. We tested PCAM's proposed schedule (from statistics on 44 days) on a further 6 days of data, using an off-the-shelf false alarm classifier to predict which alerts are real and which ones are false. Moreover, we show experimentally that PCAM is robust to various kinds of errors in the statistics used.

ACM Reference Format:
Haipeng Chen, Andrew Duncklee, Sushil Jajodia, Rui Liu, Sean McNamara, and V.S. Subrahmanian. xxxx. PCAM: A Data-Driven Probabilistic Cyber-Alert Management Framework. *ACM Trans. Internet Technol.* 1, 1, Article 1 (January xxxx), 23 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Most Cyber Security Operations Centers (CSOCs) are flooded by alerts. For instance, while speaking about the Sony breach in 2015, [13] says that "While the tools were able to identify the malicious

Authors listed in alphabetical order. Work by all authors (except Jajodia) was primarily done while they were at Dartmouth College.

Authors' addresses: Haipeng Chen, hpchen@seas.harvard.edu, Center for Research on Computation and Society, Department of Computer Science, Harvard University, Boston, MA, USA; Andrew Duncklee, aduncklee@clarku.edu, Information Technology & Consulting, Clark University, Wooster, MA, USA; Sushil Jajodia, jajodia@gmu.edu, Center for Secure Information Systems, George Mason University, Fairfax, VA, USA; Rui Liu, Rui.Liu.GR@dartmouth.edu, Dept. of Computer Science and Institute for Security, Technology, and Society, Dartmouth College, Hanover, NH, USA; Sean McNamara, Sean.R.McNamara@dartmouth.edu, Information Technology & Consulting, Dartmouth College, Hanover, NH, USA; V.S. Subrahmanian, Dept. of Computer Science and Buffet Institute for Global Affairs, Northwestern University, Evanston, IL, USA, vss@northwestern.edu..

1533-5399/xxxx/1-ART1 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

activity, those alerts were lost in a sea of 40,000 other alerts that same month." Other sources state that "The security operations center (SOC) is drowning in cybersecurity alerts"[1] — they go on to state that banks see over 100,000 alerts per day.

We propose PCAM, a framework that uses statistics to manage this flood of alerts, while satisfying workplace requirements within a given work shift. Figure 1 shows the architecture of PCAM. (i) As seen on the left side of Figure 1, before a shift starts, PCAM uses historical statistics to compute a schedule of work specifying the analysts' lunch and other breaks. Because each"true alert" must be examined by an analyst, PCAM tries to ensure that this schedule minimizes the expected number of "uncovered" true alerts. (ii) Then, as shown on the right of the figure, during a shift, security products generate alerts. PCAM dynamically uses an off the shelf classifier [6, 10, 14, 20] to predict if an alert is likely to be real ("true" alert) or not ("false" alert). "True" alerts are dynamically assigned to analysts for manual inspection in accordance with the probability (of being a true alert) returned by the classifier. False alarms are discarded.

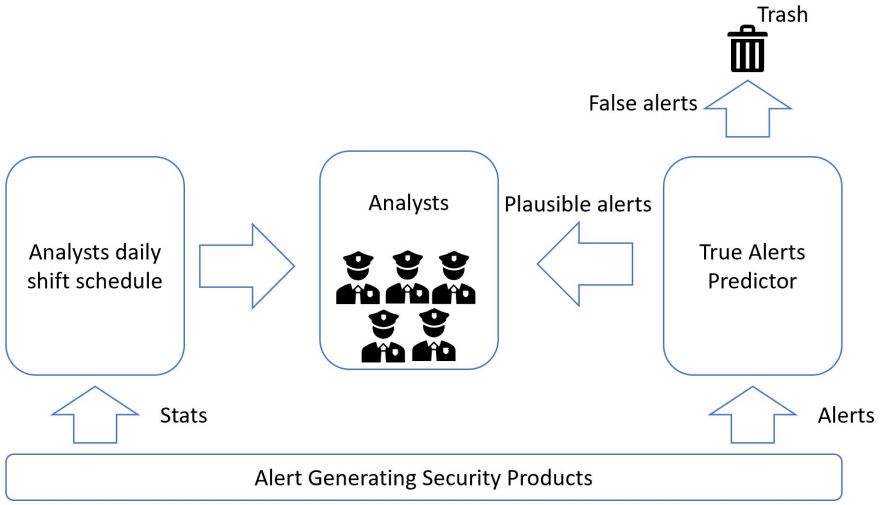


Fig. 1. PCAM Architecture

The organization of the paper and the main contributions are as follows. Section 2 describes explains the differences between past work and ours. Section 3 describes the data used by PCAM and provides a high level overview of how PCAM works. Section 4 describes the technical details behind PCAM. We formalize the problem of maximizing expected true alert coverage and show how it can be expressed via a non-linear bi-level integer programming. Section 5 addresses two major problems that make the above integer program challenging: nonlinear integer programs are hard to solve, and so are bilevel integer programs. We show how to linearize the problem and then additionally show how to modify the bilevel optimization into a single level optimization. This makes the problem amenable to solution by off the shelf integer program solvers. Section 7 describes our experimental validation of PCAM. We used 30 days of real-world data from Dartmouth College's Information Technology and Consulting (ITC) organization which maintains all of Dartmouth's networks and servers. We then used the real world alert data for the next 14 days to see how our system would perform if the schedule for the next shift was created at the end of the previous shift. In addition,

[1]https://bricata.com/blog/how-many-daily-cybersecurity-alerts/

we ran robustness experiments using simulated data whose statistics varied from those learned from the prior history. In all these cases, PCAM performed extremely well.

2 RELATED WORK

In this section, we introduce related works that study *security personnel scheduling for manual analysis of security alarms.*

The idea that managed security services need to optimize the allocation of analysts was studied in [11] - the paper developed a combination of game theory and probabilistic temporal logic to reason about scheduling analysts in the presence of an adversary who takes advantage of the fact that lots of security alerts might be false. The problem was addressed by Dunstatter et al. [4, 5] from a Markov games perspective using deep reinforcement learning. Wang et al. [19] presented a survey of game theoretic methods that have been applied to improve cyber security. They are focused on a game theoretic setting with an adversary, which is not the focus of this paper.

Ganesan et al. [8, 9] introduced the CSOC workforce optimization problem and used a reinforcement learning method to allocate the incoming alerts to security staff. This is extended in [18] into a setting with distributed CSOCs. Their work does not address what happens before a shift.

Shah et al. [17] proposed an optimization framework to allocate cyber alert detection sensors to alert analysts. This was extended in [16] which focused on fairness issues in the sensor-analyst allocation problem. Okimoto et al. [15] proposed a system to optimize the cyber-security systems involved with multiple criteria, e.g., risk (security), surveillance (privacy) and cost. Altner et al. [1] investigated optimizing staffing and shift scheduling decisions given unknown demand on weekly shifts with an on-call mechanism. In two less related (as they do not work on analyst scheduling) but still relevant works, Franklin et al. [7] proposed a visualization tool to help with the cyber analysts to understand the analytic process and to structure their analytic thinking. Larriva-Novo et al. [12] proposed a dynamic risk management system with the capability of reacting to those rapid changes in the context of the organization with various types of sensors.

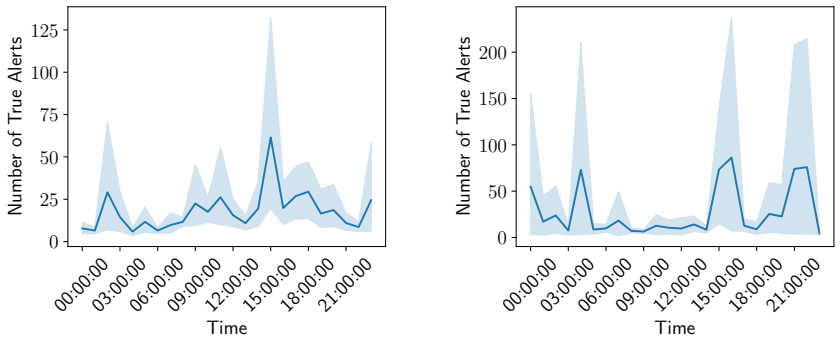
In contrast to all of these efforts that do not rely on a mechanism to separate real from false alerts, PCAM combines scheduling with prediction to (i) comes up with an *a priori* (e.g. just before a shift starts) schedule that minimizes the *expected number of uncovered true alerts* (i.e. true alerts that are missed by analysts) given some resource constraints and some workplace constraints, (ii) come up with a real-time mechanism to handle alerts that come in, once the shift schedule has been created, and (iii) considers variant of PCAM involving robustness to various natural or adversarial scenarios. Moreover, unlike the past work, PCAM has been tested on real-world data and shows high quality performance. In addition, our formulation of the analyst scheduling problem is closer to the realistic setting with more practical constraints (e.g., capacity of analysts, constraints on break time, etc.), and therefore introduces a more complex mixed integer nonlinear bi-level optimization problem which has not been addressed in previous works.

3 PCAM DATA

Figure 1 shows the proposed architecture of the PCAM system. PCAM takes as input, the data coming from alert generating products used in enterprises. In our experiments, we used alerts coming from two well-known commercial security products used at Dartmouth College. We used two pieces of information from these products.

(1) *Alert data statistics.* We obtained 44 days of real alert data. [2] This data contained 8,119,858 alerts, leading to a daily average of 180,441 alerts per day. From this entire set of alerts, according to Dartmouth's security analysts, there were just 21,916 true alerts, i.e. about 0.27%

[2] An additional 6 days of data was later used for live testing.



