Advances in Cognitive Systems 10 (2022) 1-6 Submitted 09/2022; published X/20XX

Towards an Artificial, General Episodic Memory via Learning in Noisy, Perceptually-Aliased Environments

Braeden Lane LANEB22@UP.EDU
Connor Morgan MORGANCO23@UP.EDU
Kai Vickers VICKERS23@UP.EDU
Max Woods WOODSM25@UP.EDU
Andrew Nuxoll NUXOLL@UP.EDU
Department of Computer Science, University of Portland, Portland, OR 97203 USA

Abstract

When an agent is acting in an environment, it uses a set of sensors to detect its current state. Contemporary machine learning algorithms often rely upon being in environments where the ratio of states to sensor readings is at or near 1:1. The reality in natural environments is that there can be perceptual aliasing (i.e., multiple states "feel" the same to the agent). In previous research, an episodic memory has been used to allow it to uniquely identify the current state. However, these approaches either have to be tuned to each new environment, have difficulty leveraging multiple sensors, or handle noise sensors poorly. In this research, we describe a new episodic memory system and learning agent that overcomes all three of these issues. We demonstrate this agent's effectiveness compared to three others by using a blind finite-state machine environment that exhibits both perceptual aliasing and a noise sensor.

1. Spectrum of Environment Sensors

In many artificial environments, an agent is given sensors that map 1:1 with possible states. Furthermore, researchers typically craft the sensors to be reliably informative and relevant. While this is convenient and appropriate, it is hardly typical of a natural environment.

For example, an AI agent playing chess can sense which piece is in which square, which pieces have been taken, whose turn it is, etc. A human in a natural environment must extract this information from a visual field and short term memory. Furthermore, she must contend with extraneous sensing ranging from background noise, the fact that a salt shaker has filled in for a missing pawn, intrusive thoughts of unrelated matters, etc.

These extraneous sensors can rapidly overwhelm traditional machine learning algorithms which rely on an approximate 1:1 mapping of sensing to state (Gupta & Gupta, 2019).

Another dimension of this spectrum is perceptual aliasing. In this scenario environment sensors are insufficient and, as a result, multiple states in an environment might feel the same to the agent. While this can't entirely happen in a natural environment, related situations can. For example, navigating an unfamiliar building with many identical hallways or finding your glasses in a dark hotel room.

Perceptual aliasing frequently arises in robot navigation, particularly when multiple locations appear similar to the robot (Qamar et al., 2022).

Additionally, sensors can be noisy, unreliable or simply extraneous. While this can occur anywhere, it has also been a particular challenge in robotics where computers must interface in the real world using lidar and cameras which don't provide consistently correct information (Welch et al., 1995; Tawiah, 2020).

In this paper, we focus on a simple environment that has both perceptual aliasing and extraneous sensors.

2. Artificial, General Episodic Memory

In the past, agents have employed episodic memory to overcome perceptual aliasing (McCallum, 1994; Bakker et al., 2003; Rodriguez et al., 2017; Regier et al., 2020). Rather than relying upon a 1:1 mapping between sensing and acting these agents use sequences of past experiences to uniquely identify the current state.

In humans, episodic memory is a type of long-term memory you use to remember specific events such as the last time you heard live music, or what hotel you are staying in (Tulving, 1983). Thus, these artificial episodic memories used to overcome perceptual aliasing are quite simple in comparison. They are, in effect, a database of discrete events (episodes) indexed in temporal order. Each episode consists of the agent's sensing and selected action.

A similar approach has been used in general agents (Vere & Bickmore, 1990; Chaudhry et al., 2018), game NPCs (Brom & Lukavsky, 2008), emotionally-aware robots (Dodd & Gutierrez, 2005), and problem solvers (Tecuci & W. Porter, 2007). Notably, more complicated implementations exist in some cognitive architectures (Bölöni, 2011; Laird, 2012; Menager & Choi, 2016).

Our long-term aim with this research is to move toward a general episodic memory. Such a system would have these properties:

- It would function in any environment without having to be tuned for that environment. Among other factors, this includes environments at any point on the spectrum of environmental sensors described above. https://www.overleaf.com/project/62f18db18641c4f8102e3c3c
- It could record episodes of any scope or content.
- It would respond effectively to any reasonable memory cue.
- It would function reasonably within the boundaries of given, defined, finite resources. Specifically, this will require an effective forgetting mechanism. See (Nuxoll et al., 2010; Nagy et al., 2020; Yalnizyan-Carson & Richards, 2022) for some existing work in this area.

