Adapting to the task environment: Explorations in expected value

Wayne D. Gray *, Michael J. Schoelles, Chris R. Sims

Rensselaer Polytechnic Institute, Carnegie Building, 110 8th St. Troy, NY 12180, USA

Received 16 September 2004; accepted 16 September 2004

Abstract

Small variations in how a task is designed can lead humans to trade off one set of strategies for another. In this paper we discuss our failure to model such tradeoffs in the Blocks World task using ACT-R’s default mechanism for selecting the best production among competing productions. ACT-R’s selection mechanism, its expected value equation, has had many successes (see, for example [Anderson, J. R., & Lebiere, C. (Eds.). (1998). Atomic components of thought. Hillsdale, NJ: Lawrence Erlbaum Associates.]) and a recognized strength of this approach is that, across a wide variety of tasks, it tends to produce models that adapt to their task environment about as fast as humans adapt. (This congruence with human behavior is in marked contrast to other popular ways of computing the utility of alternative choices; for example, Reinforcement Learning or most Connectionist learning methods.) We believe that the failure to model the Blocks World task stems from the requirement in ACT-R that all actions must be counted as a binary success or failure. In Blocks World, as well as in many other circumstances, actions can be met with mixed success or partial failure. Working within ACT-R’s expected value equation we replace the binary success/failure judgment with three variations on a scalar one. We then compare the performance of each alternative with ACT-R’s default scheme and with the human data. We conclude by discussing the limits and generality of our attempts to replace ACT-R’s binary scheme with a scalar credit assignment mechanism.

Keywords: Expected utility; Expected value; Credit assignment; Strategy acquisition; Adaptive behavior; Reinforcement learning

1. Introduction

Few tasks are so new as to require the invention of strategies that have never been used by the task performer. Hence, in many situations, settling on a
strategy or set of strategies for performing a task is not so much a matter of learning new strategies as it is learning which strategy, out of a set of already acquired strategies, is best adapted to the current task environment.

Two remarkable aspects of this adaptation are that it is usually continuous and often unguided. Strategy selection continues to change and evolve even when the task performed is a routine act such as making photocopies of a book chapter (Agre & Shrager, 1990). This process occurs despite the absence of supervision or explicit guidance. In fact, performance improves far beyond what would be expected if, for each step, the choice among n possible alternatives were based solely on local considerations of utility. Generally, the class of non-local cumulative-effects models (Davis, Staddon, Machado, & Palmer, 1993) required to explain this behavior is known as unsupervised learning (Sutton & Barto, 1998). This is in contrast to supervised learning (such as is used in most neural networks) where the learning agent is told not only when it errs, but also how it should have behaved differently.

This paper is motivated by our attempts to model strategy selection in a Blocks World paradigm using ACT-R. First we introduce Blocks World and the empirical phenomena we seek to model. Second, in ACT-R a type of non-local cumulative effects model referred to as the expected value equation (Anderson, Bothell, Byrne, & Lebiere, in press; Anderson & Lebiere, 1998) determines which of two or more alternative strategies will be selected. After introducing the expected value equation, we present data from two variations of a model that uses the default equation. The variations differ by whether we update expected value after each strategy is executed, or whether we update after the entire task is completed. We then discuss reasons why the ACT-R mechanism is inadequate for modeling Blocks World. Third, we present three variations of ACT-R’s expected value equation and present data from the original model run with each variation. For each, we discuss how the variation influenced model behavior as well as its fit or misfit to the empirical data. Fourth, we present results from two variations of an abstract model that uses ACT-R’s expected value equation, but replaces nearly all else with estimates obtained directly from the human data. (As before, the variations differ in terms of when expected value is computed.) Fifth and finally, we summarize our work and draw conclusions regarding the Blocks World task specifically, our variations for calculating expected value, as well as the implications of our results for ACT-R.

2. Blocks World

Blocks World is a simple task that has been used to study the tradeoff between interaction-intensive and memory-intensive strategies (Ballard, Hayhoe, & Pelz, 1995; Fu & Gray, 2000; Gray & Fu, 2000; Gray, Sims, Fu, & Schoelles, in preparation). The task is to copy a pattern of colored blocks shown in the Target window to the Workspace window, using the colored blocks in the Resource window (for our version see Fig. 1).