(a) Weekdays (b) Weekends

Fig. 2. Temporal distribution of true alerts for weekdays and weekends. The x-axis represents the time and y-axis represents the number of true alerts per hour. The shadowed areas represent the standard deviation.

of the alerts were real, while about 99.73% of the alerts were false alarms. This constitutes an average of 498 true alerts per day, lost in a sea of 180,441 alerts being generated every day. We obtained statistics about the distribution of these alerts (both all alerts and true alerts) within 10 minute windows during the course of any given day. In other words, we broke the day down into 10-minute windows and obtained statistics (mean and standard deviation of the number of alerts per window, mean and standard deviation of the number of true alerts per window) and used these in Section 4 to generate data-driven schedules for analysts for the next shift so that the shift schedules satisfy various workplace requirements. Figures 2a and 2b show the temporal distribution of the number of true alerts during the weekdays and weekends. The results show that the distribution of true alerts is very different during weekdays and weekends. During weekdays. the true alerts largely occur during the 12 noon to 3pm window, but during weekends, we see three spikes: during the 3am-4am window, during the 12 noon - 3pm window (as was the case during weekdays) as well as during the 6pm-9pm window. All times are US East Coast times.

(2) *Raw Alerts.* In addition, once workers are on a shift, new alerts keep coming in from the security products used by the enterprise. In this case, we developed a "True Alert Predictor" to predict which of the alerts coming in in real time were true and which were false. We were able to do this with high accuracy.

As a consequence, our PCAM architecture allows us to automatically schedule analyst shifts in a manner that is consistent with the arrival of both alerts and true alerts in 10-minute windows prior to the start of a shift — and during a shift, the PCAM framework can present to the analysts, those alerts that are predicted to be true in descending order of the probability of the truth.

4 PCAM ANALYST SHIFT SCHEDULING AS A NONLINEAR BI-LEVEL OPTIMIZATION

In this section, we describe the problem of scheduling an analyst's shift for Maximal True Alert Coverage (MTAC). Past work [1] suggests that most shifts are either 8-hour shifts or 12-hour shifts. Because our PCAM framework is the same irrespective of the duration of the shift, without loss of generality, we assume the shift is 12-hours long. 12-hour shifts usually run from 7am-7pm (day shift) and 7pm-7am (night shift).

minimize $\Sigma_{j=1}^{n}\left(\max\{0, TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i\}\right)$
subject to

$$\Sigma_{j=1}^{n} v_{i,j} \leq MT \qquad \forall a_i \tag{1}$$

$$\Sigma_{j=h}^{j=CT+h} v_{i,j} \leq CT - 1 \qquad \forall a_i, \forall h = 1, \ldots, n - CT \tag{2}$$

$$u_{i,h} = \Pi_{k=h}^{k=L+h}(1 - v_{i,k}) \qquad \forall a_i, \forall h = 1, \ldots, n - CT \tag{3}$$

$$\Sigma_{h=LS}^{LE-L} u_{i,h} = 1 \qquad \forall a_i \tag{4}$$

$$v_{i,j} \in \{0, 1\} \qquad \forall a_i, \forall t_j \tag{5}$$

$$u_{i,h} \in \{0, 1\} \qquad \forall a_i, \forall h = LS, \ldots, LE - L \tag{6}$$

Fig. 3. Nonlinear Optimization problem for Shift Scheduling

We assume a shift is divided up into contiguous time slices $t_1, \ldots, t_n$ where each time slice t_j represents a user-selected interval of time that is sufficiently small for their purposes. In our work, we chose the time slices to be 10 minutes long and hence a shift of 12-hours corresponds to 72 time slices, i.e. $n = 72$. We assume that the company employs m analysts $a_1, \ldots, a_m$. Each analyst is either a junior analyst, a senior analyst, or a principal analyst and their capabilities in terms of processing alerts are given below — the statistics below are reproduced from the study of [1]. We assume that analyst a_i is capable of analyzing c_i alerts per time slice in accordance with the above statistics. In addition, we use the following notation.

- MT is a constant integer which denotes the maximum number of time slices that an analyst may work during a shift.
- CT is a constant integer which denotes the maximum number of contiguous time slices that an analyst may work before taking a break for one time slice.
- L is a constant integer which says that each analyst must have a lunch break consisting of at least L contiguous time units. (We recognize that the "lunch break" in the night shift is a break for a meal rather than for lunch, but will abuse notation and call it a lunch break).
- LS, LE where $LS \leq LE$ are constant integers which bound the period when each analyst gets their lunch break. For instance, during the day shift, we may set LS to 37 (i.e., 11:00-11:10 am) and LE to 51 (i.e., 2:20-2:30 pm), and this imposes the constraint that every analyst gets his lunch break during the time interval $[LS, LE]$. In particular, we require that each analyst's lunch break be fully contained within the $[LS, LE]$ interval.
- We use A_j and TA_j as the expected number of alerts (resp. true alerts) to arrive during time interval t_j. These numbers can be estimated from historical statistics and computed as expected values.

The problem of scheduling shifts is now the problem of allocating analysts to shifts so that the expected number of "uncovered true alerts" is minimized. We first formalize this as a bi-level mixed integer nonlinear optimization problem as shown in Figure 3.

4.1 Constraints

We now explain the constraints in the mixed integer nonlinear optimization formulation provided in Figure 3.

Our formulation uses binary-valued variables $v_{i,j}$. The idea is that $v_{i,j}$ should be set to 1 if analyst a_i is assigned to work during time slice t_j and 0 otherwise, i.e.,

$$v_{i,j} = \begin{cases} 1 & \text{if analyst } a_i \text{ works during time slice } t_j \\ 0 & \text{otherwise} \end{cases}$$

Constraint (1). Constraint (1) of Figure 3 sets one constraint for each analyst saying that that analyst cannot work for more than MT time slices during any given shift.

Constraint (2). The goal of the second type of constraint is to ensure that each analyst gets a break periodically. More formally, every time interval of length CT is required to have at least one break in it. For this, suppose we set $BR_h = [h, CT + h]$ to be an interval of length CT starting at time slice t_h. Hence, we have such time intervals

$$\begin{aligned} BR_1 &= [1, CT + 1] \\ BR_2 &= [2, CT + 2] \\ &\cdots \\ BR_{n-CT} &= [n - CT, n] \end{aligned}$$

For each such interval $BR_h = [h, CT + h]$ where $1 \leq h \leq n - CT$, a break is needed for each analyst. Thus, for each $1 \leq i \leq m$ and each time slice $1 \leq h \leq n - CT$, we require that: $\Sigma_{j=h}^{j=CT+h} v_{i,j} \leq CT - 1$. By having this constraint for each and every time window BR_h of size CT, we ensure that at least one of the $v_{i,j}$'s is set to 0 which would then be the time slice in which the analyst a_i gets a break.

Constraints (3)-(4). The third constraint requires that every analyst gets a lunch break. For each analyst a_i, suppose $LT_h^i = [h, L + h]$. Intuitively, each LT_h^i interval corresponds to the possible lunch periods for analyst a_i that starts at time t_h and goes till time t_{L+h}. We require this *entire* time to be a break for the analyst, i.e. the value of the variable $v_{i,j}$ should be 0 for each $j \in LT_h^i$ if this interval happens to be the analyst's lunch break. To express this idea, we define an intermediate variable $u_{i,h}$ which is set to 1 if analyst a_i's lunch break starts at time t_h. Constraint (3) is therefore:

$$u_{i,h} = \Pi_{k=h}^{k=L+h} \left(1 - v_{i,k}\right)$$

If this is indeed analyst a_i's lunch break, then $v_{i,h} = 0$ for all $k \leq h \leq L + h$. This means that $(1 - v_{i,j}) = 1$ for all $k \leq h \leq L + h$ which in turn means that $\Pi_{k=h}^{k=L+h} \left(1 - v_{i,k}\right)$ would equal 1 *if* the interval $LT_h^i = [h, L + h]$ is indeed the analyst's lunch break.

Because the analyst gets one and only one lunch break, Constraint (4) says that one and only one of the possible $u_{i,h}$'s must be set to 1. Thus, constraints (3) and (4) jointly enforce the fact that each analyst gets a contiguous lunch break of length L during his shift. But this comes at a cost — because Constraint (3) is nonlinear.

Constraints (5)- (6). These two constraints ensure that the variables $v_{i,j}$ and $u_{i,h}$ are binary variables.

4.2 Objective Function

The formulation of the MTAC problem *assumes* that we have an estimate of the expected number TA_j of true alerts that occur during any given time period t_j, $1 \leq j \leq n$ — we can see that we do have this information from our Dartmouth College data as shown in Figures 2a and 2b.

Figures 2a and 2b show that the distribution of true alerts during weekdays is dominated (at least at Dartmouth College) by the 12pm - 3pm window. In contrast, there are multiple peaks during the weekends with attacks between the 3am - 4am windows, as well as between the 12pm -

3pm window and the 6pm - 9pm window. Though these statistics only apply to the true alerts at Dartmouth College, most medium and large enterprises can generate similar distributions.

