Neural Cognitive Modelling:
A Biologically Constrained Spiking Neuron Model of the Tower of Hanoi Task

Terrence C. Stewart (tcstewar@uwaterloo.ca)
Chris Eliasmith (celiasmith@uwaterloo.ca)
Centre for Theoretical Neuroscience, University of Waterloo
200 University Avenue West, Waterloo, ON, N2L 3G1, Canada

Abstract

We present a computational model capable of solving arbitrary Tower of Hanoi problems. All elements except visual input and motor output are implemented using 150,000 LIF spiking neurons. Properties of these neurons (firing rate, post-synaptic time constant, etc.) are set based on the neurons in corresponding areas of the brain, and connectivity is similarly constrained. Cortical components are all general-purpose modules (for storing state information and for storing and retrieving short-term memories of previous state information), and could be used for other tasks. The only task-specific components are particular synaptic connection weights from cortex to basal ganglia and from thalamus to cortex, which implement 19 context-specific rules. The model has a single free parameter (the synaptic connection weights of the input to short-term memory), and produces timing behaviour similar to that of human participants.

Keywords: Tower of Hanoi; neural engineering; cognitive architectures; computational neuroscience

Neural Cognitive Models

To explain human behaviour, cognitive scientists must identify both *what* the brain does and *how* it does it. This involves finding the algorithms underlying cognitive performance as well as determining how these algorithms are implemented within the brain through the interaction of neurons, neurotransmitters, and other physical components.

Our ongoing research is in the construction of large-scale neural models capable of exhibiting complex cognitive behaviour such as planning, rule-following, and symbolic reasoning. The intent is to bridge the gap between cognitive theory and neuroscience, allowing the fields to interact in both directions. With such a bridge, high-level cognitive theory would produce detailed low-level predictions as to the neural spiking patterns, connectivity, and so on that support particular human behaviours. Neuroscience would in turn provide constraints on high-level algorithms, indicating what operations can be performed by neurons, how accurate they can be, and how much time is needed.

Our approach uses the Neural Engineering Framework (NEF; Eliasmith & Anderson, 2003), a general method for constructing computational models whose components are simulations of spiking neurons. The NEF provides a method for defining how values can be represented in a distributed manner across a set of neurons. Most crucially, it also allows us to determine how groups of neurons can be synaptically connected such that they will compute desired functions. This allows us to take a cognitive theory expressed in terms of numerical values and transformations on those values and create a detailed neural model.

In previous work, we used this approach to create models of list memory (Choo & Eliasmith, 2010), rule induction (Rasmussen & Eliasmith, 2010), the Wason card task (Eliasmith, 2005), and action selection (Stewart, Choo, & Eliasmith, 2010a). We have also argued that our action selection model (based on the basal ganglia) is sufficient to implement the basic functionality of a production system (Stewart & Eliasmith 2010).

This paper expands on this work to present the first spiking neural model implementing rule following, planning, and goal recall. The model conforms to the known anatomy, connectivity, and neural properties of the basal ganglia, thalamus, and cortex. The components of the model are general-purpose, in that they could be used to perform different tasks *without changing any neural connections within the cortex*. To define the task to be performed, we only need to set the synaptic connection weights between the cortex and basal ganglia and between the thalamus and the cortex. This makes our work both a neural explanation of a particular high-level cognitive task, and a general set of principles for creating neural models for other such tasks.

The Tower of Hanoi

The task considered here is the Tower of Hanoi, which has a rich history in cognitive science as a problem solving task (e.g. Simon, 1975). Many symbolic (non-neural) cognitive models exist which match expert human behaviour well (e.g. Altmann & Trafton, 2002). The task involves three pegs and a fixed number of disks of different sizes with holes in them such that they can be placed on the pegs. Given a starting position, the goal is to move the disks to a goal position, subject to the constraints that only one disk can be moved at a time and a larger disk cannot be placed on top of a smaller disk.

Figure 1 shows the optimal series of steps needed to solve the four-disk Tower of Hanoi when all disks start on one peg and must all be moved onto a different peg. There are many algorithms that could be used to produce this series of steps, and cognitive research on this task involves determining which algorithm(s) people are using by examining factors such as the time taken between steps and the types of errors produced. Anderson, Kushmerick, and Lebiere (1993) provide a variety of measures of human performance on this task, and Figure 1 compares their empirical data to our model performance in terms of the time delay between movements (only conditions where no mistakes were made are considered here).

