Complex problem solving with reinforcement learning

Frédéric Dandurand
Department of Psychology, McGill University
1205 Dr. Penfield Ave., Montréal, Québec, H3A 1B1, Canada
fdandu@ego.psych.mcgill.ca

Thomas R. Shultz
Department of Psychology & School of Computer Science McGill University
1205 Dr. Penfield Ave., Montréal, Québec, H3A 1B1, Canada
thomas.shultz@mcgill.ca

François Rivest
Département d'informatique et de recherche opérationnelle Université de Montréal
CP 6128 succ. Centre Ville Montréal, Québec, H3C 3J7, Canada
francois.rivest@mail.mcgill.ca

***Abstract* - We previously measured human performance on a complex problem-solving task that involves finding which ball in a set is lighter or heavier than the others with a limited number of weighings. None of the participants found a correct solution within 30 minutes without help of demonstrations or instructions. In this paper, we model human performance on this task using a biologically plausible computational model based on reinforcement learning. We use a SARSA-based Softmax learning algorithm where the reward function is learned using cascade-correlation neural networks. First, we find that the task can be learned by reinforcement alone with substantial training. Second, we study the number of alternative actions available to Softmax and find that 5 works well for this problem which is compatible with estimates of human working memory size. Third, we find that simulations are less accurate than humans given equivalent amount of training. We suggest that humans use means-ends analysis to self-generate rewards in non-terminal states. Implementing such self-generated rewards might improve model accuracy. Finally, we pretrain models to prefer simple actions, like humans. We partially capture a simplicity bias, and find that it had little impact on accuracy.**

Index Terms – Problem Solving, Reinforcement Learning, Complex Cognition.

I. Introduction

Reinforcement learning is an important mechanism in machine learning. Learning by reinforcement is much more difficult than supervised learning. First, reinforcement learning systems must learn from limited or poor information, typically scalar signals (good or bad) in contrast to supervised learning where information consists of fully specified target vectors. Second, when solutions involve multiple steps and actions, systems have to determine which actions or set of actions lead to the final outcome because reinforcement learning systems do not get feedback after every pattern or step. As steps increase, number of possibilities grows in a combinatorial fashion.

Reinforcement learning has mastered difficult problems such as playing backgammon [1]. It has also been used in cognitive modeling, including classical and operant conditioning [2]. In addition to its ability to cover behavioral data, reinforcement learning algorithms are biologically plausible [3]. In this paper, we apply reinforcement learning to an area of cognition where it has received little attention: complex problem solving.

Problem solving is still largely dominated by classical information processing theories [4]. These approaches emphasize search and heuristics, but not learning. However, reinforcement offers a promising way to model learning in problem solving tasks because information processing and reinforcement approaches define problems in a compatible way using concepts such as states, actions, goals, constraints, costs and distances. Information processing theories provide powerful descriptive tools like task analysis to describe search spaces. Reinforcement learning algorithms provide potential mechanisms for biological and artificial agents to learn which states and actions in the search space lead to correct solutions.

We studied reinforcement learning in a complex problem solving task involving the weighing of balls. In a computer simulation, participants had to find which ball in a set of 12 identical-looking balls is heavier or lighter than the others. A virtual scale is available, but it can only be used three times on each problem. Participants could install any combination of balls on the two sides of the scale. The scale returned one of three results: left heavier, right heavier, or equal. A different ball and weight are randomly chosen for each problem [5]. Details of strategies used, explanations of why the task is difficult and presentation of a correct solution are available in [5] and [6]. Our present goal is to build a computational model of human performance on this problem, and to capture the documented human preference for symmetrical and simple ball selections [6].

In this task, there are 24 possible cases (12 balls x 2 weights) to discriminate. Accuracy is the proportion of correct responses among the cases presented. We consider the task successfully solved when the strategy used allows for correct identification (100% accuracy) of all cases. Mean human accuracy was 59.0% and no participant was able to solve the problem by feedback alone (i.e., being told if their answers were correct or not) within the 30 minutes allowed. By contrast, 16.7% (7 out of 42) of participants who watched demonstrations or read instructions were able to solve the task successfully within the same time [5], and mean accuracy was about 0.75. This is a difficult task, even with demonstrations or instructions. In this paper, we present a computational model of the most difficult experimental condition, the feedback learning group. Because participants in this group were given rewards in the form of binary feedback (correct/incorrect) at the end of each problem, reinforcement learning represents an appropriate framework for building a biologically plausible computational model of the task.