We note that we can estimate TA_j from historical data to simply be the mean of the set $\{TA_j(d)|d \in TD\}$, i.e. if j is the 10-minute time slot between 10am and 10:10am, then the mean (resp. standard deviation) is simply the mean (resp. standard deviation) of the number of true alerts generated during this time window in the historical data.[3]

The term $v_{i,j} \cdot c_i$ is the number of alerts that analyst a_i can handle (on average) in one time slice. Note that when $v_{i,j} = 0$, i.e. when the analyst is on a break, he can handle no alerts at all. Thus, the summation $\Sigma_{i=1}^{m} v_{i,j} \cdot c_i$ is the total number of alerts that the m analysts can handle in one time slice t_j and hence $TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i$ is an estimate of the total number of alerts that would be left "uncovered" in that one time slice t_j. By uncovered, we mean that this is the number of alerts that are not examined/handled by any of the analysts during this time slice. Because this number could be negative, we take the max of this and 0 in the objective function, i.e. the number of uncovered alerts in time slice t_j would be $\max(TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i)$. Finally, we sum up the number of uncovered alerts over time in order to get the total number of uncovered alerts during a given shift, i.e. the total number of uncovered alerts during the entire shift consisting of the time slices $t_1, \ldots, t_n$ is:

$$\Sigma_{j=1}^{n} \left(\max\{0, TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i\} \right)$$

The goal of our objective function is therefore to minimize the number of uncovered alerts across the shift as a whole.

Discussion: We see immediately that the objective function essentially has two levels, where the outer level optimization minimizes the total number of uncovered true alerts, and the inner level optimization ensures that the number of uncovered true alerts cannot be "negative".

5 EFFICIENT SOLUTION AS A MIXED INTEGER LINEAR PROGRAM

Because the problem of computing shifts involves optimizing an integer program that is bi-level and nonlinear, it is not directly solvable using existing commercial optimization solvers. In the rest of this section, we show how this problem can be neatly encoded as a single-level mixed integer linear optimization problem.

5.1 Non-Linear Constraints Transformation

The formulation of our problem of maximizing the expected number of true alerts that are covered is nonlinear which suggests that it can be very expensive to solve it. In this section, we show that we can represent it in a linear form.

To linearize the nonlinear constraints in Eq. (3), we replace each of the nonlinear constraints with two linear constraints:

$$u_{ih} \leq 1 - \Sigma_{k=h}^{L+h} v_{i,k}/M \quad \forall a_i, \forall h = 1, \ldots, n - CT \tag{7}$$

$$u_{ih} \geq 1 - \Sigma_{k=h}^{L+h} v_{i,k} \quad \forall a_i, \forall h = 1, \ldots, n - CT \tag{8}$$

Here $M > m$ is any constant that is larger than the total number of analysts.

The following lemma shows that the constraints expressed by equations:

Lemma 5.1. *In the formalized MTAC problem, the constraints expressed in Equations* (5)–(8) *have the exact same set of solutions as the constraints* (3)(5)(6).

(5)(6)(7)(8) *are equivalent to Eqs.* (3)(5)(6).

[3]If so desired, we can compute the mean and standard deviation using just that portion of the training data that looks at the last w time windows as opposed to all of them.

Proof. Since Eqs. (5)-(6) require both $v'_{i,j}s$ and u_j's to be binary variables, we only need to prove the equivalence of Eqs. (7)(8) and Eq. (3).

We now prove that for each analyst a_i and time slice $h = 1, \ldots, n - CT$, the same values of $v_{i,k}, k = h, \ldots, L + h$ will yield the same values of $u_{i,h}$ in both cases. For ease of reference, we denote the $u_{i,h}$'s in Eq. (3) and in Eqs. (7)(8) as $u^I_{i,h}$ and $u^{II}_{i,h}$, respectively. We split the discussion into the following two cases.

Case 1: $v_{i,k} = 0, \forall k = h, \ldots, L + h$. In this case, it is easy to see that $u^I_{i,h} = \Pi^{k=L+h}_{k=h}(1 - v_{i,k}) = 1$. At the same time, we have

$$u^{II}_{i,h} \leq 1 - \Sigma^{L+h}_{k=h} v_{i,k}/M = 1,$$
$$u^{II}_{i,h} \geq 1 - \Sigma^{L+h}_{k=h} v_{i,k} = 1,$$

which implies that $u^{II}_{i,h} = 1$. Thus $u^I_{i,h} = u^{II}_{i,h}$.

Case 2: $v_{i,k} = 1, \exists k = h, \ldots, L + h$. Denote the sum term $\Sigma^{L+h}_k v_{i,k}$ as σ. In this case, σ would be an integer value between 1 and m: $1 \leq \sigma \leq m$. For $u^I_{i,h}$, we have $u^I_{i,h} = \Pi^{k=L+h}_{k=h}(1 - v_{i,k}) = 0$. For $u^{II}_{i,h}$, we have

$$u^{II}_{i,h} \leq 1 - \Sigma^{L+h}_{k=h} v_{i,k}/M = 1 - \sigma/M,$$
$$u^{II}_{i,h} \geq 1 - \Sigma^{L+h}_{k=h} v_{i,k} = 1 - \sigma.$$

Because $1 \leq \sigma \leq m \leq M$, we have $0 < 1 - \sigma/M < 1$. So $1 - \sigma \leq u^{II}_{i,h} < 1$. Combining the constraint that $u_{i,h}$'s are binary variables, we have $u^{II}_{i,h} = 0$. Thus $u^I_{i,h} = u^{II}_{i,h}$ still holds. □

This result therefore shows a way of linearizing the constraints while preserving the same set of solutions as the solutions of the initial nonlinear integer program.

5.2 Bi-Level Objective Function Transformation

minimize $\Sigma^n_{j=1} w_j$

subject to

$$w_j \geq TA_j - \Sigma^m_{i=1} v_{i,j} \cdot c_i \quad \forall t_j \quad (9)$$
$$w_j \geq 0 \quad \forall t_j \quad (10)$$
$$\Sigma^n_{j=1} v_{i,j} \leq MT \quad \forall a_i \quad (11)$$
$$\Sigma^{j=CT+h}_{j=h} v_{i,j} \leq CT - 1 \quad \forall a_i, \forall h = 1, \ldots, n - CT \quad (12)$$
$$u_{ih} \leq 1 - \Sigma^{L+h}_{k=h} v_{i,k}/M \quad \forall a_i, \forall h = 1, \ldots, n - CT \quad (13)$$
$$u_{ih} \geq 1 - \Sigma^{L+h}_{k=h} v_{i,k} \quad \forall a_i, \forall h = 1, \ldots, n - CT \quad (14)$$
$$\Sigma^{LE-L}_{h=LS} u_{i,h} = 1 \quad \forall a_i \quad (15)$$
$$v_{i,j} \in \{0, 1\} \quad \forall a_i, \forall t_j \quad (16)$$
$$u_{i,h} \in \{0, 1\} \quad \forall a_i, \forall h = LS, \ldots, LE - L \quad (17)$$

Fig. 4. MILP formulation for Shift Scheduling

As pointed out above, the bi-level objective function is also a source of high computational complexity of the MTAC problem. To handle this problem, we introduce an auxiliary variable w_j, where for each time slice t_j, we require that $w_j \geq TA_j - \Sigma^m_{i=1} v_{i,j} \cdot c_i$ and $w_j \geq 0$. We now show that the optima are preserved as well.

Lemma 5.2. *Minimizing the objective in the optimization problem of Fig. 3 is equivalent to the following optimization problem:*

$$\textit{minimize} \qquad \sum_{j=1}^{n} w_j \tag{18}$$

$$\textit{subject to} \qquad w_j \geq TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i \qquad \forall t_j \tag{19}$$

$$w_j \geq 0 \qquad \forall t_j \tag{20}$$

Proof. Let $\mathbf{v}_j = \langle v_{i,j} \rangle, i = 1, \ldots, m$ denote the decision variable (whether they are working or not) for all the analysts i at time slice t_j. For each time slice t_j, we define $f_j(\mathbf{v}_j) \coloneqq \max\{0, TA_j - \sum_{i=1}^{m} v_{i,j} \cdot c_{i,j}\}$. The objective can be represented as $f(\mathbf{v}_1, \ldots, \mathbf{v}_n) = \sum_{j=1}^{n} f_j(\mathbf{v}_j)$.

Because for each time slice t_j, there is $w_j \geq TA_j - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i$ and $w_j \geq 0$, thus we have $w_j \geq f(\mathbf{v}_j)$. Let $w = \sum_{j=1}^{n} w_j$, we then have $w \geq f(\mathbf{v}_1, \ldots, \mathbf{v}_n)$. Therefore, we can define the epigraph of function $f(\mathbf{v}_1, \ldots, \mathbf{v}_n)$ as

$$\text{epi}f = \{(\mathbf{v}, w) | \mathbf{v} \in \text{dom}f, f(\mathbf{v}) \leq w\},$$

where $\mathbf{v} = \langle \mathbf{v}_j \rangle, j = 1, \ldots, n$ is the decision variable for all the analysts throughout all the time slices. It is easy to see that the optimization described in Eqs. (18)-(20) is the epigraph form of minimizing the objective in Fig. 3. According to [2], the epigraph form of the original optimization problem is equivalent. □

Combining Lemmas 5.1-5.2, we immediately have the following theorem.

Theorem 5.3. *The bi-level non-linear formulation of the MTAC problem described in Fig. 3 is equivalent to the single-level Mixed Integer Linear Programming (MILP) in Fig. 4.*

Important Note. The above theorem is very important because it reduces the problem of shift scheduling in order to maximize expected true alert coverage (or alternatively minimize uncovered true alerts) to a single-level mixed integer linear programming (MILP for short) problem instead of a bi-level mixed integer nonlinear programming problem. Though MILPs are in general NP-hard to solve, they have been well studied, and there are existing commercial optimization solvers (e.g., Gurobi and GLPK) that can be used to efficiently solve these types of optimization problems. For instance, in our experiments with a team of 6 junior, 8 senior and 8 principal analysts, the problem can be solved within 0.4943 seconds (averaged over 50 runs, using Gurobi).