The basic algorithm used is the "Sophisticated Perceptual Strategy" from Simon (1975). We start with the largest disk that is not in the correct location. We then examine the next smaller disk. If it blocks the move we want to make, then our new goal is to move that disk to the one peg where it will not be in the way. We then iterate this algorithm, going back to previous goals once we have accomplished the current one. The effects of this algorithm can be seen in Figure 1, since steps 1, 5, 9, and 13 show long pauses as a new set of goals are established. For example, in step 1, we have the goal of moving disk 4 to peg C, but in order to do that we must first move disk 3 to peg B, which requires moving disk 2 to peg C, which requires moving disk 1 to peg B. These goals must be generated and stored so that once disk 1 is moved to peg B we can remember to now move disk 2 to peg C, rather than re-generating the entire sequence of moves.

Non-neural symbolic cognitive models already exist which fit the empirical data extremely well (e.g. Altmann & Trafton, 2002). However, they do not provide a neural explanation of the operations involved, and they do not provide a biological grounding of the various parameters used within the model.

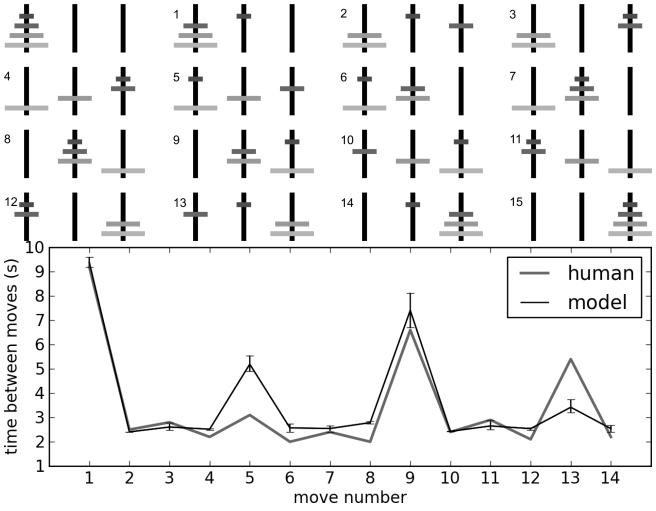

Figure 1: The sequence of moves to ideally solve the four-disk Tower of Hanoi (top). Time delay for expert human performance (Anderson, Kushmerick, & Lebiere, 1993) and our neural model is also shown (bottom).

The Neural Engineering Framework

The Neural Engineering Framework makes two key assertions. First, a group of neurons uses distributed encoding to represent a numerical vector, where different patterns of activation indicate different values for that vector. Second, synaptic connection weights between neurons can be defined so that particular operations can be computed. Thus, if one neural group is storing a vector with three elements (e.g. [x,y,z]), then these could be connected to a second neural group that would store two values calculated from those three (e.g. [x*y,cos(y)+sin(z)]). The accuracy of this calculation has been shown to depend on the properties of the neurons themselves and the complexity of the function computed, with a general result that the mean squared error is inversely related to the number of neurons in the group (Eliasmith & Anderson, 2003).

While the NEF can be used with any neuron model, the model presented here uses Leaky Integrate-and-Fire (LIF) neurons. Current entering and leaking out of each neuron affects the voltage. If this voltage reaches a threshold, the neuron fires, resetting the voltage to zero for a refractory period. When a neuron fires, it releases current to all connected neurons. This post-synaptic current decays exponentially over time at a rate τ that depends on the neurotransmitter and receptors involved (ranging from two to hundreds of milliseconds). The various parameters of the LIF model (refractory period, membrane resistance, background current, post-synaptic time constant τ, etc.) are set based on neurophsyiological measurements of different types of neurons in different brain areas. The most important of these from a functional perspective is the post-synaptic time constant τ which effectively controls how quickly the vector being represented can change.

To represent a numerical vector with a group of LIF neurons, the NEF uses preferred direction vectors, which have long been observed throughout the visual and motor cortices (e.g. Georgopoulos et al., 1986). Each neuron has a particular vector for which it will fire most strongly. The amount of current J flowing into the neuron is the dot product of the preferred vector $\boldsymbol{e}$ with the represented value $\boldsymbol{x}$, times the neuron's gain α, plus the background current J_{bias} (Eq. 1). Preferred vectors are randomly chosen, and α and J_{bias} values are distributed to match average and maximal firing rates of real neurons. We can force a group of neurons to represent a vector $\boldsymbol{x}$ by directly adding the amount of current computed via Eq. 1. This is used to provide inputs to our simulation.

