Symbolic Anomaly Detection and Assessment Using Growing Neural Gas

Matthew Paisner and Don Perlis
Department of Computer Science
University of Maryland
College Park, USA
{paisner, perlis}@cs.umd.edu

Michael T. Cox
Institute for Advanced Computer Studies
University of Maryland
College Park, USA
mcox@cs.umd.edu

Abstract— **Metacognitive architectures provide one solution to the brittleness problem for agents operating in complex, changing environments. The Metacognitive Loop, in which a system notes an anomaly, assesses the problem and guides a solution, is one form of such an architecture. This paper extends prior work on implementing the note phase of this process in symbolic planning domains using the A-distance. This extension uses a growing neural gas algorithm to construct a network which represents various normal and anomalous states. Testing shows that this technique allows for improved detection of anomalies in the note phase as well as categorization of anomalies by severity and type in the assess phase.**

Keywords: Metacognitive loop, anomaly detection, diagnosis, comprehension, neural networks.

I. Introduction And Related Work

Intelligent agents guided by machine learning algorithms are often very successful at well-defined, consistent tasks, even when those tasks are very complex. However, for autonomous agents to be effective in many real-world settings, they must be able to deal with situations that are more fluid. Thus a path-finding program might learn to navigate mazes better than a person ever could, but if it tries to apply that knowledge, unmodified, to the problem of climbing a wall, it will fail. Of course, what the agent ought to do is to temporarily forgo its maze navigation mastery and either start learning from scratch or apply some alternative algorithm that is more suited to its new task.

Anderson and Perlis [1] describe a metacognitive approach to this type of problem, in which an agent continuously monitors both the world and its own internal processes in order to *note* problems, *assess* their type and severity, and *guide* an appropriate response strategy. They call this process the metacognitive loop (MCL). Several iterations of MCL have been developed, including domain-specific versions that guide agents to improved performance in Q-learning [2] and natural language dialog [3] systems. A domain-independent version of MCL based on a Bayesian net was also developed [4] and used to guide a simulated mars rover and for other tasks.

Recently, we have started to incorporate the MCL approach into an integrated cognitive architecture called MIDCA that models action and perception along with cognition and metacognition [5, 6]. We have done so using deterministic, symbolic, planning domains like blocksworld (in which agents arrange blocks in stacks) and logistics (a package delivery scenario). Because they are defined by statements in a predicate logic, such domains are very amenable to logic- and explanation-based approaches to behavior and hence the note, assess, guide procedure. Noting anomalies in symbolic domains usually requires the explicit representation of expectations such as the expected effects of an action (e.g., see [7]). But surprising or otherwise non-explicit change is more difficult to detect despite extensive research in anomaly detection for numeric domains (e.g. [8, 9, 10, 11, 12, 13]).[1] For symbolic domains, such anomaly detection would require the adaptation of statistical techniques that were designed for real-valued, time series data.

One such technique, the A-distance [15], uses a statistical comparison between a baseline window of "normal" data and a sliding window of recent observations to detect shifts in the underlying distribution. Researchers have successfully applied this approach to domain adaptation work in natural language tasks [16, 17, 18]. More recently, we have tested an A-distance implementation of the MCL note phase within the MIDCA architecture [19, 20]. This approach was successful as an anomaly detection procedure but left some issues unexamined. First, A-distance requires the choice of a threshold value to determine how different a state must be from the baseline to be called an anomaly. One approach (used in [19]) is to try several different values. However, for real-time anomaly detection, this may not always be practical. Second, while the adaptation of A-distance to symbolic domains during the note phase is novel, the explanation of anomalies in the assess phase is currently simplistic, making no attempt to diagnose it in terms of severity or type.

This paper aims to begin the work of addressing these issues. The following sections will describe a method of using A-distance values as inputs to a growing neural gas (GNG) network [21]. This results in a network in which certain nodes correspond to anomalous data thereby providing two benefits. First, because the network grows or shrinks to fill the input space representing a given set of baseline data, it can self-calibrate to recognize anomalies of various types without the necessity for a user-selected epsilon value. Second, the nodes of the network do not simply present a Boolean "anomalous" or "normal" result

[1] A large body of work on anomaly detection in data streams also exists [14].

but can be seen as prototypes with each node representing an anomaly of a certain type and intensity. This information can then be used to help determine the most appropriate choice of strategies.