Note on Identifying the Optimal Mix of Personnel. Given a (biweekly) budget B for the day/night shift, a CISO (or similar senior leader) of an organization needs to identify the optimal mix of people to hire. This can be easily done. Suppose Sal_j, Sal_s, Sal_p respectively denote the biweekly budget for salaries of a junior, senior, and principal analyst respectively. Then the organization can hire upto n_j, n_s, n_p junior, senior, and principal analysts, respectively, where $n_j = \lfloor B/Sal_j \rfloor$, $n_s = \lfloor B/Sal_s \rfloor$ and $n_p = \lfloor B/Sal_p \rfloor$ respectively. We can then identify the optimal mix of junior, senior, and principal analysts to hire within the budget via a simple procedure. A *staffing triple* is a triple of the form (K_j, K_s, K_p) where $1 \leq K_j \leq n_j, 1 \leq K_s \leq n_s, 1 \leq K_p \leq n_p$. Given a budget B, let $ST(B)$ denote the set of all staffing triples. For each staffing triple $st_{j,s,p} = (K_j, K_s, K_p) \in ST(B)$, let MTA_{K_j,K_s,K_p} be the number of missed true alerts computed by running the Mixed Integer Linear Program shown in Figure 4. The optimal staffing is then

$$(J^*, S^*, P^*) = \underset{(K_j,K_s,K_P) \in ST(B)}{\arg\min} MTA_{K_j,K_s,K_p}.$$

6 IMPROVING ROBUSTNESS OF SCHEDULING

In the above, we use the average number of true alerts to derive a schedule for the analysts. However, the actual number of true alerts might be different from the averages due to 1) fluctuations/randomness or 2) existence of an adversary. To handle these two cases and therefore improve

the robustness of PCAM scheduling, we improve the above scheduling algorithm by taking these two factors into account. Our design is inspired by the idea of "adversarial training" in adversarial machine learning methods, where the idea is to add additional "adversarial samples" in the training set, so that the learned models are robust to adversarial attacks.

More specifically, the "adversarial sample" is a given true alert distribution $TA_1^{(l)}, \ldots, TA_{MT}^{(l)}$ denoted by a superscript l. The adversarial true alert distribution can be different from the average numbers of the distribution from which the probabilities used by PCAM are derived. For each such distribution $TA_1^{(l)}, \ldots, TA_{MT}^{(l)}$, we add it into the constraints in Eq.(9):

$$w_j \geq TA_j^{(l)} - \Sigma_{i=1}^{m} v_{i,j} \cdot c_i \quad \forall t_j \tag{21}$$

With sufficient sampling of such adversarial distributions, we can obtain a super-set of constraints which take into account changes of true alert distributions. In particular, we consider the following 3 types of changes.

PCAM-Fluct: In the first case, we consider fluctuations of true alert distributions for each time slice. This is done by sampling from a probability distribution (a Gaussian) with the mean $\overline{TA}_j$ and standard deviation std_j obtained from the historical data:

$$TA_j^{fluct} \sim \mathcal{N}(\overline{TA}_j, \text{std}_j), \quad \forall t_j \tag{22}$$

PCAM-Shift: In the second case, we consider a "shift" of the true alert distributions by τ time slices:

$$TA_j^{shift} = \begin{cases} \overline{TA}_{j+\tau}, & \text{if} \quad j+\tau \leq MT \\ \overline{TA}_{j+\tau-MT}, \quad , & \text{if} \quad j+\tau > MT \end{cases} \tag{23}$$

Here τ is an integer value which can be positive or negative.

PCAM-Mix: In the last type, we consider a mix of the above two cases, i.e., we first do a shift of the distribution and then perturb it using a Gaussian distribution:

$$TA_j^{mix} = \begin{cases} TA_{j+\tau}^{shift}, & \text{if} \quad j+\tau \leq MT \\ TA_{j+\tau-MT}^{shift}, \quad , & \text{if} \quad j+\tau > MT \end{cases} \tag{24}$$

Note that reversing the fluctuation and shift operations would result in equivalent true alert distributions.

7 EXPERIMENTAL EVALUATION

We now describe the results of our experiments. We only show results of the PCAM Analyst Shift Scheduler in the main text — as predicting whether alerts are real or fake is not a contribution of this paper, we show those results in the appendix.

7.1 PCAM Analyst Shift Scheduler

We split 24 hour periods into a day shift (7am to 7pm) and a night shift (7pm to 7am). We consider time slices of 10 minutes — hence, our shifts have 72 time slices in them.

The average number of historical true alerts during any given day (resp. night) time slice is used to estimate the number of true alerts during the same time slice in the next day (resp. night) shift.

7.1.1 Settings. As mentioned earlier, our goal is to come up with a shift schedule that satisfies various workplace constraints while minimizing the number of uncovered true alerts. As the number of uncovered true alerts obviously depends upon the number of analysts at different levels (junior, senior, and principal analysts) which in turn depends on the organization's budget, we vary these numbers in the range $\{2, 3, \ldots, 9\}$. Note that these are used to evaluate how our method

Table 1. Biweekly pay for analysts. APR denotes Alert Processing Rate/hr.

	APR	Salary	Glassdoor (NY)	Glassdoor (CA)	Glassdoor (US)
Junior	5	$ 3000	$ 2500	$ 3200	$ 2500
Senior	7.5	$ 4000	$ 2750	$ 3600	$ 2700
Principal	10	$ 6000	$ 3400	$ 4500	$ 3375

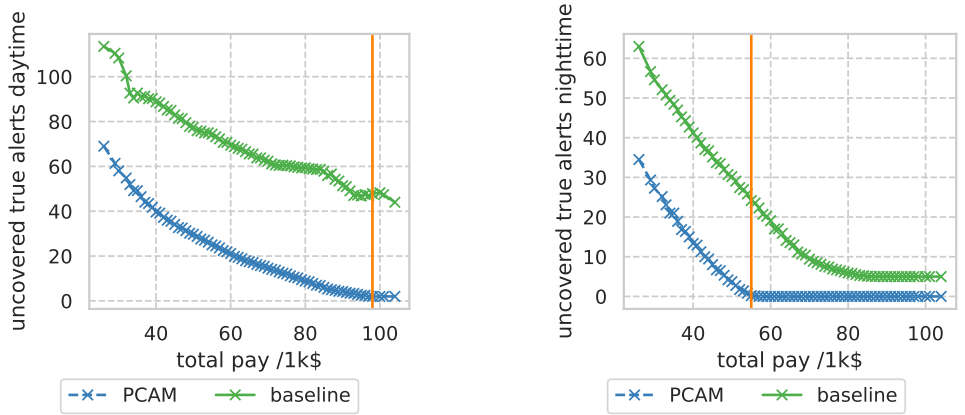
works under different settings with different numbers of analysts. Our main formulation is still the bi-level optimization introduced in Fig. 3 with fixed number of analysts of different types. Therefore, we do not include the optimization of allocation of number of analysts as part of the optimization.

For each combination tuple of junior, senior, and principal analysts, we use existing optimization solvers such as Gurobi and GLPK to solve the MILP in Figure 4 to optimize (minimize) the number of uncovered true alerts, given their pay and capabilities as described in Table 1. It is important to note that PCAM can of course also be used in settings where different true alert rates hold and where the pay rates and capabilities of analysts are different by simply plugging the new values in instead of those we use in our experiments.

Baseline: We introduce a *baseline* which maximizes the amount of time worked by each analyst, subject to the workplace constraints. Specifically, we use the result of the optimization problem posed in Figure 5 as our baseline to compare with.

maximize $\Sigma_{j=1}^{n}\Sigma_{i=1}^{m} v_{i,j}$
subject to Equation (14) to (20).

Fig. 5. Baseline Shift Scheduling



(a) For daytime shift, PCAM needs 6 junior, 8 senior, and 8 principal analysts and a budget of 98.0k total pay biweekly is needed.

(b) For nighttime shift, PCAM's solution to achieve 0 uncovered true alerts needs 3 junior, 7 senior, and 3 principal analysts — with a 55.0K total pay biweekly.

Fig. 6. Number of uncovered true alerts of optimized schedule and baseline schedule for day and night shifts. The X-axes shows the biweekly. The Y-axes shows the number of uncovered true alerts.. The vertical orange line marks when the uncovered true alerts for the PCAM scheduler is 0.

7.1.2 PCAM Shift Scheduling Results. Figure 6 compares the PCAM optimized schedule and the *baseline* for day shifts and night shifts, respectively. The x-axis shows the budget. Given a budget, PCAM finds the optimal schedule (i.e. mix of junior, senior and principal analysts) to hire and compares the number of uncovered true alerts during the day and night respectively. For the day shift, the optimal number found by PCAM to get the number of uncovered true alerts to 0 consists of 6 junior, 8 senior, and 8 principal analysts at a cost of $ 98K for a 2-week period. In contrast, for the night shift, PCAM finds that the best combination consists of 3 junior, 7 senior, and 3 principal analysts at a cost of $ 55K for a biweekly period.[4] The reason fewer analysts are needed during the night (at least for the Dartmouth College setting) is intuitively because there are fewer true alerts during the night on weekdays.

The figures show that the PCAM optimized schedule beats the baseline by a hefty margin. It is better in two respects:

(1) PCAM delivers a superior uncovered true alert rate compared to the baseline. At 98K biweekly, PCAM has 0 uncovered true alerts, while the baseline has about 45.
(2) PCAM saves organizations money — to achieve a day time rate of 0 uncovered true alerts, the baseline would need $ 147K for a biweekly period instead of the $ 98K needed by PCAM. To achieve a night time rate of 0 uncovered true alerts, the baseline would need $ 83K for a biweekly period as compared to the $ 55K required by PCAM.

