Preemptive Strategies for Overcoming the Forgetting of Goals

Justin Li and **John Laird**
University of Michigan
2260 Hayward Street
Ann Arbor, MI 48109-2121
{justinnh, laird}@umich.edu

Abstract

Maintaining and pursuing multiple goals over varying time scales is an important ability for artificial agents in many cognitive architectures. Goals that remain suspended for long periods, however, are prone to be forgotten. This paper presents a class of *preemptive strategies* that allow agents to selectively retain goals in memory and to recover forgotten goals. Preemptive strategies work by retrieving and rehearsing goals at triggers, which are either periodic or are predictive of the opportunity to act. Since cognitive architectures contain common hierarchies of memory systems and share similar forgetting mechanisms, these strategies work across multiple architectures. We evaluate their effectiveness in a simulated mobile robot controlled by Soar, and demonstrate how preemptive strategies can be adapted to different environments and agents.

Introduction

Although it is traditionally considered undesirable for agents to lose any information, the process of *forgetting* has recently attracted positive attention in the fields of both psychology and cognitive architecture. The rational analysis of memory suggests that the retention of knowledge should be predictive of the knowledge's usefulness (Anderson 1990). Forgetting is therefore critical to discarding unnecessary information. Indeed, forgetting plays a key role in reflecting the experience of the agent in heuristic judgments such as recognition and retrieval fluency (Schooler and Hertwig 2005). In artificial agents, forgetting is also a solution to the utility problem, where increased amounts of knowledge causes agent performance degradation due to search costs (Minton 1995). By selectively forgetting information, less resources are spent searching through irrelevant knowledge, thereby keeping memory and computational needs within manageable bounds (Derbinsky and Laird 2012).

With the acceptance of forgetting, however, there must also be strategies to prevent essential knowledge from being forgotten. One important type of knowledge is the goals that the agent has yet to accomplish. This is particularly problematic when the agent cannot immediately act on the goal: if the goal is forgotten, the agent does not simply fail to recognize an opportunity to act, but it fails to realize that it needs to act in the first place. For someone tasked with bringing milk home, this is the difference between failing to recognize that a nearby convenience store sells milk, and not realizing that milk must be bought. The challenge of *prospective memory* is to ensure that the goal is not forgotten when the opportunity to act arises.

The problem of prospective memory has been relatively ignored. Earlier work on opportunistic planning tackles a similar problem of resuming suspended goals (Simina and Kolodner 1995), although forgetting is not a focus and research in that field tends to treat goals as special structures. More recent work using cognitive architectures does not make such assumptions, and integrates a theory of forgetting, where knowledge can be removed from short-term memory while remaining available in the more stable long-term memory (Altmann and Trafton 2002). ACT-R (Anderson 2007) has been used to model the Intention Superiority Effect, a phenomenon where unfulfilled goals are easier to recall than fulfilled goals (Lebiere and Lee 2002). The architecture has also been used to model human reaction time data (Elio 2006), although neither of these models was tested in an agent in a complex domain. The closest precursor to this work describes how an agent might suspend and resume goals in the Tower of Hanoi puzzle (Altmann and Trafton 2002). Before the agent attempts a subgoal, it ensures that the original goal will only be partially forgotten; this allows the agent to recover the goal after the subgoal is completed. The retention of goals in the Tower of Hanoi is simplified, however, as the retention length is known through means-ends analysis. The agent's pre-programmed knowledge of the duration of a subgoal allows it to manipulate memory to facilitate later recall; this is rarely possible in less well-structured problems. Strategies for prospective memory have therefore remained untested in complex domains.

We strive to fill this gap in this paper. We present a class of *preemptive strategies* that identify key moments when the memories of goals need to be maintained, and provide two ways to overcome the forgetting of goals. This class subsumes previous work, and can take advantage of domain- and task-dependent features to improve performance. A sample of computational approaches within this class of strategies are implemented in the Soar cognitive architecture (Laird 2012) and evaluated in a simulated mobile robot. Within this

domain, we demonstrate that increasing the rate of forgetting or slowing down the temporal dynamics makes prospective memory more challenging, but that preemptive strategies can adapt to these changes.