The next section will explain the outputs generated by applying A-distance to a predicate-logic state representation and how those outputs are used as inputs to a GNG network. Subsequently, we will detail the experiments performed and present data on both the success of the GNG/A-distance combination in anomaly detection and in anomaly assessment. Finally, we conclude and discuss future research, including the ways in which we anticipate using the results of this work in a complete version of MCL.

II. Methods

A. *A-distance in planning domains*

We conducted a set of experiments using a simple logistics world [22], a symbolic planning domain in which airplanes and trucks are tasked to deliver packages. Logistics is a deterministic world in which states are defined by sets of assertions in predicate logic. As an example, consider state shown in Fig. 1. This simple configuration consists of two trucks, two planes and two packages each located in one of two cities. Each city has both a post office and an airport.

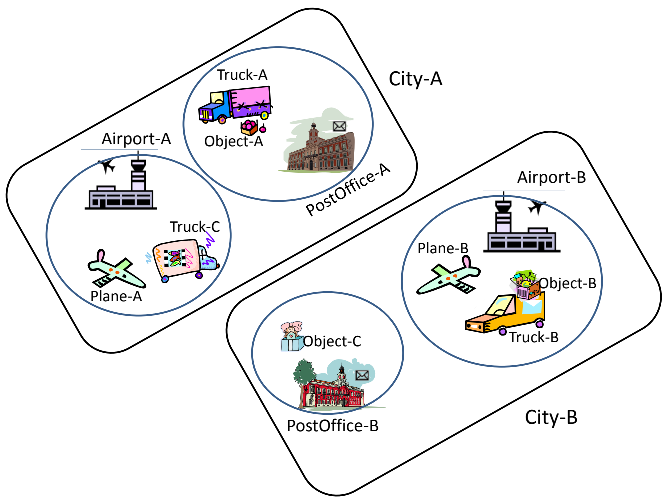

Figure 1: Pictorial representation of a state in the logistics domain

Fig. 2 shows a simplified predicate representation of the state in Fig. 1. Static predicates representing the locations of airports and post offices and their relations have been omitted.

At-Truck (TruckC, Airport_A)
At-Truck (TruckB, Airport_B)
At-Plane (PlaneA, Airport_A)
At-Plane (PlaneB, Airport_B)
At-Package (ObjectC, PostOffice_A)
Inside-Truck (ObjectA, TruckA)
Inside-Truck (ObjectB, TruckB)

Figure 2: Predicate representation of a state in the logistics domain

From such a state, a classical planner can generate a plan to achieve a given list of goals. Such plans constitute a stream of observations. Previously we introduced a method for applying A-distance to series of these plans in order to detect when changes occur in the observation stream [19] (for example, when the ability to unload airplanes has been removed). The full details of that method will not be revisited here, but it is important to the present paper to describe the output of the procedure.

First, we note that A-distance makes use of two "windows" of data, each of size *n*, to detect anomalies. The first window consists of the first *n* observations, which are assumed to come from a baseline distribution. The second window slides along the data stream and always contains the n most recent data points. The value of A-distance at time *t* is a function of the difference between the distributions of data in the two windows, [0, *n*) and (*t*-*n*, *t*]. The output of running A-distance on a data stream, then, is another data stream, with the value at step *t* indicating how anomalous were the *n* data points up to and including *t* in the initial stream.[2] In other words, if *I* is an input data stream, and *A* is the resulting A-distance stream, then *A*[*t*] measures the "anomalousness" of the range *I*[*t*-*n*…*t*].

We use multiple data streams of predicate counts to represent a series of plans [19]. For example, the state in Fig. 2 contains three At-truck predicates. If that state was the start state in a plan, then the first value for the At-truck predicate stream would be the number three, the second value would be the number of At-truck predicates in the state after one step, and so on. We generate one of these streams for each predicate in the source domain, and run A-distance separately on each stream. The output from running A-distance on these predicate count streams is a series of output streams, with the stream corresponding to a given predicate reflecting the degree to which that predicate's usage is anomalous at each time step. An anomaly is reported at time t when the value of *any* of the output streams at t is greater than a threshold parameter *epsilon*. Fig. 3 shows an example of output streams for three predicates.

