Anomaly Detection for Symbolic Representations

Computer Science Technical Report No. CS-TR-5036
UMIACS Technical Report No. UMIACS-TR-2014-03

Michael T. Cox[1], Matt Paisner[2], Tim Oates[3], and Don Perlis[2]

mcox@cs.umd.edu

[1]University of Maryland Institute for Advanced Computer Studies (UMIACS)

[2]Department of Computer Science, University of Maryland
College Park, MD 20742

[3]Department of Computer Science and Electrical Engineering,
University of Maryland, Baltimore County
Baltimore, MD 21250

Abstract

A fully autonomous agent recognizes new problems, explains what causes such problems, and generates its own goals to solve these problems. Our approach to this goal-driven model of autonomy uses a methodology called the Note-Assess-Guide procedure. It instantiates a monitoring process in which an agent *notes* an anomaly in the world, *assesses* the nature and cause of that anomaly, and *guides* appropriate modifications to behavior. This report describes a novel approach to the note phase of that procedure. A-distance, a sliding-window statistical distance metric, is applied to numerical vector representations of intermediate states from plans generated for two symbolic domains. Using these representations, the metric is able to detect anomalous world states caused by restricting the actions available to the planner.

1. Introduction

A truly intelligent agent not only can solve formal problems given to it by some oracle (e.g., an external user); it can also notice when something is amiss in the world and do something about it. That is, an autonomous agent should be able to recognize new problems as well as solve known ones; it should be able to generate its own goals as well as accept goals from a user. In this context, the first step in recognizing and representing a problem is to detect when the world is not as expected or desired. Although not all such discrepancies signal a problem, and not all problems are relevant to an individual agent, the detection of anomalous data is a key enabling step in a new kind of intelligent agency we call *goal-driven autonomy* (Cox, 2007; 2013; Klenk, Molineaux, & Aha, 2013; Munoz-Avila, Jaidee, Aha, Carter, 2010). Furthermore, anomaly detection is an outstanding technical problem in many other areas of interest to the artificial intelligence community (e.g., Albrecht, et al., 2000; Basseville, & Nikiforov, 1993; Chandola, Banerjee, & Kumar, 2009; Crook & Hayes, 2001; Fawcett & Provost, 1999; Janssens, Postma, & Hellemons, 2011; Leake, 1989; Pannell & Ashman, 2010; Schank & Owens, 1987; Sussman, 1975).

The problem of anomaly detection is instrumental to both cognitive and metacognitive processes. An *anomaly* exists when an *expected outcome* (i.e., an *expectation*) is significantly different from an *actual outcome* (or when no outcome occurs at all). At the cognitive level, noting an anomaly means that the agent understands when its own actions or other events go awry in the world. At the metacognitive level, it means that the agent notes when its own reasoning or memory fails. The ability to notice anomalies enables the deliberate evaluation of the agent's progress towards its goals and a readjustment of its action and thinking. We conceptualize this capability as a core three-phase process called the *Note-Assess-Guide procedure* (Anderson & Perlis, 2005; Perlis, 2011). The first phase of the procedure is to note an anomaly or failure. The second is to assess the nature and cause of the anomaly, and the third it to guide a response to the anomaly, either changing what the agent does in the world or learning to reason in a different way.

For example a business management agent may note that normal shipments are not getting delivered on time to customers as expected. A cursory assessment may determine that the bottleneck is due to a union longshoreman strike at the regional shipping center. The guided response may be to create a corporate goal to obtain alternate logistics options for the future. In a metacognitive example, a planning agent may expect that her car will get her to some destination, but experience an expectation failure (as well as disappointment) when the car runs out of gas. Noting this anomaly leads the agent to assess the cause of failure as forgetting to fill up when at the store before starting the journey and to learn to check the gas gauge before starting the car. The former sequence involves noting an anomaly in the world, whereas the latter concerns anomalous decisions and memory performance by the agent itself. In both cases, expectation failures drive the note phase, although these expectations may at times be implicit.

The purpose of this report is to examine in some depth the Note phase of the Note-Assess-Guide procedure and to put forth a novel method of implementing it. Our focus is on detecting anomalies in domains where states are represented symbolically with predicates, as is often the case in deliberative planning systems used in cognitive architectures. The challenge is to detect significant changes in the domain as reflected in changes in sequences of world states observed while executing plans. This is accomplished by using a numeric metric, which was developed to detect changes in real-valued data streams. We pair this metric with a representation of predicate streams that converts them to streams of integers by counting the number of instantiations of each distinct predicate in a state, ignoring predicate arguments. Experiments with a classical planner in the logistics and blocksworld domains show that the pairing of this metric and the collapsed state representation leads to a method capable of robust detection of subtle changes in the underlying domains.

The remainder of this report is organized as follows. The second section examines the metric that has been used in previous research to detect changes in numeric data streams. The third section describes a novel representation that allows the metric to be used against streams of symbolic predicates describing the world as it shifts over time in response to events and forces. Finally we report results from two empirical studies that evaluate our approach, discuss related research, and conclude.

3. The A-distance Metric as an Indication of Change

An intelligent agent not only plans for goals and executes actions to carry out plans, but it is also important to monitor such actions to assure everything is still going according to its plan. That is it needs to understand it actions in context and identify problems as they arise. A Note-Assess-Guide (NAG) procedure[1] constitutes an interpretive understanding of perceptual and conceptual input in such contexts. The procedure is to *Note* whether an anomaly has occurred; to *Assess* potential causes of the anomaly by generating hypotheses; and to *Guide* the system through a response (see Anderson, Oates, Chong & Perlis, 2006, for further details on the latter two phases). Responses can take various forms, such as (1) test the hypothesis; (2) ignore and try again; (3) ask for help; (4) change the goal; or introduce a new goal. The first phase of the NAG procedure, i.e., noting an anomaly, requires some way to represent what is normal, or what is expected. Expectations that drive reasoning can be both implicit as well as explicit. Here we focus on the former.

Done right, the Note phase naturally provides input to the Assess phase by pointing to candidate causes in the domain, the (physical) agent, or the agent's reasoning processes. We claim that algorithms for monitoring streaming real-valued time series for distributional changes can enable precisely this functionality for many aspects of the operation of intelligent, autonomous agents.

Consider a search and rescue robot with a classifier that finds people in camera images. The classifier is a Support Vector Machine (SVM) trained in the lab where labeled

[1] This procedure was identified as such in our earlier work on the Metacognitive Loop (Anderson, Oates, Chong & Perlis, 2006; Schmill et al., 2011) and is at least implicit in the still earlier Meta-AQUA system (Cox & Ram, 1999).

examples are plentiful, but it is deployed in a building filled with smoke and the robot finds none of the 10 people thought to be inside. The problem is that the change in image quality affects the accuracy of the SVM. In the absence of human feedback during the mission, what are the indications that something has gone wrong? How can the problem reliably be attributed to the change in the SVM's input distribution rather than, say, damage to the robot's wheels that prevents it from exploring quickly and fully or that the 10 people have already been found and rescued? In prior work (Dredze, Oates, & Piatko, 2010), we showed that these questions can be answered by treating the margin values produced by an SVM[2] as it classifies unlabeled instances as a univariate data stream and watching that stream for changes using a metric known as the A-distance. In the remainder of this section we describe the A-distance metric, and in the next section we show how it can be applied to domains where the underlying state representation is symbolic rather than numeric.

3.1 A-distance

The problem of finding points at which the statistical properties of a real-valued time series change has been studied extensively in statistics (Caron, Doucet, & Gottardo 2012), machine learning (Saatci, Turner, & Rasmussen 2010), and data mining (Kifer, Ben-David, & Gehrke 2004), and is often called change-point detection. Practical applications include such tasks as fault detection (an electronic circuit or an industrial process behaves differently after a fault), intrusion detection (an intruder behaves differently from the user she is pretending to be), and video segmentation (identifying the points at which an ice skater transitions between skills during her routine). Given that the vast majority of this work assumes the underlying data are real-valued and univariate, we map our symbolic representations to real-valued time series rather than invent new symbolic change-point detection algorithms.

