A Biologically Inspired Working Memory Framework for Robots

Joshua L. Phillips & David C. Noelle
({joshua.l.phillips,david.noelle}@vanderbilt.edu)
Department of Electrical Engineering and Computer Science
Vanderbilt University

Abstract

This work focuses on a particular neurocomputational account of working memory function that has been used to explain a wide range of working memory phenomena in terms of interactions between the prefrontal cortex and the mesolimbic dopamine system. Using the mechanisms prescribed by this theory, we have constructed a software toolkit for creating working memory modules for use in robotic control systems. The challenges faced by embodied robots are similar to those experienced by humans in everyday living, making this domain useful for testing the utility and scalability of this computational theory of working memory. We report the results of a feasibility study, involving a robotic version of the *delayed saccade task*, and we discuss future plans to test our working memory model in the context of robot control and learning.

Introduction

Research in the cognitive sciences has provided substantial insight into the nature of working memory and the biological mechanisms that produce it. Psychological studies have provided evidence of capacity limitations for working memory and have shown that working memory contents are immediately available for executive and deliberative processing. Electrophysiological and neuroimaging studies have implicated regions of prefrontal cortex (PFC) as being central to working memory function. These experimental findings have led to the development of biologically-based theories of working memory, and some of these theories have been explicitly instantiated in computational neuroscience models.

In the work reported here, we have focused on one particular computational account of working memory function, and we have endeavored to evaluate the ability of this account to scale to the challenges faced by human working memory systems every day. This working memory model has been previously used to explain a variety of laboratory findings, but it is uncertain whether this model is capable of addressing real-world contingencies (O'Reilly et al., 1999). Our approach to assessing the power of our working memory model is somewhat unorthodox. We have begun the process of embedding this working memory mechanism into robotic control systems, with the goal of using robotic platforms as challenging testbeds for our computational theory of working memory function. We believe that the study of working memory in robots can provide deeper insights into working memory function in humans, because the difficulties that arise when robots interact with the world are representative of the tasks that humans encounter daily. Thus, robotic tasks provide a rich and effective domain for testing the scalability of our working memory model.

This paper reviews important properties of human working memory processes, as well as our computational approach to modeling those processes. The benefits that such a working memory system might bring to robot control are briefly discussed. We then introduce an open source software library, called the *Working Memory Toolkit*, which provides an abstraction of our previous computational neuroscience models of prefrontal cortex. Through a simulation study, we show that this toolkit offers sufficient working memory functionality to learn a behavior commonly used to examine the working memory abilities of non-human primates: the *delayed saccade task*. The paper closes with a discussion of the more elaborate robot navigation and object manipulation tasks that are currently being implemented using our biologically inspired working memory system.

Background

The Nature of Working Memory

Working memory is what allows you to keep a particular book's call number in mind while you search for it in the library. Working memory is what allows you to remember what phone number the operator told you, just long enough to dial it. Working memory is what allows you to remember information that is critical to correct decision making in the current situation, but is then discarded once it has served its purpose. Due to its involvement in so many aspects of mental function, working memory is a central component of almost all theories of human cognition. Working memory is often described in terms of its storage, manipulation, updating, and executive processing components (Baddeley and Hitch, 1974). Perhaps more succinctly, working memory has been described as a system that stores a small number of "chunks" of information, protecting them from interference from other processing systems and positioning them so as to directly influence the generation of behavior (Goldman-Rakic, 1987).

While many theories of working memory exist (Miyake and Shah, 1999), they tend to agree on several key properties. One such property is the limited capacity of the working memory system. Recent estimates of this ca-

pacity suggest that the number of "chunks" that can be stored and used by working memory is approximately four (Cowan, 2001). Note that this is somewhat lower than earlier estimates which suggested a capacity of "seven plus-or-minus two" items (Miller, 1956). A second key property of working memory is that its contents are readily accessible to other cognitive processes. This property, combined with the first, suggests that working memory may be adapted to retain only the information that is most important for influencing behavior. A related key property is the volatility of working memory contents. The constituents of working memory may be updated and manipulated very quickly. Thus, working memory can allow new information to influence behavior much faster than other memory systems, which change at a slower rate (Waugh and Norman, 1965).