More specifically, we study the following questions:

1. Given failures of human participants trained by feedback, can this task be solved by reinforcement learning alone, given enough time to explore the state and action space?
2. Can we capture human accuracy using a reinforcement learning system in which training is limited to an amount comparable to humans? In other words, could a reinforcement learning algorithm be a good model of human performance in the ball weighing experiment?
3. Can we improve the model's coverage of human performance by pre-training it to exhibit the same biases that humans have when selecting actions?

II. Methods

A. Ball Weighing Experiment

As mentioned, the task modeled here requires agents to find, with three uses of a scale, the one ball that is heavier or lighter than the rest among 12 balls. A task analysis is presented in Fig. 1. Solving the ball weighing task involves alternating between two sub-tasks: deciding which balls to weigh on the scale (the selection task) and updating information about possible weights of the balls based on the result of the weighing (the labeling task).

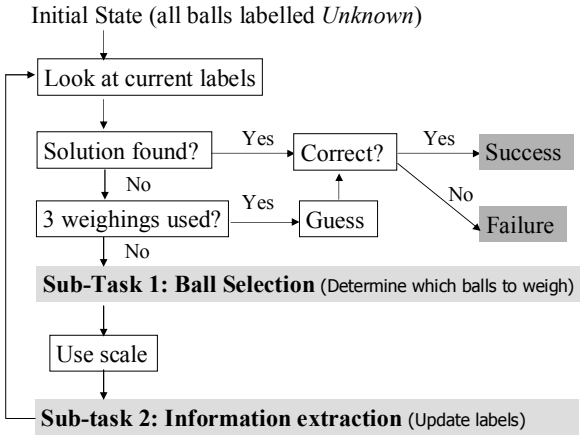

Fig. 1 Task analysis of the ball weighing experiment

Humans kept track of possible ball weights using one of seven labels: unknown (heavy, light or normal), heavy or light, heavy or normal, light or normal, heavy, light, and normal.

One of the challenges in modelling this problem is that the two sub-tasks (selection and labelling) form a loop and are thus mutually dependent: the output of one is the input of the other. Therefore, finding the optimal action depends on how the other sub-task is doing. As learning progresses on both sub-tasks, optimal actions are a moving target.

Given the complexity of the ball weighing problem, we present a simplified model here. First, we concentrate on the selection task, assuming an optimal agent for updating labels. This sidesteps the moving target issue because labelling is always correct or optimal. Second, we consider only actions that have the same number of balls on both sides of the scale, because humans selected equal numbers of balls in 98.6% of their actions. Third, we dropped the heavy-light label because humans almost never used it. After these simplifications, the search space contains a total of 6187 states and 5,671,402 actions, i.e., a mean of 916 actions for each state.

B. Reinforcement Learning

We modelled the ball weighing task using reinforcement learning, and learned it using a SARSA-based system. The goal of the system is to accurately predict the reward resulting from taking an action in a given state. Systems improve their initial inaccurate estimates by trial-and-error exploration of the problem search space. As larger portions of the search space are explored, better estimates can be learned.

As mentioned, one of the difficulties of learning by reinforcement is that the actual reward is known only at the end of the problem episode, that is, in a terminal state. To estimate rewards in non-terminal states, a possible approach is to use temporal-difference (TD) learning [7]. TD-learning uses the current state reward estimate and current rewards to correct their previous state's estimate of future reward. This allows a form of temporal backpropagation of rewards to states and actions that lead to the reward. In this paper, we used an on-policy TD control method called SARSA [7].

SARSA works by learning an estimate Q of future rewards for every state-action pair. Its learning rule is

$$Q(s_t,a_t) \leftarrow Q(s_t,a_t) + \alpha [r_{t+1} + \gamma Q(s_{t+1},a_{t+1}) - Q(s_t,a_t)] \quad (1)$$

where Q is the predicted reward, s is a state, a is an action, r is a reward, and indices t and t+1 are used for current and next states and actions respectively; α is the learning rate, and γ is the discount factor. The name SARSA is based on the quintuple that the algorithm uses (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}).