The *A-distance* (Kifer, Ben-David, & Gehrke, 2004) detects differences between two arbitrary probability distributions by dividing the range of a random variable into a set of (possibly overlapping) intervals, and then measuring changes in the probability that a value drawn for that variable falls into any one of the intervals (see Figure 2). If such a change is large, a change in the underlying distribution is declared. Let $\mathcal{A}$ be a set of real intervals and let A ∈ $\mathcal{A}$ be one such interval. For that interval, P(A) is the probability that a value drawn from some unknown distribution falls in the interval. The A-distance between P and P′, i.e., the difference between two distributions over the intervals, is defined in Equation 1 below.

$$d_{\mathcal{A}}(P,P') = 2\ supremum_{A\in\mathcal{A}}\ |P(A) - P'(A)| \qquad (1)$$

Two distributions are said to be different when, for a user-specified threshold ε, $d_{\mathcal{A}}(P,P')$ > ε. The A-distance is distribution independent. That is, it makes no assumptions about the form of the underlying distribution or about the form of the change that might occur. Unlike

[2] The margin is the distance separating the two closest points (i.e., the support vectors) between positive and negative examples (see Ben-Hur & Weston, 2010).

the L_1 norm,[3] the A-distance can be shown to require finitely many samples to detect distribution differences, a property that is crucial for streaming, sample-based approaches (Batu, et al., 2000; Kifer, Ben-David, & Gehrke, 2004).

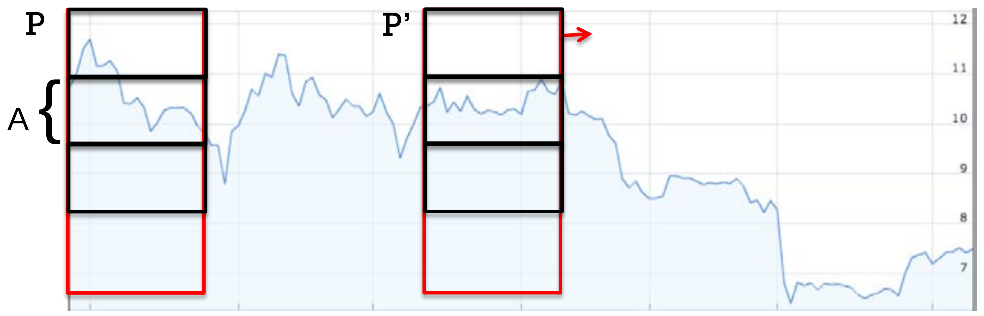

Figure 1. The A-distance compares the probability, P, that an element in a stream will fall into the "box" with height A with the probability P' it will fall into the corresponding box at a later point in time. When the probabilities differ by more than some threshold, the system infers that the underlying distribution of the stream has changed.

3.2 Applications of A-distance

In supervised classification problems, the class label produced by an SVM is based on the margin's sign and the certainty of that label is related to the margin's magnitude. As the feature distribution changes in ways that impact accuracy, the margin distribution changes as well (see Figure 3), and the A-distance can detect these changes (Dredze, Oates, & Piatko, 2010). This only requires a specification of what to monitor (the margin of the SVM that detects people), a sample of margin values obtained when the SVM works well (in the lab, yielding P), and a sliding window of margin values produced over time in the field (yielding P′). The key insight is that changes in the behavior of the deployed classification algorithms can provide useful information about both when performance may have degraded and what knowledge is at fault. That is, changes in the margin distribution (i.e., changes from P to P′) both indicate a problem and point to the SVM's input data distribution as the cause.

[3] This norm is simply the length of a vector in Manhattan distance.

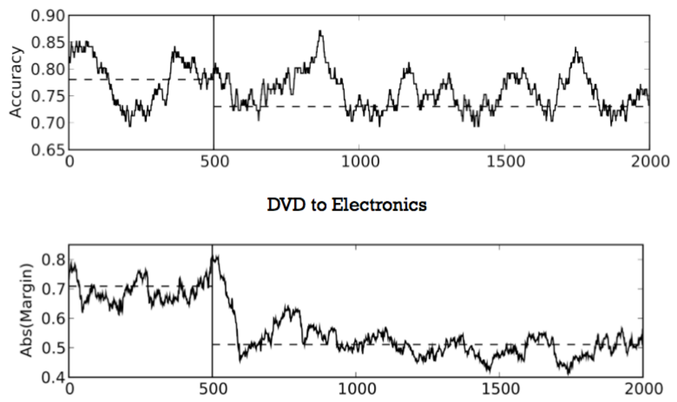

Figure 2: Plots of accuracy (upper) and margin values (lower) for an SVM trained to classify sentiment of DVD reviews (Dredze, Oates, & Piatko, 2010). The first 500 test instances are from the DVD domain while the next 1500 instances are reviews of electronics. There is a clear drop in accuracy and margin values (dotted lines show the means before and after the change), both of which can be reliably detected with A-distance trackers. The advantage of using margin values is that they do not require labeled instances yet can be used to detect changes in classification accuracy.

The A-distance can also be used to monitor changes in aspects of a system's performance explicitly optimized by its designers, such as rate of motion or stability (measured by summed pitch and roll) of a legged robot. The system designer merely has to specify the quantity to be monitored and an A-distance tracker can be deployed to detect changes. If any of these performance measures change co-temporally with the behavior of deployed classification/learning algorithms as indicated, for example, by an A-distance tracker monitoring margin values, a possible point of failure is suggested. Note that this idea of monitoring the behavior of algorithms is quite general and potentially powerful. Other values that can be monitored to indicate and localize problems include rewards obtained by following a reinforcement learning policy, frequencies of instances sorting to particular leaves in a decision tree, activations of hidden nodes in a neural network, weights computed for each training set instance as a lazy kernel regression model sees test instances, and rates of particular rules firing in a declarative knowledge base. But here we intend to demonstrate the monitoring of streams of symbolic relational states as they change during plan execution using the A-distance.

4. Environmental Change as Streams of Varying Predicate Representations

In a simple but novel approach to applying the A-distance metric to the detection of anomalies during the Note phase, consider a planning domain such as the logistics (i.e., package delivery) world (Veloso, 1994). As plans execute in the service of delivering various packages, the states of the world change in regular patterns. If a metacognitive system can monitor key relational states, it could apply the A-distance metric to indicate when something of interest has occurred in the world (or in the reasoner itself).

4.1 Symbolic Representations of Actions and States

An example state of the logistics domain is illustrated in Figure 4. The figure shows two cities, A and B, with a post office and airport in each. At Airport-B, Plane-B awaits the cargo contained in Truck-B (i.e., Object-B). To deliver the cargo to City-A, it is sufficient to unload the object from Truck-B, load it onto Plane-B, fly it to Airport-A, and then unload it. This description represents the execution of a simple four-step plan. However traditional planners do not provide an observing agent with the direct information necessary to track how the world changes as the plan executes.

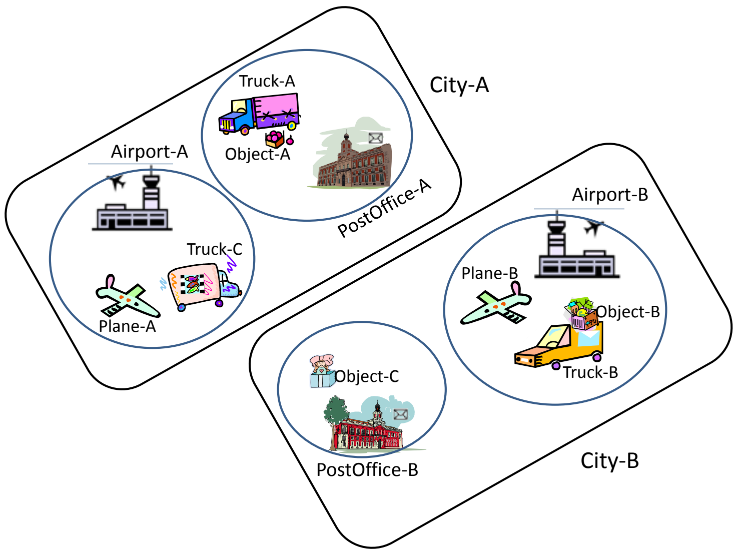

Figure 3. Example initial state in the logistics domain.

