Advances in Cognitive Systems 2 (2012) 167-184 Submitted 08/2012; published 12/2012

Noting Anomalies in Streams of Symbolic Predicates Using A-Distance

Michael T. Cox MCOX@CS.UMD.EDU
Institute for Advanced Computer Studies, University of Maryland, College Park, MD 20742 USA

Tim Oates OATES@CS.UMBC.EDU
Department of Computer Science and Electrical Engineering, University of Maryland, Baltimore County, Baltimore, MD 21250 USA

Matt Paisner MATTHEW.PAISNER@GMAIL.COM
Don Perlis PERLIS@CS.UMD.EDU
Department of Computer Science, University of Maryland, College Park, MD 20742 USA

Abstract

The MIDCA cognitive architecture has at its core a process sequence consisting of three phases. They are to *note* an anomaly, to *assess* what causes the anomaly, and then to *guide* a response to the anomaly. Here we present a novel approach to the first phase and discuss implications for the second. Our method detects a shift in streams of symbolic data that signal an anomaly and trigger in-depth understanding of the input. The approach uses a metric called the A-distance, normally used to detect shifts in distributions underlying numeric data. Instead, using a novel plan representation, we apply this metric to streams of changing predicates in various environments. Empirical results show that over a range of circumstances we are able to detect changes in the underlying domain. We suggest that these results apply to metacognitive as well as to cognitive processing.

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. Indeed, the first step in recognizing and representing a problem is to detect when the world is not as expected or desired. Although not all anomalies signal a problem, and not all problems are relevant to an individual agent, the detection of anomalous data is a key enabling step of intelligent agency. As such it is an outstanding technical problem in many 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; Klenk, Molineaux, & Aha, in press; 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. Both events serve a common function in the MIDCA (Metacognitive Integrated Dual-Cycle Architecture) architecture (Cox, Oates, & Perlis, 2011). 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. For the MIDCA architecture there exists a core process sequence called the Note-Assess-Guide (NAG) procedure wherein this processing lies. The first phase of the NAG procedure is to note an anomaly. The second is to assess the 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 (see Anderson, Chong, Oates, & Perlis, 2006, for further details on these latter two phases).

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 guiding 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 NAG 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 paper is to examine in some depth the Note phase of the NAG 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 such as MIDCA. 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 the A-distance (Kifer, Ben-David, & Gehrke, 2004), 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 the A-distance and the collapsed state representation leads to a method capable of robust detection of subtle changes in the underlying domains.

The remainder of this paper is organized as follows. The second section provides an overview of the MIDCA architecture. The third section examines the A-distance metric that has been used in previous research to detect changes in numeric data streams. The next section describes a novel representation that allows the A-distance to be used against streams of symbolic predicates describing the world as it shifts over time in response to events and forces. Then we report results from an empirical study that evaluates these representations and metric as used in the MIDCA architecture, discuss related research, and finally conclude.

2. The MIDCA Architecture

MIDCA is a metacognitive, integrated, dual-cycle architecture (Cox, Oates, & Perlis, 2011). It is based upon earlier work on computational metacognition including the MetaCognitive Loop (Anderson & Perlis, 2005; Schmill et al., 2011) and Introspective Multistrategy Learning theory (Cox & Ram, 1999). It also integrates many concepts from Norman's (1986) model of human-computer interaction (see also Lewis, 1998). For Norman, each part of the this cycle contains certain sub-processes (see the lower part of Figure 1). The output side consists of intention, planning, and action execution; the input side consists of perception, interpretation, and goal evaluation. The cycle selects a goal and commits to achieving it in a general way. The agent then creates a plan to achieve the goal and subsequently executes the planned actions to make the domain match the goal state. The agent perceives changes to the environment resulting from the actions, interprets the percepts with respect to the plan, and evaluates the interpretation with respect to the goal.

