Discovering the Top-k Unexplained Sequences in Time-Stamped Observation Data

Massimiliano Albanese, Cristian Molinaro, Fabio Persia, Antonio Picariello, V. S. Subrahmanian

Abstract—There are numerous applications where we wish to discover unexpected activities in a sequence of time-stamped observation data—for instance, we may want to detect inexplicable events in transactions at a web site or in video of an airport tarmac. In this paper, we start with a known set $\mathcal{A}$ of activities (both innocuous and dangerous) that we wish to monitor. However, in addition, we wish to identify "unexplained" subsequences in an observation sequence that are poorly explained (e.g., because they may contain occurrences of activities that have never been seen or anticipated before, i.e. they are not in $\mathcal{A}$). We formally define the probability that a sequence of observations is unexplained (*totally* or *partially*) w.r.t. $\mathcal{A}$. We develop efficient algorithms to identify the top-k *Totally* and *Partially Unexplained Sequences* w.r.t. $\mathcal{A}$. These algorithms leverage theorems that enable us to speed up the search for totally/partially unexplained sequences. We describe experiments using real-world video and cyber security datasets showing that our approach works well in practice in terms of both running time and accuracy.

Index Terms—I.2.4 Knowledge Representation Formalisms and Methods $<$ I.2 Artificial Intelligence $<$ I Computing Methodologies, I.2.4.d Knowledge base management $<$ I.2.4 Knowledge Representation Formalisms and Methods $<$ I.2 Artificial Intelligence $<$ I Computing Methodologies

1 Introduction

Identifying unexpected activities is an important problem in a wide variety of applications such as video surveillance, cyber security, fault detection in safety critical systems, and fraud detection.

For instance, airport baggage areas are continuously monitored for suspicious activities by video surveillance. In crime-ridden neighborhoods, police often monitor streets and parking lots using video surveillance. In Israel, highways are monitored for suspicious activities by a central authority. However, all these applications search for *known* activities—activities that have been identified in advance as being either innocuous or dangerous. For instance, in the highway application, security officers may look both for normal behavior (e.g. driving along the highway in a certain speed range unless traffic is slow) as well as "suspicious" behavior (e.g. stopping the car near a bridge, taking a package out and leaving it on the side of the road before driving away).

In cyber security, intrusion detection can monitor network traffic for suspicious behavior and trigger security alerts. Alert correlation methods aggregate alerts into multi-step attack scenarios. However, both techniques rely on models encoding a priori knowledge of either normal or malicious behavior. They cannot deal with events such as "zero day" attacks that have never been seen before. In practice, all these methods are incapable of quantifying how well available models explain a sequence of events observed in an observation stream.

Figure 1 shows how our framework would work in practice. We start with a set of activity models $\mathcal{A}$ for both "good" and "bad" activities. Good activities are activities that are considered appropriate (e.g., certain permitted behaviors in an airport secure baggage zone) while bad activities are ones known to be inappropriate (e.g., a baggage handler opening a suitcase, taking items out, and putting them in a different bag). Techniques already exist to find occurrences of activities in time-stamped observation data (e.g., a video, a sequence of transactions at a website, etc.) with each occurrence having an associated probability.

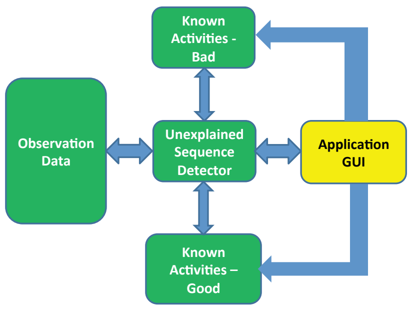


Fig. 1: Overall working of unexplained sequences

- M. Albanese is with the Department of Applied Information Technology, George Mason University, Nguyen Engineering Building, Fairfax, VA 22030. E-mail: malbanes@gmu.edu
- C. Molinaro and V. S. Subrahmanian are with the Department of Computer Science and UMIACS, University of Maryland, A.V. Williams Building, College Park, MD 20742. E-mail: {molinaro, vs}@umiacs.umd.edu
- F. Persia and A. Picariello, are with Dipartimento di Informatica e Sistemistica, Università di Napoli "Federico II", Via Claudio 21, 80125 Napoli, Italy. E-mail: {fabio.persia, picus}@unina.it

In this paper, our goal is to find an *unexplained sequence detector*, i.e. to identify subsequences of the observation data, called *unexplained sequences*, that known models are not able to "explain" with a certain confidence. In other words, what is happening in unexplained sequences is not well captured by the available activity models in $\mathcal{A}$. Once such subsequences have been identified, they can be further analyzed, e.g., to learn new activity models from them. Or, as shown in Figure 1, each unexplained sequence can be shown to a domain expert (e.g., airport security or cyber security expert) who can then add these observed sequences or generalizations thereof to the currently known list of good or bad activities.

Unexplained sequences allow an application to identify activities never seen or imagined before by experts, and to add them to an increasing body of such knowledge. For instance, a new type of terrorist attack at an airport or a zero-day attack on a computer system, may involve sequences of actions (observations) not seen before—and hence not captured by past activity models (i.e., those in $\mathcal{A}$). In this paper, we primarily focus on the unexplained sequence detector component of Figure 1.

We achieve this via a possible-worlds based model and define the probability that a sequence of observations is *totally* (or *partially*) unexplained. Users can then look for all observation sequences that are totally (or partially) unexplained with a probability exceeding a threshold that they specify. We show important properties of our mathematical model that can be leveraged to speed up the search for unexplained sequences. We define algorithms to find top-k totally and partially unexplained sequences. We develop a prototype implementation and report on experiments using two video data sets and a cyber security dataset showing that the algorithms work well in practice, both from an efficiency perspective and an accuracy perspective.

The paper starts (Section 2) with an overview of related work. Section 3 provides basic definitions of stochastic activities slightly extending [1]. Section 4 defines the probability that a sequence is totally (or partially) unexplained. We also define the problem of finding the top-k (totally or partially) unexplained sequences and classes. Section 5 derives theorems that enable fast search for totally and partially unexplained sequences. Section 6 presents algorithms for solving the problems introduced in Section 4. Section 7 describes our experiments.

2 Related Work

We are not aware of domain-independent prior work on discovering unexplained sequences. However, specific work in the domains of video and cyber-security have focused on anomalous activity detection.

2.1 Video Analysis

A Priori Definitions. Several researchers have studied how to search for specifically defined patterns of normal/abnormal activities [2]. [3] studies how HMMs can be used to recognize complex activites, while [4] and [5] use coupled HMMs. [6] uses Dynamic Bayesian Networks (DBNs) to capture causal relationships between observations and hidden states. [1] developed a stochastic automaton based language to detect activities in video, while [7] presented an HMM-based algorithm. *In contrast, this paper starts with a set $\mathcal{A}$ of activity models (corresponding to innocuous/dangerous activities) and finds observation sequences that are not sufficiently explained by the models in $\mathcal{A}$.* Such unexplained sequences reflect activity occurrences that differ from the application's expectations.

Learning and then detecting abnormality. Several researchers first learn normal activity models and then detect abnormal/unusual events. [8] suggests a semi-supervised approach to detect abnormal events that are rare, unexpected, and relevant. We do not require "unexplained" events to either be rare or relevant. [9] uses HMMs to detect rare events, while [10] defines an anomaly as an atypical behavior pattern that is not represented by sufficient samples in a training dataset and satisfies an abnormal pattern. [11] defines abnormality as unseen or rarely occurring events—an initial video is used to learn normal behaviors. [12] shows how to detect users with abnormal activities from sensors attached to human bodies. An abnormal activity is defined as "an event that occurs rarely and has not been expected in advance". The same notion of abnormal activity is considered in [13] and [14]. [15] learns patterns of activities over time in an unsupervised way. [16] detects individual anomalies in crowd scenes—an anomaly is defined as a rare or infrequent behavior compared to all other behaviors. Common activities are accepted as normal and infrequent activity patterns are flagged as abnormal. All these approaches first learn normal activity models and then detect abnormal/unusual events. These papers differ from ours as they consider rare events to be abnormal. In contrast, we consider activities to be unexplained even if they are not rare and the available models are not able to capture them. For example, if a new way to break into cars has occurred many times (and we do not have a model for it), then we want to flag sequences where those activities occur as "unexplained" even if they are not rare. In addition, if a model exists for a rare activity, we would flag it as "explained", while many of these frameworks would not.

Similarity-based abnormality. [17] proposes an unsupervised technique in which no explicit models of normal activities are built. Each event in the video is compared with all other observed events to determine how many similar events exist. Unusual events are events for which there are no similar events in the video. Hence, this work also considers unusual activity as a rare event and a large number of observations is required to verify if an activity is unusual. [18] uses a similar approach: a scene is considered anomalous when the maximum similarity between the scene and all previously viewed scenes is below a threshold. In [19], frequently occurring patterns are normal and patterns that are dissimilar from most patterns are anomalous. [20] learns trajectory prototypes and detects anomalous behaviors when visual trajectories deviate from the learned representations of typical behaviors. An unsupervised approach, where an abnormal trajectory refers to something that has never (or rarely) seen, has been proposed in [21]. A normal trajectory is intended to be one similar enough to one or more trajectories that the system already knows. In [3], activities performed by a group of moving and interacting objects are modeled as shapes and abnormal activities are defined as a change in the shape activity model. In the context of elder care, [22] proposes an approach that first analyzes and designs features, and then detects abnormal activities using a method based on the designed features and Support Vector Data Description. [23] proposes a methodology to characterize novel scenes over long time periods without a priori knowledge. A hierarchical modeling process, characterizing an activity at multiple levels of resolution, is developed to classify and predict future activities and detect abnormal behavior.

Other relevant work. In [24], unusual events are detected by monitoring the scene with *monitors* which extract local low-level observations from the video stream. The monitor computes the likelihood of a new observation with respect to the probability distribution of prior observations. If the likelihood falls below a threshold, then the monitor outputs an alert. The local alerts issued by the monitors are then combined. [25] automatically learns high frequency events (taking spatio-temporal aspects into account) and declares them normal—events deviating from these rules are anomalies. [26] learns storylines from weakly labeled videos. A storyline includes the actions that occur in a video and their causal relationships. AND-OR graphs are used to represent storyline models.

The notion of unexplained sequences used in this paper has been proposed in [27].

2.2 Cyber Security

Intrusion detection systems (IDSs) monitor network traffic for suspicious behavior and trigger alerts [28], [29], [30]. Alert correlation methods aggregate such alerts into multi-step attacks [31], [32], [33], [34], [35], [36].

Intrusion detection. Intrusion detection techniques can be broadly classified into *signature-based* [30] and *profile-based* (or *anomaly-based*) [29] methods. A signature refers to a set of conditions that characterize intrusion activities w.r.t. packet headers and payload content. Historically, signature-based methods have been used extensively to detect malicious activities. On the other hand, in profile-based methods, a known deviation from the norm is considered anomalous (e.g. HTTP traffic on a non-standard port).

In contrast, in this paper, we consider the case where we have a set $\mathcal{A}$ of known activities (both innocuous and dangerous)—and we are looking for observation sequences that cannot be explained by either (if they were, they would constitute patterns that were known a priori). These need to be flagged as they might represent "zero day" attacks—attacks that were never seen before and vary significantly from past known access patterns.

Correlation techniques. The goal of correlation is to find causal relationships between alerts in order to reconstruct attacks from isolated alerts. The main role of correlation is to provide a higher level view of the actual attacks [33], [35], [34], [37], [38], [39].

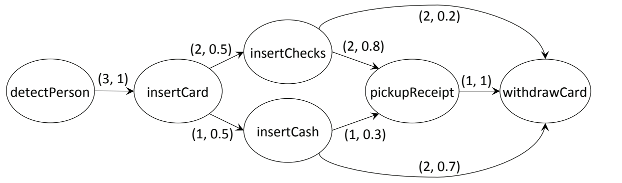


Fig. 2: Example of stochastic activity: ATM deposit

IDSs and correlation techniques rely on models encoding a priori knowledge of either normal or malicious behavior, and cannot appropriately deal with events that are not explained by the underlying models.

The framework and algorithms for identifying unexplained sequences presented in this paper are domain independent and may be applied to any domain including both activity detection in video and in cyber-security.

3 Basic Activity Model

This section extends the stochastic activity model of [1] by adding a function δ which expresses a constraint on the maximum "temporal distance" between two actions in an activity (though we make no claims of novelty for this).

We assume the existence of a finite set $\mathcal{S}$ of *action symbols*, corresponding to observable atomic actions. For instance, in the video domain, action symbols might be recognized by sophisticated image processing algorithms, while in the cyber-security domain, they may simply be read from a log file. Though our unexplained sequence detection framework is domain-independent, in some domains such as video surveillance, the problem of recognizing low-level actions in video can be a big challenge.

Definition 3.1 (Stochastic activity): A *stochastic activity* is a labeled directed graph $A = (V, E, \delta, \rho)$ where

- V is a finite set of nodes labeled with action symbols from $\mathcal{S}$;
- $E \subseteq V \times V$ is a set of edges;
- $\delta : E \rightarrow \mathbb{N}^+$ associates, with each edge $\langle v_i, v_j \rangle$, an upper bound on the time that can elapse between v_i and v_j;
- ρ is a function that associates, with each node $v \in V$ having out-degree 1 or more, a probability distribution on $\{\langle v, v' \rangle \mid \langle v, v' \rangle \in E\}$, i.e., $\sum_{\langle v, v' \rangle \in E} \rho(\langle v, v' \rangle) = 1$;
- there exists at least one *start node* in the activity definition, i.e. $\{v \in V \mid \nexists\, v' \in V \ s.t. \ \langle v', v \rangle \in E\} \neq \emptyset$;
- there exists at least one *end node* in the activity definition, i.e. $\{v \in V \mid \nexists\, v' \in V \ s.t. \ \langle v, v' \rangle \in E\} \neq \emptyset$.