As defined by Nuxoll & Laird 2010, this system would support a variety of cognitive capabilities including:

- Recognizing Novel Situations
- Avoiding Repetitive Behavior
- Modeling Outcomes of Prospective Actions
- Retroactive Learning - re-learning from past experiences given new contextual knowledge
- Explaining Past Decisions

The value of these cognitive capabilities is obvious and beyond the ability of many contemporary machine learning algorithms (Ribeiro et al., 2016; Rosenfeld et al., 2018).

3. Blind FSM Environment

A simple environment that exposes the agent to perceptual aliasing is called the blind finite state machine (FSM) environment. Here we use a finite-state machine as formally defined by (citation needed). The agent navigates the FSM to reach a goal state but its task is greatly complicated by a lack of knowledge. Specifically, the agent is unaware of the following:

- how many states the machine consists of
- how many goal states there are
- which state it is currently in
- what happens when it reaches a goal

The agent is only aware of:

- what actions it can take (the FSM's alphabet)
- when its most recent action has taken it to a goal

Blind FSM has some additional properties that are relevant but not within the agent's purview:

- Each time the agent reaches the goal state, it is immediately teleported to a random, non-goal state. Thus the agent never takes an action in the goal state.
- There is a path to the goal state from any non-goal state. The FSM is otherwise randomly generated within the boundaries of this constraint.

As should be apparent, all states "feel" the same to the agent. In particular, this is the most extreme perceptual aliasing that is practical for learning. As a result, traditional machine learning mechanisms can not perform well in this environment without some form of compensatory mechanism. Episodic memory is one such mechanism (McCallum, 1994). Other mechanisms exist (Métivier & Lattaud, 2002; Zatuchna & Bagnall, 2009; Siddique et al., 2021) but it's unclear how they would perform when perceptual aliasing is this extreme.

The amount of perceptual aliasing in the Blind FSM environment can be adjusted with the addition of more sensors. For example, if the agent had a sensor that told it which state it was in, the Blind FSM would become a trivial environment.

For this research we tested three types of binary sensors to assist and confound the agent:

- An odd sensor. All states in our implementation are numbered. This sensor is on when the agent is in an odd-numbered state.
- A noise sensor. This sensor has a 50% chance of being on or off (randomly chosen).
- A state sensor. This sensor only on when the agent is in one particular non-goal state (randomly chosen).

The agent isn't always given all of these sensors and does not know their meaning. Thus it must consider the potential value of a noise sensor as equal to that of any of the others.

4. Existing Episodic Memory Agents

We are aware of three existing agents that learn effectively in severely perceptually aliased environments like Blind FSM.

The Nearest Sequence Memory (NSM) agent (McCallum, 1994) is a Q-Learning agent (Sutton & Barto, 2018) that assigns Q-values to sequences of episodes rather than individual episodes. Like all reinforcement learning agents, it must be tuned to each new environment via a small collection of hyperparameters. Furthermore, the agent is hampered by noise sensors as we will demonstrate below.

The MaRz agent (Rodriguez et al., 2017) is designed to work with maximal perceptual aliasing (i.e., goal sensor only). It can not, by design, leverage any additional sensors given to it. Furthermore, MaRz relies upon operating in a deterministic environment which is not practical for a general episodic memory.

The ARO agent (Regier et al., 2020) is a rule-based agent that builds a tree of possible episode sequences. Based on past experiences, the agent assigned a probability of correctness to each sequence in the tree. Thus, the agent is able to calculate a path to the goal that is most likely to succeed. Similarly to MaRz it is designed to operate with exactly one non-goal sensor.

5. Phujus Agent

Given that the long-term goal is to build an artificial, general episodic memory, we take a step toward that goal by building an episodic memory system that addresses some omissions of existing systems. We've also built an episodic learning agent that uses this system. In particular, we present the Phujus (FOO-juss) agent and demonstrate some initial evidence that it has these properties that one or more of the other agents lack:

- It can work with any number of sensors given sufficient computing resources.
- It works at least as well as other agents in environments with perceptual aliasing.
- It can learn better in environments with noise sensors.
- Aside from those that limit the agent's use of available computing resources (particularly RAM) Phujus has no hyperparameters that need to be tuned for a given environment.

To date, we are not aware of any episodic memory system with these properties. This paper presents an implementation of a system which we believe is the closest we have come to achieving it and demonstrate that these properties give it an advantage in some environments.

Specifically, we will present the Phujus agent in the following sections by first reviewing its rules and then its learning algorithm. Subsequently, we will demonstrate its performance in the Blind FSM environment.