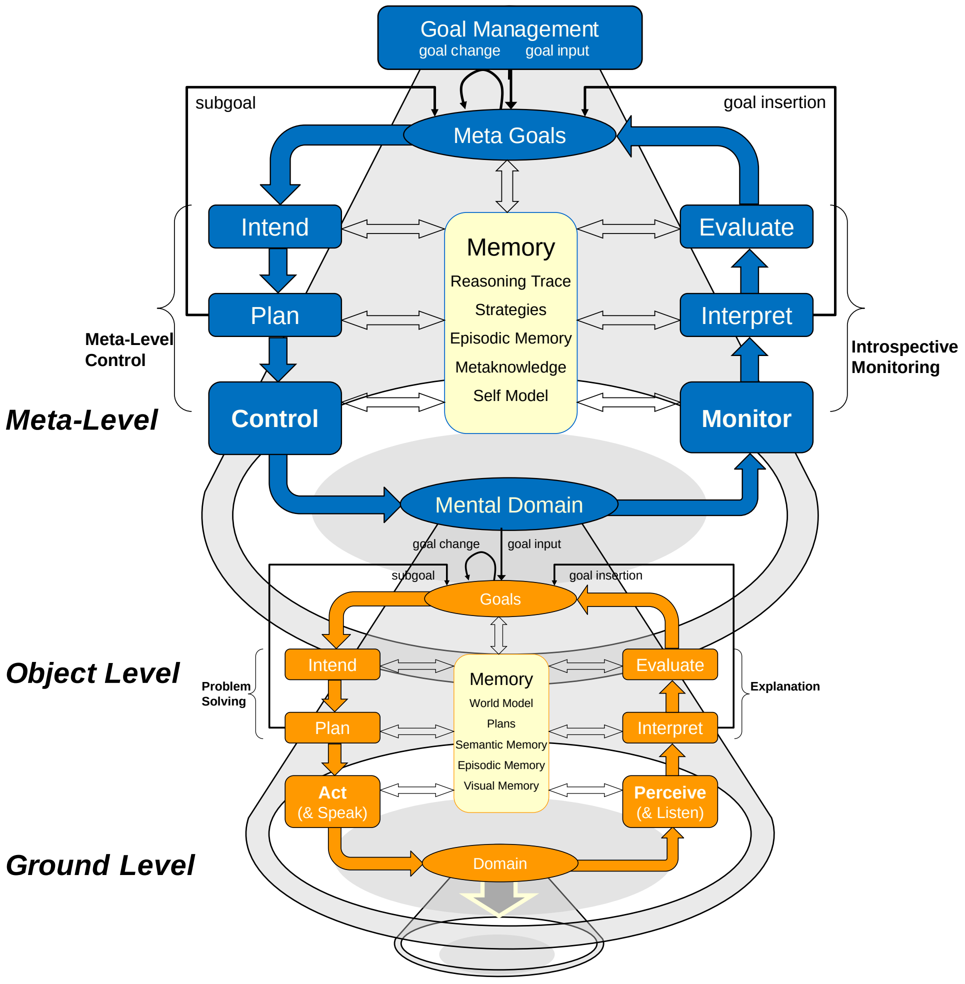

Figure 1. A metacognitive, integrated, dual-cycle architecture (MIDCA).

Figure 1 shows the structure of MIDCA with Norman-like cycles at both the cognitive (object) and metacognitive (meta) levels. The lower portion of Figure 1 represents the object and ground levels of a perception-action cycle and most closely represents Norman's conception. The figure also shows that the perception-action cycle feeds upward into a metacognitive cycle that monitors and controls the object-level, similar to the way interpretation and planning interacts with ground-level action and perception. The two loops are integrated into a novel, dual-cycle architecture.

Norman's treatment of goals is static (i.e., they are simply given). However, we contend that goals are dynamic objects that are malleable and subject to transformation and abandonment over time (Cox & Veloso, 1998; Cox & Zhang, 2007; Talamadupulay et al., 2011). Figure 1 illustrates this in the reflexive loops at the object and meta-levels that point from goals back to themselves. Goals also arise from *sub-goaling* on unsatisfied preconditions during planning (the thin black back-pointing arrows on the left of both cycles). Also new goals are formed as anomalies are detected in the environment or at the object-level. The agent recognizes the anomaly, explains what is causing it, and generates a new goal to remove or neutralize the cause (Cox, 2007). The goal-generation operation, called *goal insertion*, is indicated by the thin, black arrows that are pointing away from the interpretation processes in Figure 1.