2.1. The Blocks World studies

Each trial begins with a random placement of 8 colored blocks into empty spaces (defined by an invisible 4 x 4 grid) in the Target window. Unlike...
Fig. 1, during the study all three windows are covered by gray windows. In our studies the gray windows that cover the Resource window and the Workspace window vanish as soon as the cursor enters those windows. The between-Ss manipulation varies how effortful it is to uncover the Target window. Across three studies we have varied difficulty (a) “intuitively”, (b) by varying the Fitts’ Law Index of Difficulty, and (c) by lockout time (details of these studies are provided in Gray et al., in preparation). In each study, subjects were asked to do 40 (E1) or 48 (E2–3) trials. Each trial continued until they had correctly duplicated in the Workspace window the pattern (color and location) of blocks shown in the Target window.

The human and model data reported here are based on the version of Blocks World used in our third study. In that study, we varied the cost of opening the Target window by increasing the lockout time (i.e., the delay in uncovering the Target window after the cursor had been moved into it). The three conditions reported have a 0 (0-Lock), 400 (400-Lock), and 3200 (3200-Lock) millisecond lockout time.

2.2. Strategies

To access the information in the Target window subjects could adopt either an interaction-intensive or a memory-intensive strategy. An extreme interaction-intensive strategy would entail uncovering the Target window to obtain color information for a single block, obtaining that block from the Resource window, another uncovering of the Target window to obtain the block’s position information, followed by placing the block in the Workspace window. In contrast, an extreme memory-intensive strategy would entail one look at the Target window to encode both color and position for all eight blocks.

We did not expect to find either extreme strategy to be popular with our subjects. However, as the cost of accessing information in the Target window increased, we expected to find that subjects shifted from more interaction-intensive strategies to more memory-intensive ones.

2.3. The Blocks World results

For the current report, our measure of performance is the number of blocks correctly placed after the first, but before the second, uncovering of the Target window. At the time of the first uncovering, each of the lockout conditions has eight blocks that have to be placed. Our empirical data shows that the number of blocks placed on the first uncovering varies significantly between conditions. Hence, on subsequent uncovers, the number of remaining to-be-placed blocks differs between conditions. Likewise, as it takes some time for the models and humans to settle on stable strategies, we only report data for trials 25–48. The process of “settling in” is interesting but beyond the scope of this report.

There were 18 subjects in each of the three conditions. For these subjects, Fig. 2 shows that as lockout time increases, the number of blocks placed in the Workspace window increases. Human subjects are clearly trading off interaction-intensive for more memory-intensive strategies.

3. Failure to pick a good strategy: Issues in credit assignment and expected value

Our studies of Blocks World (Gray et al., in preparation) have shown that as the costs of opening the Target window increases subjects spend more time with the window open before going off to place the blocks. As the number of blocks placed also increases, the obvious inference is that the increased time spent with the Target window open, reflects increased time spent encoding a larger number of blocks.

To capture human adaptation to the cost of opening the Target window, we implemented a set of eight Encode-strategies. These strategies, Encode-1 through Encode-8, varied in the number of blocks they encoded per opening of the Target window. A strategy round began with choosing the strategy with the highest expected value and encoding the number of blocks specified by the strategy. The model would then go to the Resource window and attempt to retrieve the memory of an encoded, but not-yet-placed block. If a memory
element was retrieved, a block of that color was picked up from the Resource window and placed in the Workspace window. After placing a block in the Workspace window the would model try to retrieve another memory element of another encoded, but not-yet-placed block. When no more memory elements of not-yet-placed blocks could be retrieved, the model picked a new Encode-strategy according to its expected value and another round began. A trial ended with all eight blocks correctly placed in the Workspace window.

3.1. Model details

The above description generally characterizes our modeling approach. This section provides further details on the construction and operation of our model.

3.1.1. Limits on encode-strategies

On reflection, it will be clear that all eight Encode-strategies are relevant on the first uncovering of every trial, but not necessarily thereafter. For example, if on the first round of encoding, Encode-4 fired, encoded four blocks, and placed three, on the next round only five to-be-placed blocks would remain. Hence, on round 2, Encode-5, Encode-6, Encode-7, and Encode-8 would all encode five blocks. At best this would blur the distinction between Encode-strategies. At worst, it seems cognitively implausible that, for example, people would select a strategy to encode eight blocks when only one block remained to be placed. To avoid this problem we wrote our model so that an Encode-strategy would compete only if the number of to-be-placed blocks was greater than or equal to the strategy’s Encode-number (i.e., if four blocks remained, only Encode-4, Encode-3, Encode-2, and Encode-1 would be in the conflict set).