6. Phujus Rules

Regardless of type, a rule consists of a left-hand-side (LHS) – the conditions that must be true for a rule to match – and a right-hand-side (RHS) – the anticipated outcome at the next time step if the rule matches.

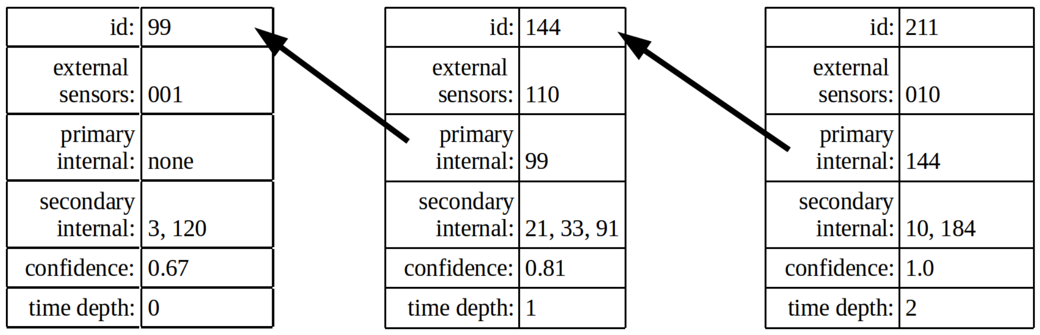

Figure 1. TFRules correspond to individual episodes and chain together to form sequences.

TFRules are each built from a single episode of the agent's experience. Broadly, an agent episode consists of the following information about one time step: what the agent was sensing, what the agent was thinking, what action the agent selected, and what sensing resulted from taking that action. Specifically, a TFRule contains these LHS properties:

- A set of external sensor values. If, for example, the goal sensor was off in the source episode then the TFRule's LHS conditions will include a requirement that the goal sensor is off.
- A selected action. In the Blind FSM environment this will be a letter from the FSM's alphabet.
- A primary internal sensor (optional). An internal sensor is defined as a sensor that does not come from the environment but, instead, is a product of the agent's thinking. For Phujus all internal sensors are binary and associated with some other TFRule. The sensor is "on" if the rule had the highest match score in the previous time step and off if it did not. Thus, TFRules can be chained together to form sequences of episodes. These are used, implicitly, in much the way that other episodic agents use episode sequences.
- A set of secondary internal sensors. These sensors do not have to be on for the rule to match but can affect its match score (see below).

The RHS of a TFRule consists of an expected set of external sensor values.

At any given timestep, multiple TFRules can match. However, the secondary internal sensors provide a way to evaluate the strength of the match. The agent uses a partial matching algorithm similar to term frequency-inverse document frequency (TF-IDF) matching to calculate a match score for a matching rule.

In particular, each condition has a "term frequency:" how often this sensor is on when the rule matches. Each sensor also has a document frequency: how often the sensor is on in general. The tf and df values are both in the range [0.0..1.0]. The condition's match score for a particular condition is a value in the range [-1.0..1.0]. It is calculated as follows:

```
match_score(double tf, double df, boolean sensor_is_on)
    IF sensor_is_on
        base_score = tf
    ELSE
        base_score = 1.0 - tf
    base_score = base_score - 0.5
    base_score = base_score * 2.0
    relevance = abs(tf - df)
    RETURN base_score * relevance
```

In addition to the secondary internal sensor conditions, partial matching algorithms are also applied to the mandatory parts of the LHS and affect the match score. The total match score for a TFRule is the sum of its conditions' scores divided-by the sum of its conditions' relevances.

Previous research shows TF-IDF can be an effective mechanism for episode matching (Vanderwerf et al., 2016).

In addition to its LHS and RHS conditions, every TFRule has two properties:

- *a confidence*: a measure of how often this rule has correctly predicted the outcome in the recent past.
- *a time depth*: how many previous TFRules are in the chain linked by the primary internal sensors. For example, consider a rule A that has rule B as a primary internal sensor and rule B has rule C as its primary internal sensor. If rule C has no primary internal sensor, then rule C has time depth 0, rule B has time depth 1 and rule A has time depth 2. To keep the agent from over-using the computer's resources, a maximum time depth is enforced.

PathRules provide a broader perspective to the agent than TFRules. The RHS of a PathRule is a sequence of episodes that the agent expects could occur. The LHS is a set of zero or more PathRules. Like TFRules, PathRules have a confidence based on how frequently they have been correct.

At any time step, the agent determines which PathRule matches its most recent experience (e.g., previous matching PathRule + sequence of actions). This matching PathRule can then be the LHS when matching some other PathRule with a particular sequence of actions. Thus, if the agent has used its TFRules to generate a sequence of actions to reach the goal it can consult the confidence level of a matching PathRule to get a strategic (rather than tactical) opinion as to whether that sequence will actually be successful.