Central to both MIDCA cycles is a Note-Assess-Guide (NAG) procedure: *Note* whether an anomaly has occurred; A*ssess* potential causes of the anomaly by generating hypotheses; and *Guide* the system through a response.[1] The note phase, part of the interpretation process, seeks to detect anomalies or discrepancies given the expectations from plans and prior reasoning. The assessment phase causally connects the anomaly to other factors hypothetically responsible for an expectation failure and is part of the evaluation process. The guide phase is part of the planning and execution processes of the cycle. Responses can take various forms, such as (1) test the hypothesis; (2) ignore (try again); (3) ask for help; or (4) change the goal. Given no anomaly, the system incorporates concepts inferred from percepts into a world model and the cycle continues.

The NAG procedure at both meta- and object-levels has two variations that represent a bottom-up, data-driven track and a top-down, knowledge rich, goal-driven track (c.f., the CLARION cognitive architecture, Sun, Zhang, & Mathews, 2006). The data-driven track we call the *D-track*; whereas the knowledge rich track we call the *K-track*. The D-track is partially implemented as a Bayesian network of ontologies (Schmill et al., 2011), and the K-track is implemented as a case-based explanation process (Cox & Burstein, 2008). Both tracks exist in each phase of the NAG procedure within both cognitive and metacognitive cycles. For the work examined here, we focus on D-track of the note phase. In the D-track, anomaly detection is to note discrepancies from the expectations implicit in the underlying distribution of states in the percepts. The expected outcome is that the probabilistic distribution will stay the same.

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

The first phase of the NAG cycle, noting an anomaly, requires some way to represent what is normal, or what is expected. Currently MIDCA generates expectations top-down from knowledge structures encoded in ontologies (see Schmill et al., 2011). But many expectations that drive reasoning and learning are implicit as well as explicit. Expectations are central to MIDCA

[1] Any architecture that has knowledge-based error-recognition and correction is likely to benefit from something like the NAG procedure; indeed, it could be seen as a direct response to the ideas put forward in Brachman (2002).

because they are the bridge between cognition and metacognition, with violations of expectations driving metacognitive reasoning about possible problems and their causes. 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 examples are plentiful, but it is deployed in a building filled with smoke and the robot finds none of the ten 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 ten people have already been found and rescued?

3.1 A-distance

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 (Kifer, Ben-David, & Gehrke, 2004). The A-distance 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 $A \in \mathcal{A}$ 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).

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

Two distributions are said to be different when, for a user-specified threshold ε, $d_{\mathcal{A}}(P,P') > \varepsilon$. 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 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).

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

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

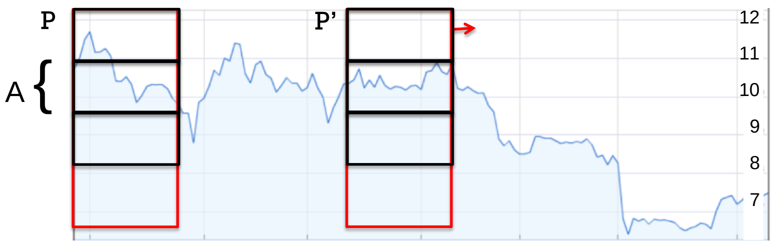

Figure 2. The A-Distance compares the probability, P, that an element in a stream will fall into the "box" with height A with the probability P' it will fall into the corresponding box at a later point in time. When the probabilities differ by more than some threshold, the system infers that the underlying distribution of the stream has changed.

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.