Forgetting and Memory Hierarchies

The goal of cognitive architecture research is to discover the fundamental computational structures and processes that underlie intelligent behavior. This paper focuses on two architectures, ACT-R and Soar, as they are fully implemented, widely used, and share similarities in design. Both architectures create behavior through the matching of if-then *rules*: the conditions test knowledge in memory, which the actions transform or remove. More crucially, the two architectures incorporate short-term and long-term memories. These memories form a hierarchy, with the levels arranged by the stability, the influence on behavior, and the amount of knowledge contained. The *memory elements* stored in this hierarchy represent different kinds of knowledge, including the pending goals of the agent; it is the forgetting of these goals that give rise to the challenges of prospective memory.

At the top of the hierarchy is *short-term* or *working* memory. This memory contains the immediate perceptions of the agent, as well as knowledge that is relevant to the current situation. The knowledge in short-term memory directly determines the behavior of the agent, as it is the only knowledge against which rules are matched. Since the size of short-term memory is a major factor in the cost of rule-matching (Forgy 1979), it is usually kept small; an architectural limit may be placed on its size (as in ACT-R), or an architectural process may remove memory elements over time (as in Soar). Both of these mechanisms can be considered "forgetting", as potentially useful knowledge is lost.

At the next level of the memory hierarchy is one or more *long-term* memories. This level of the hierarchy contains knowledge that may be useful to the agent over its lifetime, but not necessarily at the present; examples include facts about the domain and the previous experiences of the agent. Due to its potential size, rules do not match against knowledge in long-term memory. Instead, long-term knowledge is accessed through deliberate cued-*retrievals*, where the results are deposited in specialized *buffers* in working memory. Although knowledge in long-term memory does not directly impact rule-matching costs, it may still be forgotten; whether a particular memory element is lost is often a function of the agent's previous access to that knowledge.

The last level in the "memory" hierarchy is the environment, which we include for completeness. It is considered a level in the hierarchy because knowledge that is lost from long-term memory may be recoverable from the environment. Since the environment is external to the agent, access to knowledge is extremely slow as compared to other memories. In this work, we do not consider how the agent can obtain knowledge from the environment to combat forgetting.

Although other architectures may not fully implement this memory hierarchy, a small subset of abilities is sufficient for the rest of this paper to apply. In particular, the architecture must have a working memory of limited size (eg. CHREST, Polyscheme) or that is subject to forgetting (eg. CLARION, LIDA), and must support directed memory retrievals (all of the above) (Gobet and Lane 2010; Cassimatis et al. 2010; Sun 2006; Snaider, McCall, and Franklin 2011). These two capabilities are sufficient to implement some of the strategies in this work, and many of the effects apply. The rest of this paper focuses solely on ACT-R and Soar.

$$activation = \ln(\sum_{i=1}^{n} {t_i}^{-d})$$

Figure 1: The equation for base-level activation. n is the number of activation boosts, t_i is the time since the i^{th} boost occurred, and d is a free decay-rate parameter.

Forgetting in ACT-R and Soar

We now describe how ACT-R and Soar implement this memory hierarchy, and how forgetting affects the knowledge in each architecture.

In ACT-R, knowledge is represented as *chunks*. Chunks are stored and processed by *modules*, which perform specialized operations such as perception, motor actions, and memory access. These modules are only accessible through their buffers, each of which contains at most one chunk. The buffers of all the modules together act as ACT-R's working memory, and they contain the only knowledge that rules can match against and directly modify. Since the size and the number of buffers is fixed, ACT-R's working memory has limited capacity; in order to access a module, the current contents of its buffer must be replaced. To overcome this extremely rapid forgetting of information from working memory, all chunks that have existed in various buffers are automatically stored into the *declarative memory* (DM) module. This module acts as ACT-R's long-term memory, allowing knowledge no longer available in working memory to be recovered. To retrieve knowledge from declarative memory, the agent must query DM with a partial description of the desired knowledge; knowledge that best matches the description will be recreated in the buffer. A chunk in declarative memory may also be forgotten, based on its *base-level activation*, which summarizes its access history (Figure 1). Activation decreases exponentially over time, a process called *decay*, but is increased (*boosted*) whenever the chunk appears in a buffer, either due to retrieval or due to perception from the environment. If the activation of a chunk falls below a threshold, the chunk becomes unretrievable, and can only be boosted again through perception. Forgetting in ACT-R therefore occurs on two levels: knowledge may be lost in working memory due to replacement by other knowledge, and knowledge may be lost in declarative memory due to infrequent access.

