Handling Seasonality using Metacognition

Kenneth M'Balé, Darsana Josyula
Department of Computer Science
Bowie State University
Bowie, MD USA
kmbale@cs.umd.edu and darsana@cs.umd.edu

***Abstract*— This paper summarizes a work in progress in the area of the metacognitive loop (MCL). The objective of MCL is to provide a design approach supported by software to extend an intelligent system's ability to cope with perturbations. A perturbation is any deviation from optimal performance for the system. Many MCL implementations exist, each increasing in sophistication. This paper describes an approach to produce the next implementation of MCL, which we call the General Purpose Metacognition Engine (GPME). The GPME evolves the functionality of the current implementation developed at the University of Maryland, MCL2, in particular, to handle seasonality. Seasonality is a periodic or cyclic variation in conditions that causes agents to re-learn when the length of the seasonal cycle exceeds their ability to detect the cycle.**

Keywords-Metacognition;Learning;Reasoning;Situated Agents;Autonomous Agents.

I. INTRODUCTION

One of the objectives of Artificial Intelligence is to impart upon systems the ability humans have for overcoming the "brittleness problem." The "brittleness problem" is the characteristic of systems to fail when operating circumstances exceed the designer's expectations. The Metacognitive Loop is a proposed solution for addressing this problem [1].

Metacognition is cognition about cognition; reasoning about one's own reasoning. Conceptually, the solution consists of one system, referred to as the host, which is integrated with another system referred to as MCL. The host supplies information to MCL about its actions and about the expectations of the results of these actions. MCL monitors the success of the host's actions by comparing expectations and outcomes. When an outcome does not match an expectation, MCL notes the anomaly. It assesses the anomaly using its internal knowledge such as significance, priority, similarity to other anomalies and possible responses. Finally, MCL guides the host by providing a suggestion to address the anomaly. MCL applies a basic algorithm composed of these steps; note, assess, guide, repeat [2].

Consider a robot trained to perform a certain function. Its initial training occurs on a dry surface. After some time in operation, it arrives at a wet surface. It needs to learn how to function efficiently on this new surface. A slightly wet surface might require minor adjustments such as tolerating wheel slippage. A very wet surface requires major adjustments that amount to relearning how to function. Learning is a time-consuming and expensive operation. After some time operating on the wet surface, the robot moves again onto a dry surface. Ideally, it is not necessary for the robot to once again invest the same level of effort for learning how to function on a dry surface. A better option is for the robot to note the change in the environment it is situated in, to assess which of its learned procedures have the best chance to work and to proceed efficiently. While this example used a robot, MCL and GPME are intended to integrate with cognitive robots or cognitive software agents. We use the terms host, agent and robot interchangeably to mean a cognitive host.

Several approaches have been researched to address seasonality. For example, Zhang and Qi describe the inability of artificial neural networks to handle seasonality [3]. Using simulated and real trend time series data, their research concludes that neural networks are not well suited to forecast without substantial prior data processing. In another paper, Taskaya-Temizel and Casey conclude that neural networks model seasonality provided their architecture is properly configured [4]. To address seasonality, the input layer size should be equal to the longest cycle information. In this paper, we discuss how seasonality, changes that repeat periodically, can be handled using expectation violations.

In Section II, we describe the initial MCL implemented using different strategies. In section III, we describe the current implementation of MCL (MCL2). In Section IV, we describe the design of the GPME, the successor to MCL2. In Section V, we describe our testing approach. Finally, we conclude.

II. MCL-ENHANCED Q-LEARNERS

Agents equipped with MCL have the ability to recover from unanticipated failures. Several applications of MCL exist that improve the performance of the underlying cognitive agent. The earliest MCL implementations utilized simple strategies to improve a host system that consisted of a Q-Learner as the baseline [5]. The baseline Q-Learner enhanced with MCL was capable of increasingly complex responses to expectation violations (anomalies).

The Q-Learner was deployed in a basic 8x8 grid where two grid locations contained rewards. After a set number of cycles, the reward locations, magnitude and type were changed to force an anomaly. The performance of the Q-Learner to relearn the location of the rewards, by abandoning the policy it had previously learned and restarting the learning when needed, is the key measure used to compare performance between several variations of MCL, as reported in Table I.