There is substantial evidence that regions of prefrontal cortex (PFC) play an important role in working memory (Goldman-Rakic, 1987). Neurons in this brain region have been found to actively maintain high firing rates in the absence of stimuli, encoding relevant bits of information during delay periods. Many different kinds of information appear to be actively maintained in the PFC, including spatial locations (Funahashi et al., 1989), recently viewed objects (Cohen et al., 1994; Miller and Desimone, 1994), action rules (Wallis et al., 2001), and even verbal information (Demb et al., 1995). Dense recurrent connections in PFC are thought to support active maintenance of high firing rates through mutual excitation (Camperi and Wang, 1998).

Recurrent excitation is not a sufficiently flexible mechanism to account for the fluidity of working memory function. In some situations, working memory contents must be actively maintained in the face of distractions. In other situations, contents must be rapidly updated, discarding old contents in favor of new contents. In order to account for working memory performance, some intelligent mechanism must be placed in control of memory updating. But how does the brain know what information should be retained and what can be safely discarded?

This issue of intelligent updating is the focus of our computational model of working memory. Our model asserts that working memory is *adaptive* in the sense that proper control of updating is *learned* from experience. If the retention of a particular kind of informational chunk in a given situation results in reward, the system will be more likely to retain similar chunks in similar situations. The question then becomes one of how such a reinforcement learning scheme is implemented in the brain.

One candidate for a neural substrate for reinforcement learning involves the mesolimbic dopamine system. Recordings of dopamine cell firing in awake behaving animals suggest that dopamine cells fire in response to changes in expected future reward (Shultz et al., 1997). Interestingly, such a measure of change in expected future reward is a key component of a machine learning algorithm known as *temporal difference (TD) learning* (Sutton, 1988). This has led researchers to construct computational models of neural reinforcement learning, grounded in interactions between dopamine neurons and circuits in other brain areas, such as the striatum (Barto, 1994; Montague et al., 1996). These models have been able to account for biological and behavioral findings associated with conditioning and motor sequence learning.

It is important to note that midbrain dopamine neurons also project broadly to the PFC, as well as to loop-like circuits between PFC and the basal ganglia, mediated by the thalamus. Thus, the midbrain dopamine system is not only well positioned to assist in the learning of overt motor actions, but it may also contribute to the learning of the appropriate timing for covert actions, such as working memory updating (Braver and Cohen, 2000). This is the basis of the working memory model explored here. A temporal difference learning algorithm, implemented, in part, by the midbrain dopamine system, learns to identify situations in which working memory contents should be actively maintained and situations in which working memory contents should be rapidly updated. In this way, the working memory system adapts to the reward contingencies of the organism's environment. This model has been successfully used to account for a variety of working memory phenomena, including deficits seen in schizophrenia and under focal frontal lesions (Braver and Cohen, 2000; O'Reilly et al., 2002).

Temporal Difference Learning

The temporal difference (TD) learning algorithm (Sutton, 1988) is a powerful method for learning to select actions based on reinforcement signals: sporadic, scalar measures of how "good" or "bad" the current situation is. The algorithm uses these sparse measures of performance to adjust behavior over time. The central component of this algorithm is an estimator of future reward, called the *adaptive critic*. The adaptive critic is commonly a simple artificial neural network that takes information about the current state of the animal and maps it onto an estimate of how good or bad the current situation is. Importantly, this mapping is not fixed, but is learned through experience.

Every situation is assumed to be immediately evaluated by the animal, assigning the situation with some scalar amount of "reward", labeled $r(s)$ for situation s. This scalar identifies things that are inherently good (e.g., food) with positive values, inherently bad (e.g., pain) with negative values, and neutral situations with a value of zero. The goal of the adaptive critic is *not* to estimate this value, however, but to estimate the *value function*, $V(s)$, of the situation in terms of the likely reward to be received in the future. In other words, a situation's worth is not measured entirely by the amount of reward we receive at that instant. For example, when playing chess it is sometimes desirable to sacrifice one of your pieces (a pawn) in order to win the game. The adaptive critic is to estimate expected future reward. If $(s+1)$ is the situation that follows situation s in time, this expected future reward may be formalized as:

$$V(s) \;=\; \gamma^0 r(s) + \gamma^1 r(s+1) + \ldots + \gamma^n r(s+n)$$

The value of the current situation, $V(s)$, is the sum of

all of the rewards we will receive over the next n time steps. The rewards on each time step are "discounted" by a factor, γ, in the range $[0, 1]$. This discounting factor makes rewards that occur in the near future more "valuable" than those that occur much later. This equation may be rewritten in a recursive form:

$$V(s) \;=\; \gamma^0 r(s) + \gamma^1 V(s+1) \;=\; r(s) + \gamma V(s+1)$$

Any estimate of the value function that deviates from this equality is inaccurate, and the magnitude of the inaccuracy is captured by the *temporal difference error*:

$$\delta(s) \;=\; (r(s) + \gamma V(s+1)) - V(s)$$

The TD learning algorithm incrementally updates the adaptive critic's estimate of $V(s)$ in proportion to $\delta(s)$, increasing the estimate if $\delta(s)$ is positive and decreasing the estimate if $\delta(s)$ is negative.