Instead consider the predicates that are true in the initial state shown in Figure 4. Table 1 enumerates these predicates. Another predicate that exists in this domain is the inside-airplane relation, but no planes are loaded with cargo, and so the predicate is not listed in the table. To convert this data to a usable, abstract representation, one need only count the number of predicates irrespective of their argument from top to bottom in the table. The abstract state representation is thus a non-negative integer vector in which each dimension represents the number of instances of a single type-generalized state predicate (Cox & Kerkez, 2006). The abstract vector-representation of the initial state is therefore `[3 2 2 1 0]`, assuming inside-airplane as the final element. A sequence of states in this representation is thus a sequence of vectors of the same length (5 in this case), and the distribution of values in each position of these vectors can be tracked for changes using the A-distance. Any change detected in this way points to a change in the prevalence of the corresponding predicate.

Table 1. Concrete literals in the initial state of Figure 4.

Predicate	Argument 1	Argument 2
at-truck	Truck-A	Postoffice-A
at-truck	Truck-B	Airport-B
at-truck	Truck-C	Airport-A
at-airplane	Plane-A	Airport-A
at-airplane	Plane-B	Airport-B
at-obj	Object-A	Postoffice-A
at-obj	Object-C	Postoffice-B
inside-truck	Object-B	Truck-B

Table 2 illustrates a novel plan representation for the four-step execution sequence. This representation maintains all intermediate, abstract states before and after each action execution. The first vector represents the problem's initial state, and the last vector is the goal state following the unloading of the plane. Although verbose, this representation allows efficient organization of plan libraries for case-based plan recognition (Kerkez & Cox, 2003), and it enables the A-distance metric to be applied to symbolic domains for change detection.

Table 2. Intermediate abstract-state vector representation of plan.

```
[3 2 2 1 0]
Unload-tr (Truck-B)
[3 2 3 0 0]
Load-pl (Plane-B,Object-B)
[3 2 2 0 1]
Fly (Plane-B,City-A)
[3 2 2 0 1]
Unload-pl (Plane-B)
[3 2 3 0 0]
```

Static predicates for which no domain operators exist to modify the relation and hence remain fixed throughout plan execution sequences can be dropped for efficiency, although we do not take advantage of this strategy. For example the literal (same-city airport-A postoffice-A) asserts that both locations are in the same city (i.e., City-A), but no planning operator can change this relationship. Thus the predicate count for same-city will never change no matter what plan the agent executes.[4]

[4] The vector representations for the logistics domain shown in this paper are simplified for explanatory purposes. The actual predicate vector sequence used in the domain is (at-truck at-airplane part-of at-obj inside-airplane inside-truck loc-at same-city).

4.2 A-distance Applied to Predicate Streams from Vector Representations

To note major changes in the domain, an agent compares the A-distance between a sample window known to be non-anomalous and a sliding window moving through the current predicate stream. Once the A-distance is greater than a constant domain-specific threshold parameter, the agent can conclude that the underlying probabilistic distribution differs between the windows and hence an anomaly exists. For domains such as blocksworld and logistics, we use this method for each predicate stream and take the first anomaly in an arbitrary left to right order in the vector representation. Such an anomaly indicates that the corresponding predicate occurs more or less frequently in recent state representations than in the past.

Note that applying the A-distance to any stream of data requires the user to specify the window size and the tiling. In the experiments described below, the window size n=100 was chosen manually to be large enough to, on average, span all of the steps in a small number (e.g., 5 – 10) of plans. This ensures that the data in the window captures enough variation to get a good estimate of the distribution of predicate counts but is not so large as to include anomalous data initially. Also, because planning operators are intended to change the state of the world, we stream the discrete first derivative of the raw predicate counts rather than streaming the counts themselves. That way, the A-distance signals when the rate of change of a count is unusual regardless of the level of the count. Finally, all values in this stream are integers, so we tile the space of integers using intervals of the form $\mathcal{A} = \{[i - 0.5, i + 0.5]\}$ for all integers i in the observed range. In this way all discrete values have their own tile and accumulate their own counts. Tiles are created as needed (i.e., as their corresponding values are seen in the stream).

Table 3 presents the algorithms used to gather data in each predicate stream. P and P' are arrays of probabilities, one each for each interval in set $\mathcal{A}$. Input is treated as a FIFO queue as are the base window, w, and sliding window, w'.[5] The probability update function adjusts the probabilities as w' traverses the input observations. This algorithm is performed upon each predicate stream in the sequence of intermediate abstract-state vectors.

[5] Note that the procedure `deleteQ` is a function that returns a queue. This is important in the recursive call of function `IsAnomalous`.

Table 3: Algorithms for processing a single stream.

```
procedure ProcessStream (input:queue; n:int; $\mathcal{A}$:set; $\varepsilon$:0.0..1.0)
w ← createQ (w); w′ ← createQ (w′)
last ← front (input)
input ← deleteQ (input)
for i ← 2 .. n
  w ← addQ (w, |last - front(input)|)
  last ← front (input)
  input ← deleteQ (input)
w′ ← w
P ← initProbabilities (w, $\mathcal{A}$)
P′ ← initProbabilities (w′, $\mathcal{A}$)
IsAnomalous (last, input, w′, P, P′, $\mathcal{A}$, $\varepsilon$)


;;; The returned list is of the form [ head | tail ]
function IsAnomalous
  (last:int; input, w′:queue; P, P′:array; $\mathcal{A}$:set; $\varepsilon$:0.0..1.0):list
if emptyQ(input) then return []
w′ ← addQ (deleteQ(w′),|last-front(input)|)
P′ ← update (w′,$\mathcal{A}$,P′)
IsAnomalous ←
   [$d_{\mathcal{A}}$(P,P′)>$\varepsilon$ | IsAnomalous (front(input),deleteQ(input),w′,P,P′,$\mathcal{A}$,$\varepsilon$)]
```

5. Evaluation

This section presents results from two experiments. In the first, we consider only the initial instance of a reported anomaly. If the report is correct, we label that test a success. Otherwise, or if A-distance runs to completion without detecting an anomaly that is present, it is a failure. In the second experiment, we perform a more fine-grained analysis. Each time the sliding window shifts one data point, we compare the A-distance result with the true world state (normal or anomalous), and a value is stored representing a hit, miss, false positive or true negative. These data are then evaluated using several statistical performance metrics.

5.1 Experimental Design

The same basic experimental design was used for both experiments. We used the Prodigy/Agent planning software (Cox, Edwin, Balasubramanian, & Elahi, 2001; Veloso et al., 1995) to generate data for these tests. First we generated two separate series of pseudo-random problems. One series was from the logistics domain described in section 4, and the other was from the classical blocksworld domain. Blocksworld problems contained up to eight blocks in the initial state, and logistics problems contained up to five packages to deliver. Prodigy/Agent was run on both series to extract those plans that had solutions within a 300 second time bound. These plans were used as the normal data for each domain. We then removed the unstack action operator from the blocksworld domain and the unload-airplane operator from the logistics domain and re-ran the planner on the problem series. The missing operators cause a shift in the types of solutions generated, and the resulting plans constituted the anomalous data for their respective domains. We concatenated various randomizations of the plan sets to simulate a dynamic world that exhibits an anomaly. These two collections of events constituted the experimental data for our test. The experimental task was then to detect locations where change occurs.

The data for an individual test consist of an *experimental block* in the format illustrated by Figure 5. Normal plans represent solved problems using a complete and uncorrupted domain. Anomalous plans represent those solved with domains having missing operators. All experimental blocks start with a *stable area* that is guaranteed to have no anomaly (i.e., only normal plans) so that the algorithm can determine the underlying normalcy distribution. This area is then followed by the main body within which an anomalous *target sequence* may exist. As shown in the figure, the target sequence could be at any one of ten locations to the right of the safe area. The interval separation between potential target locations is called an *increment*. The size of the stable area is measured by the sum of one increment and the width of the target.

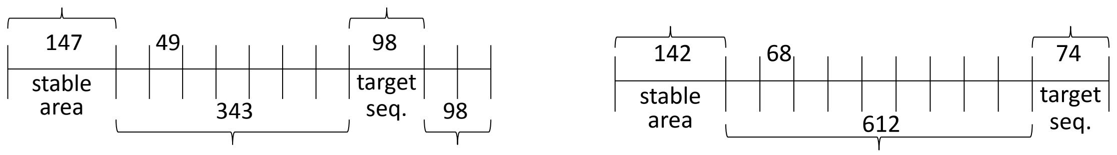

Figure 4. Experimental block structures. Numbers indicate lengths in terms of number of plans. On the left is a block for the blocksworld domain showing an anomaly target sequence two increments from the right. On the right is a block for logistics with the target flush right.