3.1.2. Calculating expect value

The Encode-strategies compete with each other based on their expected value. ACT-R’s expected value equation is:

\[ EV = PG - C \pm noise, \]

where \( P \) reflects the probability that a production has been successful in the past. \( P \) is simply calculated as the ratio:

\[ P = \frac{\text{successes}}{(\text{successes} + \text{failures})}, \]

where \( G \) is a constant expressed in units of time. \( G \) is loosely thought of as the number of seconds that a person would be willing to pursue a given goal. The default value of \( G \) is 20.
\[ C = \frac{\text{efforts}}{\text{(successes + failures)}} \]  

Finally, noise adds variability to the expected value, but rather than constituting unexplained variability it seems to be an essential element. Too little noise leads the system to prematurely settle on strategies that gain an early advantage in \( P \) and \( C \). Too much noise prevents the model from settling on any strategy, regardless of the values of \( P \) and \( C \). Indeed, in the machine learning community (where this issue is known as the exploration versus exploitation problem) noise has come to play an essential role in unsupervised learning (Sutton & Barto, 1998).

### 3.1.3. Model runs

One model was run with five different schemes for updating successes and failures. For convenience, we refer to the model when it is running a particular updating scheme as, for example, the Success-Weighted model or the All-Weighted model. However, each of these “models” used the same production rules, the same declarative memory elements, and the same settings for all ACT-R parameters. The Vanilla-Once and Vanilla-Each models differ in when expected value is updated (discussed in the next section). The Vanilla-Each and the Success-Weighted, All-Weighted, and Mixed-Weighted models differ in how expected value is computed.

With two exceptions, all ACT-R parameters were left at their defaults. Specific parameters important to our model include enable subsymbolic computations (:ese t); enable randomness (:er t); optimize learning (:ol t); parameter learning (:pl t); and base level learning (:bll 0.5). Although we make special mention of these parameters, this set is required by any model in which expected value and declarative memory activation is learned. They are all set to their default values. Our two exceptions do not have definite default values. We set activation noise (ans) to 0.23 and expected gain noise (egs) to 0.3. Activation noise is the noise added and subtracted to the activation of a declarative memory element on each retrieval attempt. The value we picked is within the normal range of this parameter and is one that we have used in other studies. Expected gain noise is the noise added and subtracted to the expected value of a production each time it appears in a conflict set (see Eq. (1)). The value we picked is within the range that we typically use in models (0.25–0.50). The setting of both egs and ans were done \textit{a priori} – neither were tuned to the particular results of our models.

### 3.1.4. Credit assignment

The credit assignment issue is “when” – when are the parameters in the expected value equation updated? These quantities could be updated for all productions once per trial; that is, after all eight blocks are placed. However, as our model interacts with the same software as our humans interact with, many hundreds of productions fire on each trial. Indeed, we counted 762 productions firings on a randomly sampled trial that took 128 s of ACT-R time to complete. (This count includes many refirings by some productions.) As all trials ended successfully, each production fired on a trial would have the value of its successes updated by one. (If it fired multiple times, it would receive multiple updates.) Each production fired on a trial would have its efforts incremented by the difference in ACT-R time between when it was selected and the end of the trial. (If it fired multiple times, by Eq. (3) its efforts would be updated for each firing by the difference between firing time and trial end time.)

Perhaps more to the point, placing eight blocks entails a number of different Encode-strategies firing a number of different times. This is the problem of structural credit assignment. Given a number of competing strategies, to what extent should each be credited with contributing to the final success of the trial? Updating all productions
at the end of each trial would make it extremely difficult for credit assignment to properly credit the success, failure, and cost of any given Encode-strategy.

We explored this issue by running two nearly identical ACT-R models that used ACT-R’s default expected value equation (Vanilla ACT-R). The only difference between the models was in where the update occurred. Vanilla-Once updated after placing all eight blocks (i.e., once per trial). Vanilla-Each updated after each firing of an Encode-strategy. Credit assignment time began ticking when an Encode-strategy was selected. Time ended when the model could no longer retrieve the declarative memory element of a not-yet-placed block. At this point another Encode-strategy was selected and a new credit assignment cycle began.