In order for the adaptive critic to make value function estimates for novel states, its estimate is computed as a parameterized function of features of the current situation. A common parameterization is an affine transformation of situation features. Thus, if the situation, s, is encoded as a vector of real valued features, s_i, the adaptive critic will estimate the value function as:

$$V(s) \;=\; w_0 \;+\; \sum_{i=1}^{n} w_i s_i$$

Thus, the adaptive critic may be implemented by a single linear connectionist processing element. In order to modify value function estimates according to the temporal difference error, weights are modified as follows:

$$\Delta w_i \;=\; \alpha\, \delta(s)\, s_i \qquad\qquad \Delta w_0 \;=\; \alpha\, \delta(s)$$

. . . where α is a learning rate parameter. Many implementations of TD learning use a technique called *absorbing reward*, in which $V(s+1)$ is forced to zero when s is a "goal" situation, marking the end of an episode or trial. Often the gradient with respect to each weight is carried over from one time step to the next, but exponentially discounted according to a parameter λ which is in the range $[0-1]$. This learning algorithm produces adaptive critics that generate good estimates of expected future reward in various situations.

Given a good estimate of the value of situations, the learning algorithm can choose actions that lead to situations of high value. In many TD learning systems, this action selection process is performed by a separate component, called the *actor* (Barto, 1994), but such a component is not needed if the situations resulting from actions can be reliably simulated. In such a case, the adaptive critic is used to estimate the values of all of the situations that can be immediately reached from the current situation, and the action that leads to the highest value situation is taken. This is the strategy taken by our adaptive working memory system, where the different situations considered involve different collections of chunks actively retained in working memory.

Methods

Our computational model of working memory updating, grounded in interactions between PFC and the dopamine system and implemented as a neural network version of TD learning, has been found to match the performance of humans and non-human primates on a variety of laboratory tasks. We hope to demonstrate the utility of such an adaptive working memory system in much more complex task domains that reflect the constraints of everyday cognition. We believe that such a demonstration might be had by integrating our working memory model into the control systems of autonomous robots, using the embedded working memory system to assist in the performance of such tasks as visual search in cluttered environments, tracking of moving and transiently occluded target objects, retention and tracking of objects that make for good landmarks for navigation, and localization of occluded objects during tool manipulation tasks. We are currently in the process of constructing such systems for both mobile robots and a stationary humanoid robot.

Contemporary robot control systems are not conducive to direct integration with computational neuroscience models. Computational neuroscience simulation software typically does not respond in real time, and the interfaces expected by robot control systems typically do not deal in the currency of neural firing rates. Thus, in order to facilitate integration, we have generated an abstraction of our working memory model in the form of an open source software library that may be embedded in robot control software. As an initial demonstration of the functionality of this library, we have simulated a robotic version of a common neuroscientific test of spatial working memory: the *delayed saccade task*.

The Working Memory Toolkit

We have developed a set of software tools for developing working memory systems that can be easily and tightly integrated into robotic control mechanisms. This set of tools, called the *Working Memory Toolkit (WMtk)*, is a software library which is general and flexible enough to be used on a variety of robotic platforms. The toolkit is written in ANSI C++ and consists of a set of classes and methods for constructing a working memory system that uses TD learning to select working memory contents.

When using the WMtk, the first step in building a robotic working memory system involves the creation of a `WorkingMemory` object. This object is configured to hold some limited number of chunks, with the capacity specified by the designer. There is no limitation on what kind of information may be grouped into a chunk. Chunks are not restricted to a particular data type, and the `WorkingMemory` object simply maintains untyped pointers to the chunks stored within it. When the robot encounters a new situation, its control systems are expected to generate a list of candidate chunks. For example, object recognition systems may detect the presence of a salient object, producing candidate chunks for the existence of the object, its location, and other relevant properties. Control systems may also produce can-

didate chunks that correspond to actions or goals, such as a desire to grasp a particular object. Importantly, candidate chunks are not automatically stored in working memory. Instead, the list of candidates is passed to the `WorkingMemory` object, and the object then uses the TD learning algorithm to learn which chunks to retain and which to discard.