For the blocksworld domain shown in the figure, the increment width is 49 plan sequences and the target width is 98, so the stable area consists of 147 normal plans concatenated together. For the logistics domain, the increment width is 68 and the target width is 74. The differences in widths exist because more blocksworld problems were solved with a missing operator than with logistics. In the blocksworld domain, all experimental blocks contain 686 plans of varying length; whereas all blocks in the logistics domain contain 828 plans.[6]

The target sequence will vary in its proportion of anomalous plans across different blocks. We differentiate the *intensity* of an anomaly represented by a target sequence by drawing from both normal and anomalous plan pools and randomizing the order of the plans within the sequence. The intensity varies in increments of 10% from 100% anomalous to 0%. At intensity 100, the target consists of only anomalous plans; at 0 intensity, it has only normal plans; and at intensity 50 the target is half anomalous and half normal.

[6] Normal blocksworld plans average 10.33 steps with a standard deviation of 5.82, and anomalous plans average 5.44 steps with a 2.56 standard deviation. Logistics statistics are 8.05 steps on average with a 3.99 standard deviation for normal plans and 3.71 steps with a 1.37 standard deviation for anomalous.

The task of the A-distance metric is to detect the presence of the target sequence while minimizing false positives. So in the left block of Figure 5, the system should begin to recognize anomalous states at a location equal to or slightly greater than plan number 147+343+1 = 491. It should then identify the states of the following 98 plans as also anomalous. However because intensities less than 100 include normal plans, the first plan in a target may not actually be anomalous. For blocks with a target sequence with intensity 0, all data are normal. Thus for all predicate streams in each 0-intensity block, the A-distance metric should find no perceptible difference in underlying distributions.

Given that ten target locations exist and for each one we vary the intensity across eleven variations, 110 total blocks of data exist in each trial of the two domains. In both experiments, we run ten such trials for each domain and average the performance. We also select 12 different threshold values (i.e., epsilon) to test the A-distance ranging from .2 to .65 in .05 increments. Additionally, the average number of steps in a block is 7425.4 for blocksworld and 5679.9 for logistics. Given that the abstract state vectors in the data produce 5 predicate streams in blocksworld and 8 streams in logistics, our second, more thorough analysis uses results from a total of 490,076,400 empirical observations in blocksworld and 599,797,440 observations in logistics.

5.2 Experiment 1

In our first analysis (Cox, Oates, Paisner, & Perlis, 2012), we looked at only one measure of performance: *accuracy*. A successful (i.e., accurate) result was recorded if the first reported anomaly actually occurred in the anomalous section of the data. If it instead occurred before the anomalous section (a false positive), it was a failure; if no hit was recorded by the end of the anomaly that also counted as a failure. The only exception to this rule was for the intensity 0 anomalous sections, which actually contained only normal data. In these instances, if A-distance reported no anomalies the run was a success and otherwise a failure.

Figure 6 shows accurate performance expressed as a percentage and graphed as a function of the intensity of the anomaly for the logistics domain. With the exception of 0-intensity blocks, the higher the intensity the greater the performance of the note phase. That is, the A-distance metric can detect persistent anomalies with near 100% accuracy and does so with few false positives or misses, and it correctly rejects normal plans as being non-anomalous.

This performance varies depending upon the value of the threshold epsilon used to determine when the A-distance is great enough to signal the anomaly. Given high thresholds, accuracy is equal to or near 100 percent for high intensity targets (i.e., for targets with mainly anomalous plans), and the performance quickly falls off for less intense anomalies. With low thresholds, the algorithm performs well across a wider span of intensity values, though less well at the end points of 100 and 0. This represents a classic tradeoff between maximizing hits and minimizing false positives in signal-detection theory.

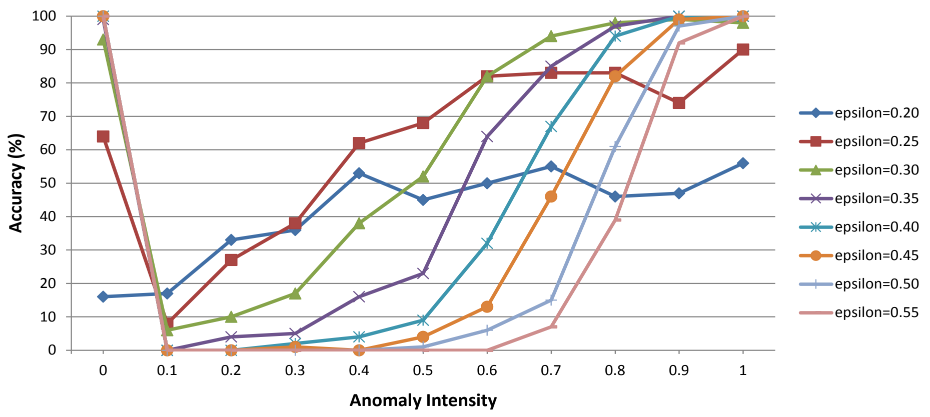

Figure 5. Logistics domain: Accuracy as a function of anomaly intensity.

The performance in the blocksworld domain is comparable to logistics as can be seen in Figure 7. The accuracy deviates somewhat from Figure 6, but the trends remain the same. The greatest difference is that similar performance graphs map to approximately .1 higher epsilon values in the blocksworld domain. See Figure 8 for the comparison.

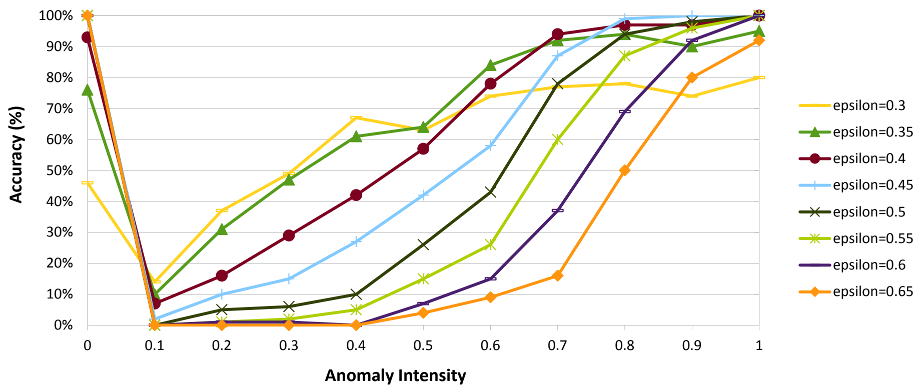

Figure 6. Blocksworld domain: Accuracy as a function of anomaly intensity.

Figure 8 illustrates the shift in performance graphically. Here accuracy averaged over all intensities is shown as a function of increasing epsilon values. In the blocksworld, overall accuracy reaches a maximum of 67.64 at an epsilon value of .35; whereas in logistics, a maximum accuracy of 62.5 occurs at an epsilon value of .3. These data show rough trends when using A-distance for symbolic representations. The next experiment examines our approach more closely.

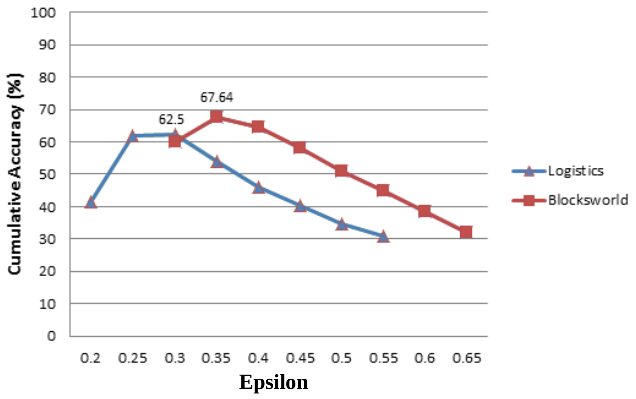

Figure 7. Average overall accuracy as a function of threshold (epsilon) value.