In response to the anomaly, simple MCL purges the policy that the Q-Learner has learned and restarts learning. The simple MCL acted after the occurrence of three

 ISBN: 978-1-61208-340-7

anomalies. The sensitive MCL considered reward values, time to reward and the expected reward per cycle. In addition, it expanded the definition of anomalies to include an unexpected reward, delays in finding rewards and the actual value of the reward compared to the expectation. The next two MCL added an effect on exploration by introducing an exploration factor called ε. A Q-learner will take the action recommended by its policy (the so-called ‘‘greedy action’’) with probability (1 – ε), and will take a random action with probability ε. This helps ensure that the agent comprehensively and continuously explores its world, learning the effect of all the actions it can take from all possible states, rather than sticking to what it already knows will bring rewards. In addition to purging, the ε value is changed to encourage the Q-Learner to explore the grid. In the steady case, the ε value is set to a fixed amount for a fixed number of cycles and then returned to the baseline value. In the decaying case, the ε value is set to the same fixed amount but it decays linearly back to the baseline value over the same number of cycles.

TABLE I. Q-LEARNER PERFORMANCE SUMMARY.

MCL type	Performance
Baseline Q-Learner	0.530
Simple MCL	0.545
Sensitive MCL	0.546
Steady ε	0.510
Decaying ε	0.526
Sophisticated MCL	0.567

Finally, the sophisticated MCL carried out an analysis of the anomaly and incorporated the results in its suggestion. It measured the magnitude of the anomaly and used this calculation to affect ε, and beyond a certain threshold, to purge. It used a decision tree to classify the degree of perturbation of the anomaly, to moderate its response.

In all these approaches, when the world has changed considerably, MCL throws out the current policy and restarts learning. Tsumori and Ozawa showed that in cyclical environments, reinforcement learning performance could be enhanced with a long-term memory and a ‘‘change detector’’, which would recall stored policies when a given known environment reappeared [6]. The different “throw-out current policy and explore” MCLs discussed above act as change detectors but they do not have a memory to store policies associated with a given environment and to recall policies when a known environment reappears (seasonality).

MCL has been applied to other systems to improve their responses to anomalies. In the Air Traffic Controller (ATC) [7] and the Natural Language Processor (Alfred) [8] implementations, MCL is implemented as a component within the host agent. In the Mars Rover [9] implementation, MCL is an external component that controls the behavior of the host agent.

An important consideration from the implementation of MCL across these several domains is that a general purpose, domain independent MCL could be possible and useful.

III. MCL2

MCL2 applies the note-assess-guide cycle for metacognition using three ontologies organized as a Bayes net shown in Figure 1. Indications nodes are connected to fringe nodes associated with domain specific expectation violations. Response nodes are connected to fringe nodes associated with domain specific actions. Failure nodes connect indications to responses. MCL2 tracks the success of suggestions against anomalies classified by indications and probable causes. The responses are statically defined at initialization to ensure that the host can process them appropriately.

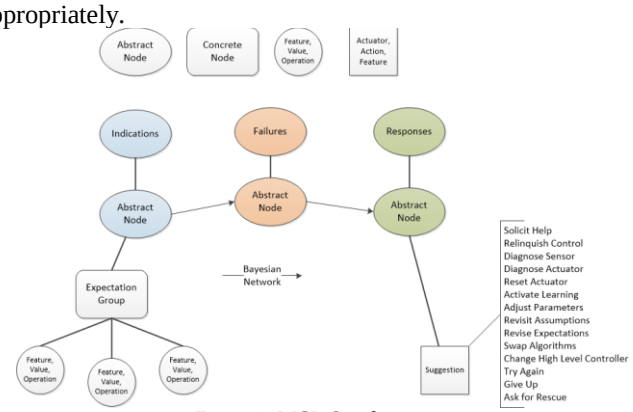

Figure 1. MCL Ontology

MCL2 succeeds in separating the MCL function from the host, as an independent process performing a general purpose function. However, MCL2 cannot handle seasonal changes efficiently since it does not have an episodic memory. Without an episodic memory, the Bayes net evolves continually without any note of time.

IV. GPME

The GPME squarely aims to remove brittleness from MCL itself and to facilitate integration with applications [10]. The communication between the host and the GPME is fully asynchronous (Figure 2).