In Soar, working memory is a graph structure with special locations for perception and motor control, as well as for accessing the long-term memories. *Episodic memory* stores the previous experiences of the agent, while *semantic memory* stores knowledge and facts about the world. We focus on

semantic memory in this paper, due to its similarity with ACT-R's declarative memory. As with its ACT-R counterpart, the agent can only access semantic memory through description queries, again with the memory element that best matches the description being returned. Only knowledge that originates from long-term memory is subject to forgetting from working memory, as it is guaranteed to be recoverable. This forgetting is also based on base-level activation, where activation is boosted by retrievals from semantic memory and by the matching of rules. Within semantic memory, knowledge experiences no dynamics, and will not be forgotten. There is therefore only one type of forgetting in Soar: the loss of knowledge from working memory, where it is guaranteed to be recoverable from semantic memory.

From these descriptions, it is clear that the memory systems of ACT-R and Soar share many similarities. Both architectures allow agents to boost the activation of a memory element to length the time until it is forgotten; this may intuitively be called *rehearsing*. Alternately, if knowledge is lost from working memory, it can be retrieved from long-term memory, allowing the agent to recover from forgetting. These two observations form the basic building blocks to overcoming forgetting.

A final observation is that both architectures treat suspended goals no differently from other memory elements, and are therefore subject to forgetting. Aside from being consistent with psychological theories (Anderson and Douglass 2001), the forgetting of goals also offer functional benefits. In ACT-R, the limited size of working memory does not leave space for non-active goals; in Soar, the goals are forgotten to avoid increased rule-matching costs due to a large working memory. Thus far, ACT-R and Soar agents have mostly dealt with domains with hierarchical goals, where problem solving determines the next goal to achieve. In less-structured domains, more elaborate strategies are required as suspended goals are removed from working memory.

Preemptive Strategies

The management of suspended goals is called prospective memory. Previous work has identified five stages of this process (Ellis 1996):

encoding The goal is stored into the long-term memory of the agent

retention The agent waits for an opportunity to act on the goal.

initiation An opportunity arises, and the agent must recognize that the goal is applicable

performance The agent acts to complete the goal.

completion The goal is marked as completed, such that the agent will not continue to pursue the goal.

The problem of forgetting most severely effects the initiation stage of prospective memory, which is the sole focus on this work.

In order for the agent to recognize that a goal is applicable during the initiation stage, the goal must be in working memory such that rules can identify the opportunity. If the goal is forgotten during the retention interval, it must be deliberately retrieved from long-term memory — except that for the agent to have initiated a retrieval, it must have recognized that the goal is applicable in the first place. This chicken-and-egg problem is solved by humans in multiple ways (McDaniel and Einstein 2000). For example, the goal may be retrieved from long-term memory spontaneously, without deliberate effort by the agent; this may occur because the goal is somehow associated with the agent's perceptions or reasoning. Another solution involves metamemory judgments, which signal to the agent that something of significance has just been perceived. The agent may then decide to search its memory for the source of this significance, and find that an opportunity to act on a goal has arisen. Both these solutions, however, may require additional memories and mechanisms than those discussed; metamemory judgments, in particular, remain largely unexplored in cognitive architectures (Li, Derbinsky, and Laird 2012). Modifying the architecture to enable these strategies has impact beyond the forgetting of goals; we leave the exploration of these strategies for future work.