We used a learning rate of 0.5. The discount factor reduces the value of distant rewards. We used a discount factor of 1.0 because correct solutions are known to take the maximal number of scale uses, and we do not want to encourage the system to seek shorter solutions.

C. Cascade-correlation

Because the state-action pair space is too large to be exhaustively sampled in an explicit manner, we used cascade-correlation (CC) neural networks as function approximators of the expected reward function. The precise shape of expected reward function may depend on problem coding. CC is described in detail in [8] and has been successfully used to model learning and cognitive development in numerous tasks [9]. We used a score threshold value of 0.075. This score threshold (2.5% of the output range) was empirically found to be satisfactory. As shown in equation 1, SARSA updates its approximation function Q after every action, i.e., it is an online technique. In contrast, CC works in batch, i.e., it learns multiple patterns (input-output pairs) at once. We therefore buffer the patterns in a cache until there are enough to make a batch to train CC, as in [10]. To ensure that SARSA uses up-to-date patterns, our system first looks for patterns in the cache. If the pattern is not found, CC is used to generate the Q value for the state-action pair, which is also inserted in the cache. All updates are made in the cache until the whole batch is processed. After each batch, the cache is emptied.

D. Hardmax and Softmax

Hardmax is a greedy algorithm, always selecting the action with highest expected reward, which results in less exploration. By contrast, Softmax considers all possible actions: the higher the expected reward, the higher the probability of selecting an action, resulting in more exploration. For this task, we used a technique similar to Softmax, except that we truncated the number of alternatives considered to the *n* best. We present results for *n* varying between 1 (i.e., Hardmax) and 10, and found n=5 to work best. The number of alternatives that our algorithm considers is about the same as the number of active elements in human working memory [11, 12].

E. Rewards

The following rewards were given to the system: +1 for a correct answer, -2 for an incorrect answer, and -1 when the system did not give any answer after three weighings. Accordingly, the cascade-correlation output unit was a sigmoid with a range of -2 to +1. Networks, like participants, were not allowed to use more than 3 weighings, but they could use fewer.

F. Input and output coding

State and action are concatenated as inputs to a cascade-correlation neural network. Networks learn to accurately predict the expected reward (output) for each state-action combination (input).

There are 24 inputs: 6 to code the state and 18 to code the action. The state indicates the number of balls marked using each label type. The 6 inputs for state code the proportion of balls of each label type in order: U, HN, LN, H, L, N. For example, to indicate 6 balls, the input value is 6/12 = 0.5.

The action indicates how many balls of each label type to install in each container. There are three containers: ball bank, left side of scale and right side of scale. For each container, the proportion of balls of each label is given in the same order as for the state. For example, if all 12 balls are labelled as unknown, and the selected action consists in weighing 6 balls on the left side of the scale (1/2 of 12 = 6) against 6 balls (1/2) on the right side, leaving no ball in the bank, the input is: 1 0 0 0 0 0; 0 0 0 0 0 0; 0.5 0 0 0 0 0; 0.5 0 0 0 0 0.

G. Biases

We have previously shown that humans tend to use simple and symmetrical selection arrangements [6]. Humans may have difficulty solving this problem because solutions sometimes require complex and asymmetrical actions.

To compare performance with humans, we computed indices of asymmetry and complexity in our networks. To measure complexity, we summed the total number of labels present on each side of the scale. To measure asymmetry, we counted the total number of differences in labels between left and right sides of the scale, i.e., whenever a label was present on one side of the scale but not on the other, one unit of asymmetry was added.

Table I shows examples of complexity and of asymmetry measures.

TABLE I
EXAMPLE COMPLEXITY AND ASYMMETRY INDEX CALCULATIONS

	Example	Index
Complexity	HN HN (1) vs. N N (1)	2
	HN LN LN (2) vs. HN LN N (3)	5
Asymmetry	HN HN vs. N N	2
	LN LN LN vs. LN LN LN	0
	HN LN LN vs. HN LN N	1

III. RESULTS

We assessed if a reinforcement learning system considering different number of alternatives (working memory size) could learn the ball weighing task by reinforcement alone, that is, with no additional information such as demonstration or instruction.

A. Working memory size

Table II presents results of the working memory experiments. It shows how many networks were trained to success within 10000 trial epochs or 100 recruits. An epoch is a pass through the 24 different problems.