In practice, we've found that PathRules give the agent the perspective (or meta-reasoning) necessary to avoid repetitive behavior. It is notable that, intuitively, human memories have a similar structure. A particular event (e.g., jumping into a lake) takes place in the context of a broader event (e.g., a summer hike).

7. Phujus Learning Algorithm

The Phujus agent follows these steps at each time step:

1. The agent receives new values from its external sensors.
2. The agent matches its TFRules to the previous episode – including matching its current external sensors to each rule's RHS – to determine which internal sensors will be on for the current

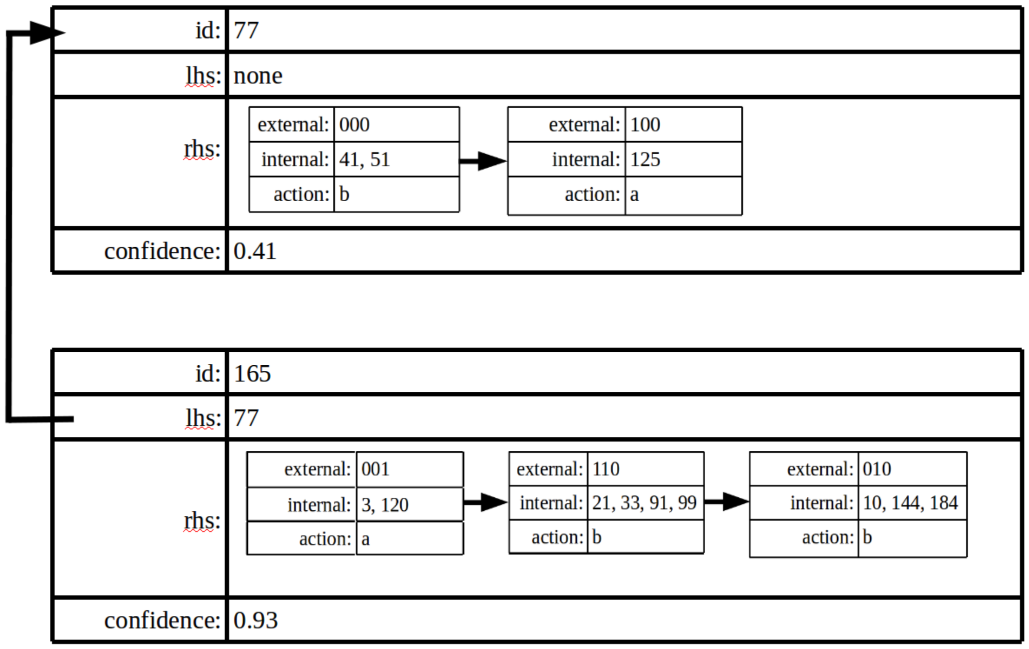

Figure 2. PathRules represent a broader context of episode sequences.

time step. For each time depth, the sensors associated with the matching TFRule(s) that have the highest match score are turned on.

3. The agent creates one or more new TFRules based on the episode from the previous time step. A maximum of one new rule is created for each active internal sensor which is used as the new rule's primary internal sensor. Redundant rules are not created.

4. The confidence value is adjusted for every TFRule whose LHS matched in the previous timestep.

5. The tf and df values are updated for every sensor.

6. The agent creates a new PathRule based on the previous matching PathRule and the sequence it has completed to this point. A redundant PathRule is not created.

7. The agent creates a new PathRule based on the sequence of episodes the agent is attempting to follow. This may be different that the PathRule created in the previous step because the agent's prediction about the sequence may have incorrectly predicted what occurred. A redundant PathRule is not created.

8. The confidence level of all matching PathRules is updated.

9. The agent sometimes selects a random action (see below). It also tracks how frequently a random action takes it to a goal state. Thus, if it took a random action in the previous time step that statistic is updated at this point. This random action success rate is used in search (below).

10. If the agent is partway through a planned sequence of actions it continues that sequence.

11. If the agent has just completed a planned sequence of actions, just reached the goal, or just took a random action, it uses its TFRules build a search tree to find a new sequence (see the Search Tree Algorithm below).

12. If a sequence was found (or is underway), the agent takes the next action from this sequence. Otherwise, it takes a random action.

8. Search Tree Algorithm

1. A root node is constructed consisting of the current external and internal sensors

2. For any given node, successor nodes can be created for each TFRule whose LHS external and internal sensors match the given node. The associated LHS action becomes the action taken to reach one or more new child nodes.