[0.13	**0.24**	$\underline{\mathbf{0.30}}$	$\underline{\mathbf{0.36}}$	$\underline{\mathbf{0.36}}$**]**	**(At-Package)**
[0.16	**0.20**	**0.20**	**0.20**	**0.18]**	**(Inside-Plane)**
[0.13	**0.21**	**0.26**	$\underline{\mathbf{0.32}}$	$\underline{\mathbf{0.34}}$**]**	**(Inside-Truck)**

Figure 3: A-distance streams for three predicates

The underlined values are those that would be considered anomalous at an epsilon value of 0.3. The last three time steps in the sequence shown would then be considered part of an anomaly. In this example, it seems likely that the anomaly has to do with packages being transferred to and from trucks in unusual numbers, since the At-Package predicate (which reflects packages being located at airports or post offices) and the Inside-Truck predicate are the ones that generate the anomaly.

Sets of predicate-associated output vectors like those in Fig. 3 will be the inputs to the algorithm described below.

[2] For t < n, A-distance is defined to be 0 since the baseline window is not yet full.

B. *Growing Neural Gas*

The algorithm we apply to the A-distance output vectors is a modified version of a growing neural gas network [21]. A basic neural gas network [23] consists of only two layers, input and output. The algorithm uses one-shot single-winner competitive learning, in which the winning output node is picked based on Euclidean distance from the input vector. Equation (1) shows the learning rule for the winner.

$$dw = n(a - w) \tag{1}$$

Here a is the input vector, n is the learning rate and w is the node's weight vector. Neighbors of the winning node also learn, but with a smaller value of n. The output of the network also reflects a single-winner dynamic, with only that node turned on.

Growing neural gas (GNG) adds the behavior of allowing output nodes to be created and eliminated to better fit the structure of the input data. In Fritzke's [21] classic model, an error value is stored for each node, based on the total squared distance from that node to all inputs for which it was the winner. New nodes are then added near the nodes with the highest error rates at fixed time intervals. Each new node has an edge connecting to the high error node near to which it was placed.

This approach is intended to induce higher sensitivity in areas where a greater input density exists. However, it is less effective at accounting for small anomalous data clusters that are disconnected from the larger mass of normal data, especially when those clusters are present only during a short interval and then cease. One reason why this pattern is less easily captured is that anomalous data will occur much less frequently, so that it is difficult for a node to move all the way out to the location of the actual cluster without being pulled back towards the center by the more frequent input patterns. Also, even if a node does reach the correct location, if the outlying input patterns stop being expressed it will drift away due to the pull of its neighbors, and the network will "forget" about the anomaly rather quickly.

The solution to this issue employed in the present experiments is two-fold. First, the method of adding nodes has been changed. For the purposes of anomaly detection, it is actually counterproductive to add more nodes in a small area of high concentration, as the old algorithm tends to do. Therefore, the version of GNG used herein adds nodes based on distance – when an input pattern is near no currently existing node, a new node is created at the point specified by the input. This allows brief or unusually anomalous events to be detected. Second, the neighbor learning rate has been reduced to zero, preventing the pull of outlying nodes back to the center. This prevents GNG from effectively performing its function as a topological map but makes it more useful as a flexible clustering method. Table 1 shows the modified GNG update algorithm. Note that this algorithm does not track error values, nor does it add nodes at any specific time interval. Additionally, the step of updating neighbors of the selected node is omitted.

Table 1. GNG update algorithm

```
Function GNG_update(inputV, maxDistance, learnRate):
    closestNode = get_closest_to(inputV)
    if distance(closestNode, inputV) > maxDistance:
        create_node_at(inputV)
    else:
        move_towards(closestNode, inputV, learnRate)
    for node in nodes:
        if node.time_since_nearby_update > maxAge:
            delete(node)
```

C. *A-distance as input to growing neural gas*

As described above, the note phase produces a series of A-distance vectors as output with each vector reflecting the abnormality of the distribution of a single predicate at each time step. The input data to GNG is the series of vectors created by, at each time step, concatenating the values from all input streams. So, for a system with k predicate streams, observed at T time steps, the input to GNG would be T vectors of length k, with each vector corresponding to a single time step t and reflecting the A-distance values for all predicates at t. GNG's input space is then a k-dimensional space in which each dimension represents the relative contribution to an anomaly for a particular predicate.