Eq. 1 allows us to convert a value $\boldsymbol{x}$ into neural activity. We can also do the reverse and use the neural activity to estimate the value $\boldsymbol{x}$ that is being represented by computing the optimal decoding vectors $\boldsymbol{d}$ using Eq. 2. a_i is the average firing rate for neuron i for a given value of $\boldsymbol{x}$, and integration is over all values of $\boldsymbol{x}$. To estimate $\boldsymbol{x}$, we add together the output current of each neuron, weighted by $\boldsymbol{d}$. This is the optimal least-squares linear estimate of $\boldsymbol{x}$.

Most crucially, we can use $\boldsymbol{d}$ to calculate the synaptic connection weights that will compute particular operations. To compute a linear operation where $\boldsymbol{x}$ is represented by one group and a second group should represent $\boldsymbol{Mx}$, where $\boldsymbol{M}$ is an arbitrary matrix, we set the connection weights between neuron i in the first group and neuron j in the second group to ω_{ij} as per Eq. 3. For non-linear operations, we do the same, but compute a new set of $\boldsymbol{d}$ values via Eq. 4.

$$J = \alpha \boldsymbol{e} \cdot \boldsymbol{x} + J_{bias} \quad (1)$$

$$\boldsymbol{d} = \Gamma^{-1} \Upsilon \quad \Gamma_{ij} = \int a_i a_j \, dx \quad \Upsilon_j = \int a_j \boldsymbol{x} \, dx \quad (2)$$

$$\omega_{ij} = \alpha_j \boldsymbol{e}_j \boldsymbol{M} \boldsymbol{d}_i \quad (3)$$

$$\boldsymbol{d}^{f(\boldsymbol{x})} = \Gamma^{-1} \Upsilon \quad \Gamma_{ij} = \int a_i a_j \, dx \quad \Upsilon_j = \int a_j f(\boldsymbol{x}) \, dx \quad (4)$$

The NEF allows us to convert an algorithm in terms of vectors and calculations on those vectors into a neural model. To use it to create cognitive models, we need to express cognitive algorithms in terms of vectors. As a simple example, consider the case of storing state information in one group of neurons and we want another group of neurons to represent how similar that state is to some desired state. This might be used as part of an algorithm that says "IF state=A THEN...."

To convert this into vectors, we can consider the similarity measure to be a single number (a vector of dimension 1). In contrast, the state can include many possible aspects, so we represent it as a high-dimensional vector. In this paper, we use 128-dimensional vectors represented with 3000 neurons. Each neuron has a randomly chosen preferred vector $\boldsymbol{e}$, and α and J_{bias} values chosen to give an average firing rate around 40Hz and a maximum firing rate of 200Hz. We can now use Eq. 1 to force the neurons to represent whatever state $\boldsymbol{x}$ we desire.

To compute the similarity between the current state $\boldsymbol{x}$ and the particular state A, we connect these neurons to a smaller group of 40 neurons representing a single number. To compute the similarity, we want to calculate the dot product between $\boldsymbol{x}$ and A. The synaptic connection weights that will do this are given by Eq. 3, where $\boldsymbol{M}$ is the vector A. Figure 2 shows that as the state value is adjusted, the firing rate of the second group of neurons changes accordingly.

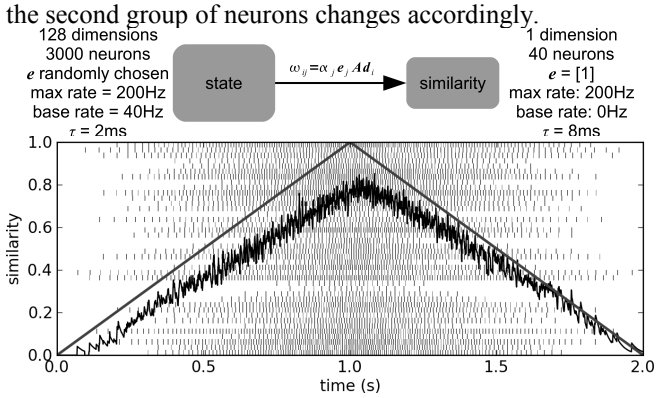

Figure 2: Computing the similarity between the current state and a specific state A. The straight line shows the correct answer as the state value varies over time. The jagged line is the value represented by the similarity neurons, decoded with Eq. 2. Spike times of these neurons are also shown.