3. The predicted value for each external sensor in a child node is generated via a voting scheme. Every TFRule whose LHS matches the parent node is allowed to vote for each sensor value on its RHS. Each vote has a "strength" based upon the match score of the best matching TFRule. Each unique external sensor prediction is assigned a confidence of correctness (in the range [0.0..1.0] based upon these votes.

4. These confidence values are scaled so that the highest confidence is 1.0. Our tests have shown this improves agent performance. We believe this is because it puts all peer nodes on equal footing.

5. Any external sensor set with a confidence that exceeds the random action success rate (see step 9 in the main algorithm) is used to create a child node. The confidence value is retained for future use.

6. Once the predicted external sensor values are set, the internal sensor values are calculated using the same mechanism as step 2 in the main algorithm (above).

7. Each node is assigned a score based upon the product of these values: its confidence, a PathRule evaluation (if there is a matching PathRule) and the inverse of the node's depth.

8. Each child node can now be expanded to continue the search. A node is not expanded if the goal sensor is on or its score is lower than the random action success rate.

9. The resulting tree is examined to find a sequence that leads to the goal. If multiple such sequences exist, the one with the highest score at its terminal node is used. If no goal sequence exists, then the search fails.

Some notes on the algorithm above:

- The agent currently uses a max search tree depth to limit the search's use of available computing resources. This was a choice of convenience and would not work with larger environments. In the future, we intend to switch to a heuristic search that has no maximum depth.
- The voting scheme described above was selected after testing and rejecting two, other less effective schemes: a) each TFRule votes with its own match score and b) only the best matching TFRule votes with its match score. The scheme we selected seems to provide the best balance of quality and quantity of votes.
- The use of 1/depth as part of a search tree node's score is a way of helping the agent decide whether to take a shorter path with a lower confidence vs longer path (i.e., a bigger investment) with a higher confidence. The use of 1/depth is based on the Sunrise Problem in mathematics (Laplace, 1814).

9. Results

In this section we show our results from testing the performance of the NSM, MaRz and PhuJus agents in a blind FSM with 24 non-goal states and 1 goal state. The alphabet size is 3 (three possible actions in each state). ARO was excluded from these experiments since it was incompatible with the test environment due to its requirement of exactly one non-goal sensor. Notably, MaRz was included but its algorithm ignores non-goal sensors entirely. Four test scenarios are presented below. In each one the agent is equipped with a different set of sensors.

In figure 3, the agent has two sensors: the goal sensor and odd sensor described above. The x-axis depicts 200 successive trips from a random start to the goal state. The agent retains its memories of all previous goals, so it has more knowledge each time it seeks a subsequent goal. The y-axis shows the number of actions required to reach the goal. This value is the average of 100 runs each with a different FSM that has the same specification.

As the figure shows, the NSM and Phujus agents are predictably able to leverage the extra knowledge provided by the odd sensor. An analysis of the FSMs used to gather these data shows that both these agents eventually reach near-optimal behavior in the long term with PhuJus reaching this level sooner. Since MaRz can not leverage non-goal sensors, it does not reach this level.

Figure 4 is identical to Figure 3 except that a noise sensor has been added. This result indicates that Phujus is better able to cope with the addition of the noise sensor and maintain near-optimal behavior.

In figure 5 a second noise sensor has been added for a total of four sensors. None of the agents can perform as well in this scenario with the NSM agent struggling the most. The noise at this point has become such liability that MaRz profits (relatively) from the fact that it ignores non-goal sensors.