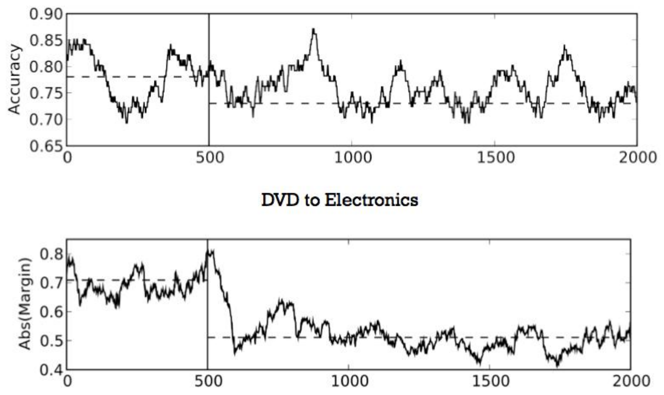

Figure 3. Plots of accuracy (upper) and margin values (lower) for an SVM trained to classify sentiment of DVD reviews (Dredze, Oates, & Piatko, 2010). The first 500 test instances are from the DVD domain while the next 1500 instances are reviews of electronics. There is a clear drop in accuracy and margin 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 knowledgebase. 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 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.

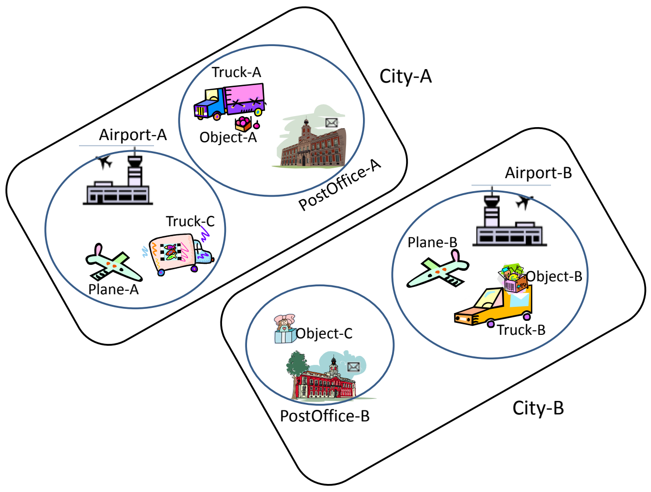

Figure 4. Example initial state in the logistics domain.

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 (five 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]

4.2 A-distance Applied to Predicate Streams from Vector Representations

To note major changes or anomalies 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.

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]

5. Evaluation

Results from an evaluation with the blocksworld and logistics domains show that the intermediate abstract-state vector representation is effective for monitoring state changes using the A-distance metric. For these domains, we apply the metric to each of the predicate streams in successive plan executions using only the intermediate vector information (dropping the planning actions themselves). To simulate a changing input distribution, we remove operators from the planning domain definitions after various periods of time. This changes the kinds of problems that can be solved in the domain and therefore the kinds of behavior observed in the stream of plans. For example, by removing the unload-airplane operator from the logistics domain, only plans that deliver packages within cities will appear subsequently.[5]

5.1 Experimental Design

We used the Prodigy/Agent planning software (Cox, Edwin, Balasubramanian, & Elahi, 2001; Veloso et al., 1995) to generate test data for this experiment. First we generated two separate series of pseudo-random problems. One series was in the classical blocksworld domain and the other problem series was in the logistics (i.e., package delivery) domain. Blocksworld problems contain 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. To generate anomalous data, we then removed the unstack action operator from the blocksworld domain and the unload-airplane operator from the logistics domain. We then re-ran the planner on the problem series. The missing operators cause a shift in the observations available in the domain, and therefore the resulting plans each constitute the

[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).

[5] One can think of the missing operator as representing a general longshoreman strike in which no planes get serviced.

anomalous data for their respective domains. We concatenated the plan sets to simulate a dynamic world that exhibits a significant change (i.e., an anomaly). These two collections of events constituted the experimental data for our test. The experimental task then is to detect the location where the change occurs.

The data for a single 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 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 Figure 5, the target sequence could be at the far end of the block. The target may also be in nine other equally separated locations from the left end of the target to the location adjoining the stable 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 target.

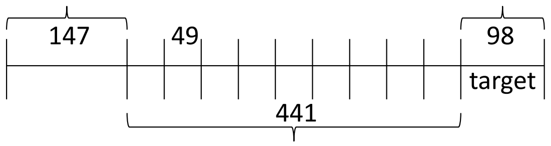

Figure 5. Experimental block structure for the blocksworld domain showing an anomaly target sequence flush right. The target can be in ten unique positions.