Action Selection

In previous work (Stewart, Choo, & Eliasmith, 2010a), we developed a model of action selection that conformed to the anatomy of the basal ganglia, thalamus, and cortex. Groups of neurons in the cortex represent state information, and connections between the cortex and basal ganglia compute the similarity between the current state and the ideal state for each action available to the agent (as in Figure 2). The role of the basal ganglia is to find the maximum of these values, and its output to the thalamus should inhibit all actions except for the one action whose ideal state is closest to the current state. Connections from thalamus to cortex implement actions in two ways. *Direct* actions involve sending a particular vector to a cortical area (implemented as in Figure 2, but with connections going the other way). *Routing* actions indicate that information should be sent from one cortical area to another. These are implemented by having a neural group that takes input from the first group and passes it on to the second (both connections computed using Eq. 3 where M is the identity matrix). We then add a group of neurons in the thalamus which inhibit all of the neurons in this middle group (ω_{ij}=-1), causing this communication channel to do nothing most of the time. The thalamus can now implement the action of passing the information between the cortical areas by inhibiting these inhibitory neurons (see Figure 3).

The basal ganglia model used is from Gurney, Prescott, and Redgrave (2001), which is expressed in terms of vectors and mathematical operations, making it natural to convert to spiking neurons using the NEF. We have shown (Stewart, Choo, & Eliasmith, 2010a) that this model can reliably detect states and implement direct and routing actions. By setting the properties of neurons to those typical of the neurons in these various brain regions, we found that direct actions are performed in 34-44ms, while routing actions require 59-73ms (Stewart, Choo, & Eliasmith, 2010b).

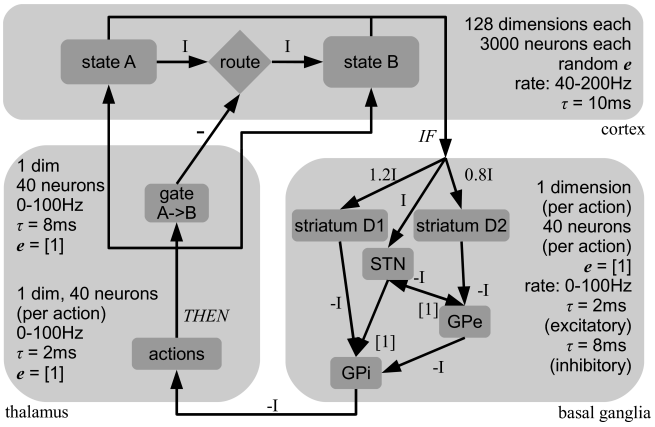

Figure 3: The basal ganglia, thalamus, and cortex. Input to basal ganglia is calculated as in Figure 2, with *IF* matrix containing ideal states for each action. Output to thalamus inhibits all actions except best one. Connections to cortex via *THEN* matrix implement actions. Gate connection has ω_{ij}=-1. All other connections calculated using Eq. 3, with I as the identity matrix and [1] as a matrix of all 1's. STN=subthalamic nucleous; GPi, GPe=globus pallidus internal, external; D1, D2 are distinct types of striatal cells.

Cortical Modules

The action selection system is capable of controlled routing of information between cortical areas. To create a cognitive model, we need to specify what these cortical areas are and what operations they perform. These areas should be general-purpose, in that they should be useful for many different tasks, not just the Tower of Hanoi, since we do not expect large-scale cortical change when learning the task.

To implement the Tower of Hanoi algorithm, we need to keep track of three variables: the thing we are currently attending to, the thing we are trying to move, and the location we are trying to move it to. For this model, we assume these are stored in three separate cortical areas referred to as ATTEND, WHAT, and WHERE.

These systems must be capable of maintaining their state. That is, given no input, they should continue to represent whatever vector they are currently representing. This requires feedback, as in Figure 4. Synaptic connections are computed with Eq. 3, where $\boldsymbol{M}$ is the identity matrix $\boldsymbol{I}$. However, given an input, the feedback loop should be disabled. This is done with inhibitory weights between the input and the feedback neurons (ω_{ij}=-1).