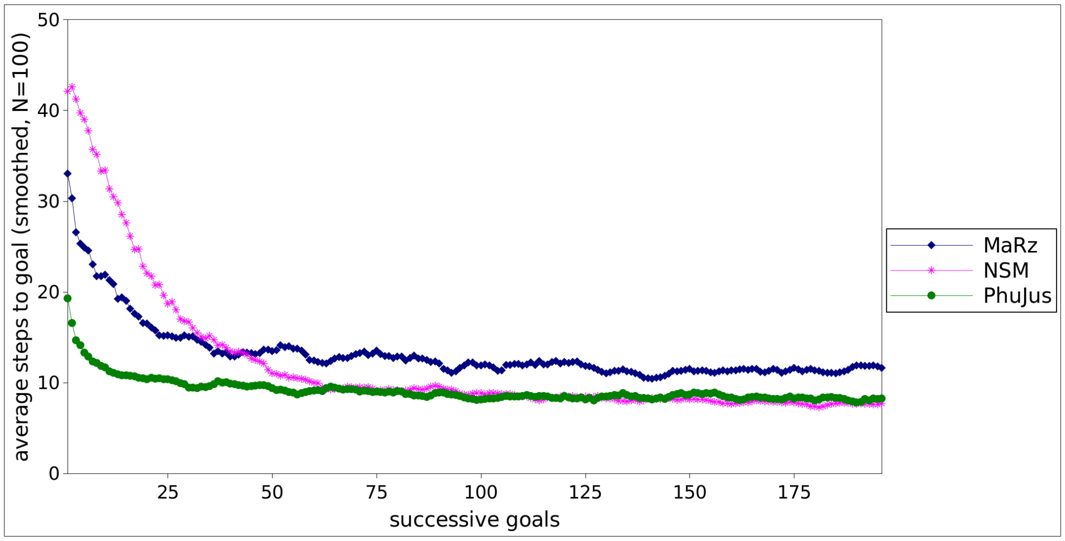

Figure 3. Agent comparison with perceptual aliasing: goal sensor and odd sensor.

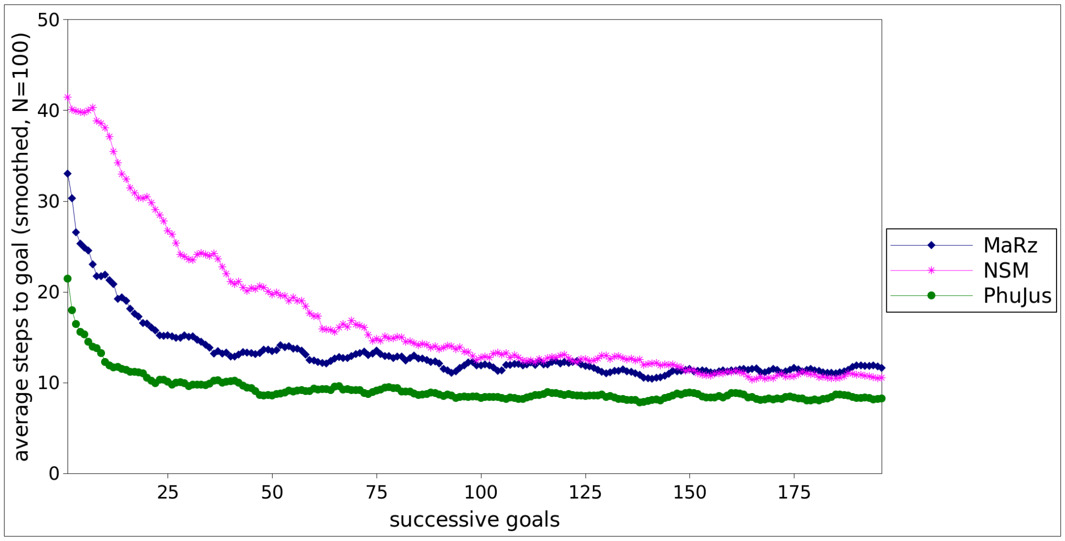

Figure 4. Agent comparison with perceptual aliasing: goal sensor, odd sensor and noise sensor.

In figure 6 the agent has one of each type of sensor: goal, odd, noise and state. The new state sensor provides additional knowledge that both NSM and PhuJus leverage to slightly improve their performance compared to figure 3. The MaRz agent performs worst in this scenario.

