Including cognitive biases and distance-based rewards in a connectionist model of complex problem solving

Frédéric Dandurand [1+]

Thomas R. Shultz [2]

Arnaud Rey [3]

[+] Corresponding author

[1] Department of Psychology, Université de Montréal

90 Vincent-d'Indy Avenue, Montreal, Quebec, H2V 2S9, Canada

Tel: ++1 514 343 4617

Email: frederic.dandurand@gmail.com

[2] Department of Psychology and School of Computer Science, McGill University

1205 Dr. Penfield Avenue, Montreal, Quebec, H3A 1B1, Canada

Tel: ++1 514 398 6139

Email: shultz@psych.mcgill.ca

[3] Laboratoire de Psychologie Cognitive, CNRS – Aix-Marseille University

3, place Victor Hugo, 13331 Marseille, France

Tel : ++33 413 550 995

Email: arnaud.rey@univ-provence.fr

Abstract

We present a cognitive, connectionist-based model of complex problem solving that integrates cognitive biases and distance-based and environmental rewards under a temporal-difference learning mechanism. The model is tested against experimental data obtained in a well-defined and planning-intensive problem. We show that incorporating cognitive biases (symmetry and simplicity) in a temporal-difference learning rule (SARSA) increases model adequacy – the solution space explored by biased models better fits observed human solutions. While learning from explicit rewards alone is intrinsically slow, adding distance-based rewards, a measure of closeness to goal, to the learning rule significantly accelerates learning. Finally, the model correctly predicts that explicit rewards have little impact on problem solvers' ability to discover optimal solutions.

Key words

Problem solving, computational modeling, reinforcement learning, temporal-difference learning, cognitive biases, distance-reduction heuristic

1. Introduction

Research in problem solving has traditionally focused on the discovery of explicit strategies using mechanisms that emphasized search (e.g., Newell, 1990), induction (e.g., Holland, Holyoak, Nisbett, & Thagard, 1986), use of heuristics (e.g., Gigerenzer, Todd, & ABC Research Group, 1999; Polya, 1957) and reasoning by analogy (e.g., Holyoak & Thagard, 1996). Computational models have reflected these emphases. There has been, however, an ever growing interest in learning, modeling how cognitive agents learn to solve problems. A number of mechanisms have been proposed, including chunking the results of look-ahead search into a new rule (Newell, 1990), compiling rules that fire successively into a new rule (Taatgen & Lee, 2003), and reinforcement learning of rules (Fu & Anderson, 2006), or strategy precedence (Rieskamp & Otto, 2006).

Learning to solve problems is the central theme of the current paper. We adopt a theoretical framework grounded in reinforcement learning theory: using a temporal difference (TD) learning mechanism (Sutton & Barto, 1998), problem solvers learn to accurately predict how much reward to expect, that is, the long-term value of taking some action in some problem state. TD-learning and reinforcement learning in general have a long history of success, both for biological and for machine learning. Regarded as biologically plausible (Houk, Adams, & Barto, 1995), TD-learning has successfully captured many classical and operant conditioning phenomena (Suri & W. Schultz, 1999; Sutton & Barto, 1990).

Although often assimilated with simple and low-level associative learning, reinforcement learning has also been used recently for the modeling of high level cognition, a qualification that applies to problem solving (e.g., Daw & Frank, 2009). Research on human problem solving is largely dominated by classical information processing theories (Holyoak, 1995). These theories rest on the problem-space hypothesis (Newell, 1980) which states that problems can be described in terms of states, operators and constraints. It is perhaps underappreciated that these concepts have direct equivalents in reinforcement learning, making information processing theories and reinforcement learning compatible and complementary.

TD-learning is a powerful mechanism by which problem solvers can learn complex tasks by trial-and-error based on infrequent and impoverished explicit rewards – as little as one binary bit of information (success or failure) for the evaluation of sequences containing multiple actions. Learning with so little information is possible, but notoriously slow. So called ‘explicit environmental rewards’ can be positive (e.g., food) or negative (e.g., electric shocks); and they are typically modeled as real numbers whose sign corresponds to the valence (positive or negative) and magnitude to the strength of the reward.

In previous work, we explored how a computational model could learn to solve a complex problem based on binary environmental rewards (Dandurand & Shultz, 2009). The model was able to learn the task using TD-learning, but exhibited two important limitations: (1) model-selected actions were more complex and asymmetrical than human-selected actions, and (2) models learned much more slowly than humans.

In the current paper, we address these limitations by extending a basic TD-learning algorithm with two novel terms: (1) cognitive biases towards simplicity and symmetry, and (2) self-generated rewards based on the closeness to goal, dubbed distance-based reward (DBR): the closer an agent gets to the goal state, the more DBR it gets. DBR is a distance-reduction heuristic similar to both hill-climbing and means-ends analysis. Our hypothesis is that the dense and rich information provided by distance-based rewards can make reinforcement learning a viable candidate in the problem-solving domain.

1.1 Models of problem solving

In this section, we review some influential systems for problem solving that use TD-learning or distance-reduction heuristics. Machine learning systems and cognitive models are both considered. Table 1 provides a summary characterization of these systems. To be designated as cognitive, systems have to be explicitly compared to experimental data of human problem solving, or at least be presented and discussed in the context of cognitive or neurological plausibility. The second column indicates whether a system uses a neural network (NN) implementation[1]. Finally, the last three columns indicate whether a system implements, respectively, TD-learning, distance-reduction heuristics, and cognitive biases. As we elaborate in the following sections, the table suggests that the cognitive model presented here offers a unique combination of TD-learning, distance-reduction, cognitive biases and connectionism.

[1] In contrast, many traditional problem solving models store knowledge as explicit rules or productions.

	cognitive model	neural networks	TD-Learning	distance-reduction heuristics	cognitive biases
(Bianchi, Ribeiro, & Costa, 2008; Polat & Abul, 2002; Provost, Kuipers, & Miikkulainen, 2006)	No	No	Yes	Yes	No
(Kaplan & Güzeliş, 2001; Parks, Levine, & Long, 1998)	No	Yes	No	No	No
(Baldassarre, 2002; Rummery, 1995; Tesauro, 1995; Thrun, 1995)	No	Yes	Yes	No	No
(Asgharbeygi, Nejati, Langley, & Arai, 2005; Nason & Laird, 2004)	Partially	No	Yes	No	No
(Sun & Sessions, 1998, 2000)	Partially	Yes	Yes	No	No
(Langley & Allen, 1993; Langley, Choi, & Rogers, 2009)	Partially	No	No	Yes	No
(Busemeyer & Myung, 1992)	Yes	No	No	Yes	No
(Kaplan & Güzeliş, 2001; Parks et al., 1998)	Yes	Yes	No	No	No
(Fu & Anderson, 2006; Rieskamp & Otto, 2006)	Yes	No	Yes	No	No
(Akyurek, 1992; Veloso et al., 1995)	Yes	No	No	Yes	No
(Cutini, Ferdinando, Basso, Bisiacchi, & Zorzi, 2008; Simen, Polk, Lewis, & Freedman, 2002)	Yes	Yes	No	No	No
(Dandurand & Shultz, 2009)	Yes	Yes	Yes	No	No
Model described in this paper (DBR)	Yes	Yes	Yes	Yes	Yes

Table 1 - Characteristics of some influential problem solving models and systems. Columns indicate if the model is cognitive, if it uses neural networks (NN), and whether it implements TD-learning, heuristics, or cognitive biases for simplicity and symmetry.

1.1.1 Connectionist systems and models for problem solving

Connectionist-based systems for reinforcement learning have been successfully developed in the machine learning community to solve complex problems, for instance, robot navigation (Baldassarre, 2002; Rummery, 1995), backgammon (Tesauro, 1995) and chess (Thrun, 1995). Other systems used neural networks to learn Tower of Hanoi problems in unsupervised fashion (Kaplan & Güzeliş, 2001; Parks et al., 1998).

While using neural networks is interesting from a psychological perspective, the primary concern of machine learning is achieving the best possible performance while biological and psychological plausibility of the mechanisms involved are often of limited interest. As far as we know, only a few connectionist models of human problem solving have been proposed (e.g., Cutini et al., 2008; Simen et al., 2002), and none address how learning can occur by reinforcement (except Dandurand & Shultz, 2009).

1.1.2 Reinforcement learning in symbolic cognitive models

There are a few symbolic-rule systems that use reinforcement learning for problem solving. An ACT-R model recently simulated nondeliberative decision making on a task involving sequential choices (Fu & Anderson, 2006). Similarly, a symbolic model called SOAR (State Operator And Result) has been proposed that uses reinforcement learning to optimize the choice of rules that maximize expected rewards (Nason & Laird, 2004). Reinforcement learning was used in another SOAR model to optimize search under limited resources when exhaustive inference could not be performed (Asgharbeygi et al., 2005). Finally, Rieskamp and Otto (2006) presented a theory in which strategy selection for solving inference problems is based on reinforcement learning.

1.1.3 Extracting rules based on reinforcement learning

In contrast to symbolic models which require prior knowledge in the form of explicit rules, Sun and Sessions (1998, 2000) proposed an approach in which explicit rules are instead extracted a-posteriori from a system trained using reinforcement learning. Expected values are learned in a backpropagation neural network. The approach was tested on two robotics tasks and on a minefield navigation task for which human performance had been previously documented (Gordon, A. Schultz, Grefenstette, Ballas, & Perez, 1994; Sun, 1997).

1.1.4 Cognitive biases

In the present model, we consider two important and well-documented cognitive biases: simplicity and symmetry. To our knowledge, no previously existing computational model of human problem solving explicitly implements such biases.

Firstly, cognitive biases towards simplicity can be found across the cognitive system, including low- and high-level perception, concept learning, categorization, language acquisition, scientific inference and high-level cognition (e.g., Chater & Brown, 2008; Feldman, 2003, 2009; Freyd & Tversky, 1984; Pizlo, 2008; Pothos & Chater, 2002; see Chater & Vitányi, 2003 for a review). A leading hypothesis to explain these simplicity biases is that much of cognition concerns compression (Wolff, 1982) and elimination of redundancy (Barlow, Kaushal, & Mitchison, 1989). Not only are the resulting representations more cognitively economical to store and process, there is also evidence that simpler representations tend to generalize better (e.g., Son, Smith, & Goldstone, 2008).

Secondly, the symmetry bias is thought to have an evolutionary basis, as generalization to mirror stimuli is often adaptive; for instance, an organism that learns to avoid a danger coming on the right side would certainly benefit from also being able to avoid this danger coming from the left side, without needing to be trained again (Rollenhagen & Olson, 2000). Symmetry biases have been directly measured in the brain; for instance, the inferotemporal neurons of monkeys trained to recognize wire-frame objects generalized their responses to stimuli rotated by 180 degrees around the vertical axis (Logothetis & Pauls, 1995). An important mechanism in this spontaneous generalization to mirror-symmetry appears to be the interhemispheric connectivity due to the corpus callosum. In fact, resection of the corpus callosum destroys this spontaneous generalization (Beale, Williams, Webster, & Corballis, 1972). While often adaptive, spontaneous mirror-symmetry generalization can produce errors and confusions, and such mirror-symmetry confusions are in fact ubiquitous (Corballis & Beale, 1976). This mirror-symmetry bias has important implications for high-level cognitive tasks; for instance, spontaneous generalization was found in naming tasks (Tarr & Pinker, 1989) and visual priming (Biederman & Cooper, 1991; Fiser & Biederman, 2001). Furthermore, children learning to read typically go through a mirror stage in which they confuse mirror letters such as b and d; and can write indifferently in both directions (Walsh & Butler, 1996). Children need to learn to break or inhibit the symmetry bias to avoid excessive generalization. In expert readers, symmetry breaking appears specific to word stimuli, and remains present for picture stimuli (Dehaene et al., 2010).

In problem solving, simplicity and symmetry biases involve the tendency to choose simple and symmetrical actions or solution steps. Such biases can be useful for guiding problem solving – for instance, a useful heuristic called divide-and-conquer involves breaking down a complex task in simpler steps, that are often regular and applied recursively (Polya, 1957). On the other hand, simplicity and symmetry may instead hinder the capacity of problem solvers to find appropriate solutions, and are often associated with functional fixedness (Duncker, 1945), problem solving sets (Luchins, 1942), and conceptual blocks (Adams, 1974). In the gizmo task studied in the present research, we previously found that humans tended to select simpler and more symmetrical solution steps than correct solutions require (Dandurand, Shultz, & Onishi, 2007).

1.1.5 Distance-reduction heuristics

A number of models have been proposed that use some distance-reduction heuristic, namely hill climbing or means-ends analysis. In a mathematical approach to human decision making, Busemeyer and Myung (1992) used hill-climbing to model how individuals learn to fine tune decision rules. Means-ends analysis has been studied and implemented in symbolic cognitive models, namely Prodigy (Veloso et al., 1995), SOAR (Akyurek, 1992; Newell, 1990) and ICARUS (Langley & Allen, 1993; Langley et al., 2009). In these models, means-ends analysis was used to guide the generation of problem solving plans or strategies based on explicit rules without involving reinforcement learning. Finally, a few proposals for combining distance-reduction heuristics and reinforcement learning have been proposed for machine learning applications (Bianchi et al., 2008; Polat & Abul, 2002; Provost et al., 2006).

1.2 Organization of the article

The rest of the article is organized as follows. We first briefly present experimental data from participants who learned to solve a complex problem, the Gizmo Problem Solving Task, using explicit environmental rewards. We also review a previously proposed computational model of these data and its limitations (Dandurand & Shultz, 2009). Second, we introduce an extended learning rule that includes cognitive biases and a distance-reduction heuristic as improvements over this previous model. Third, we make novel predictions using this resulting new model of human performance on learning to solve the Gizmo Task when provided no explicit feedback. Finally, we present new experimental data involving the Gizmo Task that tests these predictions.