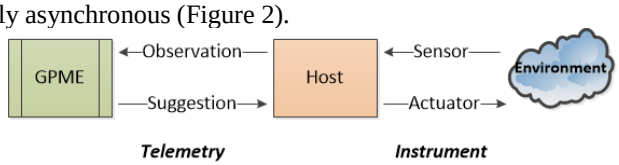

Figure 2. GPME and Host Integration

As a result, the burden is fully upon the GPME to detect anomalies from the telemetry. Therefore, some of the cognitive burden is alleviated from the host. Furthermore, the GPME defines anomalies dynamically by learning expectations from the telemetry. Consequently, the GPME uses an episodic memory to store, process and analyze its experiences.

A. *Episodic Memory*

The host supplies the GPME with a continuous telemetry stream. For the purpose of analyzing and detecting patterns in the telemetry, the GPME needs to break the telemetry into parts. We can consider each part as its own stream.

 ISBN: 978-1-61208-340-7

*1) **Segments***

Each distinct observation and suggestion is captured in an object we call a Segment (Figure 3).

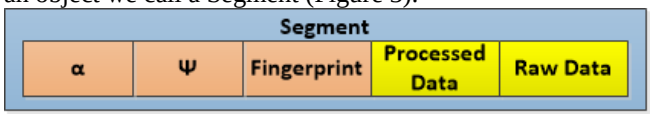

Figure 3. GPME Segment

The Processed Data and Raw Data originate from the host. Raw Data is actual sample data from the instrument. Processed Data is produced by the host using the Raw Data. For example, if the Raw Data contains a photograph, the Processed Data might contain a list of recognized faces or objects in the photograph. The α attribute is the A-distance; the probability of occurrence of this segment and its values within its own segment stream. To create the Fingerprint, the GPME samples 256 words (2 bytes) at fixed positions in the Raw Data. The fingerprint is an abbreviation of the Raw Data created from a fixed positional filter. To create the ψ attribute, the GPME measures the Hamming distance between the fingerprint and a constant fingerprint called the Prime Fingerprint. The Prime Fingerprint is randomly generated at initialization and it never changes throughout the life of the GPME.

2) Frames

We define a Frame (Figure 4) as the set of all segments received during a given GPME moment.

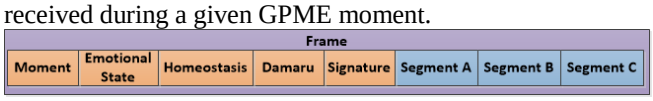

Figure 4. GPME Frame

The GPME adds a unique moment identifier to the frame. It then initializes the Damaru metric (decay value) to a special constant value called β and records the GPME's current homeostasis and emotional state (described later). The Damaru metric is a measure of the utility of the frame. It is a function of the number of times the frame is referenced in the episodic memory. The GPME generates the signature of the frame. The signature of the frame is a vector that identifies which instruments contributed segments to the frame and the number of segments each instrument contributed.

3) Links

Frames are interconnected using links. A link has a direction. The types of links are temporal, causative, attributive, spatial, order and composition. Temporal, causative and attributive links are established between adjacent frames when the frame is created. Causative and attributive links decay while temporal links do not decay. The temporal link reflects the order in which the frame arrived over time. This ordering is immutable. The causative link denotes that conditions in the successor frame are the result of preconditions in the predecessor frame. The attributive link denotes the inverse. Over time, incorrect causative and attributive links decay and disappear from the knowledge base. Spatial, order and composition links are created in response to certain anomalies. Temporal links connect successor and predecessor frames in order of arrival. There can be one of each type of link between two frames.

4) Clusters

The GPME stores each frame internally and generates additional data structures using sets of related frames. We call such a structure a Cluster. The GPME has a hierarchical knowledge base built using clusters (Figure 5). A cluster has a fixed maximum number of member frames (a prime number such as 13 or 17). The GMPE derives an artificial or abstract member called a Centroid. The centroid member contains the most significant information from each member, where significance is a function of the segment's α attributes. The centroid inherits all the links of its members. We refer to the abstract frame of a centroid as a Fragment because it is a partial or incomplete frame. A mature cluster exhibits a very tight grouping; the centroid is essentially equivalent to the members. Recall that a frame consists of several segments. The GPME uses the segments' ψ attributes to place the parent frame in a ψ cluster. The GPME uses the frame signature to place the frame in a signature cluster. A cluster can be understood as a super frame, where the member frames are treated as equivalent to each other and all of their links are available for traversal.