Figure 2 shows a stochastic activity of deposits at an Automatic Teller Machine (ATM). Each edge e is labeled with $(\delta(e), \rho(e))$. For an edge $e = \langle s_1, s_2 \rangle$, $\delta(e)$ specifies the maximum time between when s_1 is observed and when s_2 is observed. ρ specifies the probability of going from one node to another (the probability distribution associated with a node gives the transition probability from that node). For instance, the two edges starting at node insertCard mean that there is a 50% probability of going to node insertChecks and a 50% probability of going to node insertCash from node insertCard. In addition, insertChecks and insertCash must follow insertCard within 2 and 1 time units, respectively.

In general, each node of a stochastic activity definition is something that can be detected by application code. For instance, if we are tracking activities in video, each node in a stochastic activity would be something that can be detected by an image processing program, e.g. "Detect Person" in Figure 2 may be identified as holding only if a probabilistic face recognition program returns a probability over some threshold that a given frame (or block of frames) contains a face in it. Likewise, in a cybersecurity application, a node in an activity such as "Attempted login" may only be identified as occurring if a log file archives a login attempt. For the sake of simplicity, we use "high level" descriptions of nodes in our examples, as opposed to low level descriptions (e.g., the color histogram of a given image shows over 70% of the colors are a certain shade).

An instance of a stochastic activity A is a path in A from a start node to an end node.

Definition 3.2 (Stochastic activity instance): An *instance* of a stochastic activity (V, E, δ, ρ) is a sequence $\langle s_1, \ldots, s_m \rangle$ of nodes in V such that

- $\langle s_i, s_{i+1} \rangle \in E$ for $1 \leq i < m$;
- $\{s \mid \langle s, s_1 \rangle \in E\} = \emptyset$, i.e., s_1 is a start node; and
- $\{s \mid \langle s_m, s \rangle \in E\} = \emptyset$, i.e., s_m is an end node.

The probability of the instance is $\prod_{i=1}^{m-1} \rho(\langle s_i, s_{i+1} \rangle)$.

In Figure 2, $\langle$detectPerson, insertCard, insertCash, withdrawCard$\rangle$ is an instance with probability 0.35. Throughout this paper, we assume an arbitrary but fixed set $\mathcal{A}$ of stochastic activities.

The preceding definitions do not take observation sequences into account. In order to define when activity occurrences are detected in a sequence of time-stamped observation data, we first need to formally define an observation sequence. An *observation sequence* is a finite sequence of *observation IDs*. An observation ID (OID) f has an associated timestamp, denoted $f.ts$, and an associated set of action symbols, denoted $f.obs$. Without loss of generality, we assume timestamps to be positive integers. For instance, if our observation sequence is a video, then the OIDs may be frame IDs with $f.ts$ being the timestamp associated with frame f and $f.obs$ being the actions detected in frame f. On the other hand, if our observation sequence is a sequence of transactions at a website, the OIDs are transaction IDs, $f.ts$ is the timestamp associated with transaction f, and $f.obs$ are the actions associated with transaction f.

Example 3.1 (Video example): An observation sequence might be a video $v = \langle f_1, f_2, f_3, f_4, f_5 \rangle$, where the f_i's are frame IDs, $f_i.ts = i$ for $1 \leq i \leq 5$, $f_1.obs = \{\text{detectPerson}\}$, $f_2.obs = \{\text{insertCard}\}$, $f_3.obs = \{\text{insertCash}\}$, $f_4.obs = \{\text{withdrawCash}\}$, $f_5.obs = \{\text{withdrawCard}\}$. Notice that withdrawCash in frame f_4 does not appear in the stochastic activity of Figure 2. In general, action symbols may be detected in a frame even if they do not appear in the definition of a stochastic activity because it is irrelevant for that activity.

Throughout the paper, we use the following terminology and notation for (general) sequences. Suppose $S_1 = \langle a_1, \ldots, a_n \rangle$ and $S_2 = \langle b_1, \ldots, b_m \rangle$ are two sequences. S_2 is a *subsequence* of S_1 iff there exist $1 \leq j_1 < j_2 < \ldots < j_m \leq n$ s.t. $b_i = a_{j_i}$ for $1 \leq i \leq m$. If $j_i = j_{i+1} - 1$ for $1 \leq i < m$, then S_2 is a *contiguous* subsequence of S_1. We write $S_1 \cap S_2 \neq \emptyset$ iff S_1 and S_2 have a common element and write $e \in S_1$ iff e is an element appearing in S_1. The *concatenation* of S_1 and S_2, i.e., the sequence $\langle a_1, \ldots, a_n, b_1, \ldots, b_m \rangle$, is denoted by $S_1 \cdot S_2$. Finally, $|S_1|$ denotes the number of elements in S_1.

We now define an occurrence of a stochastic activity in an observation sequence.

Definition 3.3 (Activity occurrence): Let v be an observation sequence and $A = (V, E, \delta, \rho)$ a stochastic activity. An *occurrence* o of A in v is a sequence $\langle (f_1, s_1), \ldots, (f_m, s_m) \rangle$ such that

- $\langle f_1, \ldots, f_m \rangle$ is a subsequence of v,
- $\langle s_1, \ldots, s_m \rangle$ is an instance of A,
- $s_i \in f_i.obs$, for $1 \leq i \leq m$, and [1]
- $f_{i+1}.ts - f_i.ts \leq \delta(\langle s_i, s_{i+1} \rangle)$, for $1 \leq i < m$.

The probability of o, denoted $p(o)$, is the probability of the instance $\langle s_1, \ldots, s_m \rangle$.

When concurrently monitoring multiple activities, shorter activity instances generally tend to have higher probability. To remedy this, we normalize occurrence probabilities by introducing the relative probability $p^*(o)$ of an occurrence o of activity A as $p^*(o) = \frac{p(o)}{p_{max}}$, where p_{max} is the highest probability of any instance of A.

Example 3.2 (Video example): Consider the video of Example 3.1. An occurrence of the activity of Figure 2 is $o = \langle (f_1, \text{detectPerson}), (f_2, \text{insertCard}), (f_3, \text{insertCash}), (f_5, \text{withdrawCard}) \rangle$, and $p^*(o) = 0.875$. Notice that if the edge going from insertCash to withdrawCard was labeled with 1 by δ, then o would not have been an activity occurrence because withdrawCard was required to follow insertCash within at most 1 time unit, whereas it occurs after 2 time units in the video.

We use $\mathcal{O}(v)$ to denote the set of all activity occurrences in v. Whenever v is clear from the context, we write $\mathcal{O}$ instead of $\mathcal{O}(v)$.

The next section describes our framework for discovering unexplained sequences in an application-independent manner. It is worth noting that the actual input of the framework consists of an observation sequence and a set of activity occurrences (each with a probability). Though our framework is domain-independent, there can be challenges in providing the observations associated with OIDs in some domains (e.g. video surveillance, where identifying the low level actions in a video frame can be highly non-trivial).

1. With a slight abuse of notation, we use s_i to refer to both node s_i and the action symbol labeling it.

4 Unexplained Sequence Probability Model

This section defines the probability that an observation sequence is unexplained by $\mathcal{A}$. We note that the occurrence of an activity in an observation sequence can involve conflicts. For instance, consider the activity occurrence o in Example 3.2 and suppose there is a second activity occurrence o' such that $(f_1, \mathsf{detectPerson}) \in o'$. In this case, there is an implicit conflict because $(f_1, \mathsf{detectPerson})$ belongs to both occurrences, but in fact, detectPerson can only belong to one activity occurrence, i.e. though o and o' may both have a non-zero probability, the probability that these two activity occurrences coexist is 0. Formally, we say two activity occurrences o, o' *conflict*, denoted $o \nsim o'$, iff $o \cap o' \neq \emptyset$. We now use this to define possible worlds.

Definition 4.1 (Possible world): Let $\mathcal{O}$ be the set of all activity occurrences in an observation sequence v. A *possible world* for v is a subset w of $\mathcal{O}$ s.t. $\nexists o_i, o_j \in w, o_i \nsim o_j$.

Thus, a possible world is a set of activity occurrences which do not conflict with one another, i.e., an action symbol in an OID cannot belong to two distinct activity occurrences in the same world. We use $\mathcal{W}(v)$ to denote the set of all possible worlds for an observation sequence v; whenever v is clear from the context, we simply write $\mathcal{W}$.

Example 4.1 (Video example): Consider a video with two conflicting occurrences o_1, o_2. There are 3 possible worlds: $w_0 = \emptyset$, $w_1 = \{o_1\}$, and $w_2 = \{o_2\}$. Note that $\{o_1, o_2\}$ is not a world as $o_1 \nsim o_2$. Each world represents a way of explaining what is observed. The first world corresponds to the case where nothing is explained, the second and third worlds correspond to the scenarios where we use one of the two possible occurrences to explain the observed action symbols.

Note that any subset of $\mathcal{O}$ not containing conflicting occurrences is a legitimate possible world—possible worlds are not required to be maximal w.r.t. $\subseteq$. In the above example, the empty set is a possible world even though there are two other possible worlds $w_1 = \{o_1\}$ and $w_2 = \{o_2\}$ which are supersets of it. The reason is that o_1 and o_2 are uncertain, so the scenario where neither o_1 nor o_2 occurs is a legitimate one. We illustrate this below.

Example 4.2 (Video example): Suppose we have a video where a single occurrence o has $p^*(o) = 0.6$. In this case, it is natural to say that there are two possible worlds $w_0 = \emptyset$ and $w_1 = \{o\}$ and expect the probabilities of w_0 and w_1 to be 0.4 and 0.6, respectively. By restricting ourselves to maximal possible worlds only, we would have only one possible world, w_1, whose probability is 1, which is wrong.

It is worth noting that the problem of finding possible worlds corresponds to the problem of finding the independent sets of a graph: occurrences are vertices, conflicts are edges, possible worlds are independent sets. Thus, algorithms to find maximal independent sets can be directly applied to compute possible worlds—all possible worlds can be simply obtained by taking all subsets of the maximal independent sets. An efficient algorithm for generating all the maximal independent sets has been proposed in [40], processing time and memory space are bounded by $O(nm\mu)$ and $O(n+m)$, respectively, where n, m, and μ are the numbers of vertices (occurrences in our case), edges (conflicts in our case), and maximal independent sets (possible worlds in our case) of a graph.

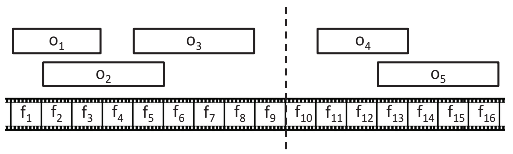


Fig. 3: Conflict-Based Partitioning of a video

We use $\overset{*}{\nsim}$ to denote the transitive closure of $\nsim$. Clearly, $\overset{*}{\nsim}$ is an equivalence relation and determines a partition of $\mathcal{O}$ into equivalence classes $\mathcal{O}_1, \ldots, \mathcal{O}_m$. Here the basic idea is to partition the observation sequence into subsequences containing occurrences that conflict directly or in a "transitive" way (we will formally define this with the notion of a *Conflict-Based Partitioning* in Definition 4.2 and illustrate it in Example 4.4). Equivalence classes that temporally overlap are collapsed into a single one.[2]

Example 4.3 (Video example): Suppose we have a video $v = \langle f_1, \ldots, f_{16}\rangle$ s.t. five occurrences o_1, o_2, o_3, o_4, o_5 are detected as depicted in Figure 3, that is, $o_1 \nsim o_2$, $o_2 \nsim o_3$, and $o_4 \nsim o_5$. There are two equivalence classes determined by $\overset{*}{\nsim}$, namely $\mathcal{O}_1 = \{o_1, o_2, o_3\}$ and $\mathcal{O}_2 = \{o_4, o_5\}$.

The equivalence classes determined by $\overset{*}{\nsim}$ lead to a conflict-based partitioning of an observation sequence.

Definition 4.2 (Conflict-Based Partitioning): Let v be an observation sequence and $\mathcal{O}_1, \ldots, \mathcal{O}_m$ the equivalence classes determined by $\overset{*}{\nsim}$. A *Conflict-Based Partitioning* (CBP) of v is a sequence $\langle v_1, \ldots, v_m\rangle$ such that:

- $v_1 \cdot \ldots \cdot v_m = v$, and
- $\mathcal{O}(v_i) = \mathcal{O}_i$, for $1 \leq i \leq m$.

The v_i's are called *segments*.

Example 4.4 (Video example): A CBP of the video in Example 4.3 is $\langle v_1, v_2\rangle$, where $v_1 = \langle f_1, \ldots, f_9\rangle$ and $v_2 = \langle f_{10}, \ldots, f_{16}\rangle$. Another partitioning of the same video is the one where $v_1 = \langle f_1, \ldots, f_{10}\rangle$ and $v_2 = \langle f_{11}, \ldots, f_{16}\rangle$.