1.3 The Gizmo Problem Solving Task

The present paper extends previous research on a complex problem, called the Gizmo Problem Solving Task, that consists in finding which gizmo in a set of 12 identical-looking gizmos is heavier or lighter than the others using at most three weighings on a two-sided balance scale (Dandurand & Shultz, 2009; previously used in: Dandurand, Bowen, & Shultz, 2004; Dandurand, Shultz, & Onishi, 2008). Our approach is consistent with Newell's (1973) recommendation to focus on a single complex problem and study it thoroughly. Variants of this class of problems are well known logical-mathematical tasks (Guy & Nowakowski, 1995; Halbeisen & Hungerbuhler, 1995) and a version of this class of problems, called the Coin problem was used in a classic psychology experiment on hints (Simmel, 1953).

1.3.1 Experiment with human participants

Among the data previously collected (Dandurand et al., 2004), we focus on a data sample of 20 participants (14 females and 6 males) consisting of McGill University undergraduate (N = 13) and graduate (N = 7) students.

On each trial, a computer program[2] (illustrated in Figure 1) randomly selected a target gizmo and assigned it a heavy or a light weight. Participants were asked to determine, with 3 uses of a balance scale, which of the 12 gizmos was either heavier or lighter than the others. Because gizmos look identical, the target gizmo could only be identified based on weight. We asked participants to keep track of their hypotheses about gizmo weights, with 7 choices for labeling a gizmo: (1) unknown (heavy, light or normal weight), (2) heavy or normal weight, (3) light or normal weight, (4) heavy or light weight, (5) heavy, (6) light, and (7) normal.

At the beginning of a trial, all 12 gizmos were labeled as unknown weight. Participants installed a certain number of gizmos on the balance scale; any combination of gizmos was allowed. They would then press the weight button and the scale would tip or remain balanced to indicate one of three results: left heavier, right heavier, or equal weight. Participants would then update the gizmo labels to reflect their updated hypotheses about possible weights of each gizmo, and typically repeat this procedure twice (for weighings 2 and 3) with different combinations of gizmos. Participants gave their answer by pressing the Answer button (usually after the third weighing, although they could use fewer). Acceptable answer states consisted of labeling one gizmo as Heavy or Light, and the 11 others as normal. Participants had to answer, and thus, they would need to guess whenever they were left with several possibilities after three weighings. Participants were given explicit feedback about the accuracy of the answer provided (correct or incorrect), but they never received any evaluative feedback for their first and second weighings. A new trial was then presented, with 12 gizmos labeled as U, and a different target selected at random. Participants worked on trials for 30

[2] A play-only version of the Gizmo Task, illustrating the reinforcement learning group, is available at:

http://lnsclab.org/html/BallsWeightExperiment/PlayVersion/play.html minutes. Appendix 1 presents an exhaustive description of the solution space, that is, how this task can be solved correctly without guessing.

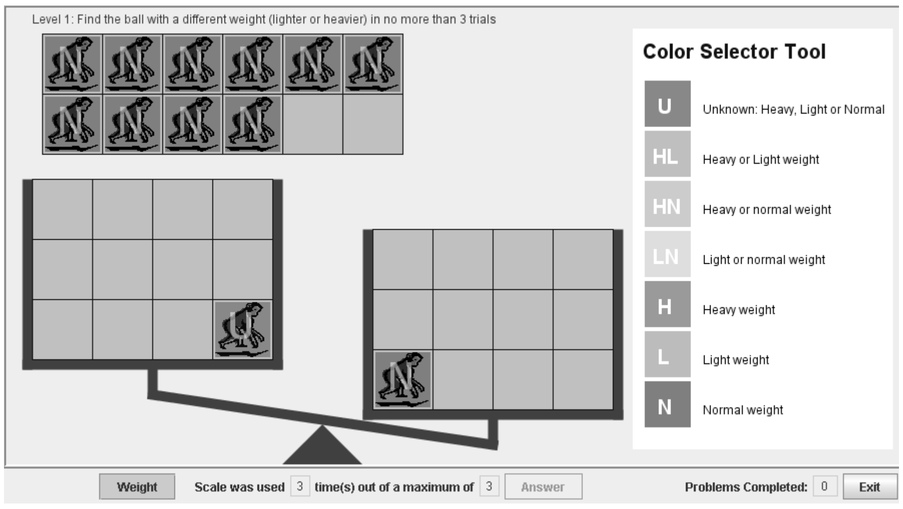

Level 1: Find the ball with a different weight (lighter or heavier) in no more than 3 trials
Color Selector Tool
U Unknown: Heavy, Light or Normal
HL Heavy or Light weight
HN Heavy or normal weight
LN Light or normal weight
H Heavy weight
L Light weight
N Normal weight
Weight
Scale was used 3 time(s) out of a maximum of 3
Answer
Problems Completed: 0
Exit

Figure 1 - The Gizmo Problem Solving task. This example shows the third weighing in which 11 gizmos have been determined to be of Normal weight, and one is of Unknown (U) weight. As a selection action, the participant decided to install 1xU vs. 1xN. The balance scale result indicates that the gizmo labeled as U is lighter than the other one.

We measured accuracy as the proportion of trials on which participants found the correct target gizmo. We also measured the asymmetry and the complexity of the selected actions. Selection actions, or simply actions, consist of two subsets of labeled gizmos to be installed respectively on each side of the balance scale; see Table 2 for some examples. To measure complexity, we sum the total number of labels present on each side of the scale. To measure asymmetry, we count the total number of differences in labels between left and right sides of the scale, i.e., whenever a label is present on one side of the scale but not on the other, one unit of asymmetry is added. Table 2 shows examples of complexity and of asymmetry measures. For this problem, the upper bound of complexity is 12 when items of each of the 6 label categories (U, LN, HN, L, H and N) are installed on both sides of the scale. The upper bound of asymmetry is 6 when items of each category are installed on the scale without a matching element on the other side.

	Example of gizmos installed		
	Left	Right	Index
Complexity	HN HN	N N	2
	HN LN LN	HN LN N	5
Asymmetry	HN HN	N N	2
	LN LN LN	LN LN LN	0
	HN LN LN	HN LN N	1

Table 2 – Example computations of complexity and asymmetry for selection actions under the column labeled "Example of gizmos installed".

1.3.2 Previous computational model

In a previous model, we used TD-learning to successfully learn to solve gizmo problems using environmental rewards only (Dandurand & Shultz, 2009). Details of the model can be found in the methods section below, as the present model extends and improves on that previous model.

1.3.3 Comparison of human and model performance

Participants completed a mean of 18.6 trials. Models were trained to a level roughly equivalent to humans. First, mean accuracy of participants was 0.58 (SE = 0.04) while model accuracy was lower 0.22 (SE = 0.02), but well above chance[3] ($t(40)=6.9$, $p<0.001$). Second, participants selected on average less asymmetrical and complex actions (asymmetry: 0.96; complexity: 2.35) than models did (asymmetry: 1.23, $t(40)=2.6$, $p<0.05$; complexity: 2.74, $t(40)=3.8$, $p<0.001$).

To address these limitations, we now present a new and improved model of the gizmo problem solving task. As mentioned, we hypothesize that the denser and richer information provided by distance-based rewards will allow models to learn faster, and that including cognitive biases will result in more human-like actions.

2. Methods

We now provide more details of the task and the model. Problem solvers need to alternate between two sub-tasks: (1) choosing which gizmos to install and weigh on the balance scale (the selection

[3] Given 12 gizmos x 2 possible weights (heavy or light), chance level of success was 1 in 24, or about 0.04.

task) and (2) keeping track of information about possible gizmo weights based on the result of the weighing (the labeling task). Figure 2 combines a task analysis of the gizmo task with a reinforcement learning framework showing relationships between the learning agent and its environment.

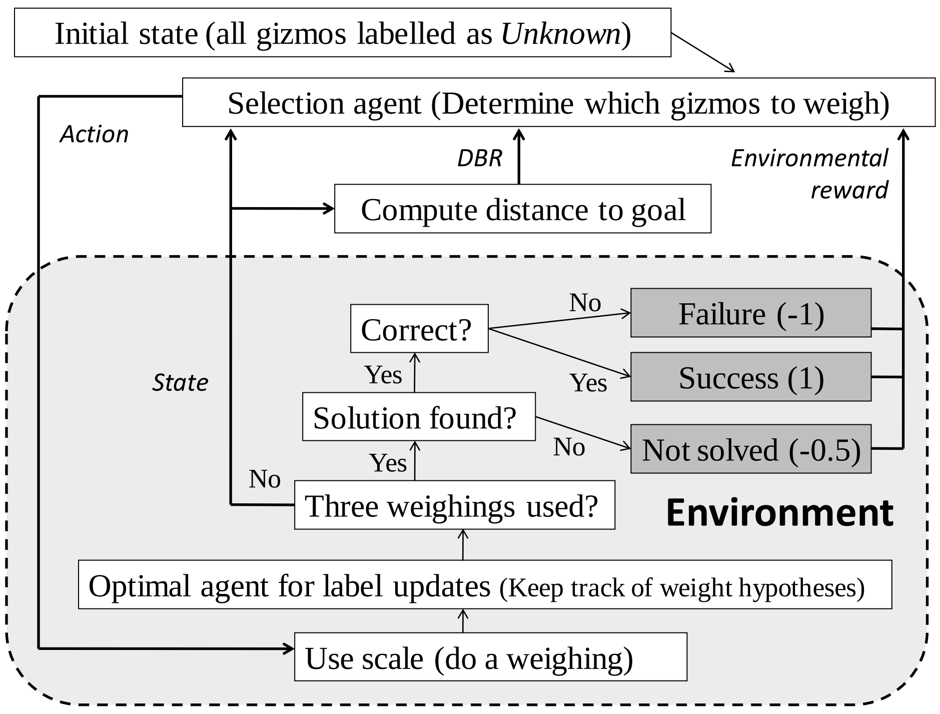

Initial state (all gizmos labelled as *Unknown*)
Selection agent (Determine which gizmos to weigh)
Action
DBR
Environmental reward
Compute distance to goal
State
Correct?
No
Failure (-1)
Yes
Yes
Success (1)
Solution found?
No
Not solved (-0.5)
Yes
No
Three weighings used?
Environment
Optimal agent for label updates (Keep track of weight hypotheses)
Use scale (do a weighing)

Figure 2 – Task analysis and agent-environment interactions for the gizmo problem solving task. From the perspective of the agent making selection decisions, the ideal agent performing label updates is part of the environment.

One of the challenges in modeling this problem is that the two sub-tasks (selection and labeling) form a loop and are thus mutually dependent: the output of one is the input of the other. Therefore, finding the optimal action in one sub-task depends on the other sub-task. Because learning progresses on both sub-tasks, optimal actions are moving targets. As previously (Dandurand & Shultz, 2009), we present a model of the selection task only, assuming optimal updating of labels, thus avoiding the moving-target issue. The ideal agent for label updates can be considered as part of the environment as far as the gizmo selection process is concerned (see Figure 2). The fact that humans performed 95.15% of labeling in accord with an optimal strategy (N = 3444) provides support for the use of such an optimal agent. Sub-optimality consisted in errors in label use (incorrect or inconsistent labeling), and in failures to update labels when new information was available[4]. Errors were often

[4] For instance, in 0.5% of cases, participants incorrectly update labels U to N on the lighter side, appearing to ignore the fact the target can be light, while correctly updating labels U to HN on the heavier side of the scale.

visibly due to GUI-interface manipulation and inattention. In incorrect labeling, participants often generated solutions whose logic strongly suggests they mentally kept track of the gizmo weights despite neglecting or forgetting to update labels explicitly.

2.1 Selection agent

The selection agent enumerates the exhaustive set of possible actions from the current state. States are characterized by the number of gizmos marked using each label type (e.g., 12xU; 4xHN 4xLN 4xN; and 11xN 1xH), and actions consist of two subsets of labeled gizmos to be installed respectively on each side of the balance scale. Actions are selected for processing under two constraints. First, we consider only actions that have the same number of gizmos on both sides of the scale, because humans selected equal numbers of gizmos in 98.6% of their actions, the remaining fraction usually due to GUI manipulation errors. This very likely reflects the fact that the participants in the experiment -- university students -- had prior knowledge and experience using a balance scale, and clearly knew they had to install the same number of gizmos on each side to get a meaningful and informative result. This human expertise with balance scales was implemented as a selection agent that considered only actions that have the same number of gizmos on both sides of the scale. Second, we drop 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.

For each action that can be taken from current state, the selection agent computes the expected reward or quality Q(s_t,a_t) of the given action (a_t) taken from the current state (s_t) using a cascade-correlation neural network (Cascor: Fahlman & Lebiere, 1990). The selection agent can only keep track of the *N* best action alternatives[5]. After the list of possible actions is completely processed, the agent selects one action to perform from its action buffer such that the probability of selecting some action is proportional to the estimate of the expected rewards of that action, a method known as Softmax (Sutton & Barto, 1998). As it progresses in the problem space, the selection agent improves its estimate of expected reward or quality. Estimates of expected rewards generated by cascor are improved using a modified version of SARSA (Sutton & Barto, 1998), which in turn become targets for cascor to learn. Each of these aspects is described in detail in the following sections.

[5] Due to various cognitive limitations, humans probably also have a limited number of alternative actions they can select from, at any given moment.