TABLE II
RESULTS OF TRAINING WITH DIFFERENT WORKING MEMORY SIZES.
20 NETWORKS WERE RUN IN EACH CONDITION.

WM size	**1**	**3**	**5**	**10**
Networks trained to success	0	14	19	7
Success networks (100% accuracy)				
Recruits	N/A	16.4	19.2	25.6
Epochs	N/A	1473	1537	4350
Unsuccessful networks (within 10000 epochs or 100 recruits)				
Recruits	12.5	18.5	31.0	71.4
Reward	-2.9	3.0	-2.0	-5.8
Accuracy	0.44	0.56	0.46	0.38

None of the networks trained using a working memory size of 1 (equivalent to hardmax) successfully learned the task. Further analyses revealed that those networks got stuck in local reward maxima at a mean of 80.2 epochs with a mean accuracy of 0.44 after having recruited a mean of 12.5 hidden units. Success was even lower in networks that consider many (10) alternatives. Those networks built complex reward functions (mean 71.4 hidden units), suggesting that they might have gotten lost in suboptimal paths while searching through too many alternatives.

The majority of networks trained with a small working memory size (3 or 5) found solutions: 70% (14/20) for size 3, and 95% (19/20) for size 5. Those networks were relatively compact: 16.4 mean recruits for size 3, and 19.2 mean recruits for size 5. They trained within 1600 epochs: means of 1473.1 for size 3, and 1537.4 for size 5. Networks that failed to find solutions also got stuck in local reward maxima, oscillating between a few solutions present in their working memory.

Our results suggest that considering a small number of alternatives is beneficial. Fewer alternatives increase the likelihood of networks getting stuck in local maxima. More

alternatives encourage exploration, but may result in systems that become excessively complex without converging. Our results are compatible with estimates of human working memory size: the traditional 7 +/- 2 [12] and the updated estimate of 4 +/- 1 [11]. Fig. 2 presents an example of exploratory behaviour for a working memory size of 5. Because of its computational advantages and psychological plausibility, we used a working memory size of 5 in all further simulations.

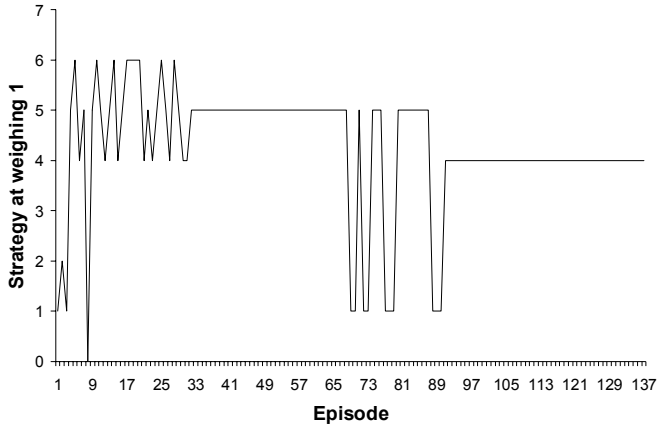

Fig. 2 Example of exploratory behavior in a network. The Y axis presents the strategy used in the first weighing (for example, 3 means the system weighed 3 balls against 3 on the first weighing). As we can see, this network explores different strategies especially at the beginning, then settles for 5/5 for a while, toggles between 1/1 and 5/5 to finally settle for the correct strategy (4/4) and successfully solve the task in 137 episodes.

B. Comparison with human performance

We compared model performance with human participants in the reinforcement learning group, i.e., those who were only told if their answers were correct or not. Those participants completed a mean of 17.0 episodes (min: 6, max: 37, std dev = 7.9) during the 30 minutes that the experiment lasted [5]. An epoch is a pass through the 24 possible cases. Thus humans had less than one epoch of training. Performance results are summarized in Table III.

TABLE III
SUMMARY OF HUMAN PERFORMANCE COMPARED WITH A SOFTMAX ALGORITHM WITH WORKING MEMORY SIZE OF 5

	Humans	Softmax (n=5) simulation
Training	17.0 episodes	1 epoch (24 episodes)
Accuracy	0.59 (std dev = 0.49)	0.21 (std dev = 0.07)
Selection complexity	2.24 (std dev = 0.31)	2.63 (std dev = 0.37)
Selection asymmetry	0.69 (std dev = 0.58)	0.90 (std dev = 0.37)

To match the amount of learning that humans received, all simulations are performed with a single epoch. In Table III, we compare humans vs. the simulation with 5 alternatives limited to one training epoch.