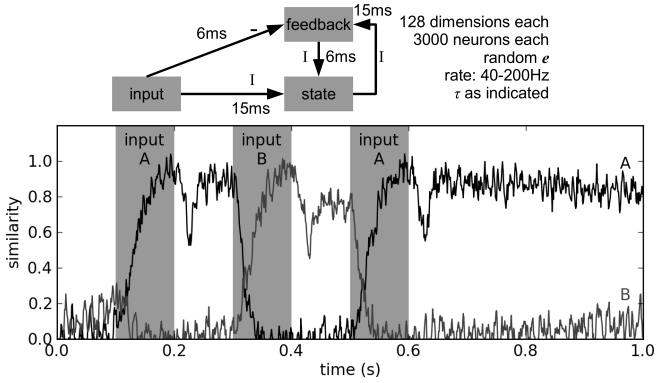

Figure 4: Maintaining and changing state. Input neurons are set to represent A from 0.1-0.2s, B from 0.3-0.4s, and A from 0.5-0.6s. Plotted similarity is between the decoded vector from the state neurons and the randomly chosen ideal vectors for A and B. The state is successfully stored over time and changes quickly when a new value is input.

The Tower of Hanoi algorithm also requires us to store and recall old goals of what disk to place where. Storing a single goal such as "disk 4 on peg C" would be easy: add the vectors together (D4+C) and store the result using a mechanism similar to that in Figure 4. However, multiple goals cannot be stored in this manner, as (D4+C)+(D3+B) cannot be distinguished from (D4+B)+(D3+C). This can be seen as an instance of the classic binding problem.

Binding using vector representations has been addressed by a family of approaches known as Vector Symbolic Architectures (VSAs; Gayler, 2003). VSAs introduce a mathematical operation that combines two vectors to produce a third that is highly dissimilar (dot product near zero) to either original vector. Since this operation is reversible, we can add together the combined vectors, store the result, and reliably extract the individual inputs.

For our model, we follow Plate (2003) and use circular convolution ⊗ to combine vectors and circular correlation ∅ as an approximate inverse. These operations can be neurally implemented using Eqs. 3 and 4 as detailed in (Eliasmith, 2005), and we have argued this approach allows for complex structured symbol manipulation in neurons (Stewart & Eliasmith, 2009).

To store a set of goals, we compute the sum of the combined vectors V=D4⊗C+D3⊗B+D2⊗C. To recall where we wanted to place a particular disk (e.g. D3), we compute V∅D3, which gives a result of approximately B (accuracy improves with increased dimensions).

For our neural model of this process, we use Figure 6. The WHAT and WHERE values are combined and fed into the MEMORY. Since the MEMORY has a feedback connection, any input will be added to its current value (MEMORY=MEMORY+α*WHAT⊗WHERE). The value α (set to 0.01) controls how quickly the memory will store new information and forget old information. Thus, any time a value is present in both the WHAT and WHERE neural groups, information about that pair will be stored in the memory. Once it is stored, to extract WHERE we planned to place a particular disk, we place its value in WHAT and nothing in WHERE. The value in RECALL should now be the vector for the peg we want to move it to.

Our neural model also needs inputs and outputs. Creating a complete model of the visual and motor systems required for this task is outside the scope of this paper. Instead, we create neural groups and directly calculate via Eqs. 1 and 2 what input currents should be fed to the vision system, and what actions are being indicated by the motor outputs. For input, we have the location of the currently attended object (ATTEND_LOC), the location of the object we are trying to move (WHAT_LOC), and the final end goal location of the object we are trying to move (GOAL_LOC). For output, we have two motor areas for indicating what disk should be moved (MOVE_WHAT) and where it should be moved to (MOVE_WHERE). We also include a simple action that causes attention to move to the largest disk (i.e. sets the value in ATTEND to the vector for the largest visible disk).