2.2 SARSA: The standard TD-learning technique

As an extension of the previous model (Dandurand & Shultz, 2009), the proposed model is based on SARSA, a standard reinforcement-learning algorithm from the class of temporal-difference techniques (Sutton & Barto, 1998). SARSA was named after the quintuple that the algorithm uses (s_t, a_t, r_{t+1}, s_{t+1}, a_{t+1}):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)\right] \qquad \text{Equation 1}$$

where *s* is a state; *a* is an action; r is a reward; indices *t* and *t+1* are used for current and next states and actions respectively; α is a learning rate; γ is a discount factor; and Q (for quality or value) indicates how much discounted rewards the agent expects to obtain for taking action *a* in state *s*. Q is thus an estimation of the long-term value of taking some action *a* from state *s*. To estimate Q, SARSA uses the reward actually obtained in the next state (r_{t+1}), and also adjusts its estimate based on the discrepancy between the current Q value and the Q value of the next state-action pair ($Q(s_{t+1},a_{t+1})$-$Q(s_t,a_t)$). In other words, if the next state-action pair has little value, then the estimate for the current state-action pair decreases. This mechanism allows some form of propagation of Q values back in time (from *t+1* to *t*), hence the name temporal-difference for this class of techniques.

In the current paper, we extend standard SARSA to cover distance-reduction heuristics and cognitive biases. Before we present the extended equation, we first describe how we compute distance to goal as well as selection complexity and asymmetry.

2.3 Distance-based rewards (DBR)

Preliminary data from think-aloud protocols of participants solving gizmo problems provides evidence for the use of a distance-reduction heuristic. Namely, participants reported trying to find the gizmos of normal weight and exclude them from possible solutions. This narrowing of possible targets allows them to get closer to the solution.

We measure distance to goal as the sum over all 12 gizmos of their individual distances, as described in Table 3. We hypothesize that problem solvers can compute, or at least estimate, the total distance to goal as the sum of individual distances. For example, a state consisting of 6xHN and 6xLN has a distance of 12. A goal state such as 1xH and 11xN has a distance of 0.5, and the initial state (12xU) has a distance of 24. Solution states have a distance of 0.5[6].

[6] If distances are set to 0 for Heavy, Light and Normal gizmos, then any combination of these three gizmo labels are valid (e.g., 4xH, 2xL and 6xN), and thus solution states are not unique.

Label	Distance
Unknown (U)	2
Heavy or Light (HL)	1
Heavy or Normal (HN)	1
Light or Normal (LN)	1
Heavy (H)	0.5
Light (L)	0.5
Normal (N)	0

Table 3 - Distance to solution used for means-ends analysis. We set distance of heavy and light labels as 0.5 to make the solution unique.

We define distance-based rewards (DBR) as follows:

$$DBR = 1 - \frac{\ln\left(1 + \sum_{i=1}^{12} d(i)\right)}{\ln\left(1 + dist_{\max}\right)} \qquad \text{Equation 2}$$

where *d(i)* are the distances to goal of each gizmo *i* based on Table 3, and dist_{max} is the maximal possible distance to goal, here 24. The equation is globally constructed as (1-(normalized distance to goal)), so that the shorter the distance to the goal the higher the distance-based reward. Normalized distance to goal uses the total sum of distances for individual gizmos, and 1 is added to avoid ln(0) which is not defined. The denominator is a normalization term to scale the range of normalized distance to goal from 0 to 1. Figure 3 presents Equation 2 as a plot.

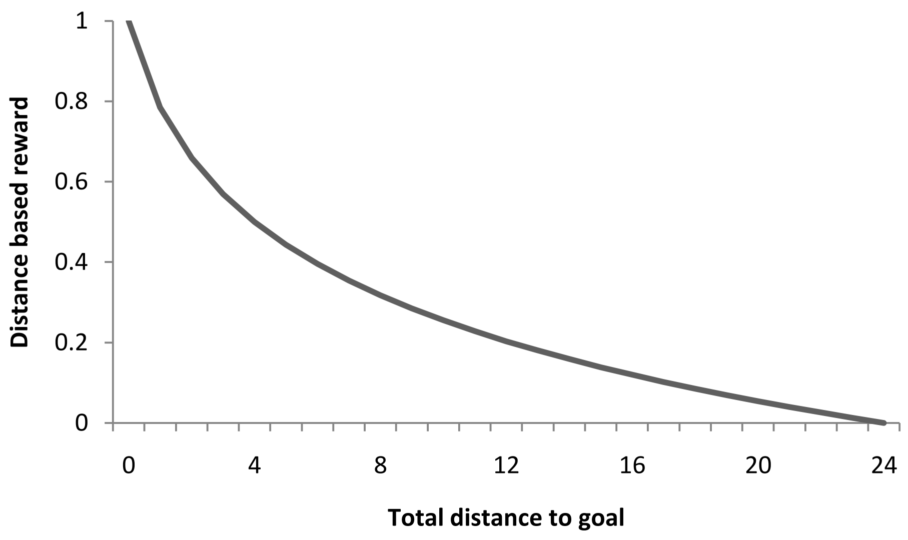


Figure 3 - Distance-based reward as a function of total distance to goal.

2.4 Measures of asymmetry and complexity

We hypothesize that complex and asymmetrical selections are more cognitively demanding to process and reason about, and thus incur more penalty. Thus, we relate complexity and asymmetry to a cognitive cost penalty (CCP) as follows:

$$CCP = -\ln(complexity + asymmetry) \qquad \text{Equation 3}$$

Computations of complexity and asymmetry were presented in the methods section of the human experiment, and examples were given in Table 2. The maximal cognitive cost penalty is 2.5, that is, ln(12) since the maximal complexity is 6 and the maximal asymmetry is 6, as described previously.

2.5 SARSA extensions

We extended the standard SARSA equation (equation 3) by adding two new terms: (1) a contribution of distance-based rewards (DBR) at time step *t+1* (d_{t+1}), and (2) a contribution of cognitive cost penalty ($c(a_t)$). Alongside with standard environmental rewards, these new terms are weighted into an overall Q-value as follows:

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha(\mu r_{t+1} + \beta d_{t+1} + \lambda c(a_t) + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)) \qquad \text{Equation 4}$$

Where:

- r_{t+1} is an environmental reward: +1 for correct answer, -0.5 when the system did not give any answer after three weighings, and -1 for an incorrect answer.
- d_{t+1} is a distance-based reward (DBR), varying between 0 and 1.
- $c(a_t)$ is the cognitive cost penalty for selecting an action.
- μ, β and λ control the respective contributions of environmental rewards, distance-based rewards and cognitive cost penalty respectively.
- α is a learning rate (set to 0.1).
- γ is a discount factor (set to 1.0)[7]

2.6 Action selection

When learning to solve the task, an agent must decide what selection action to perform. We used a Softmax approach, as done previously (Dandurand & Shultz, 2009). Under Softmax, the higher the expected reward Q(s_t,a_t) for action a_t in state s_t, the greater the probability of selecting action a_t, see Equation 5.

$$p_i = \frac{e^{Q(s_t, a_{i,t})}}{\sum_{j=1}^{buffer\ size} e^{Q(s_t, a_{j,t})}}$$

Equation 5

In other words, promising actions are taken more often, but every action has some probability of being selected. Similarly, humans do not always select actions they expect to be the best. They often try out apparently less optimal solutions to see what happens, resulting in further exploration of the

[7] This effectively means no discounting. We chose not to discount because the problem description made no mention that shorter solutions should be preferred. In fact, participants produced only about 10% of solutions involving fewer than three weighings, some of which seemed unintended and due to GUI manipulation errors. Correct and reliable solutions to this problem require the maximal number of weighings allowed (Dandurand & Shultz, 2009). And ultimately, not discounting is appropriate for episodic tasks (Sutton & Barto, 1998).

solution space. In the proposed model, Softmax is applied to a small set of the best possible alternatives, those in the action buffer.

2.7 Connectionist function approximator

To compute estimates of expected rewards, the model uses a cascade-correlation (cascor) neural network function approximator. A function approximator does not store Q values for every encountered state and action. Instead, the Q value is approximated or constructed as a function of states and actions ($Q(s_t, a_t) = f(s_t, a_t)$). The transfer function f is implemented as a neural network here. In contrast to lookup tables in which expected rewards are explicitly and exhaustively stored, neural networks exhibit interesting generalization properties. Cascade-correlation (Fahlman & Lebiere, 1990) is a constructive neural network algorithm for supervised learning. In cascor, computational units are recruited as necessary to solve some task, and installed as new hidden units. This avoids having to design a network topology a-priori, and allows the topology to change as needed. Cascade-correlation has been used successfully to model several cognitive tasks, on which it often performed better than standard backpropagation (e.g., Shultz, 2003; Shultz, Mysore, & Quartz, 2007).

Cascor learns by alternating between input and output phases. In input phases, computational units (here, sigmoid units) in a recruitment pool are trained to maximize covariance with residual network error. At the end of input phase, when covariance does not increase anymore, the unit with the highest covariance is inserted and connected to the current network structure. In this project, we use a variant of cascor called sibling-descendent cascade-correlation (SDCC: Baluja & Fahlman, 1994). SDCC can choose to install units on the current deepest hidden layer (sibling units) or on a new layer (descendent units). SDCC thus creates a greater variety of network topologies, from deep to flat, to suit the problem being learned. In output phases, connection weights feeding output units are trained to minimize network error. Cascor is trained using an algorithm for training feed-forward networks such as QuickProp (Fahlman, 1988) in both input and output phases. Details of cascor parameter settings can be found in the Appendix 2.

SARSA was interfaced with cascor using a caching system (Rivest & Precup, 2003). SARSA is an online technique, generating a data pattern after every action taken. In contrast, Cascor works in batch mode, processing multiple data patterns at once. A cache is thus required to buffer data patterns until there are enough to make a batch to train cascor (here, 100 patterns). To ensure that SARSA uses up-to-date patterns, the proposed system first looks for patterns in the cache. If a pattern is not found, cascor is used to estimate the expected reward or quality (Q) of the state-action pair, which is inserted in the cache. All Q value updates are performed in the cache. When a batch contains enough patterns, the batch is converted into a training set of 100 patterns that cascor learns, after which the cache is emptied and is ready to prepare the next batch of 100 patterns. The cache thus contains up-to-date values as calculated by SARSA for recently traversed state-action pairs.

2.8 Input and output coding

When converting cached patterns into a training set for cascor, a training pattern is generated for every state-action pair present in the cache. Inputs are built as the concatenation of state and action data, resulting in 24 inputs: 6 to code the state and 18 to code the action.

The state indicates the number of gizmos marked using each label type. We drop the heavy or light (HL) label because humans almost never used it. The 6 inputs for state code the proportion of gizmos of each label type in order: U, HN, LN, H, L, N. For example, to indicate the following state: 4U, 4HN,4LN, 0H, 0L, 0N, the input vector is 0.33, 0.33, 0.33, 0.0, 0.0, 0.0, where 0.33 = 4/12.

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

The output is a single continuous value coding the network's estimation of the expected reward or quality of the state-action pair presented at the input. The sigmoid function of the output unit is scaled to match the range of possible rewards.

2.9 A numerical example

In this section, we present a fictive example to illustrate model processing for a single trial comprising 3 weighings. To fully exercise the model, we set learning parameters as follows: $\mu = 1.0$, $\beta = 1.0$ and $\lambda = 1.0$, meaning that the model learns from both environmental (μ) and distance-based rewards (β) under cognitive cost penalty (λ). Action buffer size is set to 4. To refer to the multiple time steps involved in a trial, the following notation is used: *t, t+1* and *t+2* refer to processing that lead to the first, the second and the third weighing, respectively; and *t+3* refers to what follows the third weighing (i.e., relating to the terminal state reached).

Following Figure 2, we have, for the first weighing:

1. Initial state. A trial begins with all gizmos labeled as Unknown, i.e., s_t =12xU.
2. The selection agent determines which gizmos to weigh:
    a. List actions. The selection agent enumerates all possible actions from this state.
    b. Compute Q values for each action in the list using cascor[8].
    c. Select an action from buffer using Softmax.
3. Use scale (do a weighing): The model randomly selects and installs gizmos on the scale according to the action selected. Here, let's say gizmos are installed as follows: gizmo numbers 2, 3, 9, 1, 6 and 10 on the left side of the scale, and numbers 5, 7, 4, 12, 8 and 11 on the right side. Because gizmo 7 is heavier, the right side of the scale will be heavier.
4. Optimal agent for label updates: Gizmos on the left (lighter) side of the balance scale are updated to 6xLN; whereas gizmos on the right (heavier) side are updated as 6xHN. Thus s_{t+1} = 6xHN, 6xLN.
5. Three weighings used? Answer: No, so continue to the next weighing.
6. Compute distance to goal (see Table 3). Here, distance is 6x1+6x1 = 12, and d_{t+1} = 1- ln (1+12) / ln (25) = 0.203 using Equation 2.

The process is repeated for weighings 2 and 3, and various measures are collected, as shown in Table 4. Note that, after weighing 3, "Three weighings used?" is true, but because there are still 2xHN, "Solution Found" is false and an environmental reward of -0.5 is obtained for this trial. After the trial is completed, the model updates Q-value estimates, as shown in the last row of Table 4 (values from the current or the next column may be used depending on indices t and t+1 in Equation 4), and writes these updates back to the cache.

[8] Note that, whenever a cascor access is made (read or write), the model verifies the cache. If the cache is full (that is, contains 100 patterns), a batch is created and used for training of cascor. This consolidates recent SARSA updates into the transfer function (state-action pairs to Q values) of cascor.