We compared human and simulation accuracy using a one way ANOVA. Accuracy was significantly higher in human (0.59) than simulations (0.21), $F(1,39) = 70.8$, $p < 0.001$. Average performance for 20 networks is presented in Fig. 3. As we can see, networks take around 500 epochs to reach human performance level.

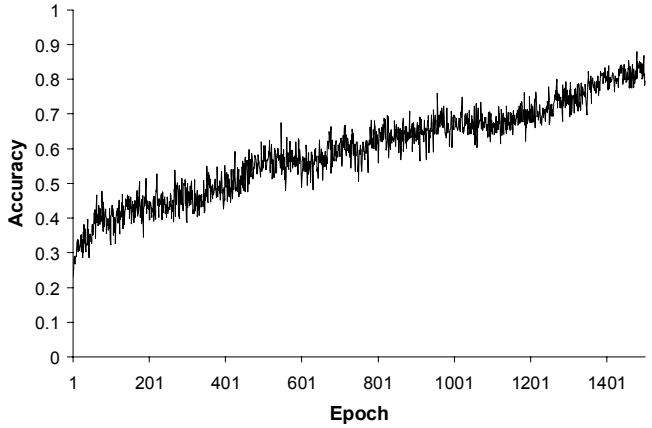

Fig. 3 Mean accuracy of 20 networks trained on the ball weighing experiment for the first 1500 episodes. Softmax working memory size was 5.

To analyse complexity and asymmetry indexes, 3-way, repeated ANOVAs were performed, with type as an independent factor with 2 levels (simulation and human), correct as a repeated factor with 2 levels (errors or correct), and weighing as a repeated factor with 3 levels (1, 2 and 3).

Selection complexity in human participants (2.24) was significantly lower than in simulations (2.63), $F(1,38) = 17.2$, $p < 0.001$. Selection asymmetry was significantly lower in humans (0.69) than simulations (0.90), $F(1,38) = 5.3$, $p < 0.05$.

C. Biases

To make our model more psychologically plausible, we introduced a simplicity bias by penalizing complexity. We went through all 5,671,402 actions in batches of 100,000 (except for the last batch which contained the remaining actions). For each action, we computed its Q value and penalized it according to:

$$Q_{penalized} = Q_{init} - 0.25 * complexity \quad (2)$$

where complexity is computed as described previously in the section on biases (examples are presented in Table I). Recruits in cascade-correlation were limited to two per batch. Networks recruited a mean of 40.6 units (S. D. 12.3) during pre-training.

We analyzed the results in a 3-way, 2-repeated-factor design with pretrain as an independent factor of three levels (human, no pretrain network, pretrain network), correct as a repeated factor of two levels (error or correct) and weighing as a repeated factor of three levels (1, 2 and 3).

To keep our analysis concise, we only report the main effect of the pretrain factor. Pretrain had a significant main effect on complexity ($F(2,53) = 11.9$, $p < 0.001$), but no significant main effect on asymmetry ($F(2,53) = 2.96$, $p > 0.05$). Post-hoc tests on complexity reveal that pretrained networks did not significantly differ from humans. Results for complexity indices are presented in Fig. 4 and Fig. 5.