Over time, the GNG network develops a set of nodes, which correspond to prototypes of normal and anomalous world states. Since the input data reflect a measurement of abnormality, nodes with larger values can be seen as more anomalous. The degree to which each prototype is anomalous, then, can be crudely determined by the distance from the node to the origin. We call this value the *severity* of the anomaly. Additionally, the anomaly *type* of any input state can be determined by referring to the network's activated output node and the dimensions along which the node exists.

To provide a Boolean anomaly detection result, this algorithm is run on a set of baseline data. The requirement for such a guaranteed set of normal data is not an issue, because A-distance already makes this assumption for the calibration of the baseline window. Fig. 4 shows an example of a small baseline network.

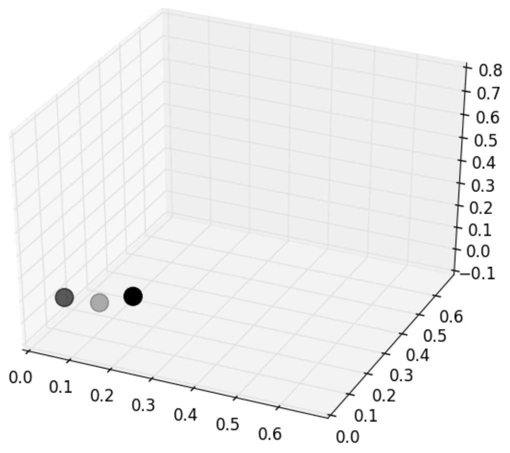

Figure 4: A baseline GNG net (no anomalous data). Darker nodes are closer to the viewer. Note that nodes are clustered close to the origin, reflecting low A-distance values for all predicates.

Once a network is generated using this data, the anomaly threshold distance D_A from the origin is defined as the maximum distance of any qualifying node in the baseline network.[3] Once D_A is calculated, the algorithm is run normally on the test data to generate a new network. If the distance from the activated node for an input to the origin is greater than D_A, that input is considered anomalous. An example of a network generated from both normal and anomalous data is shown in Fig.5. The shaded region is a sphere centered at the origin with radius D_A and represents the area in which nodes are considered normal. Note that the shaded region is the smallest sphere centered at the origin that would contain all nodes shown in Fig. 4.

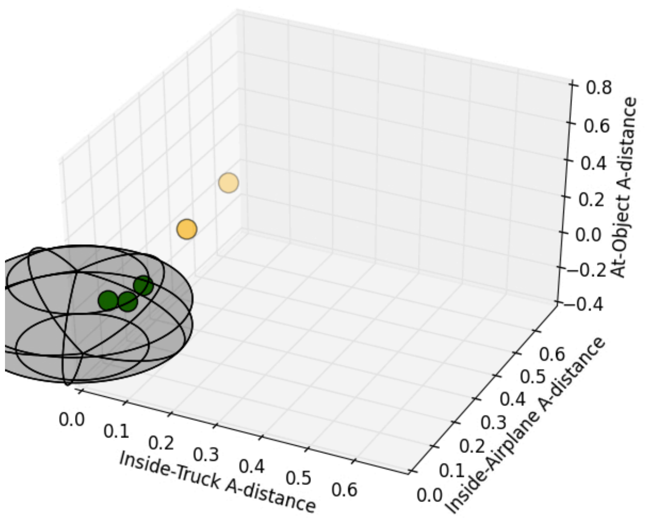

Figure 5: GNG net with anomalous data. The shaded area represents the normal region derived from baseline data. Nodes are colored to show classification as normal or anomalous.

[3] Nodes must have been updated a minimum number of times to qualify. We used 3 for this number; its purpose is to avoid extremely obvious outliers, such as nodes that have been created and then never activated again.