Thus, activity occurrences determine a set of possible worlds (different ways of explaining an observation sequence). We wish to find a probability distribution over all possible worlds that (i) is consistent with the relative probabilities of the occurrences, and (ii) takes conflicts into account. We assume the user specifies a function $Weight : \mathcal{A} \rightarrow \mathbb{R}^+$ which assigns a weight to each activity and prioritizes the importance of the activity.[3] The weight

2. Two equivalence classes $\mathcal{O}_i$ and $\mathcal{O}_j$ temporally overlap iff $[min(\mathcal{O}_i), max(\mathcal{O}_i)] \cap [min(\mathcal{O}_j), max(\mathcal{O}_j)] \neq 0$, where $min(\mathcal{O}_i) = \min\{f.ts \mid \exists o \in \mathcal{O}_i, (f, s) \in o\}$, $max(\mathcal{O}_i) = \max\{f.ts \mid \exists o \in \mathcal{O}_i, (f, s) \in o\}$, and $min(\mathcal{O}_j)$, $max(\mathcal{O}_j)$ are analogously defined.

3. For instance, highly threatening activities may be assigned a high weight.

of an occurrence o of activity A is the weight of A. We use $C(o)$ to denote the set of occurrences conflicting with o, i.e., $C(o) = \{o' \mid o' \in \mathcal{O} \wedge o' \nsim o\}$. Note that $o \in C(o)$; and $C(o) = \{o\}$ when o does not conflict with any other occurrence. Finally, we assume that activity occurrences belonging to different segments are independent events. Suppose p_w denotes the (unknown) probability of world w. As we know the probability of occurrences, and as each occurrence occurs in certain worlds, we can induce a set of nonlinear constraints that will subsequently be used to learn the values of the p_w's.

Definition 4.3: Let v be an observation sequence and $\mathcal{O}_1, \dots, \mathcal{O}_m$ the equivalence classes determined by $\overset{*}{\nsim}$. We define the non-linear constraints $NLC(v)$ as follows:

$$\begin{cases} p_w \geq 0, \quad \forall w \in \mathcal{W} \\ \sum_{w \in \mathcal{W}} p_w = 1 \\ \sum_{w \in \mathcal{W}\, s.t.\, o \in w} p_w = p^*(o) \cdot \frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}, \forall o \in \mathcal{O} \\ p_w = \prod_{k=1}^{m} \sum_{w' \in \mathcal{W}\, s.t.\, w' \cap \mathcal{O}_k = w \cap \mathcal{O}_k} p_{w'} \qquad \forall w \in \mathcal{W} \end{cases}$$

The first two types of constraints enforce a probability distribution over the set of possible worlds. The third type of constraint ensures that the probability of occurrence o—which is the sum of the probabilities of the worlds containing o—is equal to its relative probability $p^*(o)$ weighted by $\frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}$. Note that: (i) the value on the right-hand side of the third type of constraint decreases as the amount of conflict increases, (ii) if an occurrence o is not conflicting with any other occurrence, then its probability $\sum_{w \in \mathcal{W}\, s.t.\, o \in w} p_w$ is equal to $p^*(o)$, i.e., the probability returned by the stochastic automaton. The last kind of constraint reflects independence between segments. In general $NLC(v)$ might admit multiple solutions.

Example 4.5 (Video example): Consider a single-segment video consisting of frames $f_1, \dots, f_9$ (cf. Figure 3). Suppose o_1, o_2, o_3 have been detected with relative probabilities 0.3, 0.6, and 0.5, respectively. Suppose the weights of o_1, o_2, o_3 are 1, 2, 3, respectively. Five worlds are possible: $w_0 = \emptyset$, $w_1 = \{o_1\}$, $w_2 = \{o_2\}$, $w_3 = \{o_3\}$, and $w_4 = \{o_1, o_3\}$. Then, $NLC(v)$ is:[4]

$$\begin{array}{ll} p_i \geq 0 & 0 \leq i \leq 4 \\ p_0 + p_1 + p_2 + p_3 + p_4 = 1 & \\ p_1 + p_4 = 0.3 \cdot \frac{1}{3} & \\ p_2 = 0.6 \cdot \frac{1}{3} & \\ p_3 + p_4 = 0.5 \cdot \frac{3}{5} & \end{array}$$

which has multiple solutions. One solution is $p_0 = 0.4$, $p_1 = 0.1$, $p_2 = 0.2$, $p_3 = 0.3$, $p_4 = 0$. Another solution is $p_0 = 0.5$, $p_1 = 0$, $p_2 = 0.2$, $p_3 = 0.2$, $p_4 = 0.1$.

In the rest of the paper, we assume that $NLC(v)$ is solvable.[5] We say that a sequence $S = \langle (f_1, s_1), \dots, (f_n, s_n) \rangle$ *occurs in* an observation sequence v iff $\langle f_1, \dots, f_n \rangle$ is a contiguous subsequence of v and $s_i \in f_i.obs$ for $1 \leq i \leq n$. We give two semantics for S to be unexplained in a world $w \in \mathcal{W}$. Intuitively, S is *totally* (resp. *partially*) unexplained in w iff w does not explain every (resp. at least one) symbol of S. More formally:

1) S is *totally unexplained* in w, denoted $w \not\models_T S$, iff $\forall (f_i, s_i) \in S, \nexists o \in w, (f_i, s_i) \in o$;
2) S is *partially unexplained* in w, denoted $w \not\models_P S$, iff $\exists (f_i, s_i) \in S, \nexists o \in w, (f_i, s_i) \in o$.

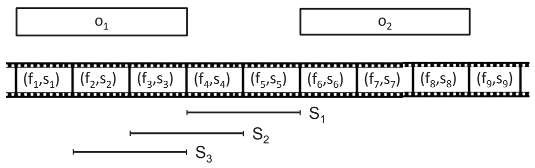


Fig. 4: Totally and partially unexplained sequences

Example 4.6 (Video example): Suppose we have a video $v = \langle f_1, \dots, f_9 \rangle$ such that $f_i.obs = \{s_i\}$, $1 \leq i \leq 9$, and two occurrences o_1 and o_2 are detected (cf. Figure 4). The four possible worlds are: $w_0 = \emptyset$, $w_1 = \{o_1\}$, $w_2 = \{o_2\}$, $w_3 = \{o_1, o_2\}$. Let $S_1 = \langle (f_4, s_4), (f_5, s_5) \rangle$, $S_2 = \langle (f_3, s_3), (f_4, s_4) \rangle$, $S_3 = \langle (f_2, s_2), (f_3, s_3) \rangle$ be sequences occurring in v. S_1 is totally (and partially) unexplained in every world. S_2 is totally unexplained in w_0 and w_2 but not in w_1 and w_3; moreover, S_2 is partially unexplained in every world. S_3 is totally and partially unexplained in w_0 and w_2 but not in w_1 and w_3.

We now define the probability of a sequence in an observation sequence being totally/partially unexplained.

Definition 4.4: Let S be a sequence occurring in an observation sequence v. The probability interval that S is totally unexplained in v is $\mathcal{I}_T(S) = [l, u]$, where:

$$\begin{array}{c} l = \textbf{minimize} \sum_{w \in \mathcal{W}\ s.t.\ w \not\models_T S} p_w \\ \textbf{subject to } NLC(v) \\ u = \textbf{maximize} \sum_{w \in \mathcal{W}\ s.t.\ w \not\models_T S} p_w \\ \textbf{subject to } NLC(v) \end{array}$$

The probability interval that S is partially unexplained in v is $\mathcal{I}_P(S) = [l', u']$, where l', u' are derived in exactly the same way as l, u above by replacing the $\not\models_T$ symbols in the above optimization problems by $\not\models_P$.

Thus, the probability that a sequence S occurring in v is totally (resp. partially) unexplained w.r.t. a solution of $NLC(v)$ is the sum of the probabilities of the worlds in which S is totally (resp. partially) unexplained. As $NLC(v)$ may have multiple solutions, we find the tightest interval $[l, u]$ (resp. $[l', u']$) containing this probability for any solution. Different criteria can be used to infer a value from an interval $[l, u]$, e.g. the MIN l, the MAX u, the average (i.e., $(l + u)/2$), etc. The only requirement is that this value has to be in $[l, u]$. We henceforth assume that such criterion has been chosen—$\mathcal{P}_T(S)$ (resp. $\mathcal{P}_P(S)$) denotes the probability that S is totally (resp. partially) unexplained.

The following proposition says that the probability that a sequence is totally (resp. partially) unexplained is no higher

4. For brevity, we do not explicitly list the independence constraints.
5. This can be easily checked via both a non-linear constraint solver, as well as methods developed in the next section.

(resp. lower) than the probability of any subsequence.

Proposition 4.1: Consider two sequences S_1 and S_2 occurring in an observation sequence. If S_1 is a subsequence of S_2, then $\mathcal{P}_T(S_1) \geq \mathcal{P}_T(S_2)$ and $\mathcal{P}_P(S_1) \leq \mathcal{P}_P(S_2)$.

We now define totally and partially unexplained sequences.

Definition 4.5 (Unexplained sequences): Let v be an observation sequence, $\tau \in [0,1]$ a probability threshold, and $L \in \mathbb{N}^+$ a length threshold. A sequence S occuring in v is:

- A *totally unexplained sequence* if (i) $\mathcal{P}_T(S) \geq \tau$, (ii) $|S| \geq L$, and (iii) S is maximal, i.e., there is no sequence $S' \neq S$ occurring in v s.t. S is a subsequence of S', $\mathcal{P}_T(S') \geq \tau$, and $|S'| \geq L$.
- A *partially unexplained sequence* if (i) $\mathcal{P}_P(S) \geq \tau$, (ii) $|S| \geq L$, and (iii) S is minimal, i.e., there is no sequence $S' \neq S$ occurring in v s.t. S' is a subsequence of S, $\mathcal{P}_P(S') \geq \tau$, and $|S'| \geq L$.

In this definition, L is the minimum length a sequence must be for it to be considered possibly unexplained. Totally unexplained sequences (TUSs for short) S have to be maximal because once we find S, any sub-sequence of it is (totally) unexplained with probability greater than or equal to that of S. On the other hand, partially unexplained sequences (PUSs for short) S' have to be minimal because once we find S', any super-sequence of it is (partially) unexplained with probability greater than or equal to that of S'.

Intuitively, an unexplained sequence is a sequence of action symbols that are observed in the observation sequence and poorly explained by known activity models. Such sequences might correspond to unknown variants of known activities or to entirely new—and unknown—activities.

An *Unexplained Sequence Problem* (USP) instance is a triple $I = \langle v, \tau, L \rangle$ where v is an observation sequence, $\tau \in [0,1]$ is a probability threshold, and $L \in \mathbb{N}^+$ is a length threshold. We want to find the sets $\mathcal{A}^{tu}(I)$ and $\mathcal{A}^{pu}(I)$ of all totally and partially unexplained sequences, respectively. When I is clear from context, we will drop it.

The following definition introduces the top-k totally and partially unexplained sequences. Intuitively, these are k unexplained sequences having maximum probability.

Definition 4.6 (Top-k unexplained sequences): Consider a USP instance and let $k \in \mathbb{N}^+$. $\mathcal{A}^{tu}_k \subseteq \mathcal{A}^{tu}$ (resp. $\mathcal{A}^{pu}_k \subseteq \mathcal{A}^{pu}$) is a set of top-$k$ totally (resp. partially) unexplained sequences iff $|\mathcal{A}^{tu}_k| = \min\{k, |\mathcal{A}^{tu}|\}$ (resp. $|\mathcal{A}^{pu}_k| = \min\{k, |\mathcal{A}^{pu}|\}$), and $\forall S \in \mathcal{A}^{tu}_k, \forall S' \in \mathcal{A}^{tu} - \mathcal{A}^{tu}_k$ (resp. $\forall S \in \mathcal{A}^{pu}_k, \forall S' \in \mathcal{A}^{pu} - \mathcal{A}^{pu}_k$) $\mathcal{P}_T(S) \geq \mathcal{P}_T(S')$ (resp. $\mathcal{P}_P(S) \geq \mathcal{P}_P(S')$).

Suppose we have a USP instance. For any $S, S' \in \mathcal{A}^{tu}$ (resp. $S, S' \in \mathcal{A}^{pu}$), we write $S =_T S'$ (resp. $S =_P S'$) iff $\mathcal{P}_T(S) = \mathcal{P}_T(S')$ (resp. $\mathcal{P}_P(S) = \mathcal{P}_P(S')$). Obviously, $=_T$ (resp. $=_P$) is an equivalence relation and determines a set $\mathcal{C}^{tu}$ (resp. $\mathcal{C}^{pu}$) of equivalence classes. For any equivalence class $C \in \mathcal{C}^{tu}$ (resp. $C \in \mathcal{C}^{pu}$) we define $\mathcal{P}_T(C)$ (resp. $\mathcal{P}_P(C)$) as the (unique) probability of the sequences in C.

Symbol	Description
$\mathcal{A}$	Set of stochastic activities
s	Action symbol
f	Observation ID (OID)
$f.ts$	Timestamp associated with observation ID f
$f.obs$	Set of action symbols associated with observation ID f
v	Observation sequence
o and $\mathcal{O}$	Activity occurrence and set of activity occurrences
w and $\mathcal{W}$	Possible world and set of possible worlds
$\langle v_1, \ldots, v_m \rangle$	Conflict-based partitioning (CBP) of observation sequence v. Each v_i is called a *segment*
$NLC(v)$	Set of non-linear constraints for observation sequence v
$LC(v)$	Set of linear constraints for observation sequence v
$w \not\models_T S$	Sequence S is totally unexplained in world w
$w \not\models_P S$	Sequence S is partially unexplained in world w
$\mathcal{I}_T(S)$	Probability interval that sequence S is totally unexplained
$\mathcal{I}_P(S)$	Probability interval that sequence S is partially unexplained
$\mathcal{P}_T(S)$	(Point) Probability that sequence S is totally unexplained
$\mathcal{P}_P(S)$	(Point) Probability that sequence S is partially unexplained

TABLE 1: Notation

Compared with the top-k unexplained sequences, the top$-k$ unexplained classes find *all* the unexplained sequences having the k highest probabilities.