Instead, we take inspiration from a third strategy that humans use, and which can be implemented without architectural changes. There is psychological evidence that people tend to recall goals when switching between task contexts, such as when walking down a hallway to a meeting (Sellen et al. 1997). If a goal is retrieved and an opportunity to act is judged to be imminent, it is kept in mind and *monitored* — that is, attentional and memory resources are set aside to continually checked for opportunities to fulfill the goal (Harris 1984). A slight variation of this strategy is to regularly retrieve and check the goal within a time period. Both strategies can be adapted for artificial agents; by temporally dissociating the retrieval of the goal from the opportunity to act, one of the dependence links of the causal loop is removed.

The two variations above share a common framework: that of using a goal-independent *trigger* — such as a periodic interval or a context switch — to manage the problem of forgetting goals. Since this class of strategies attempts to ensure that the goal is in working memory prior to the opportunity to act, we call these *preemptive strategies* for prospective memory. In addition to the type of trigger, this class of strategies can be further divided by the method used to retain a goal in memory. A *proactive* approach attempts to prevent goals from being forgotten by rehearsing each goal to boost its activation. Together with a domain-dependent trigger of subgoaling, this is the strategy used in the Tower of Hanoi work (Altmann and Trafton 2002). On the other hand, a *reactive* approach retrieves goals back into memory after forgetting has taken place. These approaches are not mutually exclusive; an agent could first retrieve goals that have been forgotten, then proceed to rehearse them to further boost their activation. Categorizing strategies by this distinction aligns them with the capabilities of the memory systems of cognitive architectures. A proactive approach applies to ACT-R's long-term memory and Soar's working memory, where base-level activation allows boosting. A reactive approach, only the other hand, only applies to the working memories of both architectures, where forgotten knowledge can be recovered

from long-term memory.

To make this concrete, an agent using preemptive strategies has a goal-independent rule which encodes the trigger as a condition. When the rule matches, the agent performs either proactive or reactive maintenance on its goals. In the proactive case, the agent boosts the activation of any uncompleted goal; in the reactive case, the agent retrieves any uncompleted goal from long-term memory. In both cases, the agent then completes any matching goals in working memory. The remaining, uncompleted goals then decay, and are potentially forgotten, until the trigger is next encountered.

A preliminary assessment of preemptive strategies suggests a remaining difficulty: that of scaling to large numbers of goals. In the psychological account of preemptive strategies, which goal to retrieve or rehearse depends on how soon the agent judges it to be relevant (Sellen et al. 1997). This, however, would require the agent to predict its future perceptions — a capability not well studied in either ACT-R or Soar. To sidestep this issue, in this work the agent retrieves all of its unfulfilled goals at the specified trigger. This temporary solution will fail with sufficiently many goals, as the first goal to be retrieved could *already have been forgotten* by the time the last goal is retrieved. Although this is not, strictly speaking, a limitation of preemptive strategies, it does constitute a restriction on the generality of this approach. The severity of this restriction is part of our evaluation of preemptive strategies below.

Empirical Evaluation

We ask three questions of preemptive strategies:

scaling Under the current approach of retrieving all deliveries at the trigger, what is the maximum number of deliveries?

rehearsal performance How do rehearsal strategies perform under different conditions?

retrieval performance How do retrieval strategies perform under different conditions?

To keep this evaluation broadly applicable, only strategies with a single trigger and a single action (either retrieval or rehearsal) are used.

We implemented the preemptive strategies in a Soar agent that controls a simulated SuperDroid robot in an indoor environment (Figure 2), which is divided into rooms. A room is a rectangular area that the agent can traverse; irregularly shaped areas, such as the one at the top of Figure 2, are divided into multiple rooms. Some rooms are separated by doorways, which constrict the agent's path. Scattered around the environment are objects of different colors, shapes, and sizes; these can be picked up and put down by the agent. The agent can carry multiple objects simultaneously, but can only perceive the objects within its field of view in the current room.