3.2. Problems in updating successes and costs

Neither of our two implementations of ACT-R’s default scheme for calculating expected values reproduced the data (see Fig. 2). By trial 25 the Vanilla-Each model always picked the Encode-1 strategy (see Table 1) whereas the Vanilla-Once model divided its attentions between Encode-1 and Encode-2 (see Table 2). Both models differ quantitatively (RMSE = 1.87 for Vanilla-Each and 1.46 for Vanilla-Once) as well as qualitatively from the human data ($r^2 = 0.33$ for Vanilla-Once; as the values for Vanilla-Each do not vary, regression cannot be calculated). Unlike the human data, neither model increases the number of initial blocks placed in the higher lockout conditions.

Unfortunately, the reason for this dilemma is as obvious as it is basic to the ACT-R calculation of expected value. For both models, as each round ends with a success, the value of $P$ stays at 1.0. The expected value is driven entirely by the costs. For the Vanilla-Each model (where credit assignment is updated once per each firing of an Encode-strategy), as we go from Encode-1 to Encode-8 the model spends more time encoding blocks, more time getting blocks from the Resource window, and more time placing blocks in the Workspace window. For the higher Encode-strategies as the costs in time soar, the expected value plummets.

The story for the Vanilla-Once model (where credit assignment is updated once per trial) is about the same but needs more elaboration. The basic insight is that the later in a trial an Encode-strategy can fire the lower, on the average, its cost will be. For example, suppose that the sequence of Encode-strategies that are selected on a trial is Encode-8, Encode-2, and Encode-1. (Due to ACT-R’s decay function, the longer the retention interval the more likely it is that one or more encoded items will not be retrieved. In this example we assume that Encode-8 can retrieve and place five blocks; whereas Encode-2 retrieves and places two, and Encode-1 retrieves and places one.) For Encode-8 to be selected there must be eight to-be-placed blocks remaining. Hence, for the Vanilla-Once model the effort for one firing of Encode-8 includes not only the time to encode eight and place five, but also the lockout, encoding, and placement time for Encode-2 as well as for Encode-1. In contrast, in this example, the effort for Encode-1 only includes one lockout plus the time to encode and place one block.

<table>
<thead>
<tr>
<th>Lockout</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>8.4</td>
<td>4.8</td>
<td>3.8</td>
<td>1.9</td>
<td>0.4</td>
<td>-1.0</td>
<td>-6.0</td>
<td>-4.7</td>
</tr>
<tr>
<td>400</td>
<td>8.1</td>
<td>5.1</td>
<td>4.0</td>
<td>3.7</td>
<td>-1.0</td>
<td>-1.5</td>
<td>-6.3</td>
<td>-6.7</td>
</tr>
<tr>
<td>3200</td>
<td>5.3</td>
<td>2.5</td>
<td>0.6</td>
<td>1.4</td>
<td>-1.0</td>
<td>-2.4</td>
<td>-9.8</td>
<td>-9.5</td>
</tr>
</tbody>
</table>

The higher the expected value the more likely the Encode-strategy is to be chosen.
The above example can be contrasted with an example in which eight successive Encode-1’s are selected. In this case, the time from beginning to end of a trial is amortized over eight firings of Encode-1 (see Eq. (3)). Each firing of each Encode-1 is assigned an effort based on the time from its firing to the end of the trial. In this sequence of eight Encode-1’s the cost of the first includes eight lockout times plus the time to get and place eight blocks, the cost of the second includes seven lockout times plus the time to get and place seven blocks, the cost of the eighth includes one lockout time plus the time to get and place one block. Averaged over all eight firings the computed costs for a trial would be less for Encode-1 than for Encode-8 in the prior example. The overall picture is that low Encode strategies can fire early or late in each trial, but the higher strategies can only fire early in each trial. Thus, the high encode strategies receive a disproportionate amount of the cost of each trial, and so the lower strategies are favored.

3.3. Implications for ACT-R

We believe that the failure to capture the qualitative trends reflects a fundamental flaw with the default ACT-R mechanism for credit assignment in tasks involving a sequence of steps over time. As Fu and Anderson (2004) point out, the current mechanism is inherently limited to binary feedback. For example, if Encode-4 is selected and successfully places four blocks, its success is updated by one. Similarly, if Encode-1 is selected and successfully places one block, its success is updated by one. Since the magnitude of the reward does not reflect the magnitude of the accomplishment, expected value is driven entirely by costs. As higher accomplishments require greater effort, ACT-R engages in a race to the bottom with the least-effort Encode-strategies winning out.