Financial Implications. In short, the use of PCAM would save $ 49K per biweekly period for day time salaries and $ 28K per biweekly period for night time salaries. As a consequence, the total saving in a year if PCAM was used instead of the baseline would be $ 77K ×24 biweekly periods which is $ 1.848M per year. Please note that this estimated saving depends upon many factors — the salaries noted in [1] and the true alert distributions shown in Figures 2b and 2a. These numbers might vary for other organizations depending upon these factors. Moreover, these numbers assume that the organization in question would like to reduce the expected number of uncovered true alerts to be near zero. If they are willing to take a risk, then the savings might drop. Nonetheless, our methods can be used by any organization as long as they can provide these numbers to PCAM as inputs.

Detailed comparison. Figures 7 to 8 show the detailed number of uncovered true alerts when varying exactly one of the three $n_{junior}, n_{senior}, n_{principal}$, while keeping the other two fixed using both the PCAM generated schedule and the *Baseline* schedule for day shifts.[5] As an example, the first subfigure in Figure 7 shows that when $n_{junior} = 2$, by varying n_{senior} and $n_{principal}$ the uncovered alerts could be reduced to 10.2 using PCAM optimized day shift. However, in Figure 8, the subfigure shows that when $n_{junior} = 2$, by varying n_{senior} and $n_{principal}$ the uncovered alerts could be reduced to 58.3 using baseline day shift. The figures show that the PCAM optimized schedule is substantially better than the *baseline* for every combination of types of analysts hired.

Humanistic practices. From a humanistic perspective, work shifts may be affected by various factors such as sickness, family issues, etc. We use the *uncovered rate* (UR) of true alerts to evaluate the performance, which is defined as the number of uncovered true alerts divided by the number of total true alerts. We define the humanistic absence rate to to be the absent work capacity divided by the total work capacity. Table 2 shows the uncovered rates for the alerts under various humanistic absence rates. The uncovered rate is obtained by the simulations based on historical data. The table

[4]Please note that the salary rates used were from [1] and from glassdoor.com as opposed to Dartmouth College salaries which were kept confidential from us. Moreover, benefits are not included in these costs, but would be around 30% of salary. Finally, please note that the number of junior, senior, and principal analysts computed above were based on the alert data for Dartmouth College's network. Though these numbers can be different for other organizations, the process followed in this paper can be directly applied to the alert statistics from those organizations.

[5]Results for night shifts are similar and we put them in the appendix for space concerns.

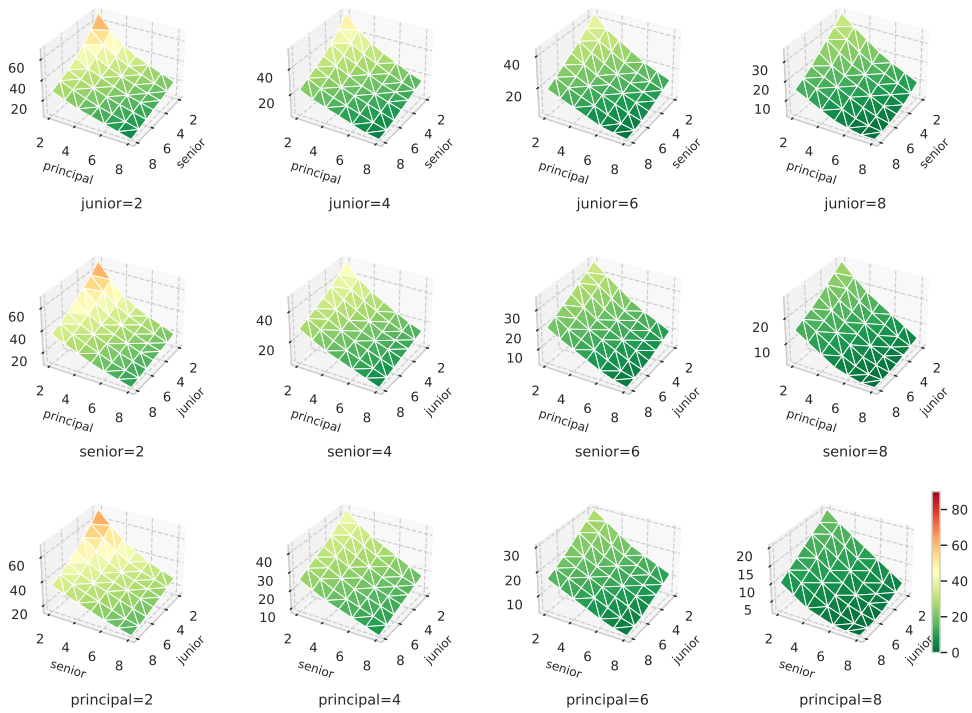


Fig. 7. Results for *PCAM* Optimized day shift. each plot shows the number of uncovered alerts when varying one number of the combination tuple (junior, senior, or principal), while the other two are fixed (the fixed values are shown in the titles of the subfigures). The first, second, third rows respectively fix the number of juniors, senior, or principal to hire. The columns indicate the number of the type of analysts that are fixed. The x and y axes represent the number of the two types of analysts that are being evaluated. The z-axis represents the uncovered alerts.

shows that PCAM is able to cover more than 75% of the alerts when the humanistic absence is 0.2 for both day and night simulations. A 20% absentee rate is quite high. The table therefore shows that even when the absentee rate of analysts is high, PCAM still achieves good performance and is robust to the humanistic absence of analysts.

Table 2. Uncovered rates under various absence rates

Humanistic absence	0	0.2	0.4	0.6	0.8	1.0
Day	1.000	0.763	0.529	0.310	0.130	0.007
Night	1.000	0.751	0.516	0.295	0.114	0.009

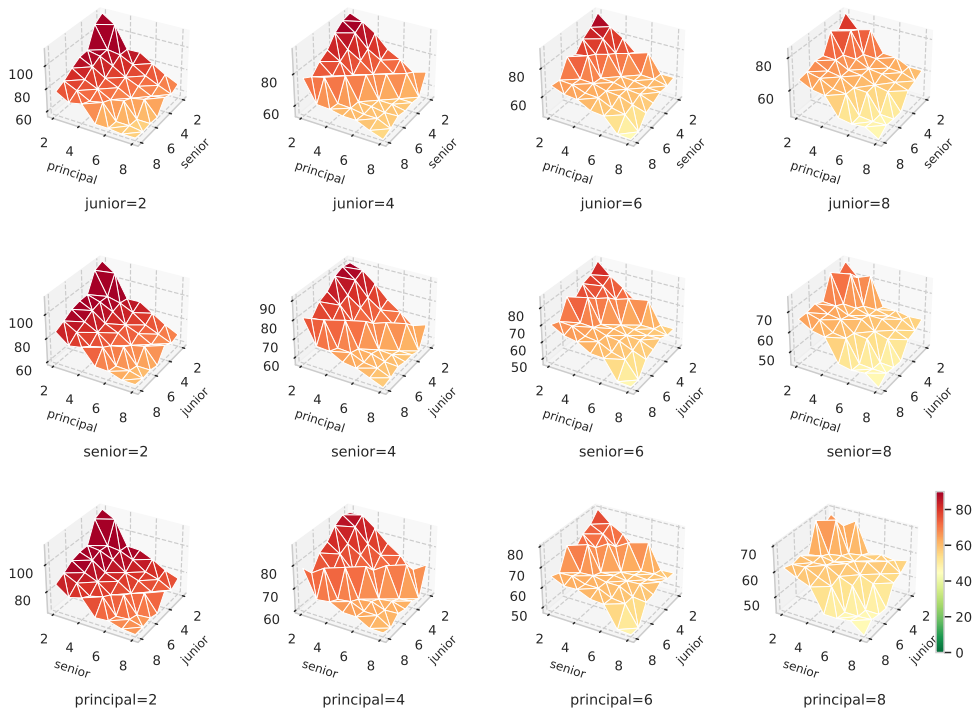


Fig. 8. Results for *baseline* day shift. Each plot shows the number of uncovered alerts when varying one number of the combination tuple (junior, senior, or principal), while the other two are fixed (the fixed values are in titles of the subfigures). The meanings of the rows, columns, and x-y-z axes are the same as in Fig. 7.

7.2 PCAM Shift Schedule Performance with Adversary

We also tested the performance of PCAM in the presence of an adversary. For each type of adversarial distribution, we also report the *Euclidean Distance*[6] from the original distribution. We use the *uncovered rate* (UR) of true alerts to evaluate the performance, which is defined as the number of uncovered true alerts divided by the number of total true alerts. **Fluctuations of true alert distributions**: In this experiment, we consider an adversarial setting where the true alert distributions are made to fluctuate around the average values of the historical data. To do this, we randomly sample 100 true alert distributions using the Normal distribution in Eq. (22) with a confidence interval (0.95). Table 3 shows the performance of PCAM-Fluct and PCAM in this setting. We immediately see that by taking the fluctuations into account in the formulated optimization problem, PCAM-Fluct performs much better than the vanilla PCAM for both day and night schedules. Figures 9a and 9d show the performance of PCAM-Fluct with respect to the Euclidean distance between the original and adversarial true alert distributions. The figures show that the uncovered rate UR increases as the distance between the distributions increases. This is

[6]We also consider other distance metrics such as Bray-Curtis Distance, Cosine Distance and Wave-Hedges Distance. We refer to [3] for the definitions of the distance metrics. The results using these other distributions are similar and are shown in the appendix.

intuitive as attacks which are more different from the original distribution will have higher success rate.