The goal of the agent is to pick up and deliver objects to other rooms, while being constrained to a predefined "patrol" route (ie. it must visit the room in a particular order). A particular instantiation of this task includes randomly generated object descriptions and locations, as well as the

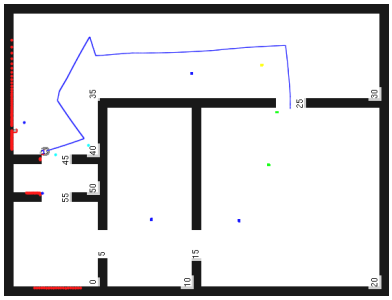

Figure 2: A visualization of the simulated robot domain. Objects, represented as dots, are scattered around the environment. The line traces the robot as it performs deliveries.

deliveries that the agent must make. A delivery is determined by the description of the object and the room to which the object should be delivered; in particular, the agent does not know the location of objects to be picked up. Before a run, the deliveries are pre-loaded into the long-term memory of the agent, and must be retrieved into working-memory. Since the deliveries originate from Soar's long-term memory, they are subject to decay while in working memory and can be forgotten. In order to make a successful delivery, the agent must recognize that an object in the room needs to be picked up, or that the current room is the destination for a delivery. Although only one goal type is used in this evaluation, the strategies can be trivially generalized to manage multiple goal types. This task allows the preemptive strategies to be tested on a complex domain.

The preemptive strategy triggers must be adapted to the robot environment. The timing trigger uses motor commands as the unit of time; an interval of 400, for example, means that the memory action is taken every 400 steps or turns of the agent. For the domain-dependent trigger, psychology and prior work suggests that context-switches would be good cues for the retrieval and rehearsal of goals. Since the agent can only perceive objects within its current room, we use the entry into a room as the trigger. Note that for this task, it is also possible to use the perception of objects as a trigger, both for retrieval/rehearsal and for searching for relevant deliveries. There are, however, two disadvantages to this trigger. First, although the trigger signals the agent that an object must be picked up, it does not signal when an object should be put down at its destination. Second, the description of an object for delivery may not be complete — a delivery may only specify the color of an object, allowing objects of any shape to fulfill the requirement. Since neither ACT-R's declarative memory nor Soar's semantic memory fully supports partial matches, especially for symbolic values like shapes, it is difficult and expensive to search through the powerset of an object's features. For these reasons, we focus only on the timing and room-based triggers.

There are two main parameters to this domain. The first parameter is the decay rate (d in Figure 1), which determines how quickly a memory element is forgotten. The decay rate is a real number between 0 and 1, exclusive: the larger the decay rate, the more quickly forgetting occurs. The second parameter is the speed of the robot, which determines how

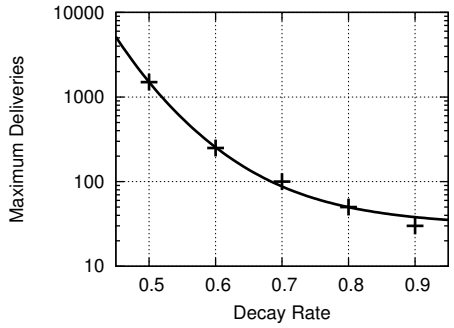

Figure 3: Number of deliveries an agent can retrieve before the first delivery is forgotten. $y = 0.0196 * e^{5.61/d} + 28.0$, where d is the decay rate. Coefficient of determination $R^2 = 0.99$.

quickly it moves forward in the simulated environment. The slower the robot's movement, the longer the goal must remain in working memory and not be forgotten. Varying the speed of the agent therefore changes the temporal dynamics of the domain, to which the strategies must adapt.

For the robot delivery task, the main measurement of performance is the percentage of deliveries that the agent successfully completes. Unless the agent completes all of its deliveries, it is stopped after the third round of patrol. Additionally, we also measure the efficiency of the agent: the ideal agent should make only the retrievals and rehearsals necessary for the task and no more. Agents that complete the same number of deliveries with less time spent on memory management are preferred.