5) Decay and Moments

In order to keep the internal knowledge base manageable, every object carries the Damaru decay attribute. When the Damaru value reaches zero, the object is deleted. A GPME moment is defined as the time required to decrement the Damaru metric of every object by one. Therefore, each moment results in one frame, and in each moment every object decays.

6) Episodes

The GPME creates an episode in response to an anomaly. An episode begins with the content of short-term memory and includes all future frames, until the anomaly is no longer detected or decays. Therefore, an episode is a set of sequential frames. Episodes can overlap by sharing frames. The anomaly is identified by its signature. The anomaly signature consists of the attributes that caused the anomaly.

7) Cases

Episodes with similar anomaly signatures and similar data patterns are clustered into a Case. A case is equivalent to a cluster, but its members are episodes (sets of sequential frames). Therefore, the case centroid is an artificial or abstract episode that contains the most significant information from each member, where significance is a function of the anomaly signature**.**

Case-based reasoning is closely related to episodic memory. A case describes a problem the system encountered and the solution to the problem [11]. The system needs to match a new problem to an existing case to arrive at a previously successful solution.

The GPME uses a variant of case-based reasoning. Traditionally, the cases result in well-known solutions. However, the GPME creates its own cases from episodes and refines them over time. As a result, each case provides several overlapping solutions to the same problem. The cases form an abstraction hierarchy above the detailed episodes.

 ISBN: 978-1-61208-340-7

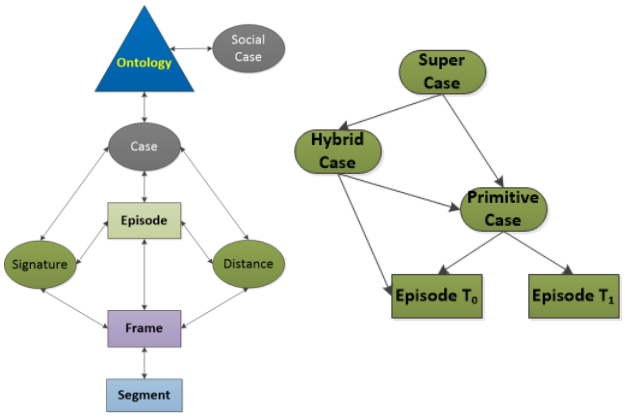

Figure 5. GPME Knowledge Base

The GPME uses cases to determine when to adjust the knowledge base by looking for cases that resemble the current anomaly circumstances.

8) Homeostasis

The GPME is an intrinsic reinforcement learner. It uses an internal reward that is a function of the number and types of anomalies that currently exist and the projection of the reward in the future. We refer to the result of this function as the homeostasis. As a result, the GPME can react to an anomaly caused by future unexpected homeostasis values.

9) Selective Imitation

In addition to being an intrinsic reinforcement learner, the GPME learns by imitation. The formalization of the knowledge base makes it possible for the GPME to share parts of its knowledge base with other instances. The recipient GPME can then select the portions of the model's knowledge base that it wants to incorporate within its own. The GPME is able to learn new cases and reasoning mechanisms from other GPME instances without needing to experience the environment first hand.

B. Episodic Memory-Driven Projections

The GPME uses its episodic memory in two ways. First, it matches the current frame to its experiences in order to project the future. Second, it matches the current anomaly to its experience to select a suggestion and project the future. Refer to Figure 6 during the description of the projection procedure.

1) Expectation Generation

The projection contains fragments and homeostasis. The GPME matches the current frame to the projection to determine whether anomalies are occurring. This approach allows the GPME to define anomalies and expectations dynamically.

2) Anomalies

The GPME creates an episode when it detects an anomaly. There are four types of anomalies; reflex, rational, context and emotional. The episode ends when the anomaly is no longer detected or it has fully decayed. A rational anomaly occurs when the projected homeostasis value is not achieved when expected, either because the actual homeostasis is over or under the projected homeostasis.