We asked ourselves whether this race-to-the-bottom situation was a necessary outcome of the ACT-R expected value formulation. Specifically, we asked whether if success and failure were defined by a scalar mechanism, rather than by a binary mechanism, the current formulation would better predict the data or whether it would still fall short. We pursued this question by constructing the series of three models that is discussed in the next section.

4. Weighting successes and failures

As success in Blocks World is defined as correctly placing eight blocks into the Workspace window, the Blocks World paradigm allows us to define partial success in terms of the number of blocks placed. Hence, if an Encode-strategy places one block into the Workspace window it is less successful than an Encode-strategy that places four blocks.

Prior models of ACT-R have apparently not had to deal with such nuances. In thinking about how to overcome this limit to ACT-R we generated a number of schemes that weight the updating of the successes and failures parameters by the number of blocks placed. Three of these schemes are discussed below and shown in Table 3.

Implementing these schemes required adding a hook to ACT-R’s parameters learning function to bypass the normal updating of the successes and failures parameters with the updates required by each scheme. The hook function is called by the parameters learning function for each production in the sequence.

4.1. Success-Weighted

Conceptually, the most basic change is to make successes a scalar vector that reflects the number of blocks correctly placed. This update is shown in the first row of Table 3. As the current model almost always successfully places at least one block, the Success-Weighted update is equivalent to dropping “failures” from the calculation of $P$ (compare Eqs. (2) and (4)) and $C$

<table>
<thead>
<tr>
<th>Weighting</th>
<th>$P$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Success</td>
<td>Failure</td>
</tr>
<tr>
<td>Success-Weighted</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>All-Weighted</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Mixed-Weighted</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
This change has the effect of setting \( P \) to one.

\[
P = \frac{\text{successes}}{\text{successes}}, \quad (4)
\]

\[
C = \frac{\text{efforts}}{\text{successes}}, \quad (5)
\]

The effect of our change is to increase the denominator of \( C \). Rather than adding one to successes each time an Encode-strategy has fired, our change adds in the number of blocks that have been successfully placed. (For example, if Encode-3 places three blocks, successes will be incremented by three. Likewise, if Encode-5 places three blocks successes will also be incremented by three.) Hence, the cost in terms of the additional time required to encode and place multiple blocks is amortized over the number of blocks actually placed.

### 4.2. All-Weighted

An alternative update would be to make both successes and failures scalar vectors. This alternative is shown in the second row of Table 3.

For All-Weighted, the equations for \( P \) and \( C \) are the same as the default equations shown in Eqs. (2) and (3). However, All-Weighted differs from the default in two ways. First, both successes and failures can be updated on a given round. Second, the number of successes and failures is weighted by the amount of the goal accomplished or attempted. As per the Success-Weighted scheme, All-Weighted increases the denominator of \( C \) and \( P \) by the number of blocks correctly placed (columns 2 and 4 of Table 3). However, unlike Success-Weighted, failures are also credited (columns 3 and 5). Failures are defined as the difference between the number of blocks encoded versus the number of blocks placed (Encode-number minus number-placed).

If, for example, Encode-8 fires and encodes eight blocks but places only three, then Encode-8 will be credited with three successes and five failures. Unlike Success-Weighted, All-Weighted affects the value of \( P \) by differentially changing both the numerator and denominator (as per Eq. (2)). An Encode-strategy that encodes more blocks than it can retrieve from memory will be punished by a decrease in \( P \) (the denominator increases faster than the numerator).

On the other hand, regardless of the number of successes and failures, for a given Encode-strategy, All-Weighted equally increments the denominator of \( C \) (see Eq. (3)). For example, if Encode-6 fires, encodes six and places six, six successes will be added to the denominator for \( C \). If the next time Encode-6 fires it encodes six but places three, the denominator will again be incremented by six (3 successes + 3 failures).

It would be one thing if All-Weighted were neutral with respect to the effect of success and failure on \( C \); however, it seems to reward failure. If six blocks are encoded and only one is placed, then the time (and therefore effort) between initiating the strategy and finishing the strategy is less than if six blocks were placed, but the effect on the denominator is the same. Counterintuitively, for the same Encode-strategy, costs are reduced more by an early failure than by an eventual complete success.