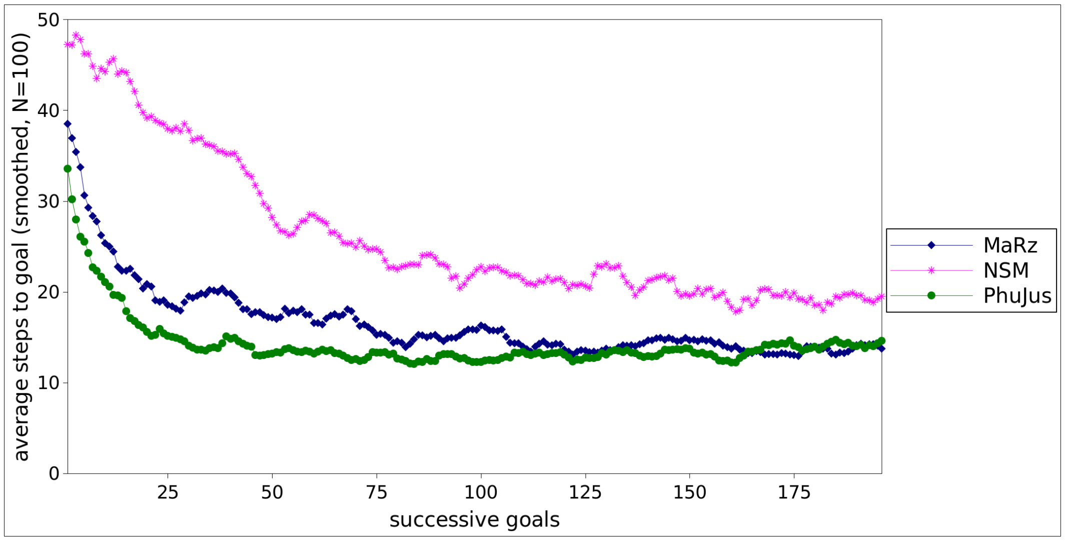

Figure 5. Agent comparison with perceptual aliasing: goal sensor, odd sensor and two noise sensors.

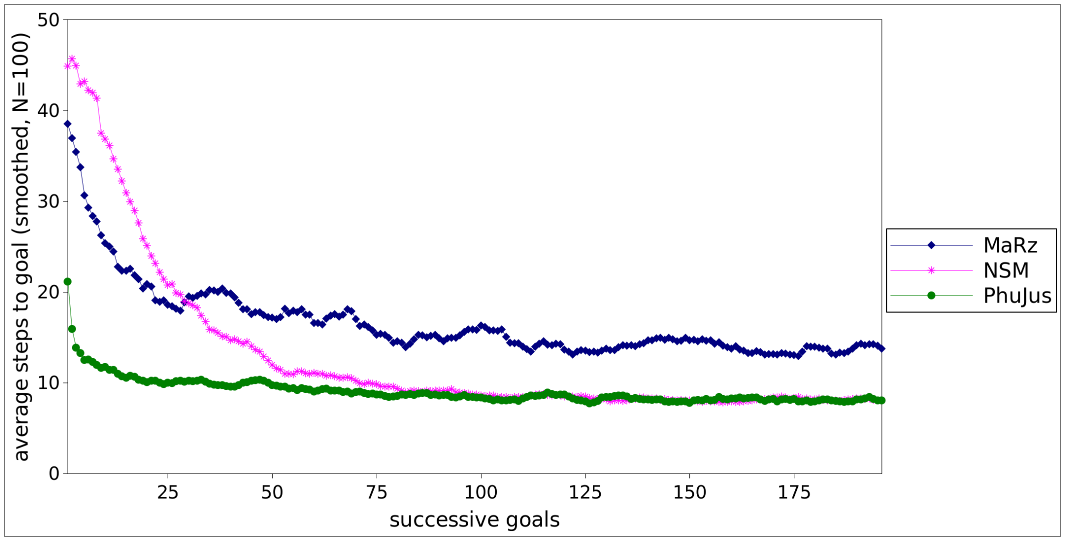

Figure 6. Agent comparison with perceptual aliasing: goal sensor, odd sensor, noise sensor and state sensor.

10. Summary of Contribution

In this paper we introduced the Phujus agent which we believe brings us closer to the goal of an artificial, general episodic memory as it is better able to operate in environments with perceptual aliasing and noisy sensors. In particular, Phujus includes the following innovations compared to previous approaches:

- The agent uses the concept of internal sensors to allow rules to represent sequences of episodes in an episodic memory and, thus, uniquely identify states in an environment with perceptual aliasing.
- The agent uses a partial matching algorithm that gives it the ability to cope with noise sensors.
- It introduces PathRules, a broader second layer of episodic memories that allows the agent to avoid repetitive, incorrect behavior. As such, PathRules form a basis for a simple form of meta-reasoning.
- It avoids the use of environment-specific hyperparameters and thus can work in any environment compatible with its architecture.

11. Future Work

We see three main thrusts to future improvements to the Phujus agent:

1. Phujus' rule database grows exponentially and must be limited to a fixed maximum size. While we can apply more computing resources as needed, it would be more practical to have Phujus adapt to a given finite capacity. We currently see two complementary avenues for this: a) generalizing similar rules likely via a partial matching approach with the external sensors and b) removing rules that have low confidence or are used infrequently.

2. We want to apply Phujus to larger and more complicated environments. In particular, we are eager to experiment with non-deterministic environments, non-static environments, and environments with large numbers of extraneous sensors.

3. We want explore the possibility of having Phujus learn to use its episodic memory to explain its decisions.

References

Bakker, B., Zhumatiy, V., Gruener, G., & Schmidhuber, J. (2003). A robot that reinforcement-learns to identify and memorize important previous observations. *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)* (pp. 430–435 vol.1).

Bölöni, L. (2011). An investigation into the utility of episodic memory for cognitive architectures. *AAAI Fall Symposium - Technical Report*.