Shifts of true alert distributions: In this experiment, the adversarial samples are generated by shifting the original true alert distribution by 0.5, 1, 1.5, . . . , 6 (i.e. with 12 types of shift periods). Table 4 shows the performance of PCAM-Shift and PCAM in this scenario. As in the preceding experiment, we see that the use of adversarial training enables PCAM-Shift to outperform vanilla PCAM. Figures 9b and 9e show the performance of PCAM-Shift as the lengths of the time shift increases. In this setting, we can see that there is no clear pattern of UR with respect to the length of the temporal shifts. Our conjecture is that the true alert distribution with larger hours of shift to the original distribution is not necessarily further away from the original distribution.

Mixed adversarial true alert distribution: In this experiment, we consider both fluctuations and time shifts in the true alert distribution. Table 5 shows the performance of PCAM-Mix and PCAM in this scenario. We can see that PCAM-Mix consistently outperforms PCAM by a large margin. Figures 9c and 9f show the performance of PCAM-Mix with respect to the distance to the original true alert distribution. As in the case of PCAM-Fluct, the figures show that the Uncovered Rate increases as the distance increases.

Table 3. Performance of PCAM-Fluct and PCAM with fluctuations in true alert distribution. Results are averaged from 100 samples for both day and night schedules.

	Total true	UR_{PCAM}	$UR_{PCAM-Fluct}$	Distance
Day	502.270	0.717	0.340	132.437
Night	498.220	0.772	0.416	154.811

Table 4. Performance of PCAM-Shift and PCAM with shift in true alert distribution. Results are averaged for 12 samples for both day and night schedules.

	Total true	UR_{PCAM}	$UR_{PCAM-Shift}$	Distance
Day	279.000	0.434	0.021	58.348
Night	213.000	0.425	0.033	42.795

Table 5. Performance of PCAM-Mix and PCAM with both fluctuations and shifts in true alert distribution. Results are averaged for 100 samples for both day and night schedules.

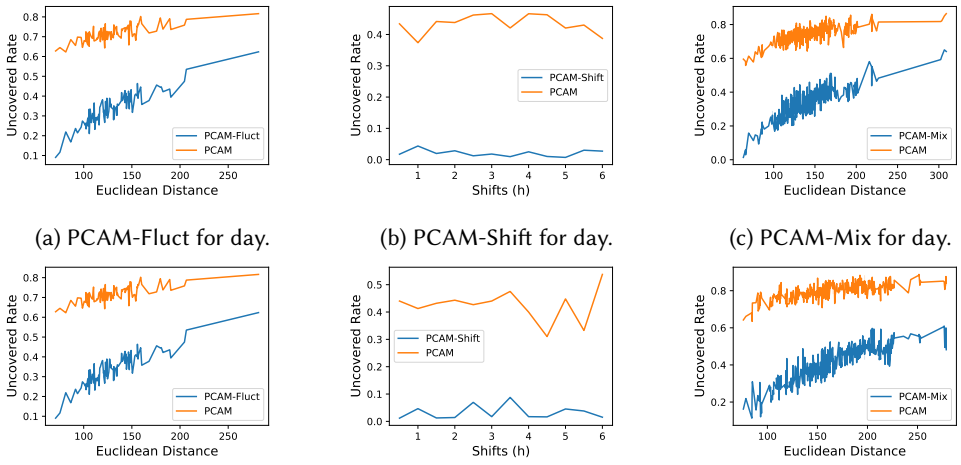
	Total true	UR_{PCAM}	$UR_{PCAM-Mix}$	Distance
Day	502.270	0.745	0.340	143.456
Night	498.220	0.801	0.409	166.163

7.3 Live test of PCAM

The acid test of the PCAM architecture would be to see how the entire PCAM architecture holds up. If analysts were scheduled in accordance with the schedule computed by PCAM and then the real world alerts occurred as they did occur in the live test, how many true alerts would be uncovered.[7]

We also did an end-to-end test using the previously determined numbers of analysts for the day and night shifts respectively on the live test (6 days). In this live test, we created the analyst

[7]The schedule assumed 6 junior, 8 senior, and 8 principal analysts for day time schedules and 3 junior, 7 senior, and 3 principal analysts for night time schedules.



(a) PCAM-Fluct for day. (b) PCAM-Shift for day. (c) PCAM-Mix for day.

(d) PCAM-Fluct for night. (e) PCAM-Shift for night. (f) PCAM-Mix for night.

Fig. 9. Performance of PCAM-Fluct, PCAM-Shift, and PCAM-Mix with respect to distance of the original and adversarial true alert distributions, for day (upper row) and night (lower row) schedules. The x-axis represents the *Euclidean Distance* or number of shifts. The y-axis represents the UR by each schedule. Results of other distance metrics are shown in the appendix.

schedules before the day and night shifts and then played back the actual arrival of alerts during the 6 days of the testing period. *Not a single true alert went uncovered by the schedules generated by the PCAM model.*

8 CONCLUSION

Most cyber-security operations centers experience a huge flood of security alerts. This deluge is typically managed by a small team which is usually overworked. Because of the sheer volume of alerts, true alerts can be missed from a sea of false alarms.

In this paper, we propose a data driven bi-level optimization algorithm to solve the following problems. First, given a distribution of true alerts during the day, how best should a given set of analysts be scheduled in order to minimize the expected number of true alerts that are not handled by an analyst. Second, given a desire to reduce the expected number of true alerts that are uncovered to be below a desired upper bound, how many analysts of different types do we need? Third, how can a predictor that predicts whether a given alert is real or merely a false alarm be incorporated into the system?

In order to solve these problems, we propose the PCAM (Probabilistic Cyber Alert Management) system. PCAM has a "before the shift" mode in which distributions of true alerts over time are used to create a schedule, and a "during the shift" mode in which alerts predicted to be true by PCAM are handled by analysts who are working during that time period. We show that PCAM's end-to-end performance is excellent. Using a 44-day training period with real alert data to train the PCAM model and to capture the desired distributions of true alerts during the day, we are able to show that the end to end PCAM system created schedules that did not miss a single true alert during a 6 day test window. Moreover, our predictive model also generated excellent results during live testing over the 6-day window, outperforming the performance in training. Last but not least, we also show that PCAM does well under various kinds of adversarial (or natural) fluctuations/shifts

between the statistics used during training and the statistics seen when the system is subsequently used.

ACKNOWLEDGEMENTS

Parts of this work were funded by ONR grants N00014-18-1-2670, N00014-16-1- 2896, and N00014-20-1-2407 and ARO grant W911NF-13-1-0421.

REFERENCES

[1] Douglas S Altner, Anthony C Rojas, and Leslie D Servi. 2018. A two-stage stochastic program for multi-shift, multi-analyst, workforce optimization with multiple on-call options. *Journal of Scheduling* 21, 5 (2018), 517–531.
[2] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.
[3] Sung-Hyuk Cha. 2007. Comprehensive survey on distance/similarity measures between probability density functions. *City* 1, 2 (2007), 1.
[4] Noah Dunstatter, Mina Guirguis, and Alireza Tahsini. 2018. Allocating security analysts to cyber alerts using Markov games. In *2018 National Cyber Summit (NCS)*. IEEE, 16–23.
[5] Noah Dunstatter, Alireza Tahsini, Mina Guirguis, and Jelena Tešić. 2019. Solving cyber alert allocation Markov games with deep reinforcement learning. In *International Conference on Decision and Game Theory for Security*. Springer, 164–183.
[6] Meisam Eslahi, Habibah Hashim, and Nooritawati Tahir. 2013. An efficient false alarm reduction approach in HTTP-based botnet detection. In *IEEE Symposium on Computers & Informatics (ISCI)*. 201–205.
[7] Lyndsey Franklin, Meg Pirrung, Leslie Blaha, Michelle Dowling, and Mi Feng. 2017. Toward a visualization-supported workflow for cyber alert management using threat models and human-centered design. In *2017 IEEE Symposium on Visualization for Cyber Security (VizSec)*. IEEE, 1–8.
[8] Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2017. Optimal scheduling of cybersecurity analysts for minimizing risk. *ACM Transactions on Intelligent Systems and Technology (TIST)* 8, 4 (2017), 52.
[9] Rajesh Ganesan, Ankit Shah, Sushil Jajodia, and Hasan Cam. 2019. Optimizing alert data management processes at a cyber security operations center. In *Adversarial and Uncertain Reasoning for Adaptive Cyber Defense*. Springer, 206–231.
[10] Neminath Hubballi and Vinoth Suryanarayanan. 2014. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications* 49 (2014), 1–17.
[11] Sushil Jajodia, Noseong Park, Edoardo Serra, and VS Subrahmanian. 2016. Using temporal probabilistic logic for optimal monitoring of security events with limited resources. *Journal of Computer Security* 24, 6 (2016), 735–791.
[12] Xavier Larriva-Novo, Mario Vega-Barbas, Victor A Villagrá, Diego Rivera, Mario Sanz, and Manuel Álvarez-Campana. 2020. Dynamic risk management architecture based on heterogeneous data sources for enhancing the cyber situational awareness in organizations. In *Proceedings of the International Conference on Availability, Reliability and Security*. 1–9.
[13] Greg Masters. June 9 2017. Crying wolf: Combatting cybersecurity alert fatigue. *SC Magazine* (June 9 2017). https://www.scmagazine.com/home/security-news/in-depth/crying-wolf-combatting-cybersecurity-alert-fatigue/
[14] Yuxin Meng and Lam-For Kwok. 2013. Enhancing false alarm reduction using voted ensemble selection in intrusion detection. *International Journal of Computational Intelligence Systems* 6, 4 (2013), 626–638.
[15] Tenda Okimoto, Naoto Ikegai, Katsumi Inoue, Hitoshi Okada, Tony Ribeiro, and Hiroshi Maruyama. 2013. Cyber security problem based on multi-objective distributed constraint optimization technique. In *2013 43rd Annual IEEE/IFIP Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE, 1–7.
[16] Ankit Shah, Rajesh Ganesan, and Sushil Jajodia. 2019. A methodology for ensuring fair allocation of CSOC effort for alert investigation. *International Journal of Information Security* 18, 2 (2019), 199–218.
[17] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, and Hasan Cam. 2018. Optimal assignment of sensors to analysts in a cybersecurity operations center. *IEEE Systems Journal* 13, 1 (2018), 1060–1071.
[18] Ankit Shah, Rajesh Ganesan, Sushil Jajodia, Pierangela Samarati, and Hasan Cam. 2019. Adaptive alert management for balancing optimal performance among distributed CSOCs using reinforcement learning. *IEEE Transactions on Parallel and Distributed Systems* (2019).
[19] Yuan Wang, Yongjun Wang, Jing Liu, Zhijian Huang, and Peidai Xie. 2016. A survey of game theoretic methods for cyber security. In *2016 IEEE First International Conference on Data Science in Cyberspace (DSC)*. IEEE, 631–636.
[20] Mingtao Wu and Young Moon. 2019. Alert correlation for cyber-manufacturing intrusion detection. *Procedia Manufacturing* 34 (2019), 820–831.