	Time t	t+1	t+2	t+3
s_t	12xU	6xHN, 6xLN	6xHN, 6xN	2xHN, 10xN
a_t	(B) 0xU; (L) 6xU; (R) 6xU	(B) 6xHN; (L) 3xLN; (R) 3xLN	(B) 5xHN, 1xLN; (L) 3xLN; (R) 2xLN, 1xHN	N/A
Scale result	Right side heavier	Equal weight	Left side heavier	N/A
r_t	N/A	0	0	-0.5
d_t	0.0 = 1-ln(1+24)/ln(1+24)	0.203 = 1-ln(1+6x1+6x1)/ln(1+24)	0.396 = 1-ln(1+6x1)/ln(1+24)	0.659 = 1-ln(1+2x1)/ln(1+24)
$c(a_t)$	-0.693 = -ln(2)	-0.693 = -ln(2)	-1.386 = -ln(1+3)	N/A
Initial $Q(s_t,a_t)$	0.4	0.5	0.3	N/A
Updated $Q(s_t,a_t)$	0.361 = 0.4+0.1*(0+0.203-0.693+0.5-0.4)	0.450 = 0.5+0.1*(0+0.396-0.693+0.3-0.5)	0.147 =0.3+0.1*(-0.5+0.659-1.386+0-0.3)	N/A

Table 4 - Numerical value for this example of the measures used by SARSA, and other information about states, actions and outcomes.

2.10 Model testing

Network performance is assessed after each learning episode (that is, a pass across all 24 trials). The model is tested on all 24 possible cases (12 gizmos x 2 weights). We measure accuracy, complexity and asymmetry of solutions generated. When tested, models always select the action associated with the highest expected reward. A similar behavior may be expected of humans: when tested, they would do their best (i.e., pick actions with highest expected reward), but they would explore more alternatives when learning (i.e., use a technique analogous to the proposed modified Softmax).

3. Results

The present model has four important parameters: (1) the action buffer size, (2) the new SARSA term for distance-based rewards, (3) the new SARSA term for cognitive cost penalty to enforce simplicity and symmetry biases, and (4) the usual SARSA term for environmental rewards. To avoid a combinatorial explosion of parameter combinations, we devised a systematic approach that varies one parameter at a time.

In a first simulation, we investigate whether models can learn the task with distance-based rewards, i.e., without any explicit environmental rewards and cognitive biases. We manipulate the action buffer size to study its effect on performance, and seek an appropriate size for further simulations. In a second simulation, we compare learning under various combinations of environmental rewards and distance-based rewards with an appropriate action buffer size and no cognitive biases. Finally, in the third simulation, we explore how the addition of cognitive biases for symmetry and simplicity improves model coverage.

3.1 Simulation 1 – Learning using distance-based rewards (DBR)

In the first simulation, we ask whether models can learn the task with distance-based rewards only ($\beta = 1.0$), i.e., without any environmental rewards ($\mu = 0.0$). To study its effect on learning, we vary the action buffer size (i.e., the number of options n considered by Softmax) from 1 to 10. We use no cognitive bias ($\lambda = 0.0$). Networks are trained to perfect accuracy (1.0), or for a maximum of 1000 episodes. We perform 20 replications per buffer size level with different initial conditions, for a total of 200 simulations.

Accuracy results are shown in Figure 4, the number of cascor recruits in Figure 5, and number of episodes in Figure 6 to reach the performance showed in Figure 4. As we can see in Figure 4, models successfully learned the task with distance-based rewards only to near-perfect accuracies (98% to 100%) for action buffer sizes above 3.

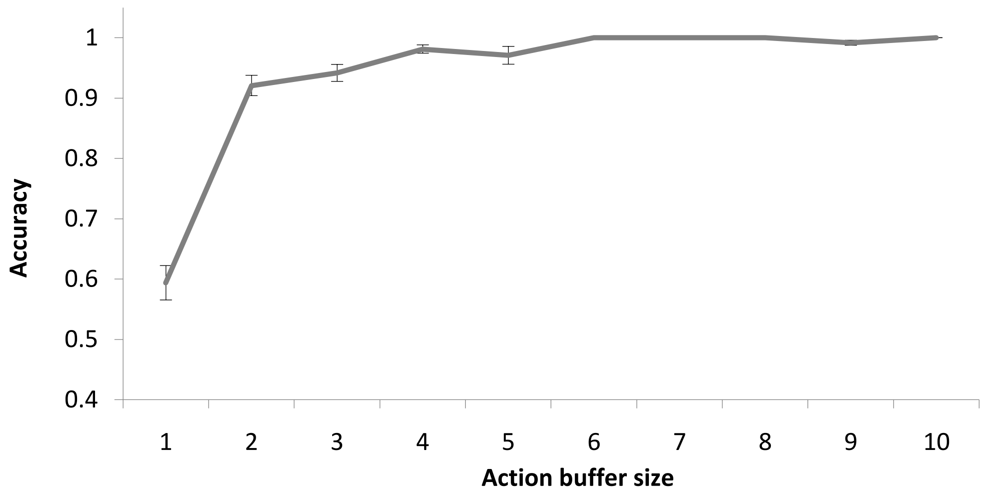


Figure 4 – Model accuracy learning with distance-based rewards only as a function of action buffer size, with standard errors.

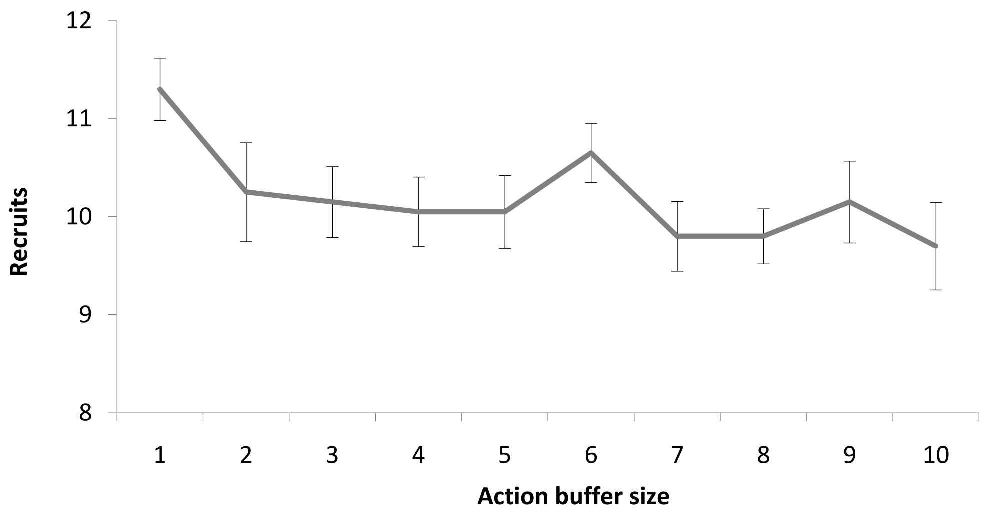


Figure 5 – Number of cascor recruits to reach near-perfect performance (see Figure 4) in model learning with distance-based rewards only as a function of action buffer size, with standard errors

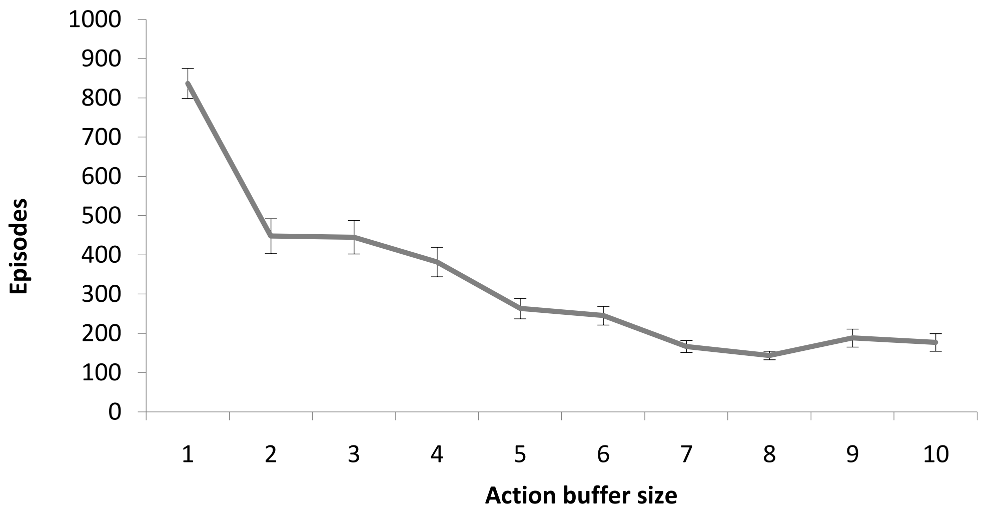


Figure 6 - Number of training episodes to reach near-perfect performance (see Figure 4) in model learning with distance-based rewards only as a function of action buffer size, with standard errors.

We perform one-way ANOVAs with action buffer size (10 levels) as an independent factor. First, an analysis of arcsine-transformed accuracies reveals a significant effect of action buffer size, *F*(9,190) = 23, *p* < 0.001. A Tukey HSD post-hoc test (*p* = 0.05) further reveals two homogeneous subsets, the first one containing an action buffer size of 1, and the second containing action buffer sizes from 2 to

10. Thus, there is no significant difference in accuracy for action buffer sizes of 2 and more. Second, an analysis of cascor recruits reveals no significant effect of action buffer size, *F* < 1, as we can see in Figure 5. Finally, an analysis of the number of training episodes reveals a significant effect of action buffer size, *F*(9,190) = 12, *p* < 0.001. A complementary analysis reveals the linear trend is significant, *F*(9,190) = 12, *p* < 0.001, suggesting training becomes faster as action buffer size increases, as we can see in Figure 6.

What action buffer size is appropriate for learning by distance-based rewards only in these simulations? While the principle of economy favors small sizes, results suggest that increasing buffer size improves learning speed and may also increase accuracy, although accuracy is already near ceiling for buffer sizes as small as 2. The resulting tradeoff suggests a buffer size between 2 and 5 appears appropriate for this problem. In following simulations, we use a buffer size of 4, which offers a good tradeoff between accuracy and learning time.

3.2 Simulation 2 – Learning with distance-based and environmental rewards

In the second simulation, we ask whether environmental or distance-based rewards are better for learning this task. To study learning under different combinations of rewards in Equation 4, we run simulations as a two-independent-factors design:

(1) Contribution of environmental rewards, 2 levels: used ($\mu = 1.0$) and unused ($\mu = 0.0$)

(2) Contribution of distance-based rewards, 2 levels: used ($\beta = 1.0$) and unused ($\beta = 0.0$)

Dependent variables are accuracy and number of cascor recruits. The action buffer size is set to 4, and no cognitive cost is included in these simulations ($\lambda = 0$).

The condition in which models are given environmental rewards only ($\mu = 1.0$, $\beta = 0.0$) replicates a condition reported previously (Dandurand & Shultz, 2009). Adjustments of the cascor parameters to reduce phase shifting result in faster learning. The condition in which environmental rewards and distance-based rewards make no contribution to learning ($\mu = 0.0$, $\beta = 0.0$) acts as a control. Although these control networks are not learning to solve the task, they nevertheless adjust quality of the current state-action pair based on the quality of the next pair, according to Equation 4. These quality values are due to random initial conditions, and thus do not bear any meaningful information.

We find that any combination of environmental or distance-based rewards enables models to learn the task given sufficient training. Figure 7 shows the first 200 episodes of training. For comparison purposes, we also plot average performance of human participants in the control and the reinforcement learning groups (*M* = 0.47), see section 3.2.1 (Testing predictions about learning with and without explicit feedback) for details. It is important to recall that human performance data are available only for less than 1 episode[9], and that the dashed line shows this as a constant. We could certainly expect human performance to increase with training as models do, but testing this hypothesis would require prohibitively long periods of training (extrapolating the data available for 30 minutes, we can estimate that humans would complete one episode of training in about 45 minutes).

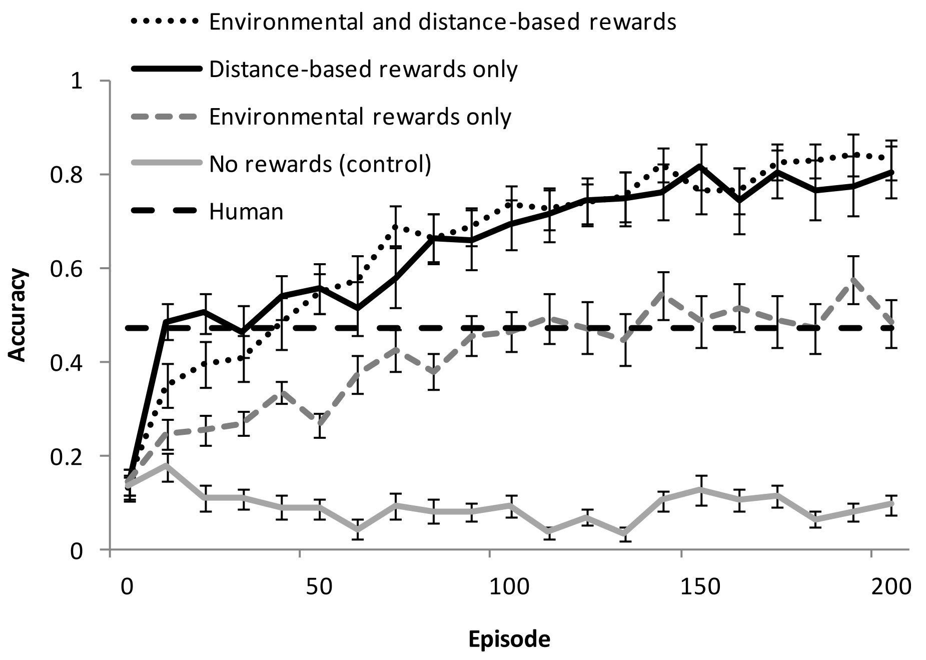


Figure 7 - Accuracy as a function of training episode for the different learning conditions (plotted every 10th episode), with standard error. Human performance is a point value (0.47) shown as a constant for comparison purposes.