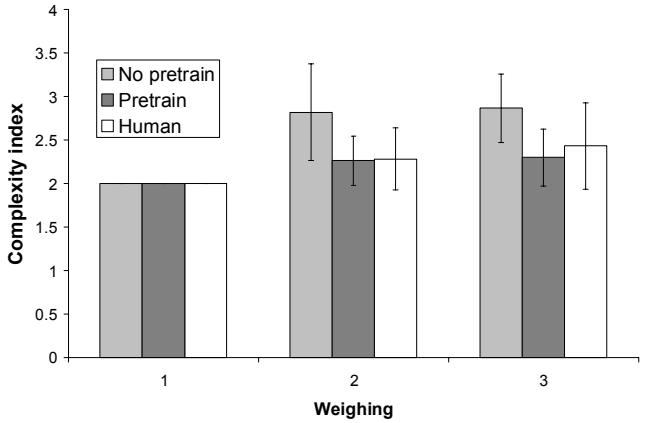

Fig. 4 Selection complexity index for error trials

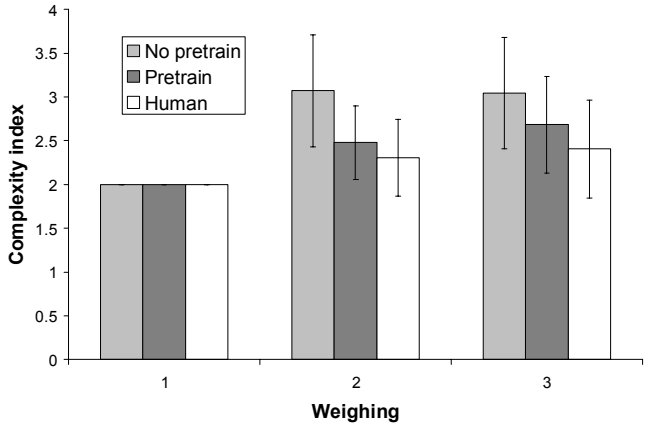

Fig. 5 Selection complexity index for correct trials

Pretrain had a significant effect on accuracy ($F(2,58)=49.0$, $p<0.001$). Tukey post-hoc tests reveal that accuracy of non-pretrained (0.21) and pretrained (0.21) simulations are not significantly different. Human accuracy (0.59) was significantly higher than simulation accuracy. Networks recruited a mean of 40.6 hidden units (std error = 2.6) to learn bias.

To sum up, we were able to capture human preference for selection simplicity by penalizing complexity in a pretraining phase. This bias had no impact on simulation accuracy.

IV. Discussion

We found that a SARSA-based simulation using cascade-correlation as a function approximator is less accurate (0.21) than human participants (0.59). We hypothesize that this accuracy difference could be caused by the following:

1. Humans, as opposed to the systems presented here, use reasoning or mental rehearsing. As they search for the action to take, humans can mentally play different alternatives and predict rewards they would get without needing to take those actions. By contrast, the system presented here lacks such capacity: in order to learn, all actions have to be overtly taken because the only source of rewards for our system is the environment. Therefore, humans can probably forego some overt search but nevertheless learn by reasoning and self-generated reinforcement. Incidentally, this task can be solved on paper without explicit feedback at all. Evidence gathered using Think Aloud Protocols also suggests that many participants can correctly assess the quality of their solution without any explicit reinforcement signal.
2. In combination with reasoning, humans probably also use means-ends analysis [13]. Humans often select actions that move them closer to goal states [6]. Think Aloud Protocols further suggest that they actively monitor distance to goal states to guide their choice of action. In particular, humans report seeking to exclude as many balls as possible. Balls that can be excluded as possible targets are those labelled as normal. Humans are thus likely to use distance to goal states to self-generate rewards in non-terminal states. Actions that lead to states closer to goal states get higher rewards.

Humans might get relatively rich information due to self-generated rewards based on distance to goals. In contrast, agents in our simulation only get three values of feedback on terminal states (+1 for correct, -2 for errors and -1 when they do not provide an answer) but cannot search mentally. In non-terminal states, humans can use distance as self-generated reward on every action taken. By contrast, simulations run only on estimated rewards provided by SARSA. For SARSA to get good estimates, it must explore more states and actions that humans typically do. Further analyses would be necessary to quantify precisely differences between human and network strategies.

In terminal states, humans might also use self-generated rewards to evaluate solutions. For example, they may generate higher rewards for strategies where they need to guess among 2 balls than among 5 after having used up weighings. In contrast, simulation agents always get a reward of -1 for not answering within three weighings.

We plan to model reasoning and means-ends analysis in our future models. To model reasoning, we could use techniques such as TD-leaf which combines search with reinforcement learning in a single system [14]. We could also use closeness to goal states as a reward term for every action, in addition to the current rewards provided by the environment. A combination of look-ahead and self-generated rewards could potentially make a closer fit to human behavior and also improve model performance.

A. Biological plausibility

The basal ganglia, with dopamine and the striatum may well learn using a SARSA-like mechanism [2, 3, 15]. SARSA requires some mechanism to enumerate actions available in a given state. In our simulation, we used an explicit, exhaustive and serial enumeration of actions. However, the brain is likely to work on a different principle. For example, sets of concurrently active neurons could be representing different options or actions [16]. Softmax selection could be the result of competition among those neurons. Such pattern of activity has been observed in the frontal cortex before a decision is taken [16].