A PCAM ALERT CLASSIFIER

We also briefly introduce our alert classifier that distinguishes true alerts from false ones. A number of threat detection systems [6, 10, 14, 20] have sought to reduce the false alarm rate. As we can use any such framework within PCAM, we do not discuss this further here. Note that we are not claiming technical novelty in this component, but describing it nonetheless for completeness of the PCAM architecture. We used the output logs of two security products used at Dartmouth College's Information Technology and Consulting (ITC) offices who run Dartmouth's network to define 3 broad categories of features related to alerts. These include:

- *Systems-Related Features*: Systems related features included information such as the time of receipt time (of a suspect packet), the identity of the logging server, and more.
- *Traffic-Related Features*: Traffic related features contain proxies for IP addresses, communication ports, the transmission protocol used, and more.
- *Threat-Related Features*: Threat-related features include the category of the threat and the severity of the threat according to that security product.

We preprocessed data by categorizing sparse attributes, discarding single-value attributes, duplicate attributes, and attributes with too many missing values. Table 6 describes the different types of variables we finally ended up with.

Table 6. Details of variables

Serial Number	Serial number of the firewall that generated the log.
Log Forwarding Profile	The profile used for log filtering and tagging.
Source Zone	Indicator of the direction of the connection, which is either outgoing of incoming.
Ingress Interface	The network interface device that the session was sourced form.
Egress Interface	The network interface device interface that the session was destined to.
Repeat Count	Number of sessions with same Source IP, Destination IP, Application, and Subtype seen within 5 seconds.
Flags	32-bit field that provides details on session, including PCAP (packet capture), IPv6, SSL, etc.
Protocol	IP protocol associated with the session.
Direction	Indicates the direction of the attack, client-to-server or server-to-client.
Source Location	Source country or Internal region for private addresses.
Destination Location	Destination country or Internal region for private addresses.
Source Port	Source port utilized by the session.
NAT Source Port	The source port used by the connection after NAT (network address translation).
Destination Port	Destination port utilized by the session.
NAT Destination Port	The destination port used by the connection after NAT (network address translation).
Subtype	Subtype of the threat log. Vulnerability means this threat is vulnerability exploit detected via a Vulnerability Protection profile.
Action	Action taken for the session.
Category	The category of the URL.
Severity	Severity associated with the threat. Reported by threat detectors.
Rule Name	Name of the rule that the session matched.

We ran 10 off the shelf classifiers in order to learn classifiers that separate true alerts from false ones. 7 of the 10 classifiers are "traditional" classifiers: Decision Trees (DTs), Logistic Regression (LR), Bernoulli Naive Bayes (BNB), Gaussian Naive Bayes (GNB), Multinomial Naive Bayes (MNB), Random Forest (RF), Support Vector Machines (SVM). The other 3 are neural classifiers including

Multi-Layer Perceptrons (MLP), Google's Deep and Wide system (DeepWide) and Convolutional Neural Nets (CNNs). We do not make any claim of novelty about these algorithms — as they give very high F1-scores and AUCs, there was no point developing a new classifier.

B TRUE ALERT PREDICTION RESULTS

The shift schedules experiments above are used to schedule each analyst's scheduled breaks and lunch slots — but they are finalized just *before* the analyst starts his shift.

The True Alert Predictor component of PCAM looks at alerts as they come in during a shift and tries to classify each alert as true or false when the alert comes in. We ran two experiments to assess the accuracy of the True Alert Predictor. In the first experiment, we did a traditional ML analysis of 44 days of data collected by Dartmouth's ITC staff. In the second experiment, we did a six day live test of the models learned on the 44 days of data to see how well they performed. As this component only uses traditional classifiers and is not particularly novel, we do not claim this as a contribution of this paper — just a necessary part of the PCAM architecture.

Historical Data Analysis. We tested each of the 7 classifiers on the features described in Section A. Because our alerts reflect temporal data, cross validation is not an appropriate test.[8] In order to avoid the pitfalls of using cross validation for temporal data, we use *rolling window prediction*. In rolling window prediction, we train our models for the first T days and then predict for day $T+1$ — then we learn the model from the first $T+1$ days of data and predict for day $T+2$, and so forth. We now report the results for rolling window prediction — for the sake of completeness, Table 7 contains the results of 10-fold cross validation as well. In our rolling window experiments, we train on each of the intervals $[1, 30], [1, 31], \ldots, [1, 43]$ and then predict for days $31, 32, \ldots, 44$ respectively, i.e. we train on the first T days and then make predictions for day $T+1$ with $T = 30, \ldots, 43$.

Table 7. Classifier Performance (True Alert vs. False Alert prediction) on 44 Days of Data using Rolling Window Prediction

	ROC-AUC	F1 score	Precision	Recall
Decision tree	0.9936	0.8634	0.8866	0.8595
Logistic	0.9992	0.8449	0.8918	0.8124
NB Bernoulli	0.9970	0.2621	0.1572	0.9064
NB Gaussian	0.9905	0.2507	0.1471	**0.9941**
NB multinomial	0.9967	0.2713	0.1639	0.9059
Random forest	0.9957	**0.8806**	0.9073	0.8600
SVM	0.9967	0.8538	0.9258	0.8064
DNN-MLP	**0.9998**	0.8740	**0.9350**	0.8300
DNN-DeepWide	0.9856	0.8634	0.8990	0.8466
DNN-CNN	0.9810	0.8634	0.8833	0.8300

Table 7 shows that PCAM's performance on rolling window prediction is excellent, with Random Forest yielding the best F1-Score of 88.06% with a precision of 90.73% and a recall of 86%. In terms of AUC, the results were almost a dead heat between Random Forest, SVM, with Google's Deep

[8]This is because K-fold cross validation would *randomly* split the data into K chunks. It would then do K iterations: in each iteration, one of the K chunks would be the test set, while the remaining $(K-1)$ chunks would be used for training the model. The performance of a classifier would then be obtained by aggregating the performance over the K folds. However, this is inappropriate for temporally sensitive data because the training folds can (with $\frac{K-1}{K}$ probability) contain data points from the future and those might be used to predict outcomes for data in the test fold which might be from the past.

and Wide coming out very marginally ahead. We therefore feel that Random Forest was the best of the 10 classifiers tested. [9]

Live Testing Experiments. We also conducted a live test. In this test, we used the best model (Random Forest) obtained via training on the 44-day window — but then tested it on 6 days of live data to assess the predictive efficacy of our models. Table 8 shows the result of rolling window prediction in this case.

Table 8. Live Test: Performance of Random Forest Classifier on True Alert vs. False Alert Prediction on days of live data Random Forest has the best F1 among all the classifiers.

	ROC-AUC	F1 score	Precision	Recall
Day 1	0.9931	0.9677	0.9633	0.9722
Day 2	0.9786	0.9649	0.9910	0.9402
Day 3	0.9998	0.9672	0.9987	0.9365
Day 4	0.9913	0.9652	0.9652	0.9652
Day 5	0.9758	0.9374	0.9450	0.9299
Day 6	0.9867	0.9364	0.9717	0.9035

Table 9. Experimental Results Showing Classifier Performance in Predicting whether an Alert is Real or False using Cross Validation

	ROC-AUC	F1 score	Precision	Recall
Decision tree	0.9895	0.9065	0.9488	0.8678
Logistic	0.9992	0.8898	0.9455	0.8404
NB Bernoulli	0.9967	0.2537	0.1476	0.9018
NB Gaussian	0.9906	0.2510	0.1437	**0.9915**
NB multinomial	0.9968	0.2643	0.1548	0.9013
Random forest	0.9923	**0.9085**	0.9519	0.8688
SVM	0.9974	0.8887	0.9488	0.8357
DNN-MLP	**0.9998**	0.9044	**0.9614**	0.8539
DNN-DeepWide	0.9825	0.8616	0.9576	0.7830
DNN-CNN	0.9840	0.8765	0.9256	0.8378

We see that the predictive accuracy of Random Forest actually *increased* — F1-scores for each of the days went up to 93.64%—96.77% with precision also increasing to lie between 94.5%—99.87% and recall to lie between 90.35%—97.22%. The fact that the trained models *improved* during live testing suggests that over-fitting did not occur during the training phase.