We see, first, that models trained only by environmental rewards take much more training (more than 100 episodes) to reach human level accuracy than the two models for which training includes distance-based rewards. Second, we see that models with distance-based rewards only initially perform better than those that also include environmental rewards. The likely explanation is that environmental rewards can sometimes reinforce incorrect strategies that involve guessing because guessing sometimes leads to the correct answer. This gives problem solvers a training signal with higher variance: while sub-optimal strategies are on average rewarded less than optimal strategies,

[9] During the 30 minutes they worked on problems, participants completed an average of about 16 trials. One episode comprises 24 trials (that is, all combinations of 12 gizmos and 2 possible weights, light or heavy).

both can be equally rewarded on some of the trials, namely those in which guessing yields a correct answer. In contrast, the distance-based rewards approach reliably reinforces strategies that most narrow down the set of possible answers, irrespectively of actual outcomes of guessing. The resulting variance of the training signal is thus smaller. In sum, by partially rewarding guessing, models learning with environmental rewards perform worse initially. However, despite the larger variance in rewards obtained from the environment, solutions would stabilize to their correct values (optimality) over the long-run.

In short, all models require more training (that is, several episodes) than humans (less than one episode) to reach equivalent accuracy (about 0.47). However, models learn much faster with distance-based rewards than with environmental rewards. In fact, when distance-based rewards are used, environmental rewards appear at best redundant and provide no additional benefit, and at worse can impair performance by partially rewarding strategies that involve guessing. In the following section, we use this result to make a prediction of performance for humans who are given no explicit feedback.

3.2.1 Testing predictions about learning with and without explicit feedback

We designed an experiment to compare the performance of human participants who are given or not given explicit feedback on this task. The prediction of our model is that participants given no feedback should perform at least as well as participants who are told if their answers are correct or not.

3.2.1.1 Design

The experimental design consisted in manipulating one independent factor, availability of explicit feedback, over two levels: (1) a reinforcement learning group in which participants were told if their answers were correct or not, and (2) a control group receiving no feedback.

3.2.1.2 Participants

The data sample contained forty participants (N=20 per condition); 23 females and 17 were males. Mean participants’ age was 23.2, ranging from 18 to 47 years of age. Participants were recruited through the McGill University subject pool (N=22) or were unselected web users (N=18).

Participants were assigned to one of the conditions (reinforcement or control) in a round-robin fashion, starting with group with the fewest completed participants.

3.2.1.3 Procedure

The experimental procedure is similar to the one used previously (Dandurand et al., 2004) except that the experiment was performed online. For this purpose, the program previously used was adapted as a Java applet that could run online in standard web browsers. The online method was found to be valid for this experiment (Dandurand et al., 2008).

Upon pressing the “Answer” button after the third weighing, the program displayed a message to the participants in the reinforcement learning group indicating if the gizmo they identified as heavy or light was the correct one. In contrast, participants in the control group were told that their answer was recorded and that they would get their accuracy score at the end of the experiment, but they were not told if individual trials were correct or not.

3.2.1.4 Results

Participants completed a mean of 15.5 trials in the reinforcement learning group, and 16.7 trials in the control group. We found that accuracy of the control group (M = 0.48, SE = 0.06) was higher than accuracy of the reinforcement learning group (M = 0.46, SE = 0.05), as we expected. However, this difference in accuracies was not statistically significant, *t* (38) < 1. This nevertheless confirms our prediction about the role of explicit feedback, namely that humans do not need explicit feedback to learn this task, and that performance without explicit feedback is at least as high as performance with explicit feedback. Note that average accuracy of the reinforcement learning group was lower than previously measured in the lab (M = 0.58), see (Dandurand et al., 2008) for more details about differences between the lab and the online methods for the Gizmo task.

These experimental results support the prediction made by the model: when problem solvers can estimate their distance to goal and use this estimation as self-generated rewards, explicit environmental rewards are, at best, redundant and do not further improve learning.

3.3 Simulation 3 - Cognitive bias for symmetry and simplicity

In this last simulation, we assess how symmetry and simplicity biases ($\lambda = 1$ vs. $\lambda = 0$) improve the utility of actions selected. We focus on data of the control group described earlier. Models and humans can be matched in two ways: given equivalent training, or reaching equivalent accuracies. We test whether cognitive biases are effective in both cases.

We manipulate two factors:

1. Amount of model learning (4 levels: 1, 2, 12 and 25 episodes). There are two ways to match human performance. First, as done in previous models, we can give models approximately the same amount of overt training as humans had. We have seen that, in this condition, model accuracy tends to be lower than humans. Second, we can train models more than humans so that accuracies of humans and models approximately match. One episode corresponds approximately to human overt training[10], whereas we empirically find that 25 episodes of training yield model accuracy that approximately matches human-level accuracy (see Figure 7); see the discussion section for further details.

2. Cognitive bias for symmetry and simplicity (2 levels: $\lambda = 0$ for unbiased model; $\lambda = 1$ for biased model). The unbiased model is identical to the one presented in simulation 2.

To assess the effect of biases on selection actions, we perform mixed ANOVAs on complexity and asymmetry measures with model type as an independent factor (2 levels: model with bias and unbiased model) and training level as a repeated factor (4 levels: 1, 2, 12 and 25 episodes). We ignore the first weighing because initial state (12xU) yields a fixed and predictable selection complexity of 2 (Labels U on each side of the scale) and an asymmetry of 0.

A pre-training period is necessary for models to effectively learn to prefer simple and symmetrical solutions from the penalty term λ. During this pre-training period lasting 50 episodes, biased models are rewarded for selecting simple and symmetrical actions, but not for solving the task. That is, they receive no environmental nor distance-based reward (that is, $\mu = 0$, $\beta = 0$, and $\lambda = 1$, in equation 4). When pre-training is finished, models learn to solve the task with distance-based rewards only but retain a preference for simple and symmetrical solutions ($\mu = 0$, $\beta = 1$, and $\lambda = 1$).

Figure 8 shows human and model accuracy as a function of training. Recall that humans got the equivalent of about one episode of training, hence the single point for human accuracy. We can see that model accuracy is below human accuracy given equivalent training, but reaches a comparable level by 25 episodes for both the biased and unbiased models.

We analyze arcsine-transformed accuracies using a mixed ANOVA with training level as a repeated factor (4 levels: 1, 2, 12 and 25 episodes) and bias as an independent factor (2 levels: biased and unbiased). The ANOVA reveals a main effect of training level, $F(1,38) = 20$, $p < 0.001$, stemming from

[10] Humans completed a mean of 18.6 trials which corresponds to less than one episode in reinforcement learning terms (an episode is a pass throughout all 24 different problem trials). Because the model implementation uses the episode as the basic unit for training, we use a single episode of training to approximately match human learning.

an increase in accuracy with training. The effect of bias is not significant, *F*(1,38) < 1. We also find a significant interaction between training and bias, *F*(1,38) = 5.5, *p* < 0.01. This interaction suggests that models that select simple and symmetrical actions are more accurate than unbiased networks, but only early in training (1 episode of training). We return to the possible implications of this interaction in the discussion.

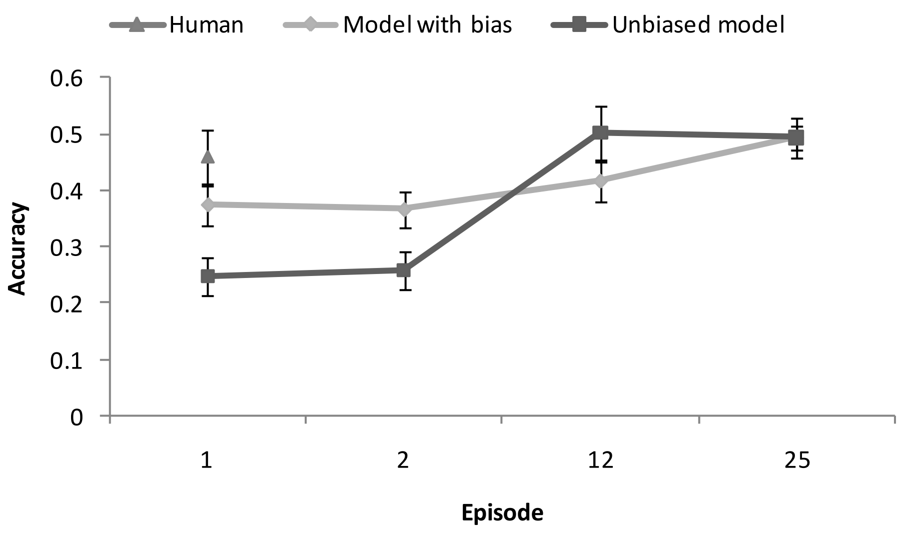


Figure 8 – Human and model accuracy as a function of learning time and biases for simple and symmetrical solutions.

Complexity measure results are presented in Figure 9 and Figure 10 for weighings 2 and 3 respectively. As we can see, unbiased models generate more complex solutions than humans. However, complexity of the solutions generated by the biased models remain comparable to human throughout training levels for both weighings.

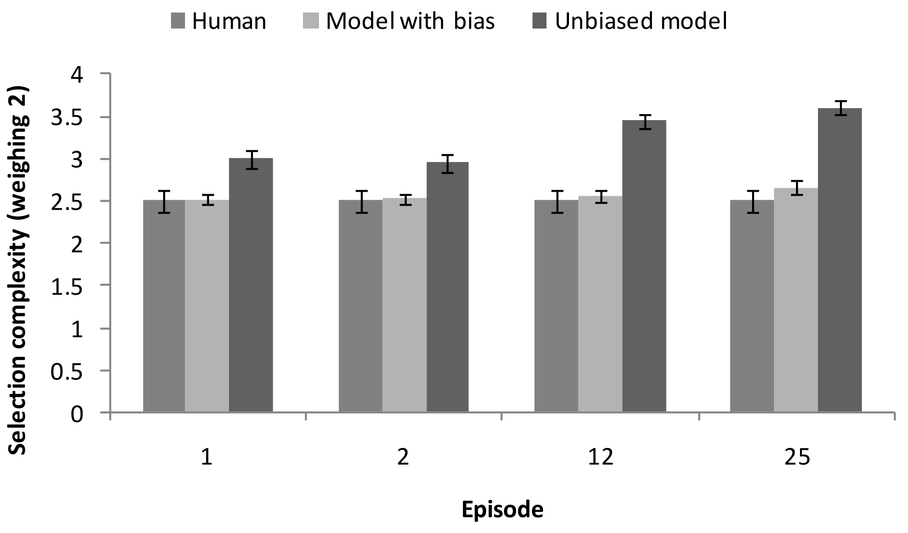


Figure 9 - Complexity of selection actions in weighing 2 for humans and models. Models are trained between a single episode, which is approximately equivalent to over human training, and 25 episodes, which yields an accuracy approximately equal to human level.

We analyze the measure of complexity using a mixed ANOVA with training level as a repeated factor (4 levels: 1, 2, 12 and 25 episodes) and bias as an independent factor (2 levels: biased and unbiased). For weighing 2, the ANOVA on complexity reveals a main effect of model type, $F(1,38) = 62$, $p < 0.001$, stemming from larger complexity in the unbiased model (M= 3.2) than the biased ones (M=2.6). We also found a significant effect of training, $F(1,38) = 16$, $p < 0.001$, stemming from an increased complexity between one episode of training (M=2.75) and 25 episodes (M=3.1), and an interaction, $F(1,38) = 7.7$, $p < 0.001$, stemming from a larger increase in the unbiased model than the biased one.

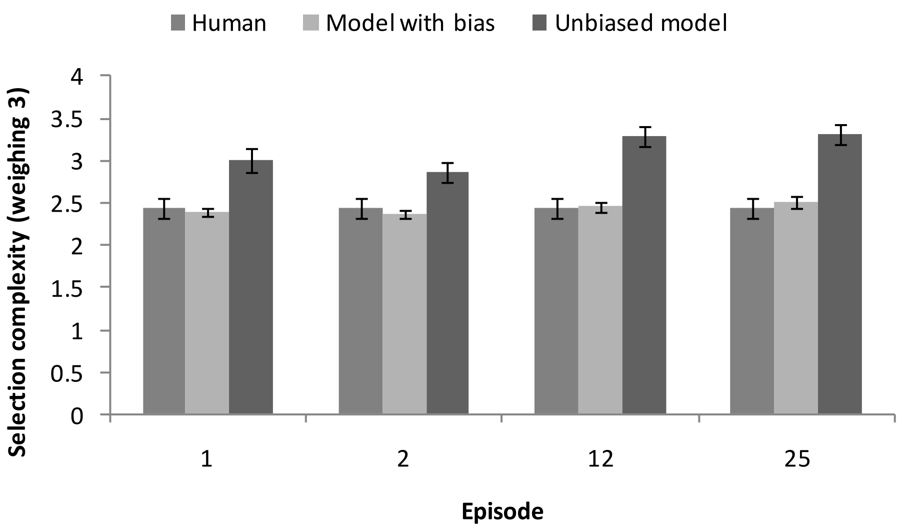


Figure 10 - Complexity of selection actions in weighing 3 for humans and models. Models are trained between a single episode, which is approximately equivalent to over human training, and 25 episodes, which yields an accuracy approximately equal to human level.

For weighing 3, the ANOVA on complexity reveals a main effect of model type, *F*(1,38) = 63, *p* < 0.001, stemming from larger complexity in the unbiased model (M= 3.1) than the biased ones (M=2.4). The effect of training level was also significant, *F*(1,38) = 237, *p* < 0.001, which indicates that selections were less complex early in training (1 episode: M=2.7) than late in training (25 episodes: M=2.9). Finally, the interaction was not significant, *F*(1,38)=1.8 , *p* > 0.05.

Asymmetry measure results are presented in Figure 11 and Figure 12 for weighings 2 and 3 respectively. We observe the same pattern as for complexity: unbiased models generate more asymmetrical solutions than humans; and again solutions generated by the biased models are comparable to humans in terms of asymmetry.