The other types of anomaly detection use the concept of Bandwidth. A bandwidth is the projected range of a certain value. The projection is based on historical value contained in the short-term memory. The value is projected to occur within an upper and lower bound. An anomaly occurs whenever the value falls outside the band. The anomaly is resolved when the value returns to its original projection. Since the short-term memory changes over time, the bandwidth also changes. Therefore, it is possible for the bandwidth to catch up to the projected value. When this situation occurs, the anomaly is aborted.

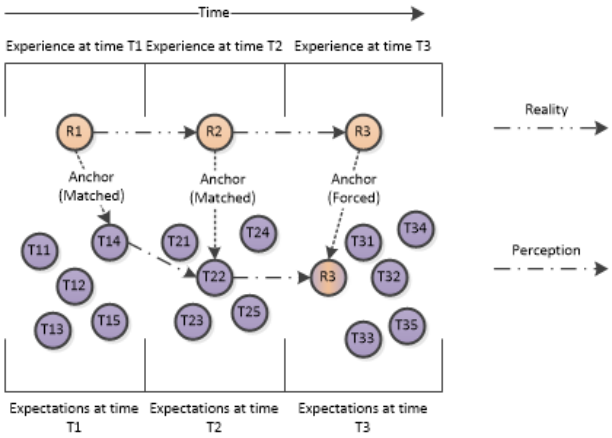

Figure 6. Projections

The reflex method projects an arrival rate of frames for each instrument stream. This projection is called the instrument arrival rate bandwidth. For example, the GPME expects the camera to provide an image every five seconds. After six seconds, if an image has not arrived, the GPME detects a reflex anomaly. The same anomaly would also be detected if the image arrived three seconds after the previous one. Since instruments are unlikely to be as regular as the example indicates, the GPME uses a range based on its experiences.

The context method detects an anomaly in two different ways. The first way relies on segment significance. The anomaly occurs when a segment that should be significant is not, or, when a segment that should not be significant is found to be. When the significance of a segment does not match its projected significance, the GPME detects the anomaly and creates an episode. The second way projects the accuracy of the projection. This expectation is called the projection accuracy bandwidth.

The emotional method relies on the homeostasis value. The GPME projects the homeostasis value to be within a certain range, called the homeostasis bandwidth. If the homeostasis value is outside the band, the GPME detects the anomaly and creates an episode. The bandwidth is the range between the highest and lowest homeostasis value in short-term memory, however, it is further adjusted by the emotional state.

3) Deadlines

When the GPME provides a suggestion, it also projects the effect the suggestion will have in the future, based on its

 ISBN: 978-1-61208-340-7

experience. As a result, the GPME also establishes deadlines by which results are expected. Deadlines allow the GPME to respond to anomalies of absence.

4) Reasoning Mechanisms

While responding to anomalies, the links allow the GPME to traverse the knowledge base in search of experiences that link current conditions to goal conditions. The algorithm the GPME uses to search the knowledge base is called a Reasoning Mechanism. The reasoning mechanism is specified in an internal language called Amri. The GPME is deployed with a number of reasoning mechanisms; deductive, qualitative, probabilistic, inductive, abductive, analogical, reactive, look-ahead and creative. The GPME can create new reasoning mechanisms by applying a Programming Method. The methods are random, mutation, interleaving and grafting.

For example, the deductive reasoning mechanism looks for the highest homeostasis value in linked cases, following the causal type links only. The abductive reasoning mechanism uses the causative or order link between super cases only. The look-ahead reasoning mechanism uses the temporal link through the clusters to find co-occurring centroids to build projections. The creative reasoning mechanism generates a result using its internal representation of the environment. The interleaving programming method blends two reasoning mechanisms into a new one.

5) Bayes Ontology

The GPME retains the Bayes ontology from MCL2. However, it is used to manage the success of reasoning mechanisms in producing a successful suggestion. In MCL2, it was used to track the success of specific suggestions.

C. GPME Processes

In principle, the GPME is similar to the Ouroboros model [12] at a high level. Comparisons between the models will be possible once there is an Ouroboros implementation. At the core of the Ouroboros model lays a self-referential recursive process with alternating phases of data acquisition and evaluation. In comparison, the GPME is a highly parallel system with several distinct processes operating concurrently on the episodic memory and the projection. There are eight distinct GPME processes named after Hindu mythology based on their function. Figure 7 depicts the GPME processes and their high-level data flows.