For the blocksworld domain shown in the figure, the increment width is 49 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[6]; whereas all blocks in the logistics domain contain 828 plans.

Within each of the ten potential target locations, the target sequence has varying proportions of anomalous plans. We differentiate the *intensity* of the 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. 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. We run ten such trials for each domain and average the performance.

The task of the A-distance metric is to detect the start location of the target sequence while minimizing false positives. So in Figure 5, the system should return a number equal to or slightly greater than 147+441+1 = 589. Because intensities less than 100 include normal plans, the first plan in a target may not be anomalous. However the delay in detecting the target should not be

[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.

greater than the width of the target itself. Therefore we (generously) count as a detection (i.e., a hit) any location within the target. The one exception is the target with intensity zero. For all predicate streams in each zero-intensity block, the A-distance metric should find no perceptible difference in underlying distributions. This is considered a true negative. We calculate the *accuracy* of the algorithm as the sum of hits and true negatives divided by the number of experimental blocks.

5.2 Experimental Results

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 or very low threshold values (i.e., epsilon of 0.20 or 0.25), the higher the intensity the greater the performance of the note phase in general. 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 zero. This represents a classic tradeoff between maximizing hits and minimizing false positives in signal-detection theory.

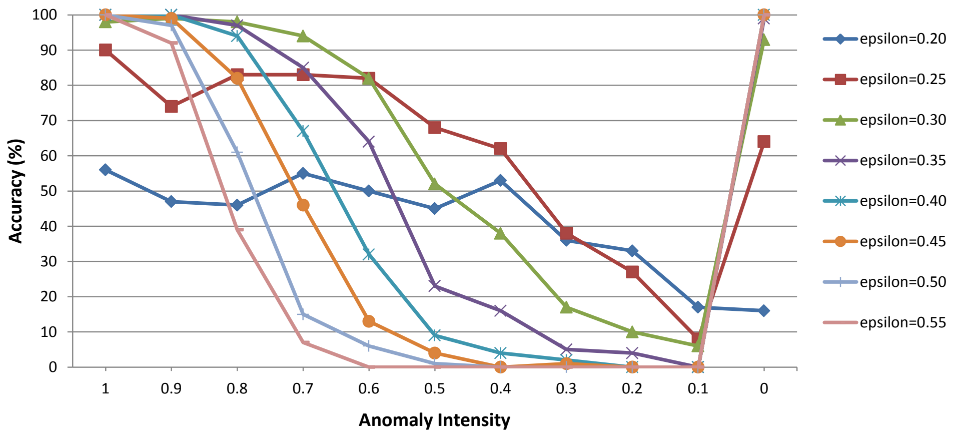

Figure 6. Logistics domain: Accuracy as a function of anomaly intensity.

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 0.1 higher epsilon values in the blocksworld domain. Figure 8 illustrates this shift in performance graphically.

5.3 Discussion

Generally we have shown that our approach works well across multiple domains and with anomalies of varying intensity. However these data are limited in a number of ways. Anomalies

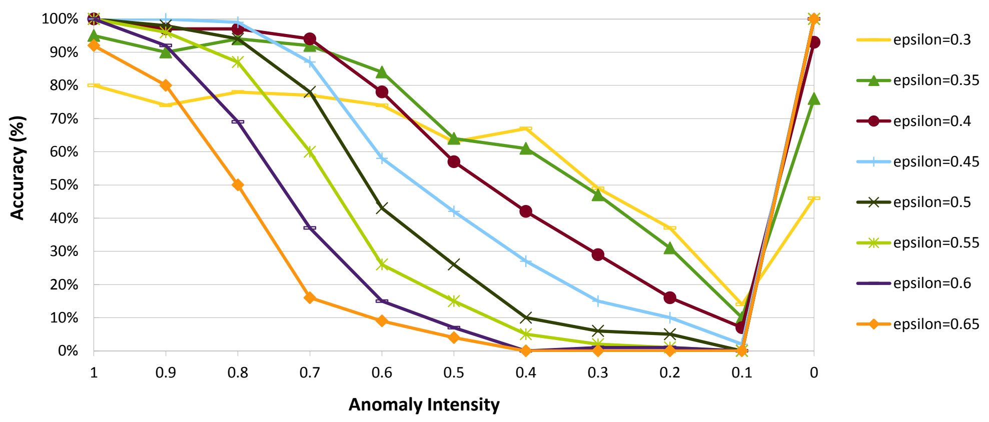

Figure 7. Blocksworld domain: Accuracy as a function of anomaly intensity.