For weighing 2, the ANOVA of asymmetry reveals a main effect of model type, *F*(1,38) = 173, *p* < 0.001, stemming from larger asymmetry in the unbiased model (M=1.2) than the biased ones (M=0.5). The effect of level of training was also significant, *F*(1,38) = 3.7, *p* < 0.05, which indicates that selections were more asymmetrical early in training (1 episode: M=1.0) than later in training (25 episodes: M=0.8). Finally, the interaction was not significant, *F*(1,38)=1.7, *p*>0.05.

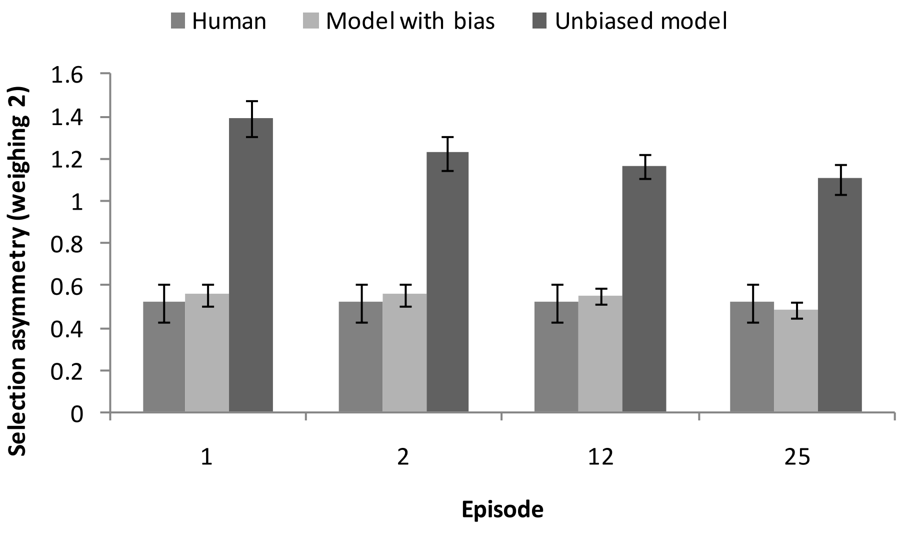


Figure 11 - Asymmetry of selection actions in weighing 2 for humans and models. Models are trained between a single episode, which is approximately equivalent to over human training, and 25 episodes, which yields an accuracy approximately equal to human level.

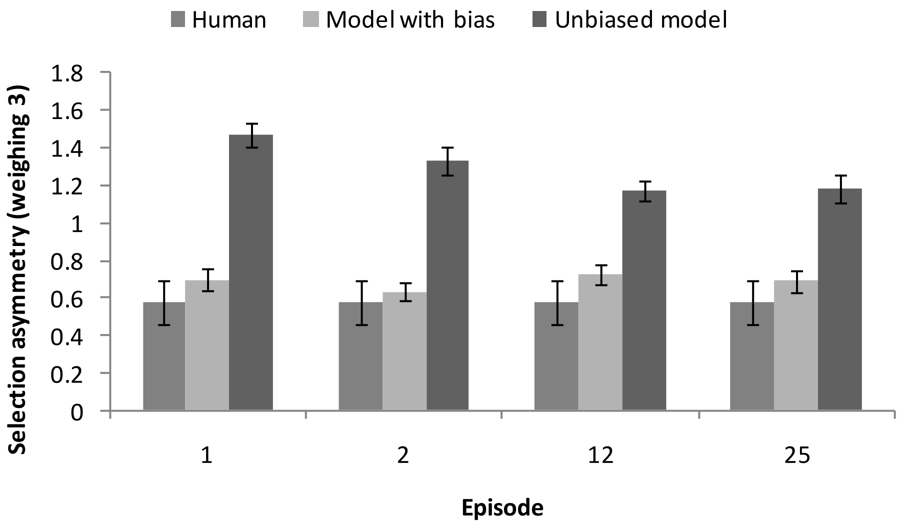


Figure 12 - Asymmetry of selection actions in weighing 3 for humans and models. Models are trained between a single episode, which is approximately equivalent to over human training, and 25 episodes, which yields an accuracy approximately equal to human level.

For weighing 3, the ANOVA on asymmetry reveals a main effect of model type, $F(1,38) = 120$, $p < 0.001$, stemming from larger asymmetry in the unbiased model (M=1.3) than the biased ones

(M=0.7). The effect of weighing was also significant, $F(1,38) = 2.9$, $p < 0.05$, which indicates that selections were more asymmetrical early in training (1 episode: M=1.1) than later in training (25 episodes: M=0.9). Finally, the interaction was significant, $F(1,38)=3.9$, $p < 0.05$, which suggests that the reduction in asymmetry was larger for unbiased models than biased ones.

4. Discussion

We implemented a distance-reduction heuristic as distance-based rewards. By generating rewards based on closeness to goal, models learn to prefer actions that lead to states closer to goal. We saw that distance-based rewards were sufficient to learn the Gizmo task. In fact, when distance-based rewards were available, environmental rewards appeared redundant and provided no additional benefit. This allowed us to make a prediction that was confirmed in human performance on this task: participants who obtain explicit feedback (rewards) on their performance do no better than control participants.

Concepts of symmetry and simplicity proved useful to discriminate between human solutions and model solutions. By default, models typically selected more complex and asymmetrical actions than humans did. However, the addition of a penalty term for complexity and asymmetry in model selections, in conjunction with a pre-training period, yielded solutions of equivalent complexity and asymmetry.

These two additions, a distance-reduction heuristic and cognitive biases, greatly improved the coverage of the model, although the accuracy of the model is still below human accuracy given equivalent overt training.

4.1 Why do distance-based rewards work better than environmental rewards?

Why do problem solvers learn better with distance-based rewards than environmental rewards? First, distance-based rewards are more frequent. In this problem, solvers can compute or estimate distance to the goal on every weighing. In contrast, environmental rewards are available only after the third weighing. Second, distance-based rewards are richer. While environmental rewards only indicate if a solution was found or not (binary value), distance-based rewards are graded as a function of distance. Such gradation allows problem solvers to compare failed solutions: the solution that moved the problem solver closer to the goal will generate more rewards and thus be judged as more desirable than the one that resulted in a larger distance to goal.

4.2 Why cognitive biases may be beneficial?

One can ask why humans tend to prefer simple and symmetrical actions. Of course, simplicity and symmetry are generally cognitively less demanding (they take less time to plan and execute, less inference to interpret results, etc.). Because it is probably adaptive for agents to minimize use of resources, starting with simple and symmetrical solutions is preferred.

Simulation results suggest an additional, less obvious explanation. We found that models that tend to select simple and symmetrical actions were more accurate than unbiased models, but only when training was short. In fact, the bias towards simplicity and symmetry appears to ultimately impede the generation of correct solutions to this problem in humans (Dandurand et al., 2007). This suggests that symmetrical and simple solutions may be better on average when not much is known about the problem. Selecting complex and asymmetrical steps is perhaps necessary to generate better solutions, but it is not sufficient. In fact, the set of asymmetrical and complex solutions is logically larger than the set of simple and symmetrical solutions, and may contain many poor actions[11]. With little a-priori knowledge of the task, starting with simple and symmetrical solutions not only saves resources, but it also yields better (yet suboptimal) solutions on average. If this example is typical of a wide variety of tasks, it may make sense for humans as problem solvers to have evolved a general preference for symmetry and simplicity, especially in a context where a reasonable (but not necessarily optimal) solution must be generated quickly with minimal effort, c.f., satisficing (Gigerenzer et al., 1999; Simon, 1957).

4.3 What is likely missing in the model?

In this research, we focused on a distance-reduction heuristic as a mechanism for learning using self-generated rewards. However, human problem solving likely involves more cognitive processes than the ones implemented in this model, including mechanisms or strategies for learning without rewards. For example, the present model lacks an explicit module to perform look-ahead reasoning. By mentally simulating the task, generating weight hypotheses and predicting outcomes, a problem solver could greatly save on overt trials. In fact, this task can be solved on paper by reasoning only. Paper here would act as an external aid because most humans probably could not completely solve the task mentally. However, the point is that humans can probably save on overt weighings by using covert, mentally simulated weighings. This may explain why humans require less overt training than current models. This could be tested by looking at the time-course of task performance. For instance,

[11] The fact that, as training proceeds, the accuracy of the biased models increases in a more stable and steady fashion compared to the unbiased models (see Figure 8) is consistent with the hypothesis that biased models explore a smaller portion of the problem space.

if we assume that it takes about the same time to obtain information by mental simulations and overt use of the computer program, participants should on average improve at the same rate, with respect to time, regardless of the number of trials overtly made. If we assume that information can be obtained faster using mental simulations than overt use of the program, participants who take longer per trial (and thus may engage in more mental simulations) may outperform those who complete trials quickly.

Work on algorithms such as TDLeaf, which combines search and reinforcement learning (Baxter, Tridgell, & Weaver, 1998), could serve as inspiration for developing a mentally-simulated, look-ahead module. Future research could also explore how other standard heuristics could be implemented in the model.

4.4 Similarities and differences with other models

In the introduction, we described a number of systems related to the present model. Here, we further describe the present work in the context of two particularly relevant research lines.

First, Polat and Abul (2002) developed a system for learning a Constrained Blocks World problem in a multi-agent context. Agents needed to find an action sequence that transforms an initial configuration of blocks into a goal configuration. In their system, rewards for non-terminal states were calculated by scaling (dividing) environmental rewards by an estimated distance to goal, as described in the introduction. While this approach allows closeness to goal to amplify the effect of environmental rewards, the latter are still needed, thus not addressing how learning can occur without environmental rewards. In contrast, the proposed approach based on distance based rewards (DBR) uses an additive scheme in which closeness to goal linearly combines with environmental rewards to allow problem solvers to learn from any combination of means-ends analysis or environmental rewards. The two systems also differ in the mechanisms they use to learn or store state-action-reward information – the DBR-based system uses neural networks to approximate rewards, whereas the Polat and Abul system uses lookup tables to store reward values. Recall that neural networks generalize to unvisited states and actions based on similarity, whereas standard lookup tables do not.

Second, Bianchi and collaborators (2008) developed a Heuristically Accelerated Reinforcement Learning (HARL) system for autonomous robots to learn to navigate an environment. HARL is very similar to DBR in using a weighted sum of explicit rewards and contributions of a heuristic function. The most important difference between HARL and DBR resides in where and how the combination of rewards and heuristic values occurs. In HARL, the heuristic score directly influences the choice of action (that is, the policy). The expected long term value of actions (i.e., Q values) is unaffected by the heuristic. As a consequence, the heuristic values influence, but cannot substitute for, explicit environmental rewards. In contrast, for DBR, heuristic values affect the choice of action throughout the value function (i.e., Q), which leaves the policy unchanged: probabilistically select one of the four actions with highest expected value. Because self-generated distance-based rewards are just another source of rewards for DBR, it can learn using any combination of self-generated heuristic rewards and explicit environmental rewards (even when no explicit reward is given). In contrast, HARL needs explicit rewards, and thus would not be able to learn with heuristics only.

Finally, the proposed connectionist approach to approximating expected rewards and implicitly learning rules or strategies contrasts with the production system approach of Fu and Anderson (2006). While both models learn from environmental rewards using SARSA and Softmax, only the present model includes distance-based rewards and cognitive biases. Another difference lies in the complexity of the problem undertaken. Whereas their problem can be expressed with about 4 production rules, the Gizmo problem comprises 6187 states and 5671402 actions, which would require many more rules.

4.5 Symbolic and connectionist models of problem solving

There are no explicit rules in the present model - computation of expected rewards is done in a connectionist system. This mapping provides implicit rules for solving problems. It is better described as rule following rather than rule use (Shultz & Takane, 2007), or here, strategy following rather than strategy use. In contrast, symbolic systems manipulate explicit, symbolic rules. While these symbolic models can explain how problem solvers select and combine the most appropriate rules for a given problem, they leave unanswered an important question: how did these rules or strategies enter the system in the first place?

The Clarion-based approach of Sun and Sessions (2000) proposes an interesting avenue in which two levels of representation co-exist, an implicit one and an explicit one. Explicit rules are inferred after learning at the implicit level, rather than needing to be given to the model.

While the implicit and distributed nature of the knowledge in connectionist models may make interpretation more challenging, it offers two important advantages over the explicit rules used in symbolic systems. Firstly, connectionist representations are more compact. Symbolic models typically function as look-up tables with an expected reward value uniquely associated with each rule, and independently of all other rules. For the gizmo problem, an explicit lookup table would require as many as 5,671,402 entries (one for each distinct state-action pairs). In contrast, our connectionist model learns a mapping of state-actions to values, and require about 10 hidden units, that is about 330 connection weights[12]. Secondly, generalization in connectionist models is more realistic than in symbolic counterparts – that is, unvisited states and actions will yield similar expected rewards to similar and known states and actions, but in a graded fashion (Shultz, 2001, 2003). In symbolic models, expected rewards are typically computed for and assigned to individual rules independently of other rules, and therefore we can expect symbolic models to require more learning.

Despite not explicitly manipulating rules, we acknowledge that the present system is not fully connectionist, particularly in the module that lists possible actions from some given state and the action buffer. Future work could investigate connectionist implementations of these modules (Rougier & O'Reilly, 2002; e.g., Rougier, Noelle, Braver, J. D. Cohen, & O'Reilly, 2005).

4.6 Relationships between DBR, hill-climbing and means-ends analysis

DBR implements a distance-reduction heuristic. How does this relate to other important distance-reduction techniques, namely hill-climbing and means-ends analysis? For this discussion, it is important to recall that, because DBR is an extension of an important TD-learning technique (SARSA), most characteristics of TD are also present in DBR.