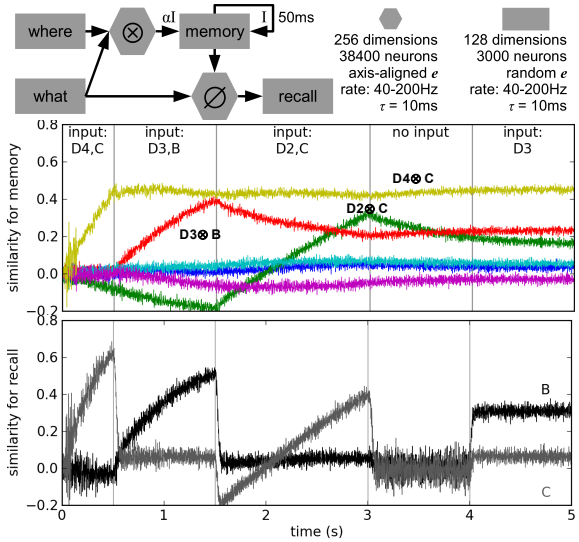

Figure 5: Goal memory. Vector pairs (D4,C; D3,B; D2,C) are presented in sequence to WHAT and WHERE neurons for first 3 seconds, loading the memory (top). After 1 second of no input, a recall is performed by putting D3 in WHAT and nothing in WHERE. The RECALL neurons (bottom) now represent B, successfully recalling the goal.

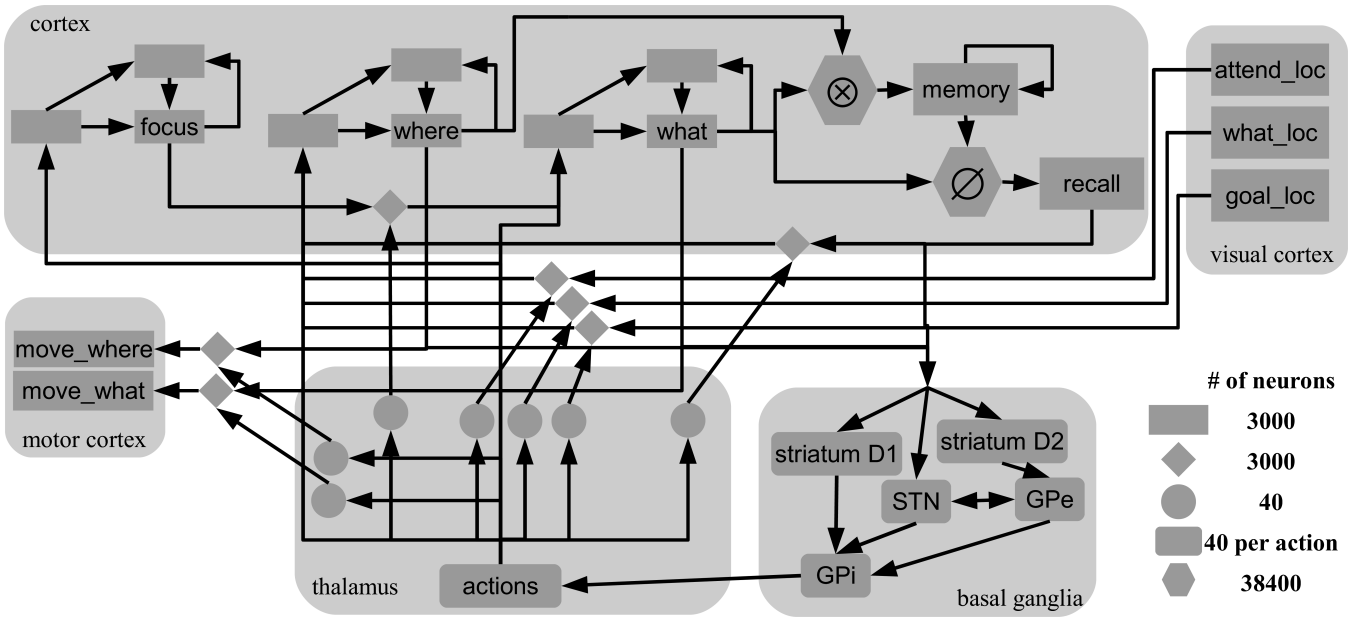

Figure 6: The Tower of Hanoi model. Connections are calculated as in Figures 3, 4, and 5. All state values project to basal ganglia for action selection. Basal ganglia chooses best action, releasing inhibition on that one action in thalamus. Thalamus projects values to state inputs (direct actions) and controls gates (routing actions) to implement the action. Inputs and outputs are provided via visual and motor cortex, which are the only elements not done in spiking neurons. Total # neurons: 150,640.