Definition 4.7 (Top-k unexplained classes): Consider a USP instance and let $k \in \mathbb{N}^+$. $\mathcal{C}^{tu}_k \subseteq \mathcal{C}^{tu}$ (resp. $\mathcal{C}^{pu}_k \subseteq \mathcal{C}^{pu}$) is the set of top-$k$ totally (resp. partially) unexplained classes iff $|\mathcal{C}^{tu}_k| = \min\{k, |\mathcal{C}^{tu}|\}$ (resp. $|\mathcal{C}^{pu}_k| = \min\{k, |\mathcal{C}^{pu}|\}$), and $\forall C \in \mathcal{C}^{tu}_k, \forall C' \in \mathcal{C}^{tu} - \mathcal{C}^{tu}_k$ (resp. $\forall C \in \mathcal{C}^{pu}_k, \forall C' \in \mathcal{C}^{pu} - \mathcal{C}^{pu}_k$) $\mathcal{P}_T(C) > \mathcal{P}_T(C')$ (resp. $\mathcal{P}_P(C) > \mathcal{P}_P(C')$).

Table 1 summarizes the main notation used in the paper.

5 Properties of USPs

This section derives properties that can be leveraged (in the next section) to devise efficient algorithms to solve USPs. We first show an interesting property concerning the solution of $NLC(v)$ (some later results rely on it); the following two subsections consider specific properties for totally and partially unexplained sequences.

For a given observation sequence v, we show that if $\langle v_1, \ldots, v_m \rangle$ is a CBP, then we can find solutions of the *non-linear* constraints $NLC(v)$ by solving m *smaller sets of linear constraints.*[6] Let $LC(v)$ be the set of linear constraints of $NLC(v)$ (i.e., all constraints of Definition 4.3 except for the last kind). Henceforth, we use $\mathcal{W}$ to denote $\mathcal{W}(v)$ and $\mathcal{W}_i$ to denote $\mathcal{W}(v_i)$, $1 \leq i \leq m$. A solution of $NLC(v)$ is a mapping $\mathcal{P} : \mathcal{W} \rightarrow [0,1]$ which satisfies $NLC(v)$. Likewise, a solution of $LC(v_i)$ is a mapping $\mathcal{P}_i : \mathcal{W}_i \rightarrow [0,1]$ which satisfies $LC(v_i)$. It is important to note that $\mathcal{W} = \{w_1 \cup \ldots \cup w_m \mid w_i \in \mathcal{W}_i, 1 \leq i \leq m\}$.

Theorem 1: Let v be an observation sequence and $\langle v_1, \ldots, v_m \rangle$ a CBP. $\mathcal{P}$ is a solution of $NLC(v)$ iff $\forall i \in [1,m]$ there exists a solution $\mathcal{P}_i$ of $LC(v_i)$

6. This yields two benefits: (i) It allows us to solve a smaller set of constraints. (ii) It allows us to solve linear constraints which are easier to solve than nonlinear ones. Moreover, it allows us to drastically reduce the space of possible worlds considered, as we can consider each segment v_i (and its corresponding possible worlds) individually, thereby avoiding the blow up we would get by combining possible worlds of different segments. This also applies to Theorems 2 and 4.

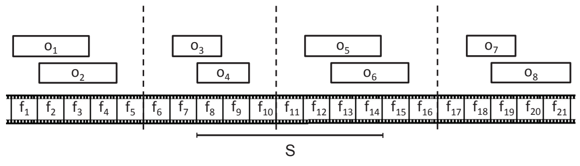


Fig. 5: Conflict-Based Partitioning of a video

s.t. $\mathcal{P}(\bigcup_{i=1}^{m} w_i) = \prod_{i=1}^{m} \mathcal{P}_i(w_i)$ for every $w_1 \in \mathcal{W}_1, \ldots, w_m \in \mathcal{W}_m$.

The following example illustrates the previous theorem.

Example 5.1 (Video example): Consider the video v of Example 4.3 (cf. Figure 3). As shown in Example 4.4, one possible CBP of v is $\langle v_1, v_2 \rangle$, where $v_1 = \langle f_1, \ldots, f_9 \rangle$ and $v_2 = \langle f_{10}, \ldots, f_{16} \rangle$. Theorem 1 says that for each solution $\mathcal{P}$ of $NLC(v)$, there is a solution $\mathcal{P}_1$ of $LC(v_1)$ and a solution $\mathcal{P}_2$ of $LC(v_2)$ s.t. $\mathcal{P}(w_1 \cup w_2) = \mathcal{P}_1(w_1) \times \mathcal{P}(w_2)$ for every $w_1 \in \mathcal{W}_1, w_2 \in \mathcal{W}_2$, and vice versa.

Consider an observation sequence v and let $\langle v_1, \ldots, v_m \rangle$ be a CBP. Given a sequence $S = \langle (f_1, s_1), \ldots, (f_q, s_q) \rangle$ occurring in v, we say that $v_i, v_{i+1}, \ldots, v_{i+n}$ ($1 \leq i \leq i+n \leq m$) are the segments *containing* S iff $f_1 \in v_i$ and $f_q \in v_{i+n}$. In other words, S spans the segments $v_i, v_{i+1}, \ldots, v_{i+n}$: it starts at a point in segment v_i (as v_i contains the first OID of S) and ends at some point in segment v_{i+n} (as v_{i+n} contains the last OID of S). S_k denotes the *projection* of S on the k-th segment v_k ($i \leq k \leq i+n$), that is, the subsequence of S containing all the pairs $(f, s) \in S$ with $f \in v_k$.

Example 5.2 (Video example): Suppose we have a video $v = \langle f_1, \ldots, f_{21} \rangle$ such that $f_i.obs = \{s_i\}$ for $1 \leq i \leq 21$. In addition, suppose 8 occurrences are detected as shown in Figure 5. Consider the CBP $\langle v_1, v_2, v_3, v_4 \rangle$, where $v_1 = \{f_1, \ldots, f_5\}$, $v_2 = \{f_6, \ldots, f_{10}\}$, $v_3 = \{f_{11}, \ldots, f_{16}\}$, and $v_4 = \{f_{17}, \ldots, f_{21}\}$. Consider now the sequence $S = \langle (f_8, s_8), \ldots, (f_{14}, s_{14}) \rangle$ occurring in v. Then, v_2 and v_3 are the segments containing S. Moreover, S_2 denotes $\langle (f_8, s_8), \ldots, (f_{10}, s_{10}) \rangle$, and S_3 denotes $\langle (f_{11}, s_{11}), \ldots, (f_{14}, s_{14}) \rangle$.

5.1 Totally unexplained sequences

The following theorem says that we can compute $\mathcal{I}_T(S)$ by solving LC (which are linear constraints) for each segment containing S (instead of solving a non-linear set of constraints for the whole observation sequence).

Theorem 2: Consider an observation sequence v. Let $\langle v_1, \ldots, v_m \rangle$ be a CBP and $\langle v_i, \ldots, v_{i+n} \rangle$ the segments containing a sequence S occurring in v. For $i \leq k \leq i+n$, let

$$\begin{aligned} l_k = &\ \textbf{minimize} \textstyle\sum_{w \in \mathcal{W}_k \ s.t. \ w \not\models_T S_k} p_w \\ &\ \textbf{subject to } LC(v_k) \\ u_k = &\ \textbf{maximize} \textstyle\sum_{w \in \mathcal{W}_k \ s.t. \ w \not\models_T S_k} p_w \\ &\ \textbf{subject to } LC(v_k) \end{aligned}$$

If $\mathcal{I}_T(S) = [l, u]$, then $l = \prod_{k=i}^{i+n} l_k$ and $u = \prod_{k=i}^{i+n} u_k$.

The following example illustrates the theorem above.

Example 5.3 (Video example): Consider Example 5.2, which is depicted in Figure 5. $\mathcal{I}_T(S)$ can be computed by solving the non-linear program of Definition 4.4 for the whole video v. But Theorem 2 says that $\mathcal{I}_T(S)$ can be computed as $\mathcal{I}_T(S) = [l_2 \times l_3, u_2 \times u_3]$, where l_2, u_2, l_3, u_3 are computed as defined in Theorem 2, i.e. by solving two smaller linear programs for v_2 and v_3.

The following theorem provides a sufficient condition for a pair (f, s) not to be included in any sequence S occurring in v and having $\mathcal{P}_T(S) \geq \tau$.

Theorem 3: Let $\langle v, \tau, L \rangle$ be a USP instance. Given (f, s) s.t. $f \in v$ and $s \in f.obs$, let $\varepsilon = \sum_{o \in \mathcal{O} \ s.t. \ (f,s) \in o} p^*(o) \cdot \frac{Weight(o)}{\sum_{o_j \in C(o)} Weight(o_j)}$. If $\varepsilon > 1 - \tau$, then there does not exist a sequence S occurring in v s.t. $(f, s) \in S$ and $\mathcal{P}_T(S) \geq \tau$.

If the above condition holds for a pair (f, s), then we say that (f, s) is *sufficiently explained. Note that to check whether a pair (f, s) is sufficiently explained, we do not need to solve any set of linear or non-linear constraints, since ε is computed by simply summing the (weighted) probabilities of the occurrences containing (f, s)*. Thus, this result yields a further efficiency. An OID f is *sufficiently explained* iff (f, s) is sufficiently explained for every $s \in f.obs$. If (f, s) is sufficiently explained, then it can be disregarded when identifying unexplained sequences. Moreover, this may allow us to disregard entire parts of observation sequences as shown in the example below.

Example 5.4 (Video example): Consider a USP instance $\langle v, \tau, L \rangle$ where $v = \langle f_1, \ldots, f_9 \rangle$ is s.t. $f_i.obs = \{s_i\}$ for $1 \leq i \leq 9$, as depicted in Figure 6.

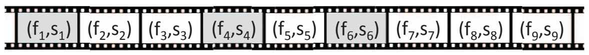


Fig. 6: Sufficiently explained frames in a video.

Suppose $L = 3$ and (f_1, s_1), (f_4, s_4), (f_6, s_6) are sufficiently explained. Even though the theorem is applicable to only a few (f_i, s_i) pairs, we see that no unexplained sequence can be found before f_7 as $L = 3$.

Given a USP instance $I = \langle v, \tau, L \rangle$ and a subsequence v' of v, v' is *relevant* iff (i) v' is a contiguous subsequence of v (ii) $|v'| \geq L$, (iii) $\forall f \in v'$, f is not sufficiently explained, and (iv) v' is maximal (i.e., there does not exist $v'' \neq v'$ s.t. v' is a subsequence of v'' and v'' satisfies (i), (ii), (iii)). We use $relevant(I)$ to denote the set of relevant observation subsequences.

Theorem 3 entails that relevant observation subsequences can be individually considered when looking for totally unexplained sequences because there is no totally unexplained sequence spanning two different relevant observation subsequences.

5.2 Partially unexplained sequences

The following theorem states that we can compute $\mathcal{I}_P(S)$ by solving NLC for the observation subsequence consisting of the segments containing S (instead of solving NLC for the whole observation sequence).

Theorem 4: Consider an observation sequence v. Let $\langle v_1, \ldots, v_m \rangle$ be a CBP and $\langle v_i, \ldots, v_{i+n} \rangle$ be the segments containing a sequence S occurring in v. Let $v^* = v_i \cdot \ldots \cdot v_{i+n}$. $\mathcal{I}_P(S)$ computed w.r.t. v is equal to $\mathcal{I}_P(S)$ computed w.r.t. v^*.

We now illustrate the use of the preceding theorem.

Example 5.5 (Video example): Consider Example 5.2 as shown in Figure 5. By definition, $\mathcal{I}_P(S)$ can be computed by solving the non-linear program of Definition 4.4 for the whole video v. Alternatively, Theorem 4 says that $\mathcal{I}_P(S)$ can be computed by solving the non-linear program of Definition 4.4 for the sub-video $v^* = v_2 \cdot v_3$.

6 Top-k Algorithms

We now present algorithms to find top-k totally and partially unexplained sequences and classes. For ease of presentation, we assume $|f.obs| = 1$ for every OID f in an observation sequence (this makes the algorithms much more concise – generalization to the case of multiple action symbols per OID is straightforward[7]). Given an observation sequence $v = \langle f_1, \ldots, f_n \rangle$, we use $v(i,j)$ $(1 \leq i \leq j \leq n)$ to denote the sequence $S = \langle (f_i, s_i), \ldots, (f_j, s_j) \rangle$, where s_k is the only element in $f_k.obs$, $i \leq k \leq j$.