Hill-climbing (HC) is a term broadly used in machine learning to refer to gradient ascent techniques (Russell & Norvig, 2003). It is also used in cognitive psychology to refer to a heuristic consisting of selecting and performing actions that move problem solvers closer to a goal state as quickly as possible (e.g., Robertson, 2001). As such, both HC and DBR use distance-to-goal information and rate as desirable being in a state closer to a solution.

An important characteristic of hill-climbing is that action selection relies on local information only, thus resembling an amnesiac trying to climb a mountain in thick fog (Russell & Norvig, 2003). Detour problems (for example, the Cannibals and Missionaries problem) provide a classical challenge to hill climbing. To find a solution, problem solvers must temporarily increase distance to goal to escape a local distance minimum, that is, a dead-end. Hill-climbing techniques remain stuck in such local optima. Furthermore, hill climbing is a greedy technique: it always selects the action that locally maximizes approach to goal. It is therefore deterministic without further exploration of the problem space: for a given problem, the sequence of actions selected will always be the same.

[12] In standard cascor, the number of connection weights equals 0.5 x hidden x (hidden + 1) + inputs x hidden + outputs x (inputs + hidden + 1)

In contrast, the fact that DBR is based on TD-learning has two important beneficial implications. First, what matters to DBR is maximizing the long-term value or sum of rewards, not only the immediate distance reduction as in standard hill climbing. For detour problems, the long-term value of actions that avoid dead-ends and lead to some global optima (that is, a goal state) will be higher than the value of actions that lead to local optima, despite the immediate rewards (i.e., distance reduction) being in the opposite direction. Second, DBR uses Softmax as a probabilistic action selection mechanism, which results in more exploration of the problem space. Thanks to these two characteristics, DBR should be able to solve detour problems. Exploring how DBR models handle such local distance minima would be an interesting avenue for future research.

Means-ends analysis involves selecting and applying an operator to transform the current problem state into a new state which is closer to some goal state in at least one dimension (Newell & Simon, 1963). Means-ends analysis is search-intensive, and combines forward and backward search (Rich, 1983). In contrast, TD-learning only uses one-step, forward-only search to list possible actions from the current state. Good estimates of the long-term rewards occur in SARSA by bootstrapping (Sutton & Barto, 1998), that is, by the gradual diffusion of Q values back by one time step (t+1 to t). It should be noted that TD does propagate some Q values signal backwards by one time step, but only to the visited states that have been reached by this forward search; it does not search backwards.

Preliminary work using think-aloud-protocols provides evidence that distance-reduction efforts are common in human solutions -- for instance, participants stating explicitly their objective of excluding as many gizmos as possible. However, more research would be needed to find direct evidence that participants are using backward search for solving the Gizmo problem. Backward search is, by necessity, a mental construction because the agent is, also by necessity, traversing the problem space in a forward fashion in reality. More research would also be needed to determine what sub-goals, if any, human participants identify in the Gizmo Task. For instance, entering the third weighing with at most 3 possibilities could be stated as a sub-goal. However, preliminary results suggest that sub-goal identification is rare. If sub-goals could be identified, future models could implement a hierarchical system that looks for appropriate sub-tasks. Studying hierarchical representations in reinforcement learning is an active area of research (e.g., Botvinick, Niv, & Barto, 2009).

5. Conclusion

To sum up, along with others, we provide additional support that reinforcement learning can be used to model human problem solving. We presented a connectionist model of problem solving based on an improved version of SARSA in which closeness to goal is rewarded, and complex and asymmetrical actions are penalized. These extensions allowed the model to better simulate human patterns of performance on the task.

DBR is a novel approach to generating rewards for SARSA. Distance-based rewards, when they can be computed, are denser and richer than their binary environmental counterparts. In this case, they are available at every weighing, and they provide a graded evaluation as a function of distance, allowing finer discriminations between action alternatives. In our proposed modification of SARSA, these rewards can be linearly combined with environmental rewards.

Finally, this research shows how multiple sources of information (here, environmental and distance-based rewards) can combine with realistic constraints (here, biases) in a unified way. To our knowledge, no other cognitive model has proposed such a unified framework based on a single learning rule with many psychologically plausible features of human problem solving (learning, means-ends analysis, cognitive biases, and an action buffer).

6. Acknowledgment

We thank François Rivest and Fermin Moscoso Del Prado for insightful comments and suggestions. We also thank Simcha Samuel for her help with experimental data collection. This work was supported by a McGill Major scholarship to F. D. and an operating grant to T. R. S. from the Natural Sciences and Engineering Research Council of Canada.

7. Appendix

7.1 Appendix 1 - Optimal solutions

The solution space was fully searched to find the exhaustive set of optimal solutions. Optimal solutions lead to the reliable identification, without guessing, of all 24 possible cases of target gizmos (12 gizmos x 2 weights -- heavy or light). Measuring accuracy of some solution as the average number of correct responses over the 24 possible cases, only optimal solutions will yield 100% accuracy. Sub-optimal solutions also lead to correct answers in some cases, but not reliably. For instance, a solution that would, on average, leave two possible gizmos to choose from after the third weighing would have an accuracy of 50% (50-50 chance).

In optimal solutions, each of the 24 possible case (12 gizmos x 2 weights) cause a distinct and unique sequence of scale results. The theoretical maximum of gizmos that can be distinguished given 3 weighings and 3 possible balance outcomes is $3^3 = 27$.

7.1.1 Optimal label updates

The following set of rules allows problem solvers to update labels optimally, at any weighing:

1. If the scale does not move, then all gizmos on it are of normal (N) weight.
2. If the scale moves, then all gizmos left in the bank are of normal (N) weight.
3. If there are gizmos of unknown (U) weight located on the side of the scale that moves up, then they are of Light or Normal (LN) weight.
4. If there are gizmos of unknown (U) weight located on the side of the scale that moves down, then they are of Heavy or Normal (HN) weight.
5. If there are gizmos of Light or Normal (LN) weight located on the side of the scale that moves down, then they are of normal (N) weight.
6. If there are gizmos of Heavy or Normal (HN) located on the side of the scale that moves up, then they are of normal weight.
7. If all gizmos are marked as of normal weight (N), except for one which is marked as Heavy or Normal (HN) or Light or Normal (LN) weight, then this gizmo is the answer, and should be relabeled as Heavy (H) or Light (L), respectively.

7.1.2 Optimal gizmo selection

In this section we present optimal gizmo selections, organized by weighing. Note that, due to symmetry, solutions are equivalent when the arrangements on the two sides of the scale would be exchanged or swapped.

7.1.2.1 First weighing

On the first weighing, the only optimal selection action is to install four versus four gizmos on the balance scale, as shown in Table 5. After optimal label updates, this optimal solution leads to two possible states (as the two unbalanced cases are in fact symmetrical): (1) 4xU, 8xN for the balanced case; and (2) 4xHN, 4xLN, 4xN for the unbalanced case. Using the distance measure described below, we see that distance to goal in both cases is 8.

Balance scale side 1	Balance scale side 2	Bank
4xU	4xU	4xU

Table 5 – Optimal solutions for weighing 1, starting from the 12xU.

7.1.2.2 Second weighing

Following the balanced case (state: 4xU, 8xN), two optimal solutions exist, as shown in Table 6. After optimal label updates, both solutions result in two possible states depending on the scale result: (1) 3x(HN and/or LN), 9xN; and (2) 1xU, 11xN.

Balance scale side 1	Balance scale side 2	Bank
3xU	3xN	1xU, 5xN
2xU	1xU, 1xN	1xU, 7xN

Table 6 – Optimal solutions for weighing 2, starting from the 4xU, 8xN state. Due to symmetry, sides 1-2 correspond equivalently to right-left and left-right.

Solutions found for the unbalanced case (state: 4xHN, 4xLN, 4xN) are presented in Table 7. One can verify that, after optimal label updates, all these solutions lead to states in which 2 or 3 gizmos are left with HN and/or LN labels (i.e., respectively, 10xN or 9xN), for all three possible scale outcomes (balanced, left side heavier or right side heavier).

Variant 1									Variant 2								
Scale side 1			Scale side 2			Bank			Scale side 1			Scale side 2			Bank		
HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N
2	1		2	1			2	4	1	2		1	2			2	4
2	1		1	1	1	1	2	3	1	2		1	1	1	2	1	3
3	2		1		4		2		2	3			1	4	2		
2	2			1	3	2	1	1	2	2		1		3	1	2	1
2	1		2		1		3	3	1	2			2	1	3		3
1	3			1	3	3		1	3	1		1		3		3	1
2	2		1	1	2	1	1	2									

Table 7 - Optimal solutions for weighing 2, starting from the 4xHN, 4xLN, 4xN state. Due to symmetry, sides 1-2 correspond equivalently to right-left and left-right. When applicable, solution variants in which HN and LN gizmos are interchanged are presented side by side.

7.1.2.3 Third weighing

Optimal solutions at the second weighing yield states in which 2 or 3 gizmos are left with HN and/or LN labels (and, respectively, 10xN or 9xN). The three scale outcomes (balanced, left side heavier or right side heavier) are used to discriminate up to 3 possibilities left (HN and/or LN). Possible solutions are shown in Table 8.

Variant 1									Variant 2								
Scale side 1			Scale side 2			Bank			Scale side 1			Scale side 2			Bank		
HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N	HN	LN	N
1			1			1		9		1			1			1	9
1			1				1	9		1			1		1		9
1	1				2		1	9	1	1				2	1		9
1	1				2			10									
1					1	1		10		1				1	1		10
1					1		1	10		1				1		1	10

Table 8 - Optimal solutions for weighing 3, starting from the states (1) 3x(HN and/or LN), 9xN; or (2) 2x(HN and/or LN), 10xN. Due to symmetry, sides 1-2 correspond equivalently to right-left and left-right.

7.2 *Appendix 2 - Cascade-Correlation parameters settings*

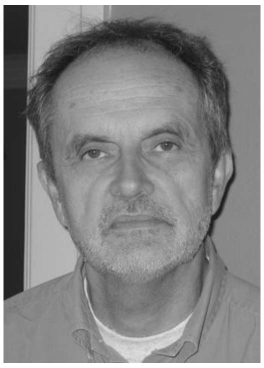
For this problem, target expected rewards can take on many different values, especially when all terms of equation 4 contribute. This requires more precise tracking of target values than typical binary classification tasks that simply require discriminating two values. While default cascor parameters were appropriate for binary classification, we empirically find that allowing cascor to remain for longer periods in input and output phases results in improved performance for this task. To allow cascor to learn for longer periods without switching phases, we selected: (1) a large value for the patience parameter (50 epochs in input and output phases rather than the default of 8); and (2) a low change threshold parameter (0.01 in input phase, and 0.002 in output phase rather than default of 0.03 and 0.01, respectively)[13]. These parameters are used to detect error reduction stagnation. Cascor switches phase if error reduced by less than "change threshold" over "patience" epochs. Cascor also switches phase after having reached maximum epochs in the current phase, set here to 200. We set the score threshold at .025 of the reward range to track target expected values with sufficient precision, as done previously (Dandurand & Shultz, 2009). No weight change is allowed to be greater in magnitude than the maximum growth factor times the previous step for that weight (Fahlman, 1988). Here, maximum growth factor was set to 2.0. The decay parameter, set to 0.0002 in output phase and to 0 in input phase, is used to keep weights from growing too big. Finally,

[13] Parameter values were the defaults in the previous model (Dandurand & Shultz, 2009).

the learning rate, which controls the amount of gradient descent used in updating weights, was set to 0.175 in output phase and to 1.0 in input phase.

8. Vitae

Frédéric Dandurand completed a PhD degree in psychology at McGill University. He is currently an NSERC-funded postdoctoral researcher at Université de Montréal. His principal research interest is the computational modeling of high-level cognitive processes such as language and problem solving. He is also a Professional Engineer in Canada and has worked as a software engineer for two major computer equipment companies.



Thomas Shultz (PhD Yale in Psychology) is Professor of Psychology and Associate Member of the School of Computer Science at McGill U. He teaches courses in Computational Psychology and Cognitive Science. He is a Fellow of the Canadian Psychological Association, and a founder and former Coordinator of McGill Cognitive Science. Research interests include connectionism, cognitive science, cognitive development, evolution and learning, and relations between knowledge and learning. He has over 200 research publications in these areas. He is a Member of the IEEE Neural Networks Society Autonomous Mental Development Technical Committee and Chair of the AMD Task Force on Developmental Psychology.

Arnaud Rey is currently a CNRS (Centre National de la Recherche Scientifique) researcher working in the domain of cognitive psychology and psycholinguistics at the Cognitive Psychology Lab (Marseille, France). He has completed a PhD in Cognitive Neuroscience at the University of Provence (Marseille, France) and a postdoc at Harvard University. Also, he has previously held an Assistant Professor position at the University of Bourgogne (Dijon, France). In 2007, he has been nominated as a Junior Member at the "Institut Universitaire de France".

9. References

Adams, J. L. (1974). *Conceptual blockbusting: A guide to better ideas*. Reading, MA: Addison-Wesley.

Akyurek, A. (1992). Means-ends planning: An example Soar system. In J. A. Michon & A. Akyurek (Eds.), *Soar: A Cognitive Architecture in Perspective* (pp. 109-167). Dordrecht, The Netherlands: Kluwer Academic Publishers.

Asgharbeygi, N., Nejati, N., Langley, P., & Arai, S. (2005). Guiding inference through relational reinforcement learning. *Proceedings of the Fifteenth International Conference on Inductive Logic Programming* (pp. 20–37). Bonn.

Baldassarre, G. (2002). *Planning with neural networks and reinforcement learning*. University of Essex, Department of Computer Science.