Using this method, a time step is classified as anomalous if the node that is activated by the vector of A-distance output at that step is outside the normal region. Because a GNG network naturally expands to fill its input space, the size of this region, and therefore the sensitivity of anomaly detection, will reflect the variability of normal data in the domain. Data on the effectiveness of this method will be presented in Section IV.

III. Experimental Setup

All experimental data were drawn from randomly generated world states and goals in the logistics domain described above. To create anomalous data, planning was done with one operator removed. The data included only those world state/goal combinations for which a plan was successfully generated.

We used three data sets for testing. The first, the "airplane anomaly" set, consisted of 500 normal plans and 100 plans with the unload-airplane operator removed. The second, designated "truck anomaly", also had 500 normal plans and 100 with the unload-truck operator removed. The third, "two anomaly", had 500 normal plans and two separate anomalous sections of 100 plans, the first of which had the unload-airplane operator removed, and the second unload-truck.

We conducted trials using variable concentrations of anomalous plans in the anomalous intervals. The varying concentrations represent anomalies of different intensities. For example at intensity 0.2, only 20% of the 100 plans in each anomalous section were actually anomalous, thus signifying a very faint anomaly. For comparison, we ran trials on each data set multiple times. Some experiments used A-distance with several epsilon values but no GNG network. Other trials used the combined A-distance/GNG method. We ran fifty trials for each pair of (data set, intensity level), where intensities ranged from 0 to 1 by units of 0.1.

IV. Results

For each method, including the A-distance/GNG combination as well as A-distance alone using several epsilon values, reported anomalies at each step were recorded and expected results calculated. The expected outcome at a time step was 'anomalous' if at least P% of the values within the sliding window came from anomalous plans, i.e., those that were generated with an operator removed. We calculated F1 (a weighted average of precision and recall) for each method and at several values of P from 1 to 50 (to clarify, P=50 means that an anomaly is defined as a step at which at least half of the data in the sliding window was generated by planning with an operator removed). Fig. 6 shows the average F1 values by test setup for each of the three data sets.

On each data set, the GNG network (the rightmost column) performed slightly worse than the best epsilon value. However, to select the best epsilon value required testing using a priori knowledge of which data came from anomalous sources; whereas GNG was initialized using only a set of baseline data.

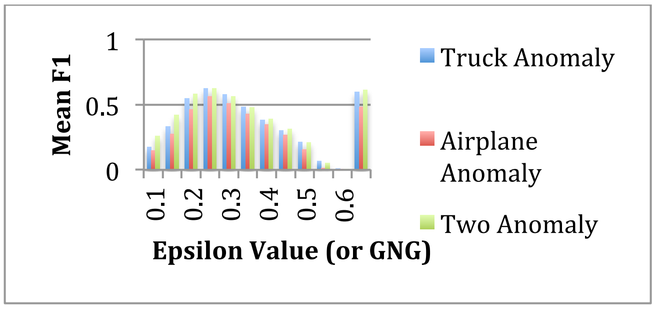

Figure 6: F1 values from testing using different epsilon values and GNG. Results were averaged across all P values.

These results therefore suggest that while the addition of processing with GNG may not be useful given a domain and anomaly set whose characteristics are well known, it does come very close to optimizing the performance of A-distance without requiring any prior information on the type of anomalies that will occur. In the context of MCL and MIDCA, which are intended to be employed in a wide spectrum of domains that may change unpredictably, this is a valuable feature.

Besides improving the flexibility of anomaly detection, this method provides useful information as the MCL cycle moves into the assessment phase. GNG's contribution to this goal is to provide groupings of anomalies by type and severity so that a newly noted anomaly can be quickly categorized, allowing a more robust diagnosis procedure to narrow down its list of options.

Fig. 7 graphs assignments of world states to nodes during the anomalous segments of the two-anomaly data set at intensity 1. The x-axis counts steps from the sliding window's entry into the anomalous section. The y-axis reflects which output node was selected at a given time step. The nodes are sorted in terms of severity from least anomalous (1) to most anomalous (8).