The Tower of Hanoi Model

The components described in the previous sections define the vast majority of the neurons and synaptic connections within our model. We next need to define the set of internal actions the model can perform, and the conditions in which it should perform each action. These rules define the *IF* and *THEN* matrices in Figure 3, and allow us to solve for the connections from cortex to basal ganglia (*IF*) and from the thalamus to cortex (*THEN*).

For each action, we determine what state the system should be in for that action to occur. We then connect the cortical state neurons to the basal ganglia using Eq. 3 where ***M*** is the vector representation of the ideal state, as in Figure 2. In addition, we can also create a neural group that will compute the dot product *between* two cortical states, allowing us to define rules that will only apply if two states are the same (or different), regardless of what they are.

To implement the effects of an action, we connect that action's thalamic neurons to the cortical neurons we want to affect. Connection weights are found using Eq. 3, where ***M*** is the vector $\mathbf{v}$ we want to send to that cortical area. If the neurons are active (representing 1), the effect will be to add the vector $1*\mathbf{v}=\mathbf{v}$ to that area. If the action neurons are inhibited by the basal ganglia (as will be true for all actions other than the current best one), the output will be $0*\mathbf{v}=0$. The same approach is used for routing actions except ***M***=-1, which will inhibit the gate which is inhibiting the communication channel between the cortical areas.

The algorithm is to ATTEND to the largest disk (placing D4 in ATTEND). Next, we form a goal to place D4 in its final location (route ATTEND to WHAT and GOAL_LOC to WHERE). We now have D4 in ATTEND and WHAT and C in WHERE. Next, we check if the object we are trying to move is in its target location. If it is (WHERE=WHAT_LOC), then we've already finished with this disk and need to go on to the next smallest disk (loading D3 in WHAT and routing GOAL_LOC to WHERE).

If the disk in WHAT is not where we are trying to move it to (WHERE is not equal to WHAT_LOC), then we need to try to move it. First, we look at the next smaller disk (send D3 to ATTEND). If we are attending a disk that is not the one we are trying to move (ATTEND is not WHAT) and if it is not in the way (ATTEND_LOC is not WHAT_LOC or WHERE), then attend the next smaller disk. If it is in the way (ATTEND_LOC=WHAT_LOC or ATTEND_LOC= WHERE), then we need to move it out of the way. To do this, set a goal of moving the disk to the one peg where it will not be in the way. The peg that is out of the way can be determined by sending the value A+B+C to WHAT and at the same time sending the values from WHAT_LOC (the peg the disk we're trying to move is on) and ATTEND_LOC (the peg the disk we're looking at is on) to WHAT as well, but multiplied by -1. The result will be A+B+C-WHAT_LOC-ATTEND_LOC, which is the third peg.

This algorithm, with a special case for use when attending the smallest disk (D1 can always be moved, since nothing is ever in its way, and if we've made it to D1 without finding anything in the way, then we can move the disk we're trying to move), is sufficient for solving Tower of Hanoi. However, it does not make use of the memory system, so it has to rebuild its plans each time. To address this, we first add a rule to do nothing if RECALL is not the same as WHERE. This occurs if we've set a new goal, but there has not been enough time for the memory to hold it (see Figure 5). Next, rules are added for the state where we have just

finished moving a disk. Instead of starting over from the beginning, we send the next largest disk to ATTEND and WHAT and route the value from RECALL to WHERE. This recalls the goal location for the next largest disk and continues the algorithm. All of this requires 19 actions.

Results

The model is able to successfully solve the Tower of Hanoi, given any valid starting position and any valid target position. It does occasionally make errors, and recovers from them (analysis of these errors is ongoing).

Figure 7 shows particular measures from the model as it solves the task. The input to the basal ganglia is the context-dependent utility of the 19 different actions it could perform. To demonstrate its successful action selection in these circumstances, the spiking output from the basal ganglia to the thalamus is shown. Different groups of neurons stop firing at the same time, releasing the inhibition in the thalamus, allowing that particular action to occur.