In order to evaluate the retention of a chunk, the adaptive critic needs to be input real valued features of the chunk that may be predictive of task success and, thus, future reward. Since the WMtk does not limit the structure of chunks, however, it cannot automatically extract meaningful features from the candidate chunks for this purpose. Thus, the WMtk requires the designer to specify a function that maps any chunk into a vector of real values that may be used by the adaptive critic to assess the value of the chunk.

A chunk rarely has intrinsic value, however, but is only worthy of retention in certain contexts. Thus, the adaptive critic must have access to a representation of the current context in which the robot finds itself. In order to provide this information, the system designer is required to specify another function which maps the robot's current situation (e.g., its sensory state) into a vector of real values.

Finally, the TD learning mechanism of the working memory system needs to be aware of the arrival of reward. This is provided by the system designer in the form of a reward function which returns the scalar reward value associated with the current situation. At each update cycle (i.e., next time step or new state), the `WorkingMemory` object calls this function to get the instantaneous reward associated with the current situation in order to compute the TD error, which drives learning in the adaptive critic.

The WMtk is designed to be easy to reconfigure but proficient when used in the default configuration. For example, the system designer may specify how the real vectors encoding chunk features and the real vector encoding the current situation are combined and preprocessed before presentation to the adaptive critic network. By default, the vectors for the situation and for each chunk being considered for retention are simply concatenated together to form the input to the adaptive critic. Other input encoding options are available. For example, there are cases in which the features of individual chunks are unimportant as long as at least one of the retained chunks possess a feature of interest. In these cases, the vector representations for the considered chunks might be combined using a logical OR operation, producing a compact "OR code" of the collection of chunks that may be presented to the adaptive critic for evaluation. Similarly, a "NOISY-OR code" option is provided, which combines considered chunk vectors using the information theoretic NOISY-OR function.

Given this collection of designer-specified functions, the `WorkingMemory` object executes the following routine. On each time step of the task, a new list of candidate chunks is given to the `WorkingMemory` object by the robot control system. Initially, these chunks are combined with the chunks that are currently stored in the working memory. The working memory system is then faced with the problem of deciding which chunks to retain. The system examines every possible subset of the collection of chunks that can fit within the limited capacity of the working memory. Each subset of chunks is translated into vectors of real valued features, and these are combined with the vector encoding of the robot's current situation to produce an input vector for the adaptive critic. The combination of chunks that yields the highest estimate of future reward is the one that is selected, and all of the chunks in that subset are retained. All other chunks are discarded. The temporal difference error is then calculated, using the reward function value for the previous time step, our estimated future reward from the previous time step, and our estimated future reward that was just calculated. This temporal difference error is then used to adjust weights in the adaptive critic. In order to encourage the adaptive critic to explore various new memory combinations from time to time, a noise parameter (ϵ) is used to specify the probability with which a random combination of chunks will be maintained in preference to the optimal subset, as determined by the adaptive critic.

Delayed Saccade Task

As an initial test of the utility of the WMtk, we implemented a software simulation of a classic working memory task known as the *delayed saccade task*. In this task, the robot is expected to fix its gaze on an object in the center of the screen (a crosshair). Then another object (a brightly colored dot) is presented in the periphery for a brief period of time. Finally, once a "go" signal is provided (the crosshair vanishing), the robot is expected to shift its gaze to where the peripheral object had appeared earlier. Rather than program the robot to perform this task, we required it to learn correct behavior via a working memory system using the WMtk.

The spatial working memory system used for this task used a simple configuration. The capacity was set to three chunks, which was more than what was needed for this task. We limited the number of screen locations at which objects could appear to five, allowing us to encode the sensory state of the robot using fifteen binary features: five for the location (if any) of a displayed crosshair, five for the location (if any) of a displayed dot, and five for the current location of the robot's gaze.

Three different kinds of chunks were considered for retention. These chunk types were "remember the location of the crosshair" (cross chunk), "remember the location of the dot" (target chunk), and "remain fixated on whatever you're looking at" (fixation chunk). Only the type of the chunk was presented to the adaptive critic, encoded over a vector of three binary features, one for each chunk type. Location information, while not available to the adaptive critic, was stored in chunk data structures. These chunks were generated by the robot control system based on the current state of the environment. The robot would sense whether there was a crosshair or a target present, and, for each object present, it would create

a corresponding candidate chunk that recorded the location of the object. Also, if the robot happened to be looking at an object, a fixation chunk was generated for consideration. Once generated, all of these chunks were provided to the `WorkingMemory` object as candidates for retention. (New chunks that duplicated current working memory contents were not considered, however.)