Algorithm 1 Top-k TUS

```
Input: USP instance $I = \langle v, \tau, L \rangle$, $k \geq 1$
Output: Top-$k$ totally unexplained sequences
 1: $TopSol = \emptyset$
 2: for all $v' \in relevant(I)$ do
 3:     $start = 1$;  $end = L$
 4:     repeat
 5:         if $\mathcal{P}_T(v'(start, end)) \geq \tau \wedge \mathcal{P}_T(v'(start, end)) > lowest$ then
 6:             $end' = end$
 7:             while $end < |v'|$ do
 8:                 $end = \min\{end + L, |v'|\}$
 9:                 if $\mathcal{P}_T(v'(start, end)) < \tau$ then
10:                     break
11:                 else
12:                     if $\mathcal{P}_T(v'(start, end)) \leq lowest$ then
13:                         $end = end + 1$
14:                         go to line 33
15:             $s = \max\{end - L, end'\}$;  $e = end$
16:             while $e \neq s$ do
17:                 $mid = \lceil (s + e)/2 \rceil$
18:                 if $\mathcal{P}_T(v'(start, mid)) \geq \tau$ then
19:                     if $\mathcal{P}_T(v'(start, mid)) \leq lowest$ then
20:                         $end = mid + 1$
21:                         go to line 33
22:                     else
23:                         $s = mid$
24:                 else
25:                     $e = mid - 1$
26:             if $start > 1 \wedge \mathcal{P}_T(v'(start - 1, s)) \geq \tau$ then
27:                 $end = s + 1$
28:                 go to line 33
29:             else
30:                 $S = v'(start, s)$;  Add $S$ to $TopSol$
31:                 $start = start + 1$;  $end = s + 1$
32:         else
33:             $start = start + 1$;   $end = \max\{end, start + L - 1\}$
34:     until $end > |v'|$
35: return $TopSol$
```

6.1 Top-k TUS and TUC

The Top-k TUS algorithm computes a set of top-k totally unexplained sequences in an observation sequence. Note that:

- at every time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_T(S) \mid S \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- On line 30, "Add S to $TopSol$" works as follows:
  - If $|TopSol| < k$, then S is added to $TopSol$;
  - otherwise, a sequence S' in $TopSol$ having minimum $\mathcal{P}_T(S')$ is replaced by S.

Leveraging Theorem 3, Top-k TUS considers only relevant observation subsequences of v individually (line 2). When it finds a sequence $v'(start, end)$ of length at least L having a probability of being totally unexplained greater than $lowest$ (line 5), it makes the sequence maximal by adding OIDs on the right (lines 7–14). Instead of adding one OID at a time, $v'(start, end)$ is extended by L OIDs at a time until its probability drops below τ (lines 9–10); a binary search is then performed to find the exact maximum length of the unexplained sequence (lines 15–25). While making the sequence maximal, if the algorithm realizes that the unexplained sequence will not have a probability greater than $lowest$ (i.e., the sequence is not a top-k TUS), then the sequence is disregarded and the process of making the sequence maximal is aborted (lines 12–14 and 19–21). This pruning allows the algorithm to move forward in the observation sequence avoiding computing the exact ending OID of the TUS thereby saving time. Throughout the algorithm, $\mathcal{P}_T$ is computed by applying Theorem 2.

Theorem 5: Algorithm Top-k TUS returns a set of top-k totally unexplained sequences of the input instance.

Algorithm Top-k TUC modifies Top-k TUS as follows to compute the top-k totally unexplained classes:

- At every time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_T(C) \mid C \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- "Add S to $TopSol$" (line 30) works as follows:
  - If there exists $C \in TopSol$ s.t. $\mathcal{P}_T(C) = \mathcal{P}_T(S)$, then S is added to C;
  - else if $|TopSol| < k$, then the class $\{S\}$ is added to $TopSol$;
  - otherwise the class C in $TopSol$ having minimum $\mathcal{P}_T(C)$ is replaced with $\{S\}$.
- On line 5, $\mathcal{P}_T(v'(start, end)) > lowest$ is replaced with $\mathcal{P}_T(v'(start, end)) \geq lowest$;
- On line 12, $\mathcal{P}_T(v'(start, end)) \leq lowest$ is replaced with $\mathcal{P}_T(v'(start, end)) < lowest$;
- On line 19, $\mathcal{P}_T(v'(start, mid)) \leq lowest$ is replaced with $\mathcal{P}_T(v'(start, mid)) < lowest$;

7. It suffices to consider the different sequences given by the different action symbols.

Algorithm 2 Top-k PUS

```
Input: USP instance $I = \langle v, \tau, L \rangle$, $k \geq 1$
Output: Top-$k$ partially unexplained sequences
 1: $TopSol = \emptyset$;   $start = 1$;   $end = L$
 2: while $end \leq |v|$ do
 3:     if $\mathcal{P}_P(v(start, end)) < \tau$ then
 4:         $end' = end$
 5:         while $end < |v|$ do
 6:             $end = \min\{end + L, |v|\}$
 7:             if $\mathcal{P}_P(v(start, end)) \geq \tau$ then
 8:                 break
 9:         if $\mathcal{P}_P(v(start, end)) \geq \tau$ then
10:             if $\mathcal{P}_P(v(start, end)) > lowest$ then
11:                 $s = \max\{end' + 1, end - L + 1\}$;   $e = end$
12:                 while $e \neq s$ do
13:                     $mid = \lfloor (s + e)/2 \rfloor$
14:                     if $\mathcal{P}_P(v(start, mid)) < \tau$ then
15:                         $s = mid + 1$
16:                     else
17:                         if $\mathcal{P}_P(v(start, mid)) \leq lowest$ then
18:                             $start = start + 1$;   $end = mid + 1$
19:                             go to line 2
20:                         else
21:                             $e = mid$
22:                 $end = e$
23:             else
24:                 $start = start + 1$;   $end = end + 1$
25:                 go to line 2
26:         else
27:             return  $TopSol$
28:     $s' = start$;  $e' = end - L + 1$
29:     while $e' \neq s'$ do
30:         $mid = \lceil (s' + e')/2 \rceil$
31:         if $\mathcal{P}_P(v(mid, end)) < \tau$ then
32:             $e' = mid - 1$
33:         else
34:             if $\mathcal{P}_P(v(mid, end)) \leq lowest$ then
35:                 $start = mid + 1$;   $end = end + 1$
36:                 go to line 2
37:             else
38:                 $s' = mid$
39:     if $\mathcal{P}_P(v(s', end - 1)) \geq \tau \wedge |v(s', end - 1)| \geq L$ then
40:         $start = s' + 1$;   $end = end + 1$
41:         go to line 2
42:     else
43:         $S = v(s', end)$;  Add $S$ to $TopSol$
44:         $start = s' + 1$;   $end = end + 1$
45: return  $TopSol$
```

The algorithm obtained by applying the modifications above is named Top-k TUC.

Theorem 6: Algorithm Top-k TUC returns the top-k totally unexplained classes of the input instance.

6.2 Top-k PUS and PUC

The Top-k PUS algorithm below computes a set of top-k partially unexplained sequences in an observation sequence. Note that:

- at each time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_P(S) \mid S \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- On line 43, "Add S to $TopSol$" works as follows:
  - If $|TopSol| < k$, then S is added to $TopSol$;
  - otherwise, a sequence in $TopSol$ having minimum $\mathcal{P}_P$ is replaced by S.

To find an unexplained sequence, Algorithm Top-k PUS starts with a sequence of length at least L and adds OIDs to its right until its probability of being partially unexplained is above the threshold. As in the case of Top-k TUS, this is done by adding L OIDs at a time (lines 5–8) and then performing a binary search (lines 9–27). When performing the binary search, if at some point the algorithm realizes that the partially unexplained sequence will not have a probability greater than $lowest$, then the sequence is disregarded and the binary search is aborted (lines 17–19 and lines 24–25). Otherwise, the sequence is shortened on the left making it minimal (lines 28–38) by performing a binary search instead of proceeding one OID at a time. If the algorithm realizes that the partially unexplained sequence will not have a probability greater than $lowest$, then the sequence is disregarded and the shortening process is aborted (lines 34–36). This allows the algorithm to avoid computing the exact starting OID of the PUS, thus saving time. Note that $\mathcal{P}_P$ is computed by applying Theorem 4.

Theorem 7: Algorithm Top-k PUS returns the set of top-k partially unexplained sequences of the input instance.

Algorithm Top-k PUC modifies Top-k PUS as follows to compute the top-k partially unexplained classes:

- At every time, $lowest$ is defined as follows:

$$lowest = \begin{cases} -1 & \text{if } |TopSol| < k \\ \min\{\mathcal{P}_P(C) \mid C \in TopSol\} & \text{if } |TopSol| = k \end{cases}$$

- "Add S to $TopSol$" (line 43) works as follows:
  - If there exists $C \in TopSol$ s.t. $\mathcal{P}_P(C) = \mathcal{P}_P(S)$, then S is added to C;
  - else if $|TopSol| < k$, then the class $\{S\}$ is added to $TopSol$;
  - otherwise the class C in $TopSol$ having minimum $\mathcal{P}_P(C)$ is replaced with $\{S\}$.
- On line 10, $\mathcal{P}_P(v(start, end)) > lowest$ is replaced with $\mathcal{P}_P(v(start, end)) \geq lowest$;
- On line 17, $\mathcal{P}_P(v(start, mid)) \leq lowest$ is replaced with $\mathcal{P}_P(v(start, mid)) < lowest$;
- On line 34, $\mathcal{P}_P(v(mid, end)) \leq lowest$ is replaced with $\mathcal{P}_P(v(mid, end)) < lowest$;

The algorithm obtained by applying the modifications above is named Top-k PUC.

Theorem 8: Algorithm Top-k PUC returns the top-k partially unexplained classes of the input instance.

7 Experimental Evaluation

We implemented Algorithms Top-k TUS, Top-k PUS, Top-k TUC and Top-k PUC, and experimentally evaluated both running time and accuracy on real-world datasets video and cyber security datasets.

7.1 Video Surveillance Domain

We evaluated our framework on two video datasets: (i) a video we shot by monitoring a university parking lot, and (ii) a benchmark dataset about video surveillance in an airport [41]. The frame observations have been generated in a semi-automatic way using both image processing libraries and human intervention.[8]

7.1.1 Parking lot surveillance video

The set $\mathcal{A}$ includes "known" normal activities such as parking a car, people passing, a person getting in a car and leaving the parking lot, and abnormal activities such as a person taking a package out of the car and leaving it in the parking lot before driving away, or a person taking an unattended package in the parking lot. Examples of detected unexplained sequences are two cars stopping next to each other in the middle of the parking lot with the drivers exchanging something before leaving the parking lot, or a person strolling around a car for a while before leaving the parking lot.

We compared Algorithms Top-k TUS and Top-k PUS against "naïve" algorithms which are the same as Top-k TUS and Top-k PUS but do not exploit the optimizations provided by the theorems in Section 5.

Figures 7 and 8 show that Top-k TUS and Top-k PUS

8. We note that identifying frame observations via the development of image processing algorithms is an extremely challenging task—the goal of our work is to present a domain-independent way of identifying unexplained sequences that builds upon domain-specific ways of recognizing actions in observation sequences. In contrast to the difficulty of detecting actions in video, in cyber-security, it is easy to identify actions in an observation sequence as they can merely be logged.

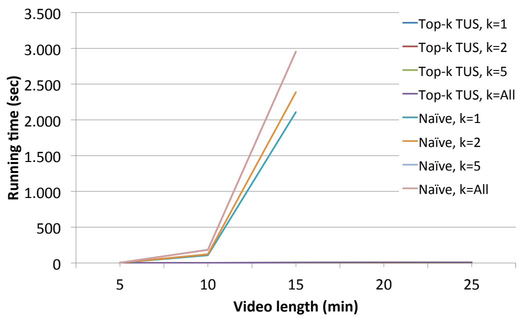


Fig. 7: Algorithm Top-k TUS vs. Naïve.

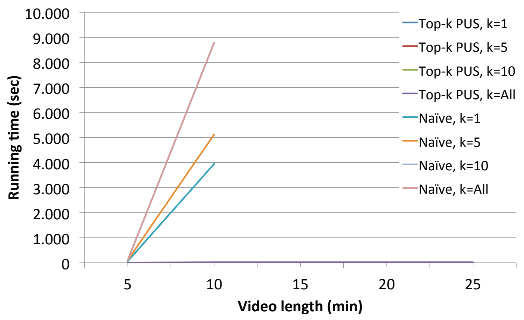
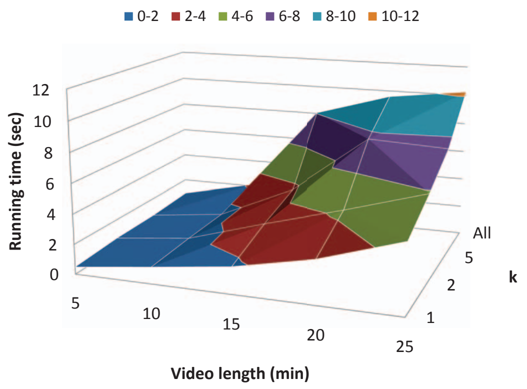
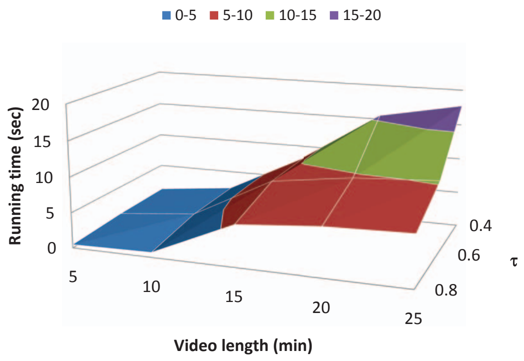


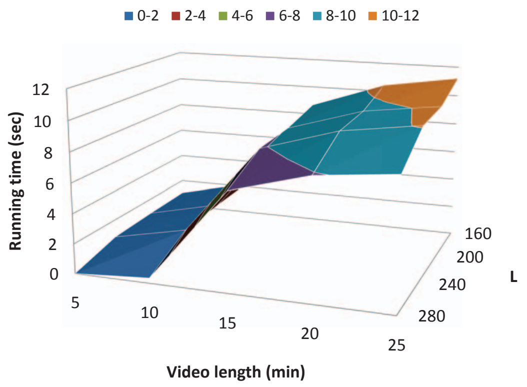
Fig. 8: Algorithm Top-k PUS vs. Naïve.