In short, the serial search process used in our simulations might be implemented as a parallel process with competition in brains. The net result is similar: only a few alternative actions are actually brought to working memory.

V. Conclusion

We have shown that the difficult selection subtask of the ball weighing problem can be learned by reinforcement alone using only rewards provided by the environment. This shows that SARSA is a powerful technique that can learn based on little information if it is given enough chance to explore the state-action space. The time and amount of exploration that was necessary is compatible with observed human failure in the experiment: too many states to explore and inadequate opportunity to do so.

With only environmental rewards, simulation accuracy was lower than humans. We argue that humans use distance to goal as additional information for solving this task. We suggest that using distances to goals as self-generated rewards is a possible way to model this source of information. We expect that enriching the density of rewards in this manner would result in better models.

We also found that a small number of alternatives (3 to 5) in the number of actions is optimal for this problem, which is compatible with estimates of human working memory size.

Finally, we have successfully captured human preference for selection simplicity by penalizing selection complexity, but this had no impact on simulation accuracy.

It would be interesting to see whether these results would generalize to other planning-intensive tasks in which problem spaces are large.

Acknowledgment

The research was supported by a McGill Major scholarship to F. D., and a Discovery grant to T. R. S. from the Natural Sciences and Engineering Research Council of Canada. We thank Yoshio Takane for proof-reading and suggestions.

References

[1] G. J. Tesauro, "Temporal difference learning and TD-Gammon," *Communications of the ACM* vol. 38, pp. 58-68, 1995.

[2] R. E. Suri and W. Schultz, "A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task," *Neuroscience* vol. 91, pp. 871-890, 1999.

[3] J. C. Houk, J. L. Adams, and A. G. Barto, "A Model of How the Basal Ganglia Generate and Use Neural Signals that Predict Reinforcement," in *Models of Information Processing in the Basal Ganglia* J.C.Houk, J. L. Davis, and D. G. Beiser, Eds.: MIT Press, 1995, pp. 249-270.

[4] K. J. Holyoak, "Problem solving," in *Thinking: An Invitation to Cognitive Science, 2nd edition.* vol. 3, E. E. Smith and D. N. Osherson, Eds. Cambridge, MA: MIT Press, 1995, pp. 267-296.

[5] F. Dandurand, M. Bowen, and T. R. Shultz, "Learning by Imitation, Reinforcement and Verbal Rules in Problem Solving Tasks," in *Third International Conference on Development and Learning (ICDL'04) Developing Social Brains*, La Jolla, California, USA, 2004, p. 26.

[6] F. D. Dandurand, T. R. Shultz, and K. H. Onishi, "Strategies, Heuristics and Biases in Complex Problem Solving," in *CogSci*, 2007, in press.

[7] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, A Bradford Book, 1998.

[8] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in neural information processing systems 2, D. S. Touretzky (ed.),* pp. 524-532, 1990.

[9] T. R. Shultz, *Computational Developmental Psychology*: Bradford Book, 2003.

[10] F. Rivest and D. Precup, "Combining TD-learning with Cascade-correlation Networks," in *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, 2003, pp. 632–639.

[11] N. Cowan, "The magical number 4 in short-term memory: A reconsideration of mental storage capacity," *Behavioral and Brain Sciences,* vol. 24, pp. 87-125, 2000.

[12] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information," *Psychological Review* vol. 63, pp. 81–97, 1956.

[13] A. Newell and H. A. Simon, *Human problem solving*. Englewood Cliffs, NJ: Prentice-Hall, 1972.

[14] J. Baxter, A. Tridgell, and L. Weaver, "TDLeaf(lambda): Combining Temporal Difference Learning with Game-Tree Search," in *In Proceedings of the Ninth Australian Conference on Neural Networks*, Brisbane QLD, 1998, pp. 168-172.

[15] K. Samejima, Y. Ueda, K. Doya, and M. Kimura, "Representation of action-specific reward values in the striatum," *Science,* vol. 310, pp. 1337-1340, 2005.

[16] P. Cisek and J. F. Kalaska, "Neural correlates of reaching decisions in dorsal premotor cortex: specification of multiple direction choices and final selection of action," *Neuron,* vol. 45, pp. 801-814, 2005.