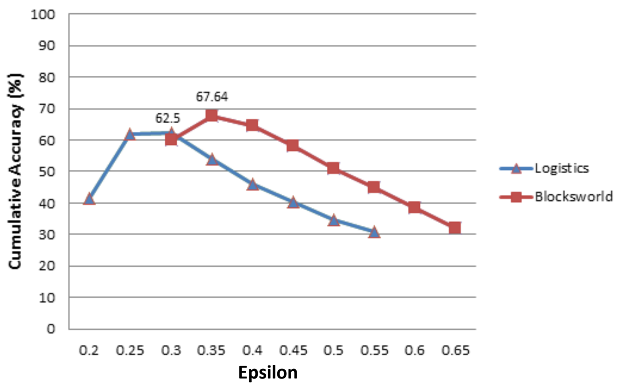

Figure 8. Average overall accuracy as a function of threshold (epsilon) value.

occur in many different contexts, and we have examined only one class. Removing an operator from a domain theory causes a change in the actions executed and hence the states observed. The effective change is all or nothing. At the current time, we have not examined the sensitivity of this approach to either the deletion or addition of multiple operators or to selective interactions that might occur when certain actions occur together. We believe that our approach is relevant to these types of anomalies, but future research remains to determine its actual performance.

Also we have shown that the A-distance metric used with our state representations will detect the rough onset of an anomaly, but it would be more informative to use this method to detect both the beginning and the end of a target sequence. Indeed, if we consider all vectors past the stable area to be independent observations throughout the input stream, we could calculate more details such as recall and precision for the entire data set. We are currently performing such tests.

One obvious problem with our method is that we drop all predicate arguments when translating first-order representations to vectors. Change may be signaled by differences in the argument values rather than by the predicate values, so the A-distance will not detect such an anomaly. For example planes may not be unloaded at a particular airport rather than at all airports, so plan sequences may simply differ by parameters and thus the pattern of intermediate abstract states will not change. To detect this directly, a system would have to monitor all concrete states at a computationally prohibitive expense. We suggest as an alternative at the K-track, to monitor just those states that directly bear on the success of the goals. These states are to be found in the goal-subgoal graph of the planning process that generated the action sequences. In past research, we identified these states as candidates for rationale-based planning monitors (Veloso, Pollack, & Cox, 1998). The problem is that as an observer of other agent planning behavior, the system will not have direct access to the graph and must infer it.

6. Anomaly Detection: Related Research

A large body of work exists on the subject of anomaly detection in data streams (see Chandola, Banerjee, & Kumar, 2009). In general, this work faces challenges posed by the huge volume of data present in many domains. Because of the size of many data sets, 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, Omiecinsky, Mark, & Nguyen, 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 a version of another approach to the problem of big data, the sliding window model. Many versions on this class of algorithm have been studied using a variety of statistical tools. In addition, several interesting innovations have been developed for customizing the parameters of the algorithm. One such 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 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 or using rules.

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, Kitamura, Sakuma, & Yokoyama, 1994). While statistical time series anomaly detection has been studied extensively, it has generally been used to analyze numeric data streams. First-order, symbolic domains such as blocksworld and logistics have received no 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 (Klenk, Molineaux, & Aha, in press).

7. Conclusion

The results presented above suggest that methods for finding anomalies in streaming real-valued data can be applied to finding anomalies in declaratively represented symbolic domains. We intend to explore the full range of activities exhibited by MIDCA-based agents, including learning and metacognition, to see if they are amenable to this approach as well. For example, changes in the distribution of rewards over time may suggest that the policy controlling behavior needs to be adapted. Note that because rewards are governed by the action policy, the fact that rewards are changing points to the policy as the place to focus attention. The reason for the change in rewards could be a change in the domain (the hardwood floors traversed by a Roomba may have been waxed), or a change in the physical agent (the Roomba's wheel bearings broke). In either case, though, adapting the policy by restarting a reinforcement learning algorithm can address the problem. In symbolic planning domains, detected anomalies point to specific predicates.