### 4.3. Mixed-Weighted

Mixed-Weighted is an alternative to All-Weighted that simply drops the count of failures from the denominator of costs. The expected value equation for Mixed-Weighted borrows its calculation of \( P \) from Eq. (2) and its calculation of \( C \) from Eq. (5). As per All-Weighted, if an Encode-strategy promises more than it can deliver, then it is punished by a reduction in \( P \). As per Success-Weighted, costs are reduced in proportion to the amount of the goal accomplished. Credit is not given for promises, only for results.

### 5. Model data: Comparing weighting schemes

The three models discussed here differ from the Vanilla ACT-R models only in the scheme they use for counting successes and failures. Like the Vanilla-Each model and unlike the Vanilla-Only

---

2 For each of the three models reported here, the model was run six times for each of the three lockout conditions.
model, the expected value is updated once per each firing of an Encode-strategy (i.e., not once per trial). In all other respects, in terms of productions, declarative memory elements, and all other parameters, the models are identical.

5.1. Success-Weighted model

As Fig. 2 shows, unlike the Vanilla ACT-R models, the Success-Weighted model consistently overshoots human performance and it seems fair to say that Success-Weighted provides a poor quantitative fit to the human data (RMSE = 1.71). Its qualitative fit, at first blush, appears quite good (\(r^2 = 0.89\)). However, this high regression value simply reflects the fact that the three points of the model data each increase as the three points of the human data do. As a more intuitive measure of qualitative fit, we propose looking at the slope of the Success-Weighted model as a percentage of the slope of the human data. By this measure the qualitative fit is 4% – a not very impressive number.

Across all three lockout conditions (see Table 4) the expected values of the smallest Encode-strategies, Encode-1 and Encode-2, is much below that of the other Encode-strategies. Post hoc comparisons show that the comparison of Encode-1 and Encode-2 versus Encode-3-8 was significant \([F(1, 105) = 1816, p < .0001]\) and accounted for 83% of the variance due to Encode-Strategy. Encode-3 is close to the higher Encode-strategies for 0-Lock, it begins diverging slightly for 400-Lock, and by 3200-Lock it is still close, but 1.3 units of expected value away from the next highest expected value. Hence, the three lockout conditions are relying on essentially the same pool of Encode-strategies with the slight increase in number placed for 400-Lock and 3200-Lock due to the less frequent use of Encode-3 in favor of a slightly increased use of the higher Encode-strategies.

5.2. All-Weighted model

Compared to the Vanilla and Success-Weighted models, the All-Weighted model is a much better fit. As shown by Fig. 2, this is the first model that comes close to capturing the qualitative and quantitative (RMSE = 0.83, \(r^2 = 0.92\), percent slope of human data = 40%) trends in the human data. Across the Encode-strategies the difference between the maximum and minimum expected value varied from 2.6 for 0-Lock (max = 9.0, min = 6.4), to 3.4 for 400-Lock (max = 8.9, min = 5.5), and 5.5 for 3200-Lock (max = 8.1, min = 2.6) (see Table 5). Post hoc comparisons showed that Encode-1 and Encode-2 had a much lower expected value for 3200-Lock than did the other Encode-strategies \([F(1, 35) = 67.7, p < .0001]\) with this comparison accounting for 79% of the variance due to Encode-strategy. This same comparison accounted for 15% of the variance for 400-Lock and 4% of the variance for 0-Lock. Hence, in contrast to the Success-Weighted model, it is clear that for the All-Weighted model a different mix of Encode-strategies was favored across the three lockout conditions.

5.3. Mixed-Weighted model

The Mixed-Weighted model is the best fitting of the three both qualitatively and quantitatively. Quantitatively it has the smallest RMSE (0.44). Qualitatively this model is as good a fit as that for the All-Weighted model (\(r^2 = 0.92\)); however, this model shows the greatest increase in blocks placed across lockout conditions (percent slope of human data = 60%). The difference between