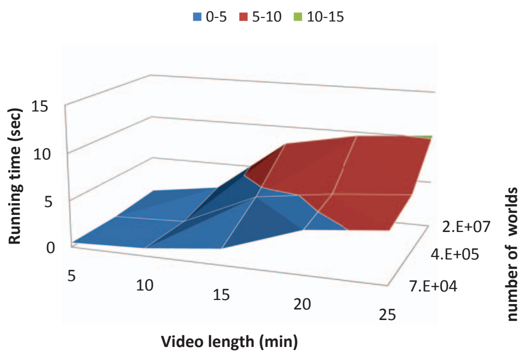
(a) Varying k ($\tau = 0.6, L = 200$)



(b) Varying τ ($k = All, L = 200$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying number of worlds ($\tau = 0.6, k = All, L = 200$)

Fig. 9: Running time of Algorithm Top-k TUS on the parking lot dataset.

significantly outperform the naïve algorithms which are not able to scale beyond videos of length 15 and 10 minutes for totally and partially unexplained sequences, respectively (with longer videos, the naïve algorithms did not terminate in 3 hours). Figures 9a and 10a zoom in on the running times for Algorithms Top-k TUS and Top-k PUS, respectively. The runtimes in Figure 7 when $k = 5$ and $k = All$ are almost the same (the two curves are indistinguishable) because, up to 15 minutes, there were at most 5 totally unexplained sequences in the video. A similar argument applies to Figure 8.

We also evaluated how the different parameters that define a USP instance affect the running time by varying the values of each parameter while keeping the others fixed to a default value.

Runtime of Top-k TUS. Table 2 reports the values we considered for each parameter along with the corresponding default value.

For example, Table 2 says that we measured the running times to find the top-1, top-2, top-5, and all totally unexplained sequences (as the video length increases) while keeping $\tau = 0.6$, $L = 200$, $\#worlds = 2E + 07$.

Varying k. Figure 9a shows that lower values of k give lower runtimes. As discussed in the preceding section, Algorithm Top-k TUS can infer that some sequences are not going to be top-k TUSs and quickly prune: this is effective with lower values of k because the probability threshold to enter the current Top-k TUSs (i.e., $lowest$ in Algorithm Top-k TUS) is higher, thus fewer candidates are added to the current Top-k TUSs,making the pruning of Algorithm Top-k TUS more effective.

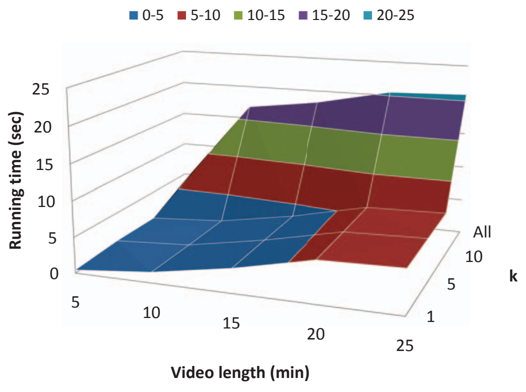
Varying τ. Figure 9b shows that the runtime decreases as the probability threshold grows. Intuitively, this is because higher probability thresholds enable Algorithm Top-k TUS to prune more.

Varying L. Figure 9c shows that higher values of L yield lower running times, though there is not a big difference between $L = 200$ and $L = 240$.

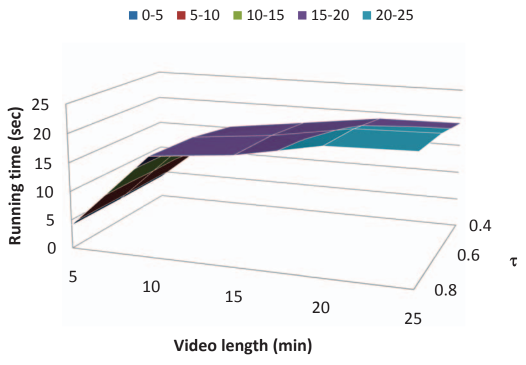
Varying Number of Possible Worlds. Finally, Figure 9d shows that more possible worlds leads to higher running times. However, note that big differences in the number of possible worlds yield small differences in running times,

Parameter	Values	Default value
k	1, 2, 5, All (Top-k TUS)	All
	1, 5, 10, All (Top-k PUS)	All
τ	0.4, 0.6, 0.8	0.6
L	160, 200, 240, 280	200
# worlds	7 E+04, 4 E+05, 2 E+07	2 E+07

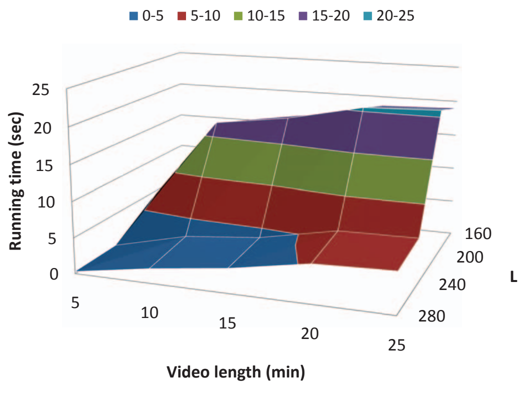
TABLE 2: Parameter values (parking lot dataset).



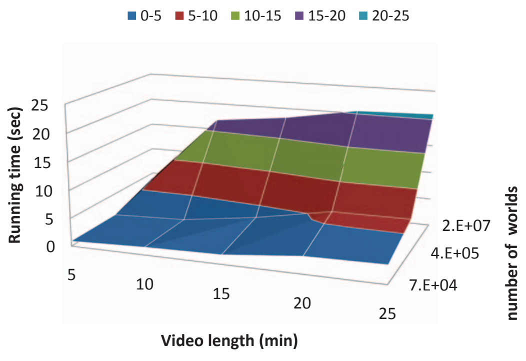
(a) Varying k ($\tau = 0.6, L = 200$)



(b) Varying τ ($k = All, L = 200$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying number of worlds ($\tau = 0.6, k = All, L = 200$)

Fig. 10: Running time of Algorithm Top-k PUS on the parking lot dataset.

hence Algorithm Top-k TUS is able to scale well (this is due to the application of Theorem 2 to compute $\mathcal{P}_T(S)$).

Runtime of Top-k PUS. The parameter values we used are reported in Table 2.

Varying k. The runtimes for $k = 1, 5, 10$ differ slightly from each other and are much lower than when all PUSs had to be found (Figure 10a).

Varying τ. Figure 10b shows that the runtimes do not change much for different values of τ.

Varying L. Figure 10c shows that higher values of L lead to lower runtimes.

Varying Number of Possible Worlds. Figure 10d shows that higher numbers of possible worlds lead to higher runtimes. As with TUSs, the runtime of Top-k PUS increases reasonably despite the steep growth of possible worlds. Runtimes of Top-k PUS are higher than for Top-k TUS because computing $\mathcal{P}_P(S)$ requires solving a non-linear program whereas $\mathcal{P}_T(S)$ requires solving linear programs.

Precision/Recall. In order to assess accuracy, we compared the output of our algorithms against ground truth provided by 8 human annotators who were taught the meaning of graphical representations of activities in $\mathcal{A}$ (e.g., Figure 2). They were asked to identify the totally and partially unexplained sequences w.r.t. $\mathcal{A}$. We ran Top-k TUS and Top-k PUS with values of τ ranging from 0.4 to 0.8, looking for *all* totally and partially unexplained sequences (L was set to 200). We use $\{S_i^a\}_{i \in [1,m]}$ to denote the unexplained sequences returned by our algorithms and $\{S_j^h\}_{j \in [1,n]}$ to denote the sequences flagged as unexplained by human annotators. Precision and recall were computed as follows: $P = \frac{|\{S_i^a | \exists S_j^h \ s.t. \ S_i^a \approx S_j^h\}|}{m}$ and $R = \frac{|\{S_j^h | \exists S_i^a \ s.t. \ S_i^a \approx S_j^h\}|}{n}$ where $S_i^a \approx_p S_j^h$ means that S_i^a and S_j^h overlap by a percentage no smaller than 75%.

Precision and recall when $\tau = 0.4, 0.6, 0.8$ are shown in Tables 3a and 3b, and show that the framework achieved a good accuracy.

τ	Precision	Recall
0.4	62.5	89.17
0.6	66.67	82.5
0.8	72.22	71.67

(a) Top-k TUS

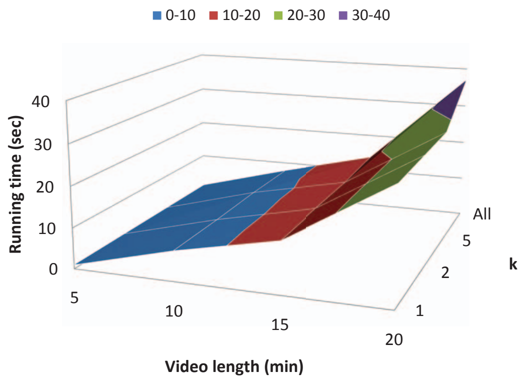
τ	Precision	Recall
0.4	59.65	77.38
0.6	64.91	74.6
0.8	70.18	71.83

(b) Top-k PUS

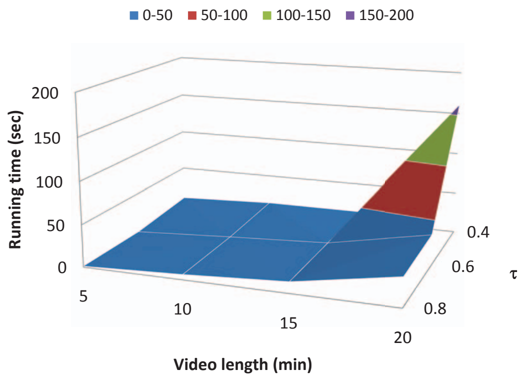
TABLE 3: Precision and recall (parking lot dataset).

7.1.2 *Airport surveillance video*

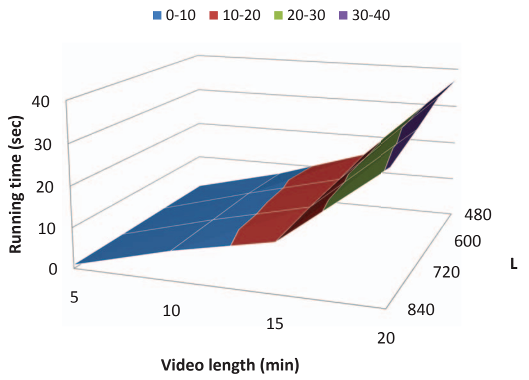
We also tested our algorithms with an airport video surveillance dataset [41]. The set $\mathcal{A}$ of known activities includes normal activities such as people passing, people chatting, people seating, and abnormal activities like a person leaving a package unattended, or a person taking an unattended package. Examples of found unexplained sequences are a



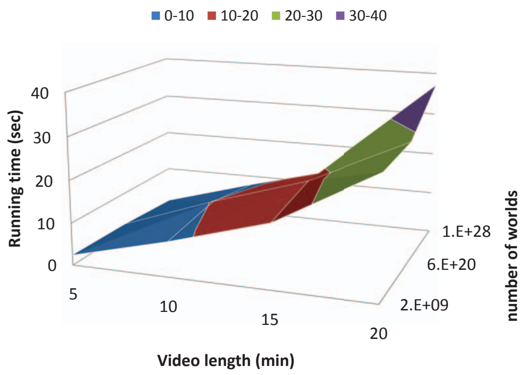
(a) Varying k ($\tau = 0.6, L = 600$)



(b) Varying τ ($k = All, L = 600$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying number of worlds ($\tau = 0.6, k = All, L = 600$)

Fig. 11: Running time of Algorithm Top-k TUS on the airport dataset.

person typing on a keypad next to a door with the door closing afterwards, or a person leaving a suitcase on the ground, standing next to it for a while looking at his watch different times, taking the suitcase, and leaving the area.

Runtime of Top-k TUS. This data set is far more complex (w.r.t. number of possible worlds) than the parking lot data set - the "naïve" algorithms did not terminate in a reasonable amount of time, even with a video of 5 minutes. Thus, we do not show the runtimes of the naïve algorithms. As in the case of the parking lot data set, we varied the k, τ, L, $\#\ worlds$ parameters, as shown in Table 4.

Varying k. Figure 11a shows that Top-k TUS's runtime varies little with k when the video is up to 15 minutes long. After that, the runtime for $k = 1, 2, 5$ are comparable, but the runtime for $k = All$ starts to diverge from them.

Varying τ. Figure 11b shows that the runtime when $\tau = 0.4$ is much higher than when $\tau = 0.6$ and $\tau = 0.8$ (the latter two cases do not show substantial differences in running time).

Varying L. Figure 11c shows that higher values of L yield lower runtimes. Though the difference is small for videos under 15 minutes, it increases for 20 minute videos.

Varying Number of Possible Worlds. Figure 11d shows that runtimes for different numbers of possible worlds are initially close (up to 15 minutes); then, the runtime for 1 E+28 possible worlds gets higher. There is only a moderate increase in runtime corresponding to a huge increase of the number of possible worlds—hence, Top-k TUS is able to scale well when the video gets substantially more complex.