Brom, C., & Lukavsky, J. (2008). Episodic memory for human-like agents and human-like agents for episodic memory. *AAAI Fall Symposium - Technical Report*, (pp. 42–47).

Chaudhry, A., Ranzato, M., Rohrbach, M., & Elhoseiny, M. (2018). Efficient lifelong learning with A-GEM. *CoRR*, *abs/1812.00420*. From `http://arxiv.org/abs/1812.00420`.

Dodd, W., & Gutierrez, R. (2005). The role of episodic memory and emotion in a cognitive robot. *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.* (pp. 692–697). IEEE.

Gupta, S., & Gupta, A. (2019). Dealing with noise problem in machine learning data-sets: A systematic review. *Procedia Computer Science*, *161*, 466–474. From `https://www.sciencedirect.com/science/article/pii/S1877050919318575`. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

Laird, J. E. (2012). *The soar cognitive architecture*. Cambridge, MA: MIT Press.

Laplace, P. S. (1814). *A philosophical essay on probabilities. (translated by f. w. truscott & f. l. emory.)*. New York: Dover.

McCallum, R. A. (1994). Instance-based state identification for reinforcement learning. *Advances in Neural Information Processing Systems*, *7*.

Menager, D., & Choi, D. (2016). A robust implementation of episodic memory for a cognitive architecture. *CogSci*.

Métivier, M., & Lattaud, C. (2002). Anticipatory classifier system using behavioral sequences in non-markov environments. *International Workshop on Learning Classifier Systems* (pp. 143–162). Springer.

Nagy, D. G., Török, B., & Orbán, G. (2020). Optimal forgetting: Semantic compression of episodic memories. *PLoS Computational Biology*, *16*, e1008367.

Nuxoll, A., Tecuci, D. G., Ho, W. C., & Wang, N. (2010). Comparing forgetting algorithms for artificial episodic memory systems. *Proceedings of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour (AISB)*.

Qamar, S. M., Khawaja, F. I., Ali, S., Qureshi, A. H., Ayaz, Y., Naveed, M., & Abbasi, A. G. (2022). An adaptive neuro-fuzzy inference system to solve perceptual aliasing for autonomous mobile robots. *Proceeedings of the 27th International Symposium on Artificial Life and Robotics*.

Regier, R., Price, O., Hadi, A., Faltersack, Z., & Nuxoll, A. (2020). Aro: A memory-based approach to environments with perceptual aliasing. *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Rodriguez, C., Marston, G., Goolkasian, W., Rosenberg, A., & Nuxoll, A. (2017). The marz algorithm: Towards an artificial general episodic learner. *International Conference on Artificial General Intelligence* (pp. 154–163). Springer.

Rosenfeld, A., Zemel, R., & Tsotsos, J. K. (2018). The elephant in the room. *arXiv preprint arXiv:1808.03305*.

Siddique, A., Browne, W. N., & Grimshaw, G. M. (2021). Frames-of-reference-based learning: Overcoming perceptual aliasing in multistep decision-making tasks. *IEEE Transactions on Evolutionary Computation*, *26*, 174–187.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement learning: An introduction*. MIT press.

Tawiah, T. A.-Q. (2020). A review of algorithms and techniques for image-based recognition and inference in mobile robotic systems. *International Journal of Advanced Robotic Systems*, *17*, 1729881420972278. From `https://doi.org/10.1177/1729881420972278`.

Tecuci, D., & W. Porter, B. (2007). A generic memory module for events. *Twentieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2007* (pp. 152–157). AAAI Press.

Tulving, E. (1983). *Elements of episodic memory*. Clarendon Press.

Vanderwerf, E., Stiles, R., Warlen, A., Seibert, A., Bastien, K., Meyer, A., Nuxoll, A. M., & Wallace, S. (2016). Hash functions for episodic recognition and retrieval. *The Twenty-Ninth International Flairs Conference*.

Vere, S., & Bickmore, T. (1990). A basic agent. *Computational Intelligence*, *6*, 41–60. From `https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.1990.tb00128.x`.

Welch, G., Bishop, G., et al. (1995). *An introduction to the kalman filter*. Technical report, University of North Carolina, Chapel Hill, NC.

Yalnizyan-Carson, A., & Richards, B. A. (2022). Forgetting enhances episodic control with structured memories. *Frontiers in computational neuroscience*, (p. 24).

Zatuchna, Z. V., & Bagnall, A. (2009). Learning mazes with aliasing states: An lcs algorithm with associative perception. *Adaptive Behavior*, *17*, 28–57.