5.3 Experiment 2

Our second analysis (Cox, Oates, Paisner, & Perlis, 2013), rather than labeling each run as either a success or a failure, looks at A-distance's performance across all observations in each data set. The analyses that follow divide the data into four standard categories: hit, miss, false positive and true negative. To accomplish this, as each data point is added to the sliding window, the Boolean result of applying A-distance (reflecting whether an anomaly was detected) is compared with a Boolean value representing whether the data point is within the anomalous section of the data, the target.

Using these data, we calculated several standard statistical performance metrics: *recall* (hits divided by the sum of hits and misses), *precision* (hits divided by the sum of hits and false positives), and *F1* (the harmonic mean of precision and recall). As such recall represents the fraction of correctly identified anomalies within all of the identifications that actually occurred; whereas precision represents the fraction of correctly identified anomalies within the total that should have occurred. Figures 9-14 show these data graphed as functions of anomaly intensity for blocksworld and logistics.

Some general trends emerge in both domains. Recall is very good at high intensities, but this number drops as the percentage of anomalous data in the target area declines, falling under 0.1 for most epsilon threshold values as the fraction of anomalous plans drops below 50%. As might be expected, there is also a consistent negative correlation between epsilon value and recall that applies without exception across both domains, with the highest tested epsilon value of 0.75 yielding the worst recall rate, and the lowest value of 0.2 generating the best performance in this area.

Precision data paint a more complicated picture. For most epsilons, precision increases to a maximum near the midpoint of the intensity range and then goes down again as intensity approaches 1. This is a perplexing result at first glance – one would expect that the number of false positives would be independent of intensity and that hit percentage would only increase as anomalies became more easily detectable, generating a constant positive slope for precision as a function of intensity. The reason that our results differ from this ideal is that there is an inherent response lag associated with sliding window algorithms.

To decide that the A-distance exceeds epsilon, enough observations need to be gathered to approximate the distribution, so the initial portion of the target sequence will be missed. Likewise, a delay exists in determining when the distribution no longer differs from the comparison (i.e., normal) window. Therefore a large number of false positives will exist immediately past the target, reflecting that most of the data points in the window are anomalous even though the current state has returned to normal. For lower intensity targets, the algorithm will "give up" on the detection earlier and thus will have fewer false positives. For greater intensities, the lag will be larger in terms of the average number of observations after the target before the first true negative. We might have reduced the number of false positives by defining the current observation as the one in the center of the

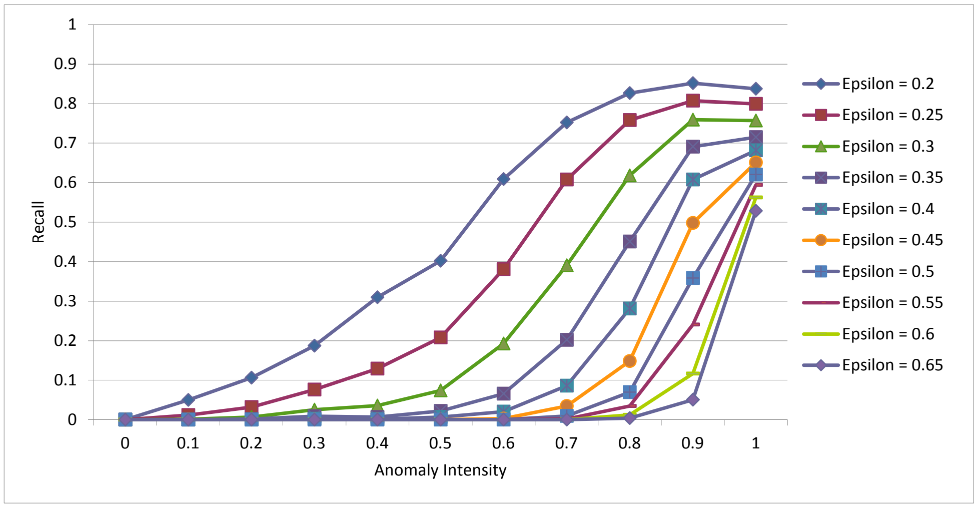

Figure 9. Recall as a Function of Anomaly Intensity [logistics]

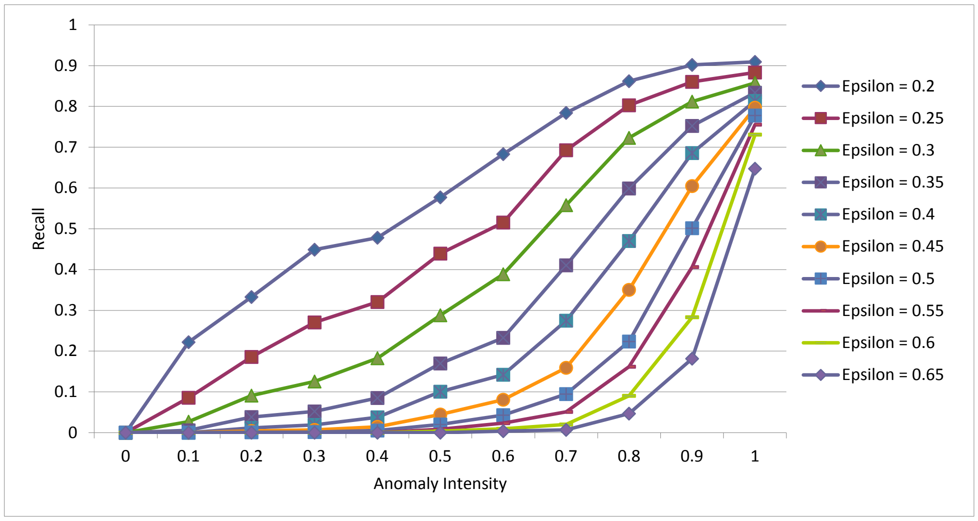

Figure 8. Recall as a Function of Anomaly Intensity [blocksworld]

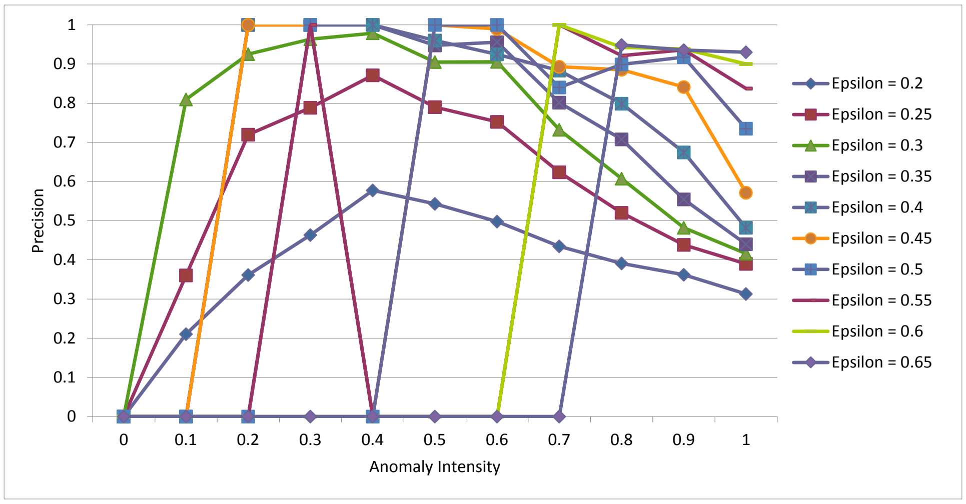

Figure 11. Precision as a Function of Anomaly Intensity [logistics]

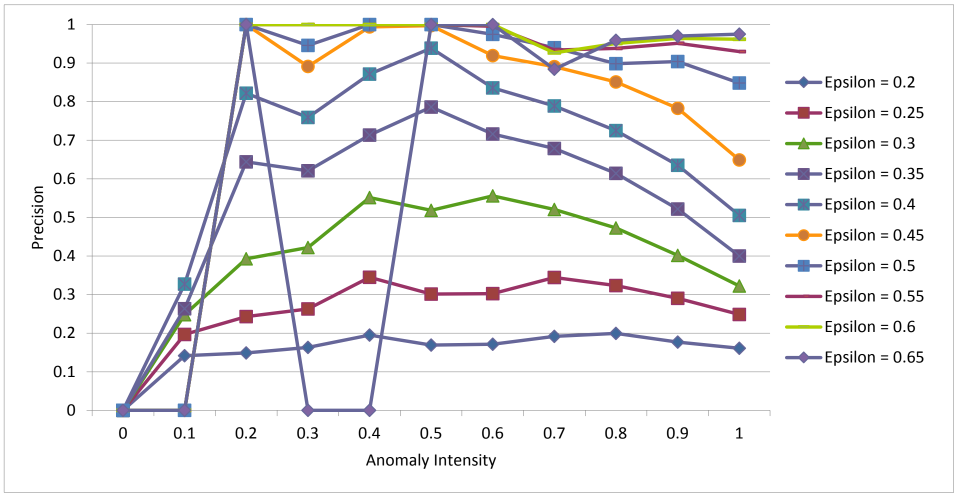

Figure 10. Precision as a Function of Anomaly Intensity [blocksworld]

window rather than on the leading edge, but this would in a sense be a false improvement - in an on-line setting we would merely have replaced the delay we have now with a predetermined response lag of half the window size.