The **Vishnu** process assembles the current frame from the telemetry and supplies it to the Brahma and the Shiva processes. Vishnu sources the knowledge base to create projections based on the current frame. These projections are not related to anomalies; they are expectations of what normally happens in the future based on current conditions. The projections and their links are referred as the Vishnu Web.

The **Shiva** process identifies fragments (projections) that match the current frame. Such a fragment is referred to as an Anchor. Anchors are placed in short-term memory. Shiva reverses the decay of useful cases (and their underlying frames) and generates anomalies related to matching.

The **Kali** process decays every object in the knowledge base by one unit. It calculates the current homeostasis value and emotional state. It publishes a new moment unique identifier. It also triggers homeostasis anomalies.

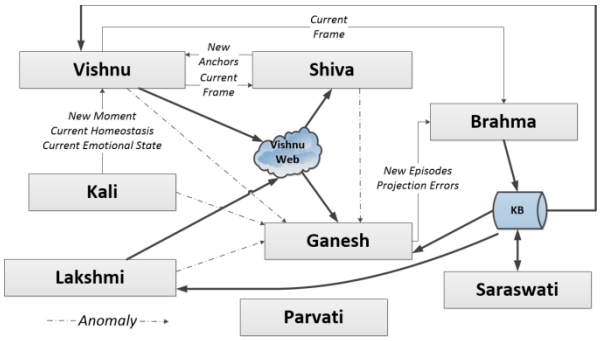

Figure 7. GPME Processes

The **Ganesh** process is responsible for responding to anomalies by doing nothing, waiting or using a reasoning mechanism to identify a suggestion. It creates an episode in the presence of an anomaly. If a suggestion is identified, it projects the expected results in the Vishnu Web.

The **Brahma** process manages the knowledge base and continually revises and optimizes clusters and cases. It also maintains the ontology.

The **Saraswati** process is responsible for communicating with other GPME instances.

The **Lakshmi** process is responsible for handling special requests from the host and for monitoring the accuracy of the Vishnu Web.

The **Parvati** process provides an instrumentation and management interface.

V. ONGOING AND FUTURE WORK

Since the GPME improves on MCL2, the testing strategy revolves around comparing the performance of systems placed in the same circumstances and scenarios. We intend to compare the system's performance when integrated with MCL2 in comparison to integrated with GPME. If possible, we will also compare performances with an established metacognitive framework such as SOAR-RL [13].

While several metrics will be collected, the principal metric is homeostasis. As we described earlier, homeostasis is the number of anomalies weighed by type. During the sampling interval, we will calculate the homeostasis of the system and examine the curve over time.

The core test involves a maze the system must navigate to obtain rewards. Periodically, the location and nature of the rewards will change in a non-random manner, cycling back through known states. The test emulates seasonality. We can envision that the system is an animal and the reward is food. Depending on the season, the animal finds the food in different locations and in different quantity. However, the location and quantity of the food is consistent with the season.

We expect that each time the season changes, the system experiences a surge in anomalies as rewards become scarce. In response, the metacognitive component helps the system find new rewards. Once a new source is found, the number

 ISBN: 978-1-61208-340-7

of anomalies should trend back towards a norm. We can measure cleverness based on the norm. The system with the lower norm is cleverer at finding rewards. We can measure adaptability based on how quickly the system responds to the change in seasons. The system that recognizes the change faster is more adaptable. Finally, we expect that as the cycle of seasons repeats itself, the GPME will outperform MCL2 and the RL in terms of total homeostasis over time.

Plotting the weighted anomalies over time, we calculate the slope of the curve from the change of season to the peak number of anomalies, to measure the efficiency of the metacognitive component's ability to recognize the change of season. The slope of the curve between the peak and the normal number of anomalies measures the efficiency of adaptation. The actual value of the normal number measures cleverness.