For the assess phase of the NAG procedure, we will explore ways of mapping from these predicates to planning operators by reasoning about which operators are either most dependent on (pre-conditions) the predicates or most likely to affect (post-conditions) the predicates. For example during the empirical evaluation above, the predicate most often implicated in the detection of change in the logistics domain 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. 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.

A second general direction is to develop a multi-variate version of the A-distance note phase. Rather than treating each element of the collapsed state vector as independent, capturing dependencies across multiple streams of data simultaneously can find structure that would otherwise be invisible. One possibility is to move from intervals over the real line to hyper-cubes in multi-dimensional spaces. The difficulty is that the computational complexity of this approach is exponential in the dimensionality of the data. An intermediate approach that is often successful is to look at all 2- or 3-combinations of variables (here predicates) and look for anomalies in them. This works because true high-dimensional interdependencies are rare, and even when they do exist, they can be seen in lower dimensional projections.

A third avenue to explore is that of integrating data-driven anomaly detection with top-down knowledge-based approaches. Although in its early stages of development, the MIDCA architecture envisions both a knowledge-rich K-track and a data-driven D-track for each major process in the cognitive and metacognitive cycles of the architecture. As mentioned in Section 2, the note phase is embedded in the interpretation processes of the perception-action cycle at the cognitive level of the architecture and of the monitoring-control cycle at the metacognitive level. But the detailed interaction between the K-track and the D-track is unclear at both levels. We can imagine for instance the A-distance metric detecting a course-grain window of change and a more knowledge-based process re-examining the region for those expectations that would have been violated if active. In any case, this integration remains to be developed, although many interesting research issues exist concerning the exact nature of low-level and high-level processing involved in the NAG procedure.

This paper has developed a novel approach to detecting anomalies and has spent most of its effort in a narrow examination of the computational details concerning the method and its efficacy. We described a number of examples of change in the environment, and we tested the A-distance approach against such changes. But our technique was restricted to the interpretation of percepts at the cognitive level (i.e., the perception-action cycle) only. The same approach should be equally effective at the metacognitive level (D-track) for detecting anomalies within cognitive states of the agent itself. If MIDCA represents mental states and processes in a predicate representation similar to world states and action operators, then the A-distance could be applied to a similar intermediate abstract-state vector representation of its own planning and interpretations.

This paper presents a very small piece of a larger effort currently underway to implement a new cognitive architecture that provides a computational accounting of the interaction between cognitive and metacognitive activities for intelligent agents. We have described an efficient, data-driven technique for detecting anomalies in symbolic representations of time-dependent events using a metric called the A-distance. This technique is used in the note phase of a larger Note-Assess-Guide procedure embedded in the interpretation process of the MIDCA architecture. Results demonstrate the effectiveness of this technique across anomalies of varying intensity in two, distinct, planning domains. Although this research is limited in its current scope, it represents a significant step toward a better understanding of the role of metacognition and related cognitive processes in complex, problem-solving behavior.

Acknowledgements

The authors thank the anonymous reviewers for their insights. This material is based upon work supported by the Office of Naval Research under Grants No. N00014-12-1-0430 and No. N00014-12-1-0172 and by the Army Research Office under Grant No. 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), 521-538.

Ahmed, E., Clark, A., & Mohay, G. (2008). A novel sliding window based change detection algorithm for asymmetric traffic. *Proceedings IFIP International Conference on Network and Parallel Computing* (pp. 168-175). New York: 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*, 1075–1093.

Anderson, M., Chong, W., Oates, T., & 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*, 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*, 21-40.

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. *Proceedings of Forty-First Annual IEEE Symposium on Foundations of Computer Science* (pp. 259-269). New York: 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. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 318-323). Menlo Park, CA: AAAI Press.