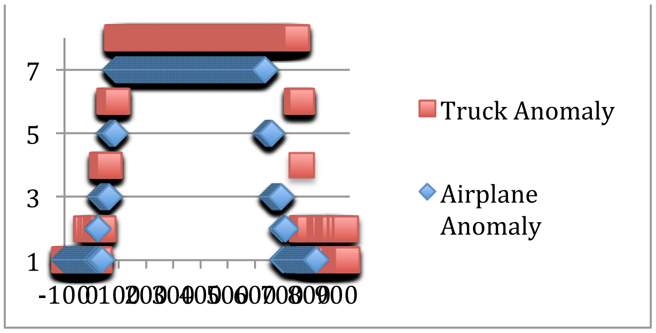

Figure 7: Activated node from GNG network for each time step from 100 before to 100 after each anomaly, for the two-anomaly data set.

This graph contains two significant features. First, other than at the very edges of each anomaly (when most of the data in the sliding window is normal), the two anomaly types map to two non-intersecting sets of nodes: the truck anomaly maps to nodes 4, 6 and 8, while the airplane anomaly maps to 3, 5 and 7. So, the GNG network is able to differentiate between anomalies of different types. Second, in both cases there is a clear progression as a larger percentage of the window slides into the anomalous region. At first GNG maps to nodes that are slightly anomalous (3, 4), then nodes that are moderately anomalous (5, 6) and finally those that are fully anomalous (7, 8) as the window's back edge enters the anomaly. This pattern is reversed as the window slides out of the anomalous section, showing that the network has generated anomaly prototypes which reflect intensity as well as type.

Fig. 8 shows the network containing these nodes. Normal nodes 1 and 2 are contained in the shaded region, while the nodes mapped to truck and airplane anomalies arc out along the axes corresponding to those A-distance values. The three red dots have much higher values along the Inside-Truck axis and only slightly larger than normal one along the other two axes, while the blue dots' values increase primarily in the direction of the Inside-Airplane axis.

V. Discussion

Both A-distance and Growing Neural Gas are established techniques. The use of GNG to perform online clustering of A-distance data, however, opens up new possibilities for an agent that is attempting to adapt to a changing world. The first of these is the ability to dispense with user-selected epsilon values. Because a GNG network grows to fill its input space, it can organically generate a threshold that provides results comparable with the best possible epsilon choice.

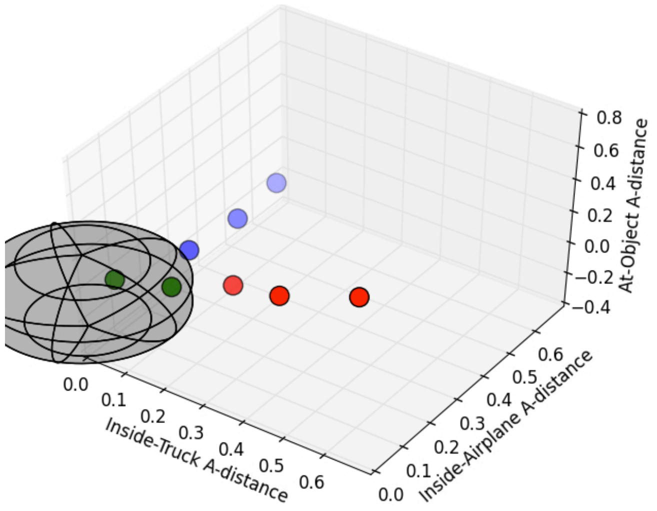

Figure 8: GNG net with two different anomalous data sets. The shaded area represents the normal region derived from baseline data (see end of section III for details). Nodes are colored to reflect classification by anomaly status: red nodes were activated during the truck anomaly, blue nodes during the airplane anomaly.

Secondly, the network generated by GNG provides a unique perspective on anomaly analysis. A simpler approach might have been to run GNG, or another online clustering algorithm, on raw state data. By using A-distance data instead, we create a network whose nodes represent not similar states but similar anomalies. Further, because the input data have a consistent meaning – a larger coordinate value corresponds to a larger disparity from normal for the corresponding predicate – we have actually gained more information than simply a set of clusters. If a selected node is just above the anomaly threshold, we can surmise that it might simply be an outlier or a weak anomaly. If the selected node has a very high value for the "inside-airplane" coordinate and a very low one for "inside truck," we know that something strange is happening with the airplane fleet, but we probably do not need to spend much time worrying about our trucks.