Runtime of Top-k PUS. We varied the k, τ, L, $\#\ worlds$ parameters as reported in Table 4.

Varying k. Figure 12a shows that the runtime decreases as k decreases.

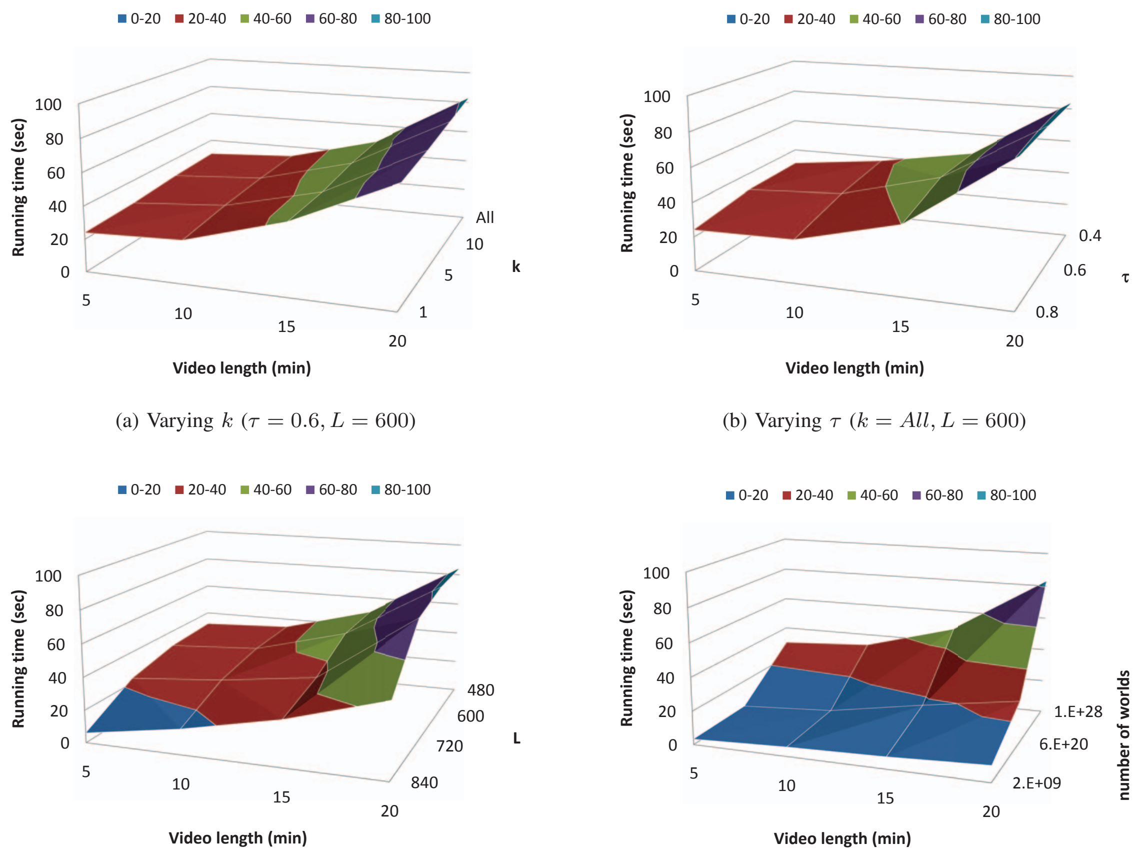
Varying τ. Figure 12b shows that the runtimes for $\tau = 0.4$ and $\tau = 0.6$ are similar and higher than the one for $\tau = 0.8$.

Varying L. Figure 12c shows that lower values of L give higher running times. The runtimes are similar for $L = 480$ and $L = 600$ (the number of PUSs found in the video are similar in both cases). Execution times are lower for $L = 720$ and much lower for $L = 800$ (in this case, the number of PUSs found in the video is approximately half the number of PUSs found with $L = 480$ and $L = 600$).

Varying Number of Possible Worlds. Figure 12d shows that though the runtime grows with the number of possible

Parameter	Values	Default value
k	1, 2, 5, All (Top-k TUS)	All
	1, 5, 10, All (Top-k PUS)	All
τ	0.4, 0.6, 0.8	0.6
L	480, 600, 720, 840	600
# worlds	2 E+09, 6 E+20, 1 E+28	1 E+28

TABLE 4: Parameter values (airport dataset).



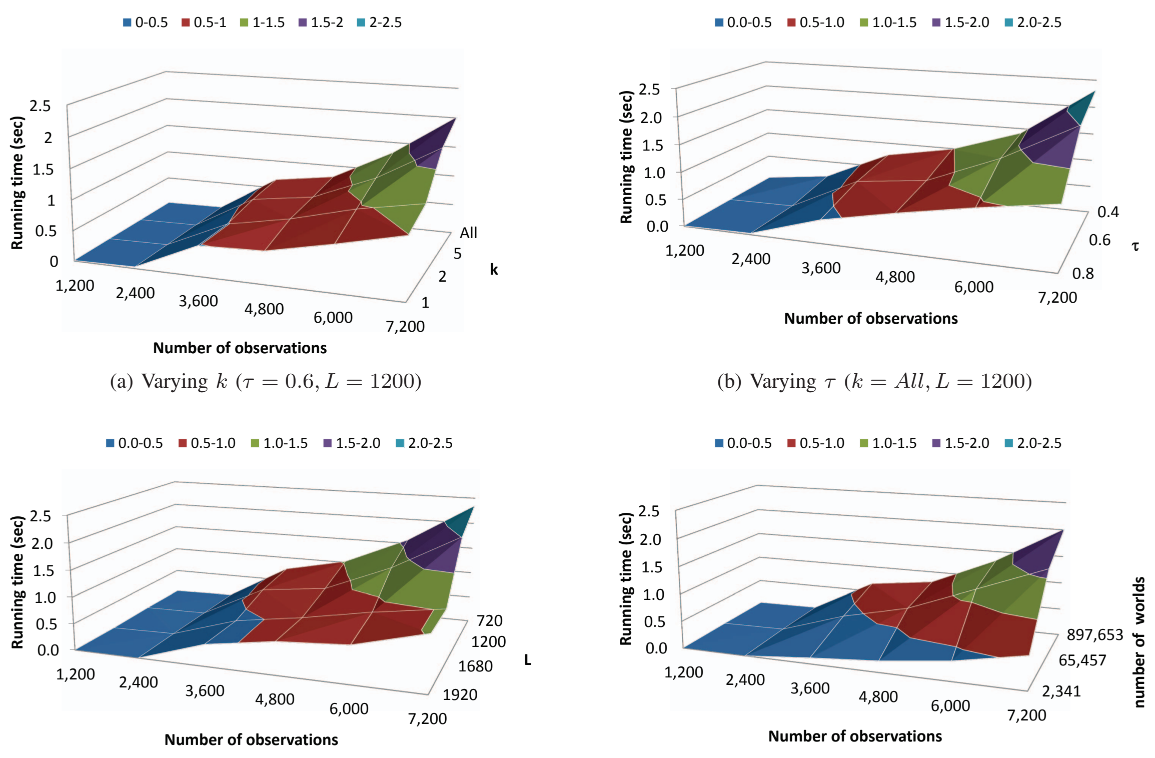
(a) Varying k ($\tau = 0.6, L = 600$)

(b) Varying τ ($k = All, L = 600$)

(c) Varying L ($\tau = 0.6, k = All$)

(d) Varying number of worlds ($\tau = 0.6, k = All, L = 600$)

Fig. 12: Running time of Algorithm Top-k PUS on the airport dataset.



(a) Varying k ($\tau = 0.6, L = 1200$)

(b) Varying τ ($k = All, L = 1200$)

(c) Varying L ($\tau = 0.6, k = All$)

(d) Varying number of worlds ($\tau = 0.6, k = All, L = 1200$)

Fig. 13: Running time of Algorithm Top-k TUS on the cyber security dataset.

worlds, Top-k PUS responds well to the steep growth of the number of possible worlds.

Precision/Recall. We evaluated the accuracy of Top-k TUS and Top-k PUS in the same way as for the parking lot data set. Precision and recall are reported in Tables 5a and 5b and show that we achieved high accuracy.

τ	Precision	Recall
0.4	56.48	80.35
0.6	78.79	76.25
0.8	81.82	73.99

(a) Top-k TUS

τ	Precision	Recall
0.4	72.62	77.12
0.6	75	73.59
0.8	76.19	71.5

(b) Top-k PUS

TABLE 5: Precision and recall (airport dataset).

7.2 Cyber Security Domain

We also ran evaluations with a cyber security dataset consisting of network traffic from a university network. We used (i) Wireshark (http://www.wireshark.org/) to capture network traffic and generate packet sequences, and (ii) Snort (http://www.snort.org/) as the set of activity models $\mathcal{A}$.

Runtime. We varied the k, τ, L, $\#\ worlds$ parameters as shown in Table 6 and used *all* Snort rules as the set of activity models.

Running times for Algorithm Top-k TUS and Algorithm Top-k PUS are shown in Figures 13 and 14, respectively. They confirm the trend already seen with video data: runtime decreases as L and τ (resp. k and the number of possible worlds) increase (resp. decrease).

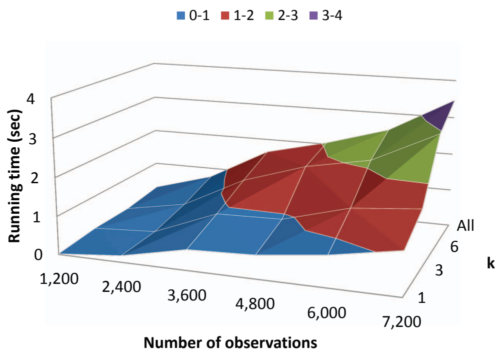
Parameter	Values	Default value
k	1, 2, 5, All (Top-k TUS)	All
	1, 3, 6, All (Top-k PUS)	All
τ	0.4, 0.6, 0.8	0.6
L	720, 1200, 1680, 1920	1200
# worlds	2341, 65457, 897653	897653

TABLE 6: Parameter values (cyber security dataset).

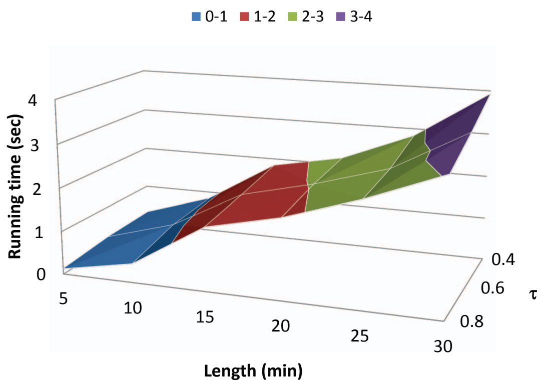
Accuracy. We measured accuracy as follows. Let $\mathcal{A}$ be the set of *all* Snort rules. First, we detected all occurrences of $\mathcal{A}$ in the data stream. We then ignored a certain subset $\mathcal{A}'$ of $\mathcal{A}$ and identified the unexplained sequences. Clearly, ignoring models in $\mathcal{A}'$ is equivalent to not having those models available. Thus, occurrences of ignored activties are expected to have a relatively high probability of being unexplained as there is no model for them. We measured the fraction of such occurrences that have been flagged as unexplained for different values of τ.

Specifically, we considered two settings: one where only *ICMP rules* in $\mathcal{A}$ were ignored, and another one where only *preprocessor rules* in $\mathcal{A}$ were ignored.[9] The results

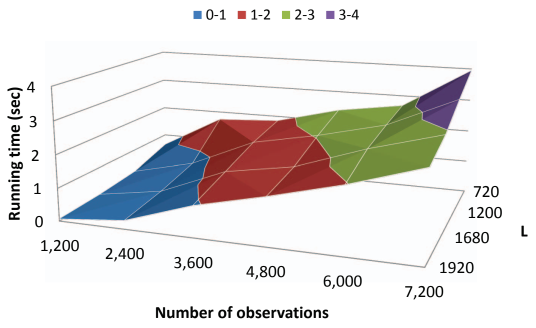
9. ICMP rules are Snort rules designed to analyze ICMP packets (e.g., echo request a.k.a. ping) and alert on suspicious or malformed ICMP packets. For instance, a ping sweep is a passive reconnaissance attack that uses multiple echo requests to establish which IP addresses map to live hosts. Some packets and applications have to be decoded into plain text for Snort rules to trigger. Prepocessor rules are designed to handle such situations. For instance, the *arpspoof* preprocessor is fed a list of IP:MAC addresses. When it detects a layer-2 attack, it triggers an alarm for a layer-2 event, such as multiple MAC addresses from a single IP.



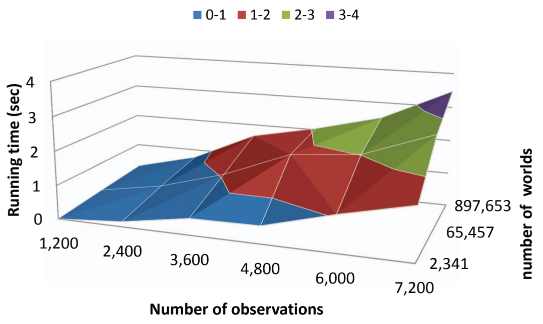
(a) Varying k ($\tau = 0.6, L = 1200$)



(b) Varying τ ($k = All, L = 1200$)



(c) Varying L ($\tau = 0.6, k = All$)



(d) Varying number of worlds ($\tau = 0.6, k = All, L = 1200$)

Fig. 14: Running time of Algorithm Top-k PUS on the cyber security dataset.

are reported in Tables 7 and 8 for Top-k TUS and Top-k PUS, respectively, and show that our framework achieved good accuracy. When ICMP rules were ignored, unexplained sequences were sequences where ICMP activities were occurring, and likewise when preprocessor rules were ignored.