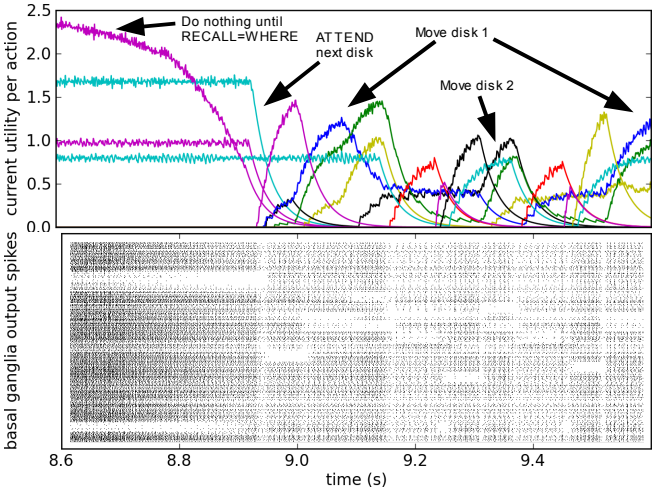

Figure 7: Input to basal ganglia (top) and action selection output to thalamus (bottom) during typical moves 1 and 2 of the Tower of Hanoi task. Changing input reflects changing similarity between cortex states and ideal states for each action. Utilities for four example rules are indicated. Spiking output inhibits actions from being executed. At any given time, one group of output neurons (corresponding to the largest input) will stop firing (as seen in bottom spike raster), releasing inhibition and allowing the action to occur.

The average time delay between moves when no errors occur is shown in Figure 2, compared to human performance. The model does not exactly match the empirical data. However, it should be noted that there is only one parameter in the entire model: the scaling factor α in the memory system. All other timing parameters are taken from the neurophysiology of the various brain regions. (The number of neurons in each group is also freely chosen, but affects accuracy rather than timing).

Our main result is that we can create a neural model at this level of complexity, implementing symbolic algorithms in a non-symbolic manner. The fact that the model performance is in the right ballpark is highly encouraging, and we expect to improve this by exploring modifications to the set of rules. For example, Altmann and Trafton (2002) add heuristics such as "whenever you move disk 2, move disk 1 on top of it" and "don't undo your previous move", both of which are needed for their symbolic model to match empirical data.

Our ongoing research is to develop this model into a full neural cognitive architecture. This involves exploring the use of the same cortical components in multiple tasks and identifying neurological constraints. We are also adding learning rules to adjust the *IF* weights into the basal ganglia to improve performance over time, as this is where the dopamine implicated in reinforcement learning is found.

References

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

Anderson, Kushmerick, & Lebiere, 1993

Choo, F., Eliasmith, C. (2010). A Spiking Neuron Model of Serial-Order Recall. *32nd Annual Conference of the Cognitive Science Society.*

Eliasmith, C. (2005). Cognition with neurons: A large-scale, biologically realistic model of the Wason task. *27th Annual Meeting of the Cognitive Science Society.*

Eliasmith, C. & Anderson, C. (2003). *Neural Engineering: Computation, representation, and dynamics in neurobiological systems.* Cambridge: MIT Press.

Gayler, R. (2003). Vector symbolic architectures answer Jackendoff's challenges for cognitive neuroscience. *International Conference on Cognitive Science*.

Georgopoulos, A.P., Schwartz, A., & Kettner, R.E. (1986). Neuronal population coding of movement direction. *Science*, 233, 1416-1419.

Gurney, K., Prescott, T., & Redgrave, P. (2001). A computational model of action selection in the basal ganglia. *Biological Cybernetics* 84, 401-423.

Rasmussen, D., Eliasmith, C. (2010). A neural model of rule generation in inductive reasoning. *32nd Annual Conference of the Cognitive Science Society.*

Simon, H. A. (1975). The functional equivalence of problem solving skills. *Cognitive Psychology*, 7(2), 268–288.

Stewart, T.C., Choo, X., & Eliasmith, C. (2010a). Symbolic reasoning in spiking neurons: A model of the cortex/basal ganglia/thalamus loop. *32nd Annual Meeting of the Cognitive Science Society.*

Stewart, T.C., Choo, X. & Eliasmith, C. (2010b). Dynamic behaviour of a spiking model of action selection in the basal ganglia. *10th Int. Conf. on Cognitive Modeling*

Stewart, T.C,, & Eliasmith, C. (2009). Compositionality and biologically plausible models. In W. Hinzen, E. Machery, & M. Werning (Eds.), *Oxford Handbook of Compositionality.*

Stewart, T.C. & Eliasmith, C. (2010). Neural symbolic decision making: A scalable and realistic foundation for cognitive architectures. *Proceedings of the 1st Annual Meeting of the BICA Society.*