Results

Scaling

To determine the maximum number of retrievable deliveries, agents with different decay rates are given increasing number of deliveries to retrieve, until the first delivery decays sufficiently to be removed from memory. The empirical results (Figure 3) suggest that although such a limit exists, the limit increases exponentially as the decay rate decreases. This relationship is derived from the base-level activation equation, and applies to both ACT-R and Soar. After the activation of a delivery is initially boosted by its retrieval, it simply decays while other deliveries are retrieved. Over different decay rates, this results in the exponential curve shown.

Note that although a similar limit exists for rehearsals, the equivalent question is less well-defined. The number of deliveries that could be rehearsed until the first delivery is forgotten — or until the first delivery returns to its original level of activation — depends not only on the decay rate and the amount of rehearsal, but also on the level of activation prior to rehearsal. The limit is therefore also dependent on when the rehearsal is performed, as the activation increases and decreases over the lifetime of the agent. Furthermore, since neither decay rate nor activation information is declaratively available, the agent cannot dynamically adapt the number of rehearsals. There is therefore little practical benefit in computing a limit on rehearsals.

Decay	**Timing**			**Room Entry**		
Rate	4	16	64	4	16	64
0.34	100	100	100	100	100	100
0.38	100	100	100	76.36	80.0	94.54
0.42	56.36	60.0	41.81	7.27	5.45	5.45
0.46	3.63	3.63	7.27	0.0	0.0	0.0
0.50	0.0	0.0	0.0	0.0	0.0	0.0

Table 1: Percentage of deliveries completed by agents with different decay rates. The second heading is the number of rehearsals under different triggers. The agent speed was set to 0.7; the timing trigger fired every 400 steps.

Rehearsal Performance

Although preemptive strategies can be separated into proactive and reactive approaches, it may be sufficient to only take the proactive approach. Within the robot delivery task, this is equivalent to rehearsing deliveries such that they are never forgotten, thus rendering retrievals unnecessary. This allows agents to selectively avoid forgetting memory elements, and is well suited to memories where forgotten knowledge is not easily recovered — for example, ACT-R's declarative memory.

Table 1 shows the performance of the agent at different decay rates, with different numbers of rehearsals at the trigger. For the timing trigger, these results suggest that with sufficient rehearsals, the agent is able to prevent deliveries from being forgotten. This result is due to the high correlation between the trigger and the passage of time, which ultimately causes the decay of activation. As long as the timing unit is correlated with the unit of activation decay — which is true of the motor outputs in this domain — a timing trigger can be used to selective prevent memory elements from being forgotten. This may, however, consume an unreasonable amount of effort: for an agent to rehearse 64 times (at decay rate 0.3), over 9% of the agent's time is spent in preventing goals from being forgotten. To avoid excessive rehearsals, then, the number of rehearsals should be matched to the decay rate.

The need to adjust the decay rate and number of rehearsals is more apparent when the effects of decay rate is considered while holding the amount of rehearsals constant. The results show that small increases in the decay rate causes a sharp drop-off in performance. The cause of this behavior is discussed in the next section, but it suggests that the amount of rehearsal should be determined by the agent designer.

The domain-dependent trigger also performs well below a particular decay rate; however, the decay rate is not the only factor influencing its performance. Table 2 shows the result of the same experiment on agents with different speeds, while keeping the decay rate constant at 0.4. While the timing trigger performs equally well across different agent velocities, the performance of the domain trigger decreases with the speed of the agent. Due to the agent's slower movement, there is increased time between the rehearsal of a delivery and the perception of the object, allowing the delivery to

Speed	Timing			Room Entry		
	4	16	64	4	16	64
0.3	90.9	90.9	90.9	18.1	18.1	22.7
0.5	95.4	81.8	95.4	22.7	22.7	13.6
0.7	92.7	92.7	92.7	56.3	36.3	34.5
0.9	87.2	92.7	92.7	40.0	36.3	34.5
1.1	92.7	98.1	90.9	52.7	36.3	23.6

Table 2: Percentage of deliveries completed by agents with different speeds. The second heading is the number of rehearsals under different triggers. The decay rate was set to 0.4; the timing trigger fired every 400 steps.

be forgotten while the agent is (for example) completing a different delivery.