With regard to the reward function, a scheme was used that both matched standard practice in the primate laboratory and matched reward functions found in the reinforcement learning literature. The robot was provided with a scalar reward value of 0 for all situations until the very end of a trial. If the robot's behavior was perfect for the trial — remaining centrally fixated until the "go" signal was given and only then saccading to the location of the previously presented target — the robot was provided with a reward of 20.[1] If the robot did not perform the trial correctly, a reward of 0 was provided at the end.

The final critical component of this demonstration was the control system that made use of working memory contents in order to select behaviors. This control system contained two parts. The first corresponded to relatively automatic processes that would be automatically engaged unless actively blocked by working memory contents. The second part implemented controlled behaviors driven by the presence of working memory chunks. This division is consistent with models of the role of PFC working memory circuits in cognitive control, with the working memory actively maintaining chunks that focus attention on particular goals in an effort to inhibit more automatic behaviors (Braver and Cohen, 2000).

The automatic behaviors of the robot were very basic, and they would not be able to reliably perform the delayed saccade task on their own. If there are no objects displayed, the robot will look at a random location. If objects are displayed, it will look at one of them, chosen randomly, with high probability.[2] These simple behaviors were sometimes useful for learning the task (e.g., encouraging the robot to look at the crosshair), but they would not drive correct responding by themselves.

The controlled behaviors were only invoked by particular combinations of chunks. The fixation chunk had the highest priority. If a fixation chunk was present, and the robot was currently looking at either an object or a location specified by another retained chunk, then the robot would continue to gaze at the current location with high probability. If a cross chunk or a target chunk was being maintained, the robot would consider the set of locations corresponding to all such retained chunks, as well as the location of any object that the robot was currently looking at, and randomly choose one of these locations to look at, with high probability. In short, in the absence of fixation chunks and focal visible objects, the robot would look at a remembered location.

It is important to note that correct behavior would only be consistently produced if the working memory system learned, from experience, to retain appropriate considered chunks. Specifically, failure to remember the location of a briefly presented target dot would make it virtually impossible for the robot to saccade to the correct location at the end of the trial. Conversely, the spurious retention of a cross chunk would cause the robot to prefer to look at the middle of the screen than at the location of the previously presented target. Thus, the robot was required to learn which chunks needed to be remembered and which needed to be ignored.

Results

Each trial varied in length from thirteen to twenty time steps. The first three time steps of each trial consisted of a blank screen. Then the sequence of stimuli — crosshair appearance, target dot flash, and crosshair removal — was presented, with the onset time of each event varying randomly. The last time step always consisted of a blank screen, corresponding to the point just after the the disappearance of the crosshair. At this point, the robot would either be rewarded for never looking away from the crosshair and then looking at the proper location or not rewarded for having looked away from the crosshair too early or not looking at the correct location on the last time step.

A simulation consisted of running trials back-to-back until the system performed the task correctly for twenty trials in a row. The weights of the adaptive critic were initialized to random values in the range $[-0.001, 0.001]$. The learning rate (α) was set to 0.01, the discount rate (γ) was set to 0.99, and the backup rate (λ) was set to 0.7. The working memory noise parameter (ϵ) was set to 0.05. The total number of trials taken to reach the stopping criterion was recorded for 1000 simulations. The average number of trials taken to reach criterion was 459.3 with a standard error of 25.4.

This number of trials is not unreasonable. It is much less than the number of trials typically needed to train monkeys on this task. Consider that the system must not only learn to overcome its automatic search processes, but it must also decide which informational chunks will cause proper controlled processes to be activated. An examination of the system's learning trajectory showed that it did not learn to go to the optimal solution immediately. Several other options were available. For instance, the robot sometimes simply remembered to look at the crosshair and then chose a random location at the end of the trial. This strategy produced a 20% success rate. During learning, the system often appeared to return to this strategy until it discovered the utility of retaining the target location and forgetting the location of the crosshair. Once it discovered the correct strategy, it did not abandon it, with only random noise affecting performance negatively, afterward. The system performed very well given that its only feedback was the reward or lack of reward at the end of the trial.

[1] This value was selected because it produced good performance when using default WMtk parameters.

[2] With low probability (0.001), the robot always had a chance of ignoring the objects and its memory contents, opting to look at a random location, instead.