Because the approach of MCL is to fix problems by first deducing what has gone wrong, the added semantics of a net constructed from A-distance predicate data are important. When presented with a weak anomaly, MCL might want to wait for confirmation, or pursue a conservative strategy. If the anomaly is caused by changes to the plane-related predicates, it should keep trucks running as they have been while employing reasoning or learning techniques to reevaluate airplane control strategies. The ability to use data-driven techniques like GNG and A-distance to generate semantic knowledge about the type of failure that is being experienced gives MCL a significant head start in more conventional approaches to reasoning and a clearer target for directed learning.

VI. Future Work

The simplest way to build on the work described in this paper would be to try to improve the clustering process. We have made several changes to GNG to adapt it to use in MCL, but it might be beneficial to incorporate other clustering algorithms and variations, for example as in [24].

The use of GNG with A-distance improves upon previous research with the MCL note phase. But its primary purpose is to expand those results into a useful tool for the assess phase in a push towards a working implementation of MCL for symbolic domains. Aside from optimization, then, the next step will be to move to the final stage of MCL and use the information from GNG prototypes to direct a strategy change. In the logistics domain, a simple example might feature an agent that has two choices of planning algorithms: one that emphasizes ground transport and one air transport. When an anomaly occurs, it could decide whether to switch algorithms on the basis of anomaly type and severity as indicated by the active GNG node.

Another avenue of research will focus on combining the data-driven techniques discussed herein with a knowledge-rich approach to assessment. A significant appeal of logically formulated domains like logistics is that they are by nature amenable to techniques of case-based reasoning (e.g., [25, 26]). MCL should be able to utilize this type of process alongside and with the aid of techniques like the GNG/A-distance algorithm. Once this area has been explored and combined with present work, we hope to have a version of MCL that can autonomously aid an agent in dealing with adversity across a wide range of domains and problem types.

VII. Conclusion

This research stands in a broad context that seeks to examine the mechanisms and the means under which cognitive systems make intelligent decisions and act independently over long periods of time and in situations of change and complexity [27, 28]. Our contention is that robust behavior in the face of surprise is a function of many aspects of intelligence including action and perception, and cognition and metacognition. We are in the early stages of developing an integrated cognitive architecture that specifies these components. The cognitive mechanisms include both problem-solving and comprehension and involve significant amounts of learning. This paper has examined the first two phases in the note-assess-guide procedure which supports the interpretive mechanisms of comprehension and failure recognition. Although we have concentrated upon the data-driven features in these phases, we are working towards the interaction of data-driven algorithms with knowledge-rich methods (see [5]). Current progress and empirical results as reported here lead us to expect interesting research ahead.

Acknowledgment

We thank ONR and ARO for their support through ONR Grants # N00014-12-1-0430 and # N00014-12-1-0172 and ARO Grant # W911NF-12-1-0471. We also thank Josh Brulé, Michael Maynord and our anonymous reviewers for their helpful comments and feedback.

References

[1] M. L. Anderson and D. Perlis . "Logic, self-awareness and self-improvement: The metacognitive loop and the problem of brittleness," Journal of Logic and Computation, 15.1, 2005.

[2] M. L. Anderson, T. Oates, W. Chong, and D. Perlis, "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, pp. 387-411, 2006.

[3] D. Josyula, S. Fults, M. L. Anderson, S. Wilson, and D. Perlis, "Application of MCL in a dialog agent," in Third Language and Technology Conference, 2007.

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

[5] M. T. Cox, M. Maynord, M. Paisner, D. Perlis, and T. Oates, "The integration of cognitive and metacognitive processes with data-driven and knowledge-rich structures," Proc. Annual Meeting of the International Association for Computing and Philosophy, IACAP-2013, 2013.

[6] M. Maynord, M. T. Cox, M. Paisner, and D. Perlis, "Data-driven goal generation for integrated cognitive systems," To appear in C. Lebiere & P. S. Rosenbloom (Eds.), Integrated Cognition: Papers from the 2013 Fall Symposium. Menlo Park, CA: AAAI Press, , in press.