Taken together, these two results suggest that while it is possible to retain goals in memory through rehearsals alone, this requires knowledge of both the agent's forgetting mechanism as well as the temporal dynamics of the domain; there is no magical amount of rehearsal and trigger that will work across all domains and decay rates. For timing triggers, task performance and resource consumption may be difficult to balance without experimentation, as it partially depends on how often perceptions boost the activation of goals. Domain triggers are also dependent on the domain, not only for the features that determine the trigger, but also for the time between that feature and the opportunity to act.

Retrieval Performance

We now focus on the retrieval of already forgotten elements, which applies to ACT-R's working memory, where rehearsal cannot prevent information loss.

The performance of agents with different decay rates using retrieval strategies are shown in Figure 4. Two differences between these results and those for rehearsals strategies require explanation: first, retrieval strategies do not perform as well as rehearsal strategies, even at low decay rates; and second, retrieval strategies allow the agent to perform deliveries at a greater range of decay rates. The two behaviors have the same underlying cause as the sharp decline seen in rehearsal strategies: that rehearsal strategies can either maintain deliveries indefinitely if the decay rate is sufficiently low, or deliveries are lost relatively quickly above the threshold. If the amount of rehearsal is not sufficient to retain the deliveries in memory, all deliveries will be eventually forgotten. Since the rehearsal agents do not retrieve forgotten deliveries in this evaluation, they fail to complete any more deliveries — hence the sharp drop-off in performance for the rehearsal agents. In contrast, the decay rate does not need to be as carefully calibrated for retrieval strategies, as the agent simply recovers when a delivery is forgotten. Thus, although deliveries may be forgotten earlier due to the lack of rehearsals, this does not effect agent task performance until it reaches extreme levels. For the same reason, agent performance remains stable across velocities: even if a delivery is forgotten after its retrieval, the agent has a second opportunity to complete the delivery when it enters the room again.

Examination of the results also show that while the timing

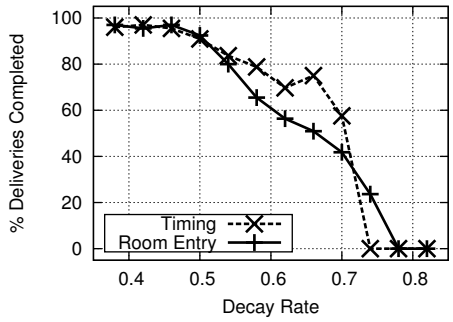

Figure 4: Percentage of deliveries completed by retrieval agents with different decay rates and triggers. The agent speed was set to 0.7; the timing trigger fired every 400 steps.

trigger outperforms the room entry trigger, it is performing more retrievals. At decay rate 0.54, the timing trigger requires 1.5 times as many retrievals per delivery completed than the room entry trigger (19.3 vs. 12.3), and this ratio grows as the decay rate is increased. The efficiency of the room entry trigger is due to it being highly predictive of opportunity; many of the retrievals due to timing triggers do not lead to action on the part of the agent. This may occur if deliveries are retrieved when the agent does not perceive any objects, or when all deliveries have already been made for the room. With the room entry trigger, the embedded domain knowledge makes it more likely that an opportunity to act is present. The high probability of a relevant retrieval allows the agent to reduce the number of retrievals without comprising its performance.

Conclusion

In this paper, we introduced a class of preemptive strategies to overcome the forgetting of goals. These strategies do not require architectural modification, and in their simplest form can be implemented by any forgetful architecture with memory retrievals. These strategies were tested in a Soar agent in a simulated robot domain, showing that they perform well when the decay rate is low and when the trigger accurately predicts opportunities to act, and can continue to operate as these ideal conditions degrade. While this behavior is tied to the mechanism of forgetting, the qualitative behavior holds for any architecture where the likelihood of forgetting increases over time. As implemented, these strategies are sensitive to both the forgetting mechanism and the domain, as they require parameters in the form of environmental context-switch tiggers and timing triggers that match the forgetting mechanism. Further research may allow agents to dynamically select between triggers and adapt them to the environment.