Conclusion

Our theory of working memory function, based on the biology of the PFC and midbrain dopamine system, has been used to account for many phenomena observed in the laboratory, but it has yet to be validated in the context of more complex real-world working memory tasks. In order to test this theory, we have begun to incorporate an associated computational neuroscience model into robot control systems. We have demonstrated that the robotic version of this model is functional and capable of simulating performance on a simple standard spatial working memory task.

The next step will involve using the WMtk for more complex aspects of robot control. We are currently integrating the WMtk with an object recognition system embedded in a mobile robot. This working memory system will be rewarded if it remembers the location of environmental objects that make for good landmarks — objects that are easily reacquired and are useful for localization. We hope to show that the adaptive working memory mechanisms of our model are capable of learning to identify the features of good landmarks from experience. Such a result will provide further evidence of the power and scalability of this computational account of the neural basis of working memory.

Acknowledgments

This work has been supported by the National Science Foundation under grant EIA-0325641. The authors extend their thanks to their collaborators on this NSF ITR project, the members of the Computational Cognitive Neuroscience Laboratory at Vanderbilt University, and three anonymous reviewers.

References

Baddeley, A. D. and Hitch, G. J. (1974). Working memory. In Bower, G. A., editor, *Recent Advances in Learning and Motivation*, volume 8, pages 47–90. Academic Press, New York.

Barto, A. G. (1994). Adaptive critics and the basal ganglia. In Houk, J. C., Davis, J. L., and Beiser, D. G., editors, *Models of Information Processing in the Basal Ganglia*, pages 215–232. MIT Press.

Braver, T. S. and Cohen, J. D. (2000). On the control of control: The role of dopamine in regulating prefrontal function and working memory. In Monsell, S. and Driver, J., editors, *Control of Cognitive Processes*, volume 18 of *Attention and Performance*, chapter 31, pages 713–737. MIT Press.

Camperi, M. and Wang, X.-J. (1998). A model of visuospatial working memory in prefrontal cortex: Recurrent network and cellular bistability. *Journal of Computational Neuroscience*, 5:383–405.

Cohen, J. D., Forman, S. D., Braver, T. S., Casey, B. J., Servan-Schreiber, D., and Noll, D. C. (1994). Activation of prefrontal cortex in a nonspatial working memory task with functional MRI. *Human Brain Mapping*, 1:293–304.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Brain and Behavioral Sciences*, 24(1):87–185.

Demb, J. B., Desmond, J. E., Wagner, A. D., Vaidya, C. J., Glover, G. H., and Gabrieli, J. D. E. (1995). Semantic encoding and retrieval in the left inferior prefrontal cortex: A function mri study of task difficulty and process specificity. *Journal of Neuroscience*, 15:5870–5878.

Funahashi, S., Bruce, C. J., and Golman-Rakic, P. S. (1989). Mnemonic coding of visual space in the monkey's dorsolateral prefrontal cortex. *Journal of Neurophysiology*, 61:331–349.

Goldman-Rakic, P. S. (1987). Circuitry of the prefrontal cortex and the regulation of behavior by representational knowledge. In Plum, F. and Mountcastle, V., editors, *Handbook of Physiology*, pages 373–417. American Psysiological Society, Bethesda, MD.

Miller, E. K. and Desimone, R. (1994). Parallel neuronal mechanisms for short-term memory. *Science*, 263:520–522.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63:81–97.

Miyake, A. and Shah, P., editors (1999). *Models of Working Memory: Mechanisms of Active Maintenance and Executive Control*. Cambridge University Press, Cambridge.

Montague, P. R., Dayan, P., and Sejnowski, T. J. (1996). A framework for mesencephalic dopamine systems based on predictive hebbian learning. *Journal of Neuroscience*, 16:1936–1947.

O'Reilly, R. C., Braver, T. S., and Cohen, J. D. (1999). A biologically based computational model of working memory. In (Miyake and Shah, 1999), chapter 11, pages 375–411.

O'Reilly, R. C., Noelle, D. C., Braver, T. S., and Cohen, J. D. (2002). Prefrontal cortex and dynamic categorization tasks: Representational organization and neuromodulatory control. *Cerebral Cortex*, 12:246–257.

Shultz, W., Dayan, P., and Montague, P. R. (1997). A neural substrate of prediction and reward. *Science*, 275:1593–1599.

Sutton, R. S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, 3:9–44.

Wallis, J. D., Anderson, K. C., and Miller, E. K. (2001). Single neurons in prefrontal cortex encode abstract rules. *Nature*, 411:953–956.

Waugh, N. C. and Norman, D. A. (1965). Primary memory. *Psychological Review*, 72:89–104.