[7] M. Klenk, M. Molineaux, and D. Aha, "Goal-driven autonomy for responding to unexpected events in strategy simulations," Computational Intelligence, 29.2, pp. 187–206, 2013.

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

[9] M. Basseville and I. V. Nikiforov, Detection of abrupt changes - theory and application. Englewood Cliffs, NJ, Prentice-Hall, 1993.

[10] P. Crook and G. Hayes, "A robot implementation of a biologically inspired method for novelty detection," in Proc. Towards Intelligent Mobile Robots Conference, Edinburgh, Scotland, Division of Informatics, University of Edinburgh, 2001.

[11] T. Fawcett and F. Provost, "Activity monitoring: noticing interesting changes in behavior," in Proc. 5th International Conference on Knowledge Discovery and Data Mining (SIGKDD99), pp. 53-62, New York: Association for Computing Machinery, 1999.

[12] J. Janssens, E. Postma, J. Hellemons (Eds.), Proc. International Workshop on Maritime Anomaly Detection 2011, Tilburg, The Netherlands: Tilburg Center for Cognition and Communication, Tilburg University, 2011.

[13] G. Pannell and H. Ashman, "Anomaly detection over user profiles for intrusion detection," in Proc. 8th Australian Information Security Management Conference, Perth, Western Australia: Edith Cowan University, 2010.

[14] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys 41(3), 2009

[15] D. Kifer, S. Ben David, and J. Gehrke, "Detecting change in data streams," In Proc. Thirtieth Very Large Databases Conference, pp. 80-191, 2004.

[16] J. Blitzer, R. McDonald, and F. Pereira, "Domain adaptation with structural correspondence learning," Proc. 2006 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2006.

[17] J. Blitzer, M. Dredze, and F. Pereira, "Biographies, Bollywood, boom-boxes and blenders: domain adaptation for sentiment classification," in Association for Computational Linguistics Vol. 7, pp. 440-447, 2007.

[18] M. Dredze, T. Oates, and C. Piatko, "We're not in Kansas anymore: detecting domain changes in streams," in Proc. EMNLP '10, pp. 585-595, 2010.

[19] M.T. Cox, T. Oates, M. Paisner, D. Perlis. "Noting anomalies in streams of symbolic predicates using A-distance," Advances in Cognitive Systems 2, pp. 167-184, 2012.

[20] M.T. Cox, T. Oates, M. Paisner, and D. Perlis, "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, University of the Azores, Portugal: CMATI, pp. 179-190, 2013.

[21] B Fritzke, "A growing neural gas network learns topologies," in Advances in Neural Information Processing Systems 7, MIT Press, Cambridge MA, 1995.

[22] M. M. Veloso, "Learning by analogical reasoning in general problem solving," Doctoral Dissertation, Carnegie Mellon University, Pittsburgh, PA, 1992.

[23] T. Martinez and K. Schulten, "A neural-gas network learns topologies," in T. Kohonen, K. Makisara, O. Simula, and J. Kangas (Eds.), Artificial Neural Networks, B.V., North Holland: Elsevier Science Publishers, 1991.

[24] B. Hammer, M. Strickert, T. Villman, "Supervised neural gas with general similarity measure," in Neural Processing Letters 21(1), pp. 21-44, 2005.

[25] J. Kolodner, "Case-based reasoning," San Francisco, Morgan Kaufmann, 1993.

[26] R.L. de Manteras, D. McSherry, D. Bridge, D. Leake, B. Smith, S. Craw, B. Faltings, M.L. Maher, M.T. Cox, K. Forbus, M. Keane, A Aamodt, and I. Watson, "Retrieval, reuse and retention in case-based reasoning," Knowledge Engineering Review, 20(3), pp. 215-240, 2006.

[27] M.T. Cox, "Perpetual self-aware cognitive agents," AI Magazine 28(1), pp. 32-45, 2007.

[28] D. Perlis, M. T Cox, M. Maynord, E. McNany, M. Paisner, V. Shivashankar, J. Shamwell, T. Oates, T.-C. Du, D. Josyula, and M. Caro "A broad vision for intelligent behavior: Perpetual real-world cognitive agents," unpublished.