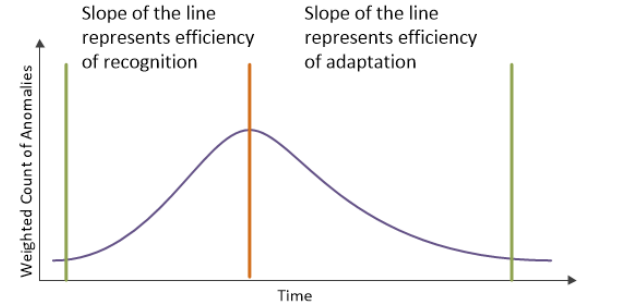

Figure 8. Projected Performance Curve over a Season

The test will execute several thousand seasonal cycles. Analyzing the performance curves of all cycles, we expect that the GPME will outperform MCL2 and RL by substantially minimizing the slopes over time.

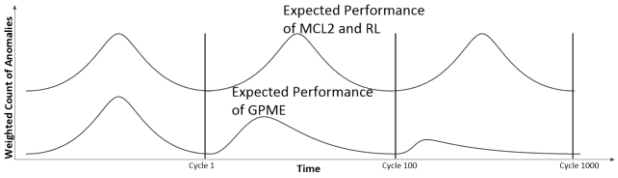

Figure 9: Projected Aggregate Performance Comparison

VI. CONCLUSION

The domain-generality of the GPME and its open and flexible interface will allow designers to build robust systems more rapidly, accelerating the application of metacognition enabled solutions in a larger number of domains.

The GPME builds on MCL2 by introducing the ability to deal with seasonality. To support this ability, the GPME develops its own expectations to supplement those the designer specifies. As a result, the foresight of the designer is no longer a limiting factor and it is free to discover and handle anomalies the designer did not anticipate. The GPME uses its episodic memory to match attributes of anomalies to cases it generates. This approach substantially lowers the need to relearn when dealing with seasonal changes. The introduction of decay means that experiences and cases that are not valuable eventually exit the episodic memory in order to keep its performance constraints manageable.

ACKNOWLEDGMENTS

This research is supported in part by the Office of Naval Research grant ONR #N00014-12-1-0430.

REFERENCES

[1] [1] M. T. Cox, "Metacognition in computation: A selected research review," *Artif. Intell.*, vol. 169, no. 2, pp. 104–141, Dec. 2005.

[2] [2] M. D. Schmill, M. L. Anderson, S. Fults, D. P. Josyula, T. Oates, D. Perlis, H. Shahri, S. Wilson, and D. Wright, "The Metacognitive Loop and Reasoning about Anomalies." p. 17, 2011.

[3] [3] G. P. Zhang and M. Qi, "Neural network forecasting for seasonal and trend time series," *Eur. J. Oper. Res.*, vol. 160, no. 2, pp. 501–514, 2005.

[4] [4] T. Taskaya-Temizel and M. C. Casey, "A comparative study of autoregressive neural network hybrids.," *Neural Netw.*, vol. 18, no. 5–6, pp. 781–9, 2005.

[5] [5] 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," *J. Exp. Theor. Artif. Intell.*, vol. 18, no. 3, pp. 387–411, Sep. 2006.

[6] [6] K. Tsumori and S. Ozawa, "Incremental learning in dynamic environments using neural network with long-term memory," *Proc. Int. Jt. Conf. Neural Networks, 2003.*, vol. 4, pp. 2583–2588, 2003.

[7] [7] D. P. Josyula, H. Vadali, B. J. Donahue, and F. C. Hughes, "Modeling metacognition for learning in artificial systems," *2009 World Congr. Nat. Biol. Inspired Comput.*, pp. 1419–1424, 2009.

[8] [8] D. P. 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.

[9] [9] D. Wright, "Finding a Temporal Comparison Function for the Metacognitive Loop," Doctoral Dissertation. University of Maryland, College Park. 2011.

[10] [10] K. M. M'Balé and D. P. Josyula, "Integrating Metacognition into Artificial Agents," in *AAAI 2013 Fall Symposium Series*, 2013, pp. 55–62.

[11] [11] R. C. Schank, *Dynamic Memory Revisited*, 2nd ed. New York, NY: Cambridge Press, 1999, p. 302.

[12] [12] K. Thomsen, "The Cerebellum in the Ouroboros Model, the ' Interpolator Hypothesis ,'" in *COGNITIVE 2013*, 2013, pp. 37–41.

[13] [13] R. P. Marinier and J. E. Laird, "Emotion-Driven Reinforcement Learning," *Cogn. Sci.*, pp. 115–120, 2008.

 ISBN: 978-1-61208-340-7