Baluja, S., & Fahlman, S. E. (1994). *Reducing network depth in the cascade-correlation* (No. CMU-CS-94-209). Pittsburgh: Carnegie Mellon University.

Barlow, H. B., Kaushal, T. P., & Mitchison, G. J. (1989). Finding minimum entropy codes. *Neural Computation*, *1*(3), 412–423.

Baxter, J., Tridgell, A., & Weaver, L. (1998). TDLeaf(lambda): combining temporal difference learning with game-tree search. *the Proceedings of the ninth Australian Conference on Neural Networks* (pp. 168-172).

Beale, I. L., Williams, R. J., Webster, D. M., & Corballis, M. C. (1972). Confusion of mirror images by pigeons and interhemispheric commissures. *Nature*, *238*(5363), 348-349.

Bianchi, R. A. C., Ribeiro, C. H. C., & Costa, A. H. R. (2008). Accelerating autonomous learning by using heuristic selection of actions. *Journal of Heuristics*, *14*, 135-168.

Biederman, I., & Cooper, E. E. (1991). Evidence for complete translational and reflectional invariance in visual object priming. *Perception*, *20*(5), 585–593.

Botvinick, M. M., Niv, Y., & Barto, A. G. (2009). Hierarchically organized behavior and its neural foundations: A reinforcement learning perspective. *Cognition*, *113*, 262–280.

Busemeyer, J. R., & Myung, I. J. (1992). An adaptive approach to human decision making: Learning theory, decision theory, and human performance. *Journal of Experimental Psychology: General*, *121*(2), 177-194.

Chater, N., & Brown, G. D. A. (2008). From universal laws of cognition to specific cognitive models. *Cognitive Science*, *32*(1), 36-67.

Chater, N., & Vitányi, P. (2003). Simplicity: a unifying principle in cognitive science? *Trends in Cognitive Sciences*, *7*(1), 19-22.

Corballis, M. C., & Beale, I. L. (1976). *The psychology of left and right*. New-York: Erlbaum.

Cutini, S., Ferdinando, A. D., Basso, D., Bisiacchi, P. S., & Zorzi, M. (2008). Visuospatial planning in the travelling salesperson problem: A connectionist account of normal and impaired performance. *Cognitive Neuropsychology*, *25*(2), 194-217.

Dandurand, F., & Shultz, T. R. (2009). Connectionist models of reinforcement, imitation, and instruction in learning to solve complex problems. *IEEE Transactions on Autonomous Mental Development*, *1*(2), 110 - 121.

Dandurand, F., Bowen, M., & Shultz, T. R. (2004). Learning by imitation, reinforcement and verbal rules in problem solving tasks. In J. Triesch & T. Jebara (Eds.), *Proceedings of the Third International Conference on Development and Learning: Developing social brains* (pp. 88-95). La Jolla, CA: University of California, San Diego, Institute for Neural Computation.

Dandurand, F., Shultz, T. R., & Onishi, K. H. (2007). Strategies, heuristics and biases in complex problem solving. *Proceedings of the Annual Conference of the Cognitive Science Society (CogSci 2007)* (pp. 917-922). New-York: Lawrence Erlbaum Associates, Inc.

Dandurand, F., Shultz, T. R., & Onishi, K. H. (2008). Comparing online and lab methods in a problem-solving experiment. *Behavior Research Methods*, *40*(2), 428-434.

Daw, N. D., & Frank, M. J. (2009). Reinforcement learning and higher level cognition: Introduction to special issue. *Cognition*, *113*, 259-261.

Dehaene, S., Nakamura, K., Jobert, A., Kuroki, C., Ogawa, S., & Cohen, L. (2010). Why do children make mirror errors in reading? Neural correlates of mirror invariance in the visual word form area. *Neuroimage*, *49*(2), 1837-1848.

Duncker, K. (1945). On problem solving. *Psychological Monographs*, *58*(5), 1-110.

Fahlman, S. E. (1988). Faster-learning variations on back-propagation: An empirical study. In T. J. Sejnowski, G. E. Hinton, & D. S. Touretzky (Eds.), *the Proceedings of the 1988 Connectionist Models Summer School*. San Mateo, CA: Morgan Kaufmann.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2* (pp. 524-532). Los Altos, CA: Morgan Kaufmann.

Feldman, J. (2003). The simplicity principle in human concept learning. *Current directions in psychological science*, *12*(6), 227-239.

Feldman, J. (2009). Bayes and the simplicity principle in perception. *Psychological Review*, *116*(4), 875-887.

Fiser, J., & Biederman, I. (2001). Invariance of long-term visual priming to scale, reflection, translation, and hemisphere. *Vision Research*, *41*(2), 221–234.

Freyd, J., & Tversky, B. (1984). Force of symmetry in form perception. *American Journal of Psychology*, *97*(1), 109-126.

Fu, W.-T., & Anderson, J. R. (2006). From recurrent choice to skill learning: a reinforcement-learning model. *Journal of Experimental Psychology: General*, *135*(2), 184-206.

Gigerenzer, G., Todd, P. M., & ABC Research Group. (1999). *Simple heuristics that make us smart*. Oxford, UK: Oxford University Press.

Gordon, D., Schultz, A., Grefenstette, J., Ballas, J., & Perez, M. (1994). *NRL task: navigation and collision avoidance*. Washington DC: Naval Research Lab.

Guy, R. K., & Nowakowski, R. J. (1995). Coin-weighing problems. *American Mathematical Monthly*, *102*(2), 164-167.

Halbeisen, L., & Hungerbuhler, N. (1995). The general counterfeit coin problem. *Discrete Mathematics*, *147*(1), 139-150.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. (1986). *Induction - Processes of inference, learning and discovery*. Cambridge, MA: MIT Press.

Holyoak, K. J. (1995). Problem solving. In E. E. Smith & D. N. Osherson (Eds.), *Thinking: An Invitation to Cognitive Science, 2nd edition* (pp. 267-296). Cambridge, MA: MIT Press.

Holyoak, K. J., & Thagard, P. (1996). *Mental leaps - Analogy in creative thought*. Cambridge, MA: MIT Press.

Houk, J. C., Adams, J. L., & Barto, A. G. (1995). A model of how the basal ganglia generate and use neural signals that predict reinforcement. In J. C. Houk, J. L. Davis, & D. G. Beiser (Eds.), *Models of information processing in the basal ganglia* (pp. 249-270). Cambridge, MA: MIT Press.

Kaplan, G. B., & Güzeliş, C. (2001). Hopfield networks for solving Tower of Hanoi problems. *ARI: An Interdisciplinary Journal of Physical and Engineering Sciences*, *52*(1), 23-29.

Langley, P., & Allen, J. A. (1993). A unified framework for planning and learning. In S. Minton (Ed.), *Machine learning methods for planning*. San Mateo, CA: Morgan Kaufmann.

Langley, P., Choi, D., & Rogers, S. (2009). Acquisition of hierarchical skills in a unified cognitive architecture. *Cognitive Systems Research*, *10*, 316-332.

Logothetis, N. K., & Pauls, J. (1995). Psychophysical and physiological evidence for viewer-centered object representations in the primate. *Cerebral Cortex*, *5*(3), 270-288.

Luchins, A. S. (1942). Mechanization in problem solving—the effect of< xh: i> Einstellung</xh: i>. *Psychological monographs*.

Nason, S., & Laird, J. E. (2004). Soar–RL: Integrating reinforcement learning with Soar. *Proceedings of the Sixth International Conference on Cognitive Modeling* (pp. 208-213). Mahwah, NJ: Erlbaum.

Newell, A. (1973). You can’t play 20 questions with nature and win. In W. G. Chase (Ed.), *Visual information processing*. New York: Academic Press.

Newell, A. (1980). Reasoning, problem solving, and decision processes: The problem space hypothesis. In R. Nickerson (Ed.), *Attention and performance VIII*. Hillsdale, NJ: Lawrence Erlbaum.

Newell, A. (1990). *Unified theories of cognition*. Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1963). GPS: A program that simulates human thought. In E. A. Feigenbaum & J. Feldman (Eds.), *Computers and thought*. New York, NY: McGraw-Hill.

Parks, R. W., Levine, D. S., & Long, D. L. (Eds.). (1998). *Fundamentals of neural network modeling: Neuropsychology and cognitive neuroscience*. Cambridge, MA: MIT Press.

Pizlo, Z. (2008). *3D Shape. Its unique place in visual perception*. MIT Press.

Polat, F., & Abul, O. (2002). Learning sequences of compatible actions among agents. *Artificial Intelligence Review*, *17*, 21-37.

Polya, G. (1957). *How to solve it. 2nd ed.* Princeton, NJ: Princeton University Press.

Pothos, E. M., & Chater, N. (2002). A simplicity principle in unsupervised human categorization. *Cognitive Science*, *26*(3), 303–343.

Provost, J., Kuipers, B. J., & Miikkulainen, R. (2006). Developing navigation behavior through self-organizing distinctive-state abstraction. *Connection Science*, *18*(2), 159–172.

Rich, E. (1983). *Artificial intelligence*. New York: McGraw-Hill.

Rieskamp, J., & Otto, P. E. (2006). SSL: A theory of how people learn to select strategies. *Journal of Experimental Psychology: General*, *135*(2), 207–236.

Rivest, F., & Precup, D. (2003). Combining TD-learning with Cascade-correlation networks. *the Proceedings of the twentieth International Conference on Machine Learning (ICML)* (pp. 632–639).

Robertson, S. I. (2001). *Problem solving*. East Sussex: Psychology Press Ltd.

Rollenhagen, J. E., & Olson, C. R. (2000). Mirror-image confusion in single neurons of the macaque inferotemporal cortex. *Science*, *287*(5457), 1506-1508.

Rougier, N. P., & O'Reilly, R. C. (2002). Learning representations in a gated prefrontal cortex model of dynamic task switching. *Cognitive Science:*, *26*(4), 503-520.

Rougier, N. P., Noelle, D. C., Braver, T. S., Cohen, J. D., & O'Reilly, R. C. (2005). Prefrontal cortex and flexible cognitive control: rules without symbols. *Proceedings of the National Academy of Sciences of the United States of America*, *102*(20), 7338-7343.

Rummery, G. A. (1995). *Problem solving with reinforcement learning*. Cambridge University, Engineering Department.

Russell, S., & Norvig, P. (2003). *Artificial intelligence, a modern approach. Second edition*. Upper Saddle River, NJ: Prentice Hall.

Shultz, T. R. (2001). Assessing generalization in connectionist and rule-based models under the learning constraint. *Proceedings of the Twenty-third Annual Conference of the Cognitive Science Society* (pp. 922-927). Mahwah, NJ: Erlbaum.

Shultz, T. R. (2003). *Computational developmental psychology*. Cambridge, MA: MIT Press.

Shultz, T. R., & Takane, Y. (2007). Rule following and rule use in simulations of the balance-scale task. *Cognition*, *103*, 460-472.

Shultz, T. R., Mysore, S. P., & Quartz, S. R. (2007). Why let networks grow? In D. Mareschal, S. Sirois, G. Westermann, & M. H. Johnson (Eds.), *Neuroconstructivism: Perspectives and prospects* (pp. 65-98). Oxford: Oxford University Press.

Simen, P., Polk, T., Lewis, R., & Freedman, E. (2002). A recurrent neural network model of goal management. *Proceedings of the International Conference on Computational Intelligence and Neuroscience* (pp. 504-508).

Simmel, M. L. (1953). The coin problem: a study in thinking. *American Journal of Psychology*, *66*, 229-241.

Simon, H. A. (1957). *Models of man*. New York, NY: Wiley.

Son, J. Y., Smith, L. B., & Goldstone, R. L. (2008). Simplicity and generalization: Short-cutting abstraction in children's object categorizations. *Cognition*, *108*(3), 626–638.

Sun, R. (1997). Learning, action and consciousness: a hybrid approach toward modelling consciousness. *Neural Networks*, *10*(7), 1317–1331.

Sun, R., & Sessions, C. (1998). Learning to plan probabilistically from neural networks. *Proceedings of IEEE International Conference on Neural Networks* (pp. 4-9). Piscataway, NJ: IEEE Press.

Sun, R., & Sessions, C. (2000). Learning plans without a priori knowledge. *Adaptive Behavior*, *8*(3/4), 225-254.

Suri, R. E., & Schultz, W. (1999). A neural network model with dopamine-like reinforcement signal that learns a spatial delayed response task. *Neuroscience*, *91*, 871-890.

Sutton, R. S., & Barto, A. G. (1990). Time-derivative models of Pavlovian reinforcement. In M. Gabriel & J. Moore (Eds.), *Learning and computational neuroscience: Foundations and adaptive networks* (pp. 497-537). MIT Press.

Sutton, R. S., & Barto, A. G. (1998). *Reinforcement learning: an introduction*. Cambridge, MA: MIT Press.

Taatgen, N. A., & Lee, F. J. (2003). Production Compilation: A simple mechanism to model complex skill acquisition. *Human Factors*, *45*(1), 61-76.

Tarr, M. J., & Pinker, S. (1989). Mental rotation and orientation-dependence in shape recognition. *Cognitive psychology*, *21*(2), 233–282.

Tesauro, G. J. (1995). Temporal difference learning and TD-Gammon. *Communications of the ACM*, *38*, 58-68.

Thrun, S. (1995). Learning to play the game of chess. In G. Tesauro, D. Touretzky, & T. Leen (Eds.), *Advances in Neural Information Processing Systems 7*.

Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E., & Blythe, J. (1995). Integrating planning and learning: the PRODIGY architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, *7*(1), 81-120.

Walsh, V., & Butler, S. R. (1996). The effects of visual cortex lesions on the perception of rotated shapes. *Behavioural brain research*, *76*(1-2), 127–142.

Wolff, J. G. (1982). Language acquisition, data compression and generalization. *Language & Communication*, *2*(1), 57-89.