C PCAM DETAILED SCHEDULING RESULTS FOR NIGHT SHIFTS

Figure 10 to 11 show the detailed number of uncovered true alerts when varying exactly one of the three $n_{junior}, n_{senior}, n_{principal}$, while keeping the other two fixed using both the PCAM generated schedule and the *Baseline* schedule for night shifts. Results show that the PCAM optimized schedule is substantially better than the *baseline* for every combination of types of analysts hired during night shifts.

[9] In contrast, the results of K-fold cross validation shown in Table 9 are slightly inflated (due to the methodological flaw with using cross validation on temporal data) with Random Forest yielding the best F1-Score of 90.85% with a precision of 95.19% and a recall of 86.88%.

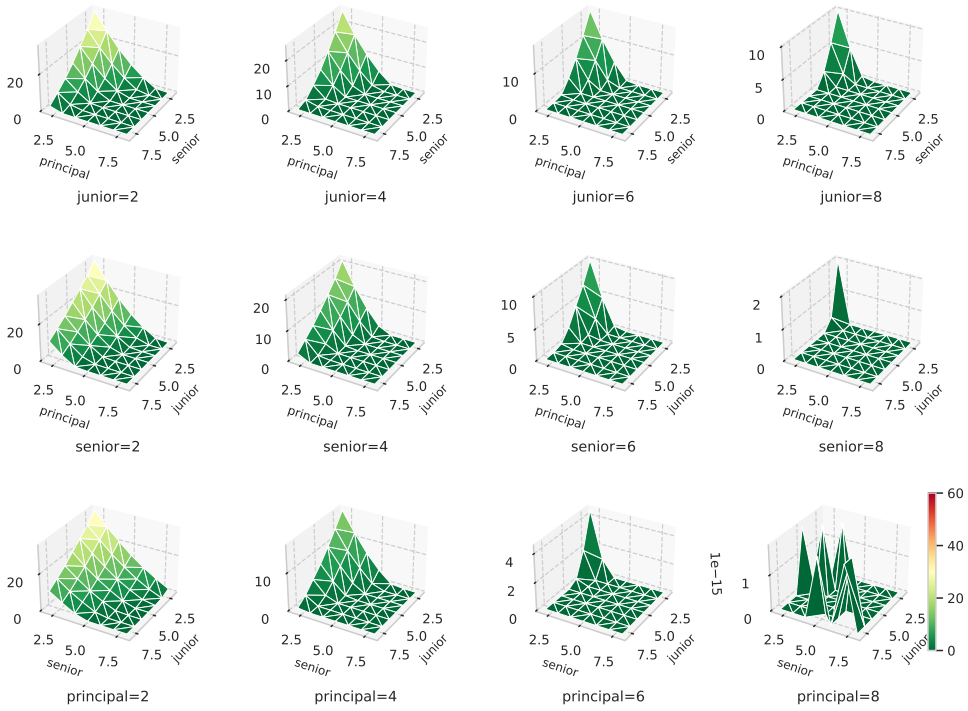


Fig. 10. Results for *PCAM* Optimized night shift. each plot shows the number of uncovered alerts when varying one number of the combination tuple (junior, senior, or principal), while the other two are fixed. The meanings of the rows, columns, and x-y-z axes are the same as in Fig. 7.

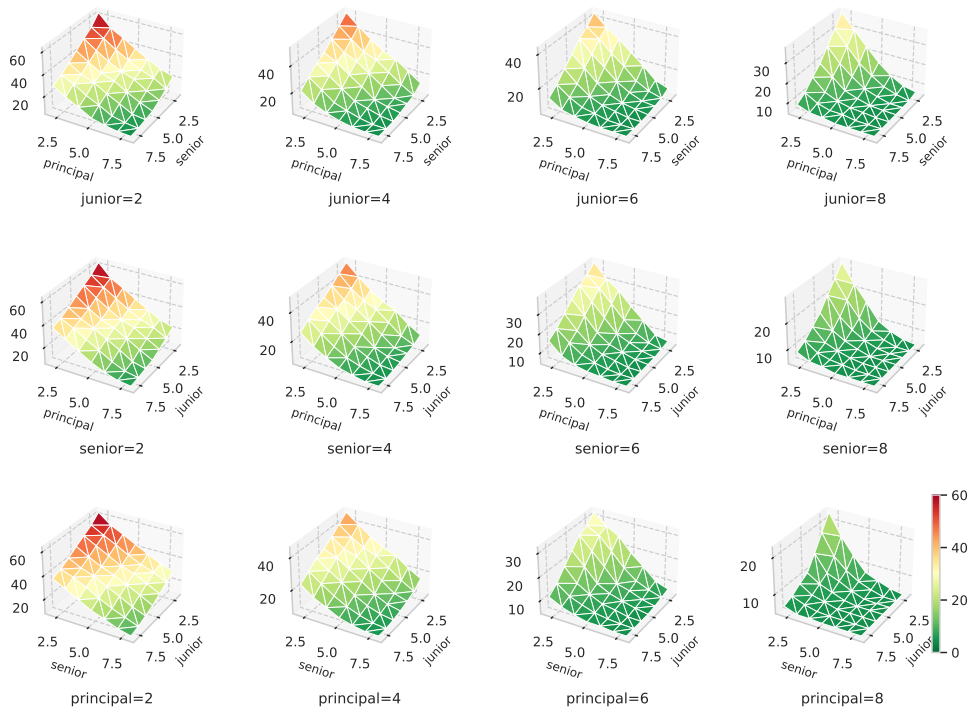
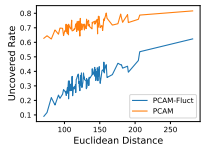
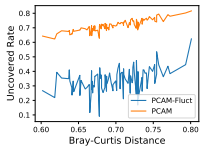


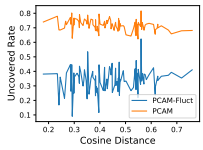
Fig. 11. Results for *baseline* night shift. each plot shows the number of uncovered alerts when varying one number of the combination tuple (junior, senior, or principal), while the other two are fixed. The meanings of the rows, columns, and x-y-z axes are the same as in Fig. 7.



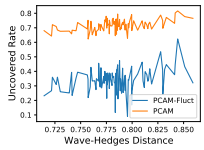
(a) PCAM-Fluct for day with *Euclidean Distance*.



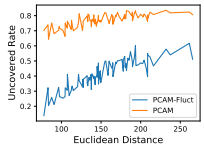
(b) PCAM-Fluct for day with *Bray-Curtis Distance*.



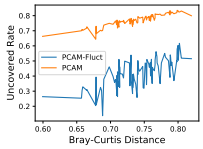
(c) PCAM-Fluct for day with *Cosine Distance*.



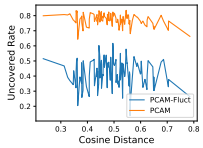
(d) PCAM-Fluct for day with *Wave-Hedges Distance*.



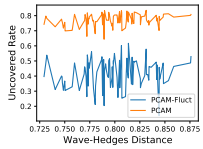
(e) PCAM-Fluct for night with *Euclidean Distance*.



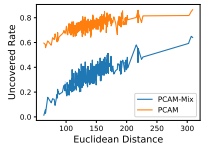
(f) PCAM-Fluct for night with *Bray-Curtis Distance*.



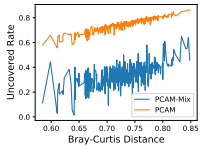
(g) PCAM-Fluct for nights with *Cosine Distance*.



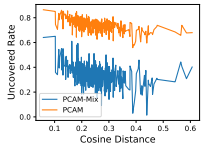
(h) PCAM-Fluct for night with *Wave-Hedges Distance*.



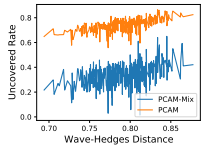
(i) PCAM-Mix for day with *Euclidean Distance*.



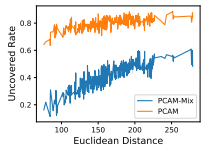
(j) PCAM-Mix for day with *Bray-Curtis Distance*.



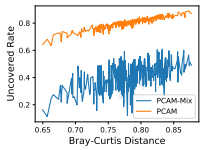
(k) PCAM-Mix for day with *Cosine Distance*.



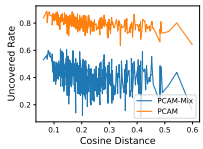
(l) PCAM-Mix for day with *Wave-Hedges Distance*.



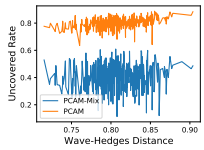
(m) PCAM-Mix for night with *Euclidean Distance*.



(n) PCAM-Mix for night with *Bray-Curtis Distance*.



(o) PCAM-Mix for night with *Cosine Distance*.



(p) PCAM-Mix for night with *Wave-Hedges Distance*.

Fig. 12. Performance for day and night schedules of PCAM-Fluct and PCAM-Mix. The x-axis represents the distance metrics, including *Euclidean Distance*, *Bray-Curtis Distance*, *Cosine Distance*, and *Wave-Hedges Distance*. The y-axis represents the *Uncovered Rate* by each schedule.