Another unusual feature of the precision graphs is the tendency to jump suddenly from 0% to 100% precision, and sometimes back again. This trend is especially pronounced at higher epsilon values and in the logistics domain. The reason for it is that the number of recorded hits is extremely small for high epsilon and low intensity. In these cases, there might only be one or two hits in the entire target space for a given intensity and no false positives, yielding a perfect precision rating. When intensity falls by 0.1 in the next observation, the anomaly becomes too faint to be detected at all and no positives of any sort are reported, dropping precision to 0. Though many overall observations are recorded, for high epsilon and low intensity we have very few positive examples, causing the variance behavior associated with a small sample size.

F1 is the harmonic mean of recall and precision and reflects the trends described previously. What we see in Figures 13 and 14 is that blocksworld's performance is somewhat more consistent across intensities, reflecting both a larger false positive rate (which is, lag bias aside, broadly constant across intensities) and a recall rate that falls less quickly with intensity than in the logistics domain. F1 performance in logistics is superior in mid-range intensities, whereas blocksworld performance is better in lower and higher intensity values. It also reveals some differences between blocksworld and logistics in terms of optimal epsilon values, which can be seen more clearly in Figures 15 and 16.

These charts graph three different F-statistics as functions of epsilon. F(x) reflects differently weighted harmonic means of precision and recall, with higher x values skewing towards recall. F values in logistics are generally highest at the lowest epsilon value tested, 0.2, and fall fairly constantly from there. In blocksworld, the optimal epsilon seems to be

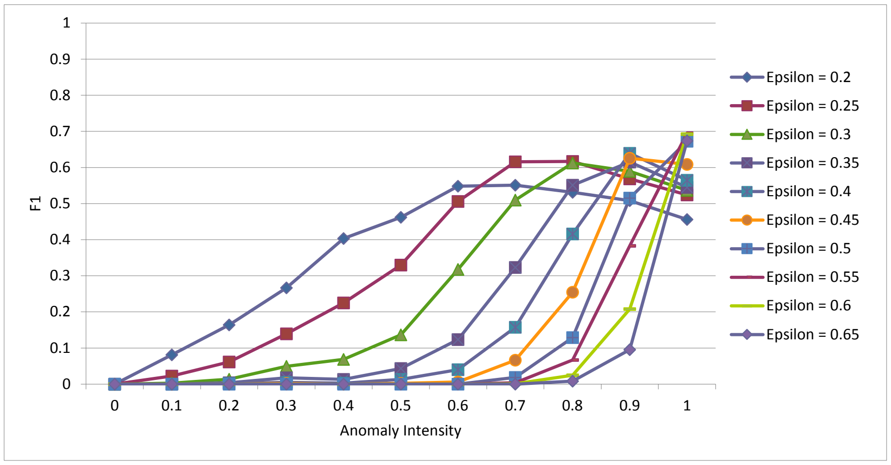

Figure 12. F1 as a Function of Anomaly Intensity [logistics]

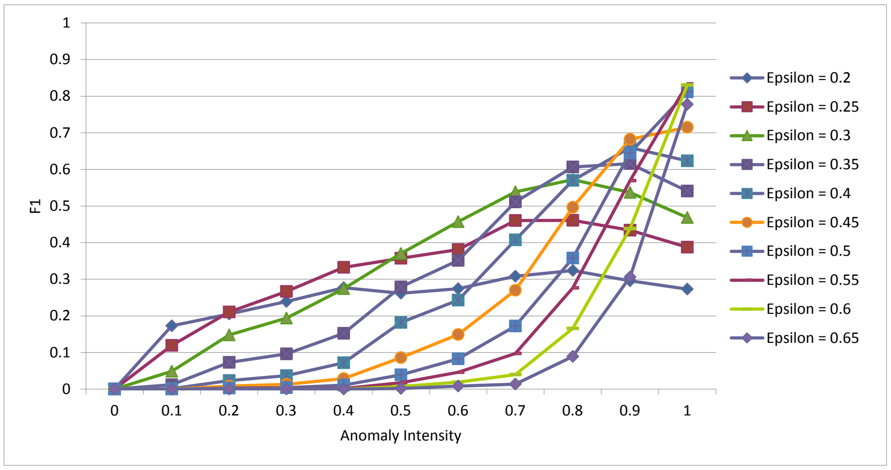

Figure 13. F1 as a Function of Anomaly Intensity [blocksworld]

closer to 0.3, depending on which f value is used. This result seems to reflect a larger base variance in blocksworld distributions, requiring a higher epsilon value to reduce false positives.

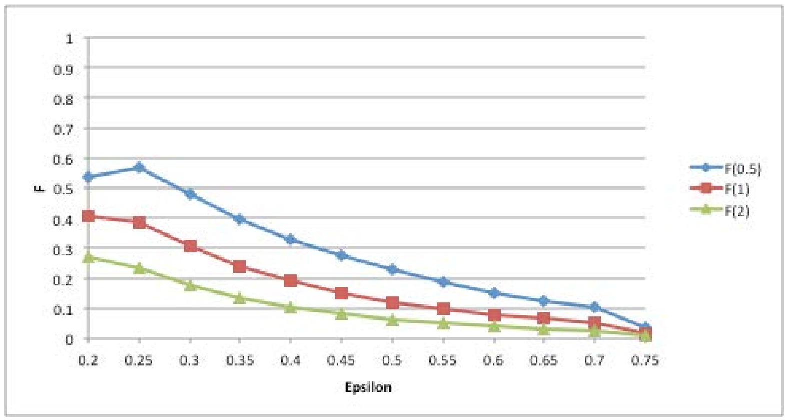

Figure 15. F Values as Functions of Epsilon [logistics].

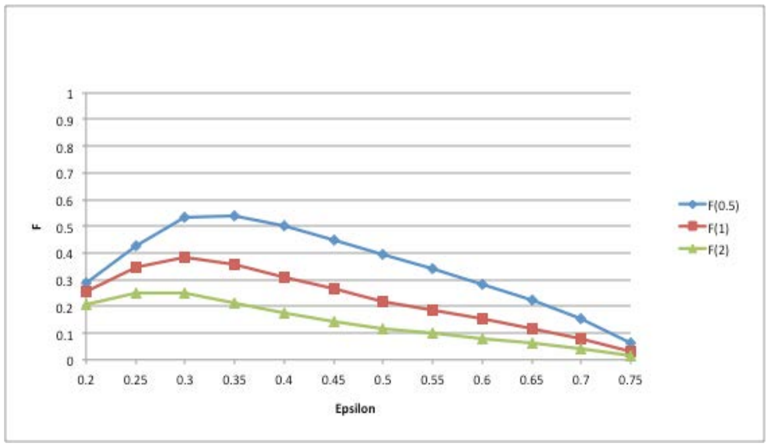

Figure 14. F Values as Functions of Epsilon [blocksworld].