τ	Accuracy
0.4	85.71
0.6	71.42
0.8	68.17

(a) Ignoring ICMP rules

τ	Accuracy
0.4	75.12
0.6	63.84
0.8	59.29

(b) Ignoring Preprocessor rules

TABLE 7: Accuracy of Top-k TUS (cyber security dataset).

τ	Accuracy
0.4	91.24
0.6	83.39
0.8	72.84

(a) Ignoring ICMP rules

τ	Accuracy
0.4	88.76
0.6	76.24
0.8	74.85

(b) Ignoring Preprocessor rules

TABLE 8: Accuracy of Top-k PUS (cyber security dataset).

7.3 Experimental Conclusions

Our experiments show that:
(i) *Runtime increases with observation sequence length* (because there are more possible worlds, causing $LC(v)$ and $NLC(v)$ to have more variables and constraints). Despite the enormous blow-up in the number of possible worlds, our algorithms perform very well showing quadratic performance in all three datasets.
(ii) *Runtime increases with the number of totally or partially unexplained sequences present in the video.* This is because determining the exact endpoints of each TUS (resp. PUS) is costly. Specifically, determining the exact end frame of a TUS requires computing $\mathcal{P}_T$ many times: when a TUS is found, Top-k TUS (and also Top-k TUC) need to go through the **while** loop of lines 7–14, the binary search in the **while** loop of lines 16–25, and the **if** block of lines 26–31. All these code blocks require $\mathcal{P}_T$ to be computed. Likewise, determining the exact start and end frames of a PUS requires $\mathcal{P}_P$ to be computed many times as Algorithm Top-k PUS (as well as Algorithm Top-k PUC) goes through different loops and binary searches (one to determine the start frame, another to determine the end frame) requiring multiple computations of $\mathcal{P}_P$.
(iii) *In general, the number of TUSs and PUSs in the observation sequence decreases as τ and L increase*, because higher values of τ and L are stricter conditions for a sequence to be totally or partially unexplained.
(iv) *Runtime decreases as k decreases* because our algorithms use k intelligently to infer that certain sequences are not going to be in the result (aborting the loops and binary searches mentioned above).
(v) *Precision increases whereas recall decreases as τ increases*. The experimental results have shown that a good compromise can be achieved by setting τ at least 0.6

and that our framework had a good accuracy with all the datasets we considered.

8 Conclusions

Suppose we have a sequence v of time-stamped observation data and a set $\mathcal{A}$ of "known" activities (normal or suspicious). This paper addresses the problem of finding subsequences of v that are not "sufficiently" explained by the activities in $\mathcal{A}$. We formally define what it means for a sequence to be unexplained by defining *totally* and *partially* unexplained sequences. We propose a possible worlds framework and identify interesting properties that can be leveraged to make the search for unexplained sequences highly efficient via intelligent pruning. We leverage these properties to develop the Top-k TUS, Top-k PUS, Top-k TUC, Top-k PUC algorithms to find totally and partially unexplained sequences with highest probabilities. We conducted experimentals over three datasets in the video and cyber security domains showing that our approach has good running time and high accuracy.

This paper represents a start towards detecting unexplained sequences in a domain independent way. Much future work is possible. For instance, we may wish to allow activity occurrences to violate the temporal constraints in the stochastic automata based activity model by penalizing such activity occurrences via diminished probabilities. This can be done in many ways. Second, we would like to increase scalability of our algorithms, possibly through the development of specialized data structures to support identification of the Top-k TUS, Top-k PUS, Top-k TUC, Top-k PUC algorithms. Third, the assumption of independence after conflict-based partitioning is convenient and follows upon much work in computer science that makes such assumptions—but it may not be appropriate for all applications. Relaxing this assumption is also an important direction for future work.

Acknowledgements. Some of the authors of this paper were funded in part by ARO grants W911NF0910206 and 4021UMUSA0525 ONR grant N000140910685.

References

[1] M. Albanese, V. Moscato, A. Picariello, V. S. Subrahmanian, and O. Udrea, "Detecting stochastically scheduled activities in video," in *IJCAI*, 2007, pp. 1802–1807.

[2] S. Hongeng and R. Nevatia, "Multi-agent event recognition," in *ICCV*, 2001, pp. 84–93.

[3] N. Vaswani, A. K. R. Chowdhury, and R. Chellappa, ""shape activity": A continuous-state hmm for moving/deforming shapes with application to abnormal activity detection," *IEEE Transactions on Image Processing*, vol. 14, no. 10, pp. 1603–1616, 2005.

[4] M. Brand, N. Oliver, and A. Pentland, "Coupled hidden markov models for complex action recognition," in *CVPR*, 1997, pp. 994–999.

[5] N. Oliver, E. Horvitz, and A. Garg, "Layered representations for human activity recognition," in *ICMI*, 2002, pp. 3–8.

[6] R. Hamid, Y. Huang, and I. Essa, "Argmode - activity recognition using graphical models," in *CVPRW*, 2003, pp. 38–43.

[7] N. P. Cuntoor, B. Yegnanarayana, and R. Chellappa, "Activity modeling using event probability sequences," *IEEE Transactions on Image Processing*, vol. 17, no. 4, pp. 594–607, 2008.

[8] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, "Semi-supervised adapted hmms for unusual event detection," in *CVPR*, 2005, pp. 611–618.

[9] M. T. Chan, A. Hoogs, J. Schmiederer, and M. Petersen, "Detecting rare events in video using semantic primitives with hmm," in *ICPR*, 2004, pp. 150–154.

[10] T. Xiang and S. Gong, "Video behavior profiling for anomaly detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 5, pp. 893–908, 2008.

[11] J. Kim and K. Grauman, "Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates," in *CVPR*, 2009.

[12] J. Yin, Q. Yang, and J. J. Pan, "Sensor-based abnormal human-activity detection," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1082–1090, 2008.

[13] D. H. Hu, X.-X. Zhang, J. Yin, V. W. Zheng, and Q. Yang, "Abnormal activity recognition based on hdp-hmm models," in *IJCAI*, 2009, pp. 1715–1720.

[14] X.-X. Zhang, H. Liu, Y. Gao, and D. H. Hu, "Detecting abnormal events via hierarchical dirichlet processes," in *PAKDD*, 2009, pp. 278–289.

[15] D. Mahajan, N. Kwatra, S. Jain, P. Kalra, and S. Banerjee, "A framework for activity recognition and detection of unusual activities," in *ICVGIP*, 2004.

[16] F. Jiang, Y. Wu, and A. K. Katsaggelos, "Detecting contextual anomalies of crowd motion in surveillance video," in *ICIP*, 2009, pp. 1117–1120.

[17] H. Zhong, J. Shi, and M. Visontai, "Detecting unusual activity in video," in *CVPR*, 2004, pp. 819–826.

[18] C. E. Au, S. Skaff, and J. J. Clark, "Anomaly detection for video surveillance applications," in *ICPR*, 2006, pp. 888–891.

[19] Y. Zhou, S. Yan, and T. S. Huang, "Detecting anomaly in videos from trajectory similarity analysis," in *ICME*, 2007, pp. 1087–1090.

[20] A. Mecocci and M. Pannozzo, "A completely autonomous system that learns anomalous movements in advanced videosurveillance applications," in *ICIP*, 2005, pp. 586–589.

[21] L. Brun, A. Saggese, and M. Vento, "A clustering algorithm of trajectories for behaviour understanding based on string kernels," in *SITIS*, 2012, pp. 267–274.

[22] J. Wang, Z. Cheng, M. Zhang, Y. Zhou, and L. Jing, "Design of a situation-aware system for abnormal activity detection of elderly people," in *AMT*, 2012, pp. 561–571.

[23] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2287–2301, 2011.

[24] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 555–560, 2008.

[25] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, "Video anomaly detection in spatiotemporal context," in *ICIP*, 2010, pp. 705–708.

[26] A. Gupta, P. Srinivasan, J. Shi, and L. S. Davis, "Understanding videos, constructing plots learning a visually grounded storyline model from annotated videos," in *CVPR*, 2009, pp. 2012–2019.

[27] M. Albanese, C. Molinaro, F. Persia, A. Picariello, and V. S. Subrahmanian, "Finding "unexplained" activities in video," in *IJCAI 2011*, Barcelona, Spain, July 2011, pp. 1628–1634.

[28] B. Mukherjee, L. T. Heberlein, and K. N. Levitt, "Network intrusion detection," *IEEE Network*, vol. 8, no. 3, pp. 26–41, May 1994.

[29] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, and E. Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, February-March 2009.

[30] A. Jones and S. Li, "Temporal signatures for intrusion detection," in *ACSAC*. New Orleans, LA, USA: IEEE Computer Society, December 2001, pp. 252–261.

[31] L. Wang, A. Liu, and S. Jajodia, "Using attack graphs for correlating, hypothesizing, and predicting intrusion alerts," *Computer Communications*, vol. 29, no. 15, pp. 2917–2933, September 2006.

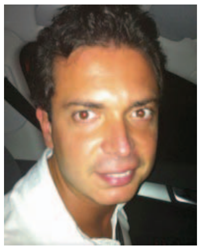
[32] S. Noel, E. Robertson, and S. Jajodia, "Correlating intrusion events and building attack scenarios through attack graph distances," in *ACSAC*, Tucson, AZ, USA, December 2004, pp. 350–359.

[33] H. Debar and A. Wespi, "Aggregation and correlation of intrusion-detection alerts," in *RAID*, ser. Lecture Notes in Computer Science,

W. Lee, L. Mé, and A. Wespi, Eds., vol. 2212. Davis, CA, USA: Springer, October 2001, pp. 85–103.
[34] P. Ning, Y. Cui, and D. S. Reeves, "Constructing attack scenarios through correlation of intrusion alerts," in *CCS 2002*. Washington, DC, USA: ACM, November 2002, pp. 245–254.
[35] S. O. Al-Mamory and H. Zhang, "Ids alerts correlation using grammar-based approach," *Journal of Computer Virology*, vol. 5, no. 4, pp. 271–282, November 2009.
[36] M. Albanese, S. Jajodia, A. Pugliese, and V. S. Subrahmanian, "Scalable analysis of attack scenarios," in *ESORICS*. Leuven, Belgium: Springer, September 2011, pp. 416–433.
[37] X. Qin and W. Lee, "Statistical causality analysis of INFOSEC alert data," in *RAID*, ser. Lecture Notes in Computer Science, G. Vigna, C. Kruegel, and E. Jonsson, Eds., vol. 2820. Pittsburgh, PA, USA: Springer, September 2003, pp. 73–93.
[38] X. Qin, "A probabilistic-based framework for INFOSEC alert correlation," PhD thesis, Georgia Institute of Technology, August 2005.
[39] A. J. Oliner, A. V. Kulkarni, and A. Aiken, "Community epidemic detection using time-correlated anomalies," in *RAID*, ser. Lecture Notes in Computer Science, S. Jha, R. Sommer, and C. Kreibich, Eds., vol. 6307. Ottawa, Canada: Springer, September 2010, pp. 360–381.
[40] S. Tsukiyama, M. Ide, H. Ariyoshi, and I. Shirakawa, "A new algorithm for generating all the maximal independent sets," *SIAM J. Comput.*, vol. 6, no. 3, pp. 505–517, 1977.
[41] "Ninth IEEE international workshop on performance evaluation of tracking and surveillance (PETS 2006) benchmark data," http://www.cvg.rdg.ac.uk/PETS2006/data.html, 2006.

Massimiliano Albanese received his Ph.D. degree in Computer Science and Engineering in 2005 from the University of Naples "Federico II". He joined the University of Maryland in 2006 as a Faculty Research Assistant. Dr. Albanese joined George Mason University in 2011 as an Assistant Professor in the Department of Applied Information Technology. His current research interests are in the areas of Modeling and Recognition of Cyber Attacks, Scalable Detection of Cyber Attacks, Network Hardening, and Moving Target Defense.

Cristian Molinaro received the PhD degree in Computer Science Engineering from the University of Calabria, Italy. He is a faculty research assistant at the University of Maryland Institute for Advanced Computer Studies. His research interests include database theory and logic programming.



Fabio Persia received the Master degree in Computer Science and Engineering from the University of Naples "Federico II", Italy, where he is currently a PhD student in Computer Science and Engineering. His current research interests lie in the field of video surveillance applications, multimedia databases, and knowledge representation and management.

Antonio Picariello received the PhD degree in Computer Science and Engineering from the University of Naples "Federico II", Italy, in 1998. In 1993, he joined the Istituto Ricerca sui Sistemi Informatici Paralleli, National Research Council, Naples, Italy. In 1999, he joined the Dipartimento di Informatica e Sistemistica, University of Naples "Federico II" and is currently an associate professor. His current research interests include knowledge extraction and management, multimedia integration, and image and video databases.

V. S. Subrahmanian is Professor of Computer Science and Director of the Center for Digital International Government and Co-Director of the Lab for Computational Cultural Dynamics at the University of Maryland where he has been on the faculty since 1989. He has worked extensively on databases and artificial intelligence and has co-authored over 200 papers as well as several books. He has served on the editorial board of several journals, has won numerous awards, and delivered invited talks at numerous conferences. His work has been extensively cited both in the academic literature as well as in the press.