<table>
<thead>
<tr>
<th>Lockout</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>7.1</td>
<td>8.8</td>
<td>10.0</td>
<td>10.2</td>
<td>10.3</td>
<td>10.4</td>
<td>10.2</td>
<td>9.9</td>
</tr>
<tr>
<td>400</td>
<td>6.3</td>
<td>8.1</td>
<td>9.4</td>
<td>10.2</td>
<td>10.4</td>
<td>10.4</td>
<td>10.1</td>
<td>9.8</td>
</tr>
<tr>
<td>3200</td>
<td>3.6</td>
<td>6.1</td>
<td>7.7</td>
<td>9.0</td>
<td>9.4</td>
<td>9.5</td>
<td>9.4</td>
<td>9.1</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Lockout</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6.4</td>
<td>8.1</td>
<td>7.8</td>
<td>9.0</td>
<td>7.9</td>
<td>7.5</td>
<td>7.0</td>
<td>6.8</td>
</tr>
<tr>
<td>400</td>
<td>5.5</td>
<td>7.8</td>
<td>8.9</td>
<td>8.5</td>
<td>7.4</td>
<td>7.1</td>
<td>7.3</td>
<td>6.3</td>
</tr>
<tr>
<td>3200</td>
<td>2.6</td>
<td>4.9</td>
<td>6.8</td>
<td>8.1</td>
<td>8.0</td>
<td>7.6</td>
<td>6.6</td>
<td>6.7</td>
</tr>
</tbody>
</table>
number of blocks placed at 0-Lock versus 3200-Lock is 1.84 for humans (see Fig. 2), 0.79 for Mixed-Weighted, 0.66 for All-Weighted, and 0.09 for Success-Weighted.

In terms of expected value, performance in the 3200-Lock condition is dominated by Encode-strategies 3–6 (see Table 6). The expected value of these Encode-strategies were quite similar. The next closest Encode-strategy was 0.50 expected value units below this range. In contrast, for 400-Lock both Encode-2 and Encode-8 were within the same range of expected value as Encode-strategies 3-6. For 0-Lock, Encode-2 fell within the range of values shown across Encode-strategies 3-6. Hence, compared to Success-Weighted and All-Weighted, for the Mixed-Weighted model as lockout time increases the extreme Encode-strategies (high as well as low) are less likely to be selected.

5.4. Comparing models

The default, or vanilla, ACT-R models simply cannot handle these data. A strategy is either a success or a failure. If it places at least one block it is successful. The Vanilla-Each model was entirely driven by costs to an exclusive use of Encode-1 over all runs of the model for trials 25–48. The Vanilla-Once model was more variable in its choice of Encode-strategies but it relied primarily on Encode-1 and Encode-2 across all three lockout conditions.

The Success-Weighted model reduced the costs of the higher Encode-strategies by the number of blocks they successfully placed. This cost reduction sufficed to boost the expected value of all higher Encode-strategies. The expected values for Encode-strategies 4-8 fell within 0.46 expected value units of each other for 0-Lock, within 0.56 for 400-lock, and within 0.52 for 3200-lock. Hence, the number placed was much higher than for the Vanilla models, but the number placed did not vary between lockout conditions.

The All-Weighted model punished strategies that encoded more than they placed by lowering their P value, but worked against itself by reducing costs based solely on the number encoded. This bias in reducing costs actually worked to favor strategies that encoded a lot but placed little. This all worked to favor Encode-strategies in the range of Encode-3 on up.

Like the All-Weighted model, the Mixed-Weighted model punished strategies that encoded more than they placed by lowering their P value. Unlike that model, it only reduced costs for the number of blocks actually placed. This combination worked to favor Encode-strategies in the range of Encode-4 to Encode-6 over both the lower and higher strategies. The Mixed-Weighted model provided the best qualitative and quantitative fit to the empirical data.

6. An alternative ACT-R model

The ACT-R models reported above were complete ACT-R models that used the same software task environment as human users. Interacting with that environment required a complex mixture of cognitive, perceptual, and action operations to move visual attention and the mouse around the Blocks World screen (see Fig. 1). Our goal in modeling was to produce a high cognitive-fidelity model of the range of interactive behavior required by humans to perform the task.

Although we ourselves do not believe it, we recognize that it might be the case that our problems in getting our model to fit the human data has nothing to do with the expected value equation, but more to do with some other aspect of the model that is constant across conditions. To check on this possibility we created simulated Vanilla-Each and Vanilla-Once ACT-R models in Lisp. The simulated models used ACT-R’s default expected value equation, but replaced most other parameters with estimates drawn directly from human data. Thus in constructing these models, we hoped to pin the blame squarely on ACT-R’s expected
value mechanism, and not some other aspect of ACT-R. In fact, we removed all productions from these models and simply divided its behavior into lockout time, encoding time, and placement time. Due to the drastic nature of our changes, the simulated models did not interact with the experimental software, but rather with an abstract model of the Blocks World environment.