6. Related Research

Detecting anomalies in the input of agents and other intelligent systems represent a significant scientific problem in many areas of interest to the artificial intelligence community (e.g., Albrecht et al., 2000; Basseville & Nikiforov 1993; Crook and Hayes 2001; Fawcett and Provost 1999; Janssens, Postma, & Hellemons 2011; Pannell, & Ashman 2010). A very large body of work exists on the subject of anomaly detection in data streams (see Chandola, Banerjee, & Kumar 2009). In general, such work faces challenges posed by the huge volume of data present in many domains. Because of this, algorithms that perform computations using all of the accumulated data at each step are often infeasible in practice. One proposed solution to this problem is the “s-monitor” approach (Huang et al. 2009). An s-monitor is a simple model of the data which is used to check for changes frequently, while a more complete analysis is performed at wider intervals. The success of this method depends on the quality of the models, but it can be both effective and efficient if these are implemented well.

A-distance employs another approach to the problem of big data, the sliding window model. Many versions of this class of algorithm have been studied using a variety of statistical tools. One study (Khalastchi et al., 2001) examined the use of one such algorithm in autonomous robots. In this study, Mahalanobis distance, which is a statistical distance metric similar to a-distance, was used to detect anomalies using a single sliding window. Several experiments were conducted using aerial ground-based and simulated agents. Results suggested that sliding window models could be usefully applied in real-time.

In addition, several innovations have been developed for customizing the parameters of the algorithm. One modification introduced variable, self-adjusting window sizes (Deypir, Sadreddini, & Hashemi 2012), while another added automatic recalibration of the threshold for declaring an anomaly (Ahmed, Clark, & Mohay 2008). These changes provide sliding window style approaches with more flexibility for dealing with different or fluctuating data sets without extensive user customization.

Anomaly detection is applied to a wide array of domains, including health, sports, meteorology, web security, and others. Many example applications take real-valued n-dimensional streams as input, but there are also some that analyze text or other forms of value-restricted or symbolic data (Agyemang, Barker, & Alhajj 2006). Like A-distance, they generally require that the data be converted to some form of numeric value or vector which can then be analyzed statistically as a stream.

Some anomaly detection algorithms also use rule-based strategies. These are especially important in cases where certain types of anomalies are considered important and others are not, so that a purely statistical model would generate many false positives. Rule-based techniques have been used in detecting disease outbreaks (Wong et al. 2002), and in computer intrusion detection using state transition analysis (Ilgun, Kemmerer, & Porras, 1995). One drawback of this type of approach compared with a statistical model is that such algorithms must be tailored to fit their particular domain, and are usually not generalizable.

Time series data streams, in which each data point follows temporally and is dependent on those that came before it, are an important subcategory of anomaly detection problems. They have many real world applications, from real-time video analysis (Lelescu & Schonfeld 2003) to searching for perturbations in the data from nuclear reactors (Kozma et al. 1994). While statistical time series anomaly detection has been studied extensively, it has generally been used to analyze numeric data streams. Symbolic domains such as blocksworld and logistics have received much less attention due to the apparent difficulty of performing statistical analyses.

In the larger context of cognitive systems, many others have also focused upon failure-driven reasoning and learning that begins with noting anomalies or expectation failures. For example, it has long been argued that failure provides both human and artificial reasoners with strong clues when deciding what needs to be learned (Birnbaum, Collins, Freed, & Krulwich, 1990; Cox & Ram, 1994; Fox & Leake, 1995; Hammond, 1986; Kolodner, 1987; Pazzani, 1990; Reason, 1992; Schank & Owens, 1987; Sussman, 1975; VanLehn, 1991). One of the earliest AI systems in this tradition is HACKER (Sussman, 1975). In planning domains, it would notice when plan execution failures occur due to negative goal interactions and respond by debugging the goal ordering of the plan. More generally, however, the NAG approach can be situated within systems exhibiting goal-driven autonomy and more generally goal reasoning (Vattam, Klenk, Molineaux, & Aha, 2013).

7. Conclusion

This report has presented the results of two sets of experiments employing A-distance in the two symbolic domains of logistics and blocksworld. This is significant as an extension of statistical anomaly detection techniques to a category of environments (i.e., planning domains) in which they had previously been absent. It is also a significant building block in our research work on metacognitive architectures (see Cox, Maynord, Paisner, Perlis, & Oates, 2013; Maynord, Cox, Paisner, & Perlis, 2013) and goal-driven autonomy (e.g., Cox, 2013; Paisner, Cox, Maynord, & Perlis, in press). More generally it is a step toward the computational recognition and analysis of problems. Not all anomalies are problems, but many problems start with some discrepancy or anomaly in the world. Recognizing and classifying anomalous changes represents a first step in understanding what goes wrong in an agent's environment and improving its performance in complex domains.

In subsequent work, we intend to link this research to the assess phase of the NAG procedure by developing methods to increase the depth of analysis when anomalies are detected. In planning domains, we will examine ways of using the anomalous predicates discovered by A-distance to reason about which operators are either most dependent on the predicates (through pre-conditions) or most likely to affect the predicates (through post-conditions). For example, in the logistics world experiments above the predicate most often implicated in the detection of change was inside-airplane. This predicate was also in the effects list of the dropped unload-airplane operator. Assessment might infer that a change in the occurrence of this action was the cause of the observed change and might justify this inference by hypothesizing an event such as a union work stoppage to account for the lack of these actions in the observation stream (using prior knowledge about such events). Unfortunately inside-truck was a second predicate that saw A-distances greater than the given epsilon value (hence correlated with change). Because this predicate is indirectly involved in the causal relationships associated with the change, assessment is neither simple nor obvious. Future research remains to determine the most effective approaches for the assessment and guide phases that leverage the results reported here, as well how to best incorporate this approach with preexisting work.

Acknowledgements

This material is based upon work supported by ONR Grants N00014-12-1-0430, N00014-12-1-0172, and N00014-13-1-0798 and by ARO Grant W911NF-12-1-0471.

References

Agyemang, M., Barker, K., & Alhajj, R. (2006). A comprehensive survey of numeric and symbolic outlier mining techniques. *Intelligent Data Analysis*, 10(6).

Ahmed, E., Clark, A., & Mohay, G. (2008). A novel sliding window based change detection algorithm for asymmetric traffic. In *Proceedings of the IFIP International Conference on Network and Parallel Computing 2008*, 168-175. IEEE.

Albrecht, S., Busch, J., Kloppenburg, M., Metze, F., & Tavan P. (2000). Generalized radial basis function networks for classification and novelty detection: Self-organization of optional Bayesian decision. *Neural Networks 13*(10): 1075–1093.

Anderson, M., Oates, T., Chong, W., & Perlis, D. (2006). The metacognitive loop I: Enhancing reinforcement learning with metacognitive monitoring and control for improved perturbation tolerance. *Journal of Experimental and Theoretical Artificial Intelligence, 18*(3), 387-411.

Anderson, M. L., & Perlis, D. (2005). Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness. *Journal of Logic and Computation 15*(1).

Basseville, M. & Nikiforov, I. V. (1993). *Detection of abrupt changes - Theory and application.* Englewood Cliffs, NJ: Prentice-Hall.

Batu, T., Fortnow, L., Rubinfeld, R., Smith, W. D., & White, P. (2000). Testing that distributions are close. In *Proceedings of 41st Annual IEEE Symposium on Foundations of Computer Science (FOCS 2000)* (pp. 259-269). IEEE.

Ben-Hur, A., & Weston, J. (2010). A user's guide to support vector machines. *Methods in Molecular Biology, 609*, 223-239.

Birnbaum, L., Collins, G., Freed, M., & Krulwich, B. (1990). Model-based diagnosis of planning failures. In *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 318-323). Menlo Park, CA: AAAI Press.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys 41*(3).

Cox, M. T. (2007). Perpetual self-aware cognitive agents. *AI Magazine 28*(1), 32-45.

Cox, M. T. (2013). Question-based problem recognition and goal-driven autonomy. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), *Goal Reasoning: Papers from the ACS workshop* (pp. 10-25). Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.

Cox, M. T., Edwin, G., Balasubramanian, K., & Elahi, M. (2001). Multiagent goal transformation and mixed-initiative planning using Prodigy/Agent. In N. Callaos, B. Sanchez, L. H. Encinas, & J. G. Busse (Eds.), *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics, Vol. VII* (pp. 1-6). Orlando, FL: International Institute of Informatics and Systemics.