Acknowledgments

The authors acknowledge the funding support of the Office of Naval Research under grant number N00014-08-1-0099.

References

Altmann, E. M., and Trafton, J. G. 2002. Memory for goals: An activation-based model. *Cognitive Science* 26(1):39–83.

Anderson, J. R., and Douglass, S. 2001. Tower of hanoi: Evidence for the cost of goal retrieval. *Journal of Experimental Psychology: Learning, Memory, and Cognition* 27(6):1331–1346.

Anderson, J. R. 1990. *The Adaptive Character of Thought*. Psychology Press.

Anderson, J. R. 2007. *How Can the Human Mind Occur in the Physical Universe?* New York, NY: Oxford University Press.

Cassimatis, N. L.; Bignoli, P. G.; Bugajska, M. D.; Dugas, S.; Kurup, U.; Murugesan, A.; and Bello, P. 2010. An architecture for adaptive algorithmic hybrids. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 40(3):903–914.

Derbinsky, N., and Laird, J. E. 2012. Competence-preserving retention of learned knowledge in Soar's working and procedural memories. In *Proceedings of the 11th International Conference on Cognitive Modeling (ICCM)*.

Elio, R. 2006. On modeling intentions for prospective memory performance. In *Proceedings of the 28th Annual Conference of the Cognitive Science Society (CogSci)*, 1269–1274.

Ellis, J. 1996. Prospective memory or the realization of delayed intentions: A conceptual framework for research. In Brandimonte, M.; Einstein, G. O.; and McDaniel, M. A., eds., *Prospective Memory: Theory and Applications*. Mahwah, New Jersey: Lawrence Erlbaum.

Forgy, C. L. 1979. *On the Efficient Implementation of Production Systems*. Ph.D. Dissertation, Carnegie Mellon University, Pittsburgh, PA.

Gobet, F., and Lane, P. 2010. The CHREST architecture of cognition the role of perception in general intelligence. In *Proceedings of the 3rd Conference on Artificial General Intelligence (AGI)*.

Harris, J. E. 1984. Remembering to do things: A forgotten topic. In Harris, J. E., and Morris, P. E., eds., *Everyday Memory, Actions and Absent-mindedness*. Academic Press. 71–92.

Laird, J. E. 2012. *The Soar Cognitive Architecture*. Cambridge, MA: MIT Press.

Lebiere, C., and Lee, F. J. 2002. Intention superiority effect: A context-switching account. *Cognitive Systems Research* 3(1):57–65.

Li, J.; Derbinsky, N.; and Laird, J. E. 2012. Functional interactions between memory and recognition judgments. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI)*, 228–234.

McDaniel, M. A., and Einstein, G. O. 2000. Strategic and automatic processes in prospective memory retrieval: A multiprocess framework. *Applied Cognitive Psychology* 14(7):S127–S144.

Minton, S. 1995. Quantitative results concerning the utility of explanation-based learning. In Ram, A., and Leake, D. B., eds., *Goal-Driven Learning*. MIT Press.

Schooler, L. J., and Hertwig, R. 2005. How forgetting aids heuristic inference. *Psychological Review* 112(3):610–628.

Sellen, A. J.; Louie, G.; Harris, J. E.; and Wilkins, A. J. 1997. What brings intentions to mind? An *in situ* study of prospective memory. *Memory* 5:483–507.

Simina, M. D., and Kolodner, J. L. 1995. Opportunistic reasoning: A design perspective. In *Proceedings of the 17th Annual Conference of the Cognitive Science Society (CogSci)*, 78–83.

Snaider, J.; McCall, R.; and Franklin, S. 2011. The LIDA framework as a general tool for AGI. In *Proceedings of the 4th Conference on Artificial General Intelligence (AGI)*.

Sun, R. 2006. The CLARION cognitive architecture: Extending cognitive modeling to social simulation. In Sun, R., ed., *Cognition and Multi-Agent Interaction: From Cognitive Modeling to Social Simulation*. Cambridge University Press.