Brachman, R. (2002). Systems that know what they're doing. *IEEE Intelligent Systems, 17*, 67-71.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*, Article No. 15.

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

Cox, M. T., & Burstein, M. H. (2008). Case-based explanations and the integrated learning of demonstrations. *Künstliche Intelligenz, 22*, 35-38.

Cox, M. T., Edwin, G., Balasubramanian, K., & Elahi, M. (2001). Multiagent goal transformation and mixed-initiative planning using Prodigy/Agent. *Proceedings of the Fifth 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*, 96-104.

Cox, M. T., Oates, T., & Perlis, D. (2011). Toward an integrated metacognitive architecture. (Technical Report FS-11-01) In P. Langley (Ed.), *Advances in Cognitive Systems, Papers from the 2011 AAAI Symposium* (pp. 74-81). Menlo Park, CA: AAAI Press.

Cox, M. T., & Ram, A. (1994). Failure-driven learning as input bias. In A. Ram & K. Eiselt (Eds.), *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.

Cox, M. T., & Veloso, M. M. (1998). Goal transformations in continuous planning. *Proceedings of the 1998 AAAI Fall Symposium on Distributed Continual Planning* (pp. 23-30). Menlo Park, CA: AAAI Press.

Cox, M. T., & Zhang, C. (2007). Mixed-initiative goal manipulation. *AI Magazine 28*(2), 62-73.

Crook, P. & Hayes, G. (2001). A robot implementation of a biologically inspired method for novelty detection. *Proceedings Towards Intelligent Mobile Robots Conference*.

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

Dredze, M., Oates, T., & Piatko, C. (2010). We're not in Kansas anymore: Detecting domain changes in streams. *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. *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining* (pp. 53-62). New York: ACM.

Fox, S., & Leake, D. (1995). Modeling case-based planning for repairing reasoning failures. *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. *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. *Data Warehousing and Knowledge Discovery: Proceedings of the Eleventh International Conference* (pp. 75-86). Berlin: Springer.

Janssens, J., Postma, E., & Hellemons, J. (Eds.) (2011). Maritime anomaly detection using stochastic outlier selection, *Proceedings of the International Workshop on Maritime Anomaly Detection* (pp. 21-23). Tilburg University, Tilburg, The Netherlands.

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

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

Klenk, M., Molineaux, M., & Aha, D.W. (in press). Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence*.

Kolodner, J. L. (1987). Capitalizing on failure through case-based inference. *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. *Proceedings IEEE World Congress on Computational Intelligence, 1994 IEEE International Conference on Neural Networks* (pp. 3207-3210). New York: 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., & Schonfeld, D. (2003). Statistical sequential analysis for real-time video scene change detection on compressed multimedia bitstream. *IEEE Transactions on Multimedia*, *5*, 106-117.

Lewis, M. (1998). Designing for human–agent interaction. *AI Magazine, 97*(2), 67–78.

Norman, D. (1986). Cognitive engineering. In D. Norman & S. Draper (Eds.), *User-centered system design: New perspectives on human-computer interaction* (pp. 31-61.). Hillsdale, NJ: Lawrence Erlbaum.

Pannell, G., & Ashman, H. (2010). Anomaly detection over user profiles for intrusion detection. *Proceedings of the Eighth Australian Information Security Management Conference* (pp. 81-94). Perth, Australia.

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.

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.

Sun, R., Zhang, X., & Mathews, R. (2006). Modeling meta-cognition in a cognitive architecture. *Cognitive Systems Research, 7*, 327-338.

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

Talamadupulay, K., Schermerhorn, P., Bentony, J., Kambhampati, S., & Scheutz, M. (2011). Planning for agents with changing goals. *Twenty-First International Conference on Automated Planning and Scheduling: Proceedings of the System Demonstrations* (pp. 71-74). Menlo Park, CA: AAAI Press.

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

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*, 81-120.

Veloso, M. M., Pollack, M. E., & Cox, M. T. (1998). Rationale-based monitoring for continuous planning in dynamic environments. *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems* (pp. 171-179). Menlo Park, CA: AAAI Press.