Cox, M. T., & Kerkez, B. (2006). Case-based plan recognition with novel input. *International Journal of Control and Intelligent Systems, 34*(2), 96-104.

Cox, M. T., Maynord, M., Paisner, M., Perlis, D., & Oates, T. (2013). The integration of cognitive and metacognitive processes with data-driven and knowledge-rich structures. In *Proceedings of the Annual Meeting of the International Association for Computing and Philosophy*. IACAP-2013.

Cox, M. T., Oates, T., Paisner, M., & Perlis, D. (2012). Noting anomalies in streams of symbolic predicates using A-distance. *Advances in Cognitive Systems 2*, 167-184.

Cox, M. T., Oates, T., Paisner, M., & Perlis, D. (2013). Detecting change in diverse symbolic worlds. In L. Correia, L. P. Reis, L. M. Gomes, H. Guerra, & P. Cardoso (Eds.), *Advances in Artificial Intelligence, 16th Portuguese Conference on Artificial Intelligence* (pp. 179-190). University of the Azores, Portugal: CMATI.

Cox, M. T., & Ram, A. (1994). Failure-driven learning as input bias. In A. Ram & K. Eiselt (Eds.), In *Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society* (pp. 231-236). Hillsdale, NJ: Lawrence Erlbaum.

Cox, M. T., & Ram, A. (1999). Introspective multistrategy learning: On the construction of learning strategies. *Artificial Intelligence, 112*: 1-55.

Crook, P. & Hayes, G. 2001. A robot Implementation of a Biologically Inspired Method for Novelty Detection. In *Proceedings of Towards Intelligent Mobile Robots Conference*, Edinburgh, Scotland: Division of Informatics, University of Edinburgh.

Deypir, M., Sadreddini, M. H., and Hashemi, S. (2012). Towards a variable size sliding window model for frequent itemset mining over data streams. *Computers & Industrial Engineering, 63*(1), 161-172.

Dredze, M., Oates, T., & Piatko, C. (2010). We're not in Kansas anymore: Detecting domain changes in streams. In *Proceedings of Empirical Methods in Natural Language Processing* (pp. 585-595). Stroudsburg, PA: Association for Computational Linguistics.

Fawcett, T., & Provost, F. (1999). Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining (SIGKDD99)* 53-62. New York: Association for Computing Machinery.

Fox, S., & Leake, D. (1995). Modeling case-based planning for repairing reasoning failures. In *Proceedings of the 1995 AAAI Spring Symposium on Representing Mental States and Mechanisms* (pp. 31-38). Menlo Park, CA: AAAI Press.

Hammond, K. J. (1986). Learning to anticipate and avoid planning problems through the explanation of failures. In *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 556-560). Los Altos, CA: Morgan Kaufmann.

Huang, W., Omiecinsky, E., Mark, L., & Nguyen, M. Q. (2009). History guided low-cost change detection in streams. In *Data Warehousing and Knowledge Discovery, Proceedings of the 11th International Conference,* 75-86. Berlin: Springer.

Ilgun, K., Kemmerer, R.A., & Porras, P.A., (1995). State transition analysis: A rule-based intrusion detection approach. *IEEE Transactions on Software Engineering, 21*(3), 181-199.

Janssens, J., Postma, E., & Hellemons, J. (Eds.) (2011). *Proceedings of International Workshop on Maritime Anomaly Detection 2011*. Tilburg, The Netherlands: Tilburg Center for Cognition and Communication, Tilburg University.

Khalastchi, E., Kaminka, G., Kalech, M., & Lin, R. (2001). Online anomaly detection in unmanned vehicles. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, 115-122. Taipei, Taiwan: International Foundation for Autonomous Agents and Multiagent Systems.

Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the 30th Very Large Databases Conference*, 180-191, VLDB Endowment

Kerkez, B., & Cox, M. T. (2003). Incremental case-based plan recognition with local predictions. *International Journal on Artificial Intelligence Tools: Architectures, languages, algorithms, 12*(4), 413-464.

Klenk, M., Molineaux, M., & Aha, D.W. (2013). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence, 29*(2), 187–206.

Kolodner, J. L. (1987). Capitalizing on failure through case-based inference. In *Proceedings of the Ninth Annual Conference of the Cognitive Science Society* (pp. 715-726). Hillsdale, NJ: Lawrence Erlbaum.

Kozma, R., Kitamura, M., Sakuma, M., & Yokoyama, Y. (1994). Anomaly Detection by Neural Network Models and Statistical Time Series Analysis. In *Proceedings IEEE World Congress on Computational Intelligence, IEEE International Conference on Neural Networks* Vol. 5, 3207-3210. IEEE Press.

Leake, D. (1989). Anomaly detection strategies for schema-based story understanding. *Proceedings of the Eleventh Annual Conference of the Cognitive Science Society* (pp. 490-497). Hillsdale, NJ: Lawrence Erlbaum.

Lelescu, D., and Schonfeld, D. (2003). Statistical Sequential Analysis for Real-time Video Scene Change Detection on Compressed Multimedia Bitstream. *IEEE Trans. Multimedia*, 5(1), 106-117.

Maynord, M., Cox, M. T., Paisner, M., & Perlis, D. (2013). Data-driven goal generation for integrated cognitive systems. In C. Lebiere & P. S. Rosenbloom (Eds.), *Integrated Cognition: Papers from the 2013 Fall Symposium* (pp. 47-54). Technical Report FS-13-03. Menlo Park, CA: AAAI Press.

Munoz-Avila, H., Jaidee, U., Aha, D. W., Carter, E. (2010). Goal-driven autonomy with case-based reasoning. In *Case-Based Reasoning. Research and Development, 18th International Conference on Case-Based Reasoning, ICCBR 2010* (pp. 228-241). Berlin: Springer.

Paisner, M., Cox, M. T., Maynord, M., Perlis, D. (in press). Goal-driven autonomy for cognitive systems. To appear in *Proceedings of the 36th Annual Conference of the Cognitive Science Society*. Austin, TX: Cognitive Science Society.

Pannell, G., and Ashman, H. (2010). Anomaly detection over user profiles for intrusion detection. In *Proceedings of the 8th Australian Information Security Management Conference*. Perth, Western Australia: Edith Cowan University.

Pazzani, M. (1990). Learning fault diagnosis heuristics from device descriptions. In Y. Kodratoff & R. S. Michalski (Eds.), *Machine learning III: An artificial intelligence approach* (pp. 214-234). San Mateo, CA: Morgan Kaufmann.

Perlis, D. (2011). There's No 'Me' in Meta - Or Is There? In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking* (pp. 15-26). Cambridge: MIT Press.

Reason, J. (1992). *Human error*. New York: Cambridge University Press.

Schank, R. C., & Owens, C. C. (1987). Understanding by explaining expectation failures. In R. G. Reilly (Ed.), *Communication failure in dialogue and discourse*. New York: Elsevier Science.

Schmill, M., Anderson, M., Fults, S., Josyula, D., Oates, T., Perlis, D., Shahri, H., Wilson, S., & Wright, D. (2011). The metacognitive loop and reasoning about anomalies. In M. T. Cox & A. Raja (Eds.), *Metareasoning: Thinking about thinking* (pp. 183-198). Cambridge, MA: MIT Press.

Sussman, G. J. (1975). *A computational model of skill acquisition*. New York: American Elsevier.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science, 15*, 1-47.

Vattam, S., Klenk, M., Molineaux, M., & Aha, D. (in press). Breadth of approaches to goal reasoning: A research survey. In D. W. Aha, M. T. Cox, & H. Munoz-Avila (Eds.), *Goal Reasoning: Papers from the ACS workshop* (pp. 111-126). Tech. Rep. No. CS-TR-5029. College Park, MD: University of Maryland, Department of Computer Science.

Veloso, M. M. (1994). *Planning and learning by analogical reasoning*. Berlin: Springer.

Veloso, M. M., Carbonell, J. Perez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: The PRODIGY Architecture. *Journal of Theoretical and Experimental Artificial Intelligence 7*(1): 81-120.

Wong, W., Moore, A., Cooper, G., & Wagner, M. (2002). Rule-based anomaly pattern detection for detecting disease outbreaks. In *Proceedings of the AAAI-02 Conference*, 217-223. Menlo Park, CA: AAAI Press.