For each lockout condition, the models used the lockout time for that condition. For encoding time we estimated a constant of 1-s per block. We derived this estimate from our empirical data and it is congruent with prior, theoretically based estimates established in our laboratory for the encoding time required in a serial attention task (Altmann & Gray, 2000). Placement time was also estimated from the empirical data. Finally, we estimated the probability that the model would place all the blocks that it encoded. In the human case, if the subject placed six blocks, it is impossible from our data to determine if they had encoded six, seven, or eight blocks in memory. Thus, we derived this parameter from our previous ACT-R models. Specifically, for each number of blocks attempted by the ACT-R model, we recorded the number of blocks actually placed, and from this data derived the probability table used in our simulations.

The simulated Vanilla-Each model produced nearly the exact behavior as the actual Vanilla-Each model (compare Fig. 2 with Fig. 3). Rather than always choosing Encode-1 it almost always chose Encode-1. During trials 25 to 48 the mean number of blocks placed ranged from 1.002 in 0-Lock, to 1.033 in 400-Lock, to 1.035 in 3200-Lock. Its RMSE was 1.85 and its slope was 1.4% that of the Human data. (The expected values for each Encode-strategy across the three lockout conditions are shown in Table 7.)

The simulated Vanilla-Once model was also very similar to the actual Vanilla-Once model (compare Fig. 2 with Fig. 3). Its RSME was 1.43 (compared to 1.46 for the actual Vanilla-Once model), its slope was better than for the actual model, but remained 40% that of the human data. (The expected value for each Encode-strategy across the three lockout conditions are shown in Table 8.)

We conclude from these simulations that the exact form of our ACT-R model did not contribute to our inability to model the data using the default expected value equations. Furthermore, these results strengthen our belief that the failure to predict the empirical results rests squarely on the expected value equation, and not some other component of the ACT-R theory.

![Fig. 3. Simulated ACT-R. Blocks placed following the first uncovering of the Target window for Humans versus two simulated ACT-R models. The simulation was written in Lisp and used encoding times per block and placement times per block that were estimated from the empirical data. (See text for fuller description.)](image-url)
7. Discussion and conclusions

We divide this section into a brief discussion of alternative changes to the expected value computations, conclusions about our work on the expected value equation, conclusions about our model of Blocks World, and the implications of our work for ACT-R.

7.1. Other changes to the expected value equation

Rather than changing how successes and failures are calculated, our initial instinct was to change $G$ – the value of the goal. In ACT-R, $G$ is nominally the time that a goal would be worth pursuing and its default is set to 20 s. It made much sense to us that if a goal was worth pursuing for, say 24 s, then the value of placing each block would be worth 3 s. A strategy that placed three blocks would have its $G$ incremented by 9 units. Whatever the merits of this scheme, unlike $P$ and $C$, $G$ does not accumulate separately for each production. Rather, $G$ is a global value that is applied equally to calculate the expected value for each item in the conflict set. To experiment with $G$ would require learning $G$. This would require more extensive changes to the current version of ACT-R (5.0) than the changes reported in this paper.

Other than our work, the only work we know that explores changes in ACT-R's calculation of expected value is that of Belavkin and colleagues. Whereas our work focuses on discriminating among strategies, Belavkin's work is focused more on changes in the range of expected values considered as the model gains expertise within a domain (Belavkin & Ritter, 2003) or on when to give up on a strategy (Belavkin, 2003).

7.2. Conclusions for calculations of expected value

This research into the parameters of the ACT-R expected value equation arose out of failure in trying to fit a model to the Blocks World data. As soon as we ran the model with ACT-R's default expected value settings (the Vanilla models) we realized that unless we could amortize costs and punish strategies that encoded more than they could place that we could not hope to fit the human data. The problem stemmed from the sparse reinforcement in the Blocks World environment, as well as the binary nature of success or failure in ACT-R. This dilemma prompted us to explore the space of expected values equations generated by changes in how successes and failures were counted and accumulated.

Our Mixed-Weighted model provides the best fit to our data and we believe it makes the most intuitive sense. For problems such as Blocks World where eventual success can be easily quantized into smaller units, it makes sense to us to reward and punish strategies based on how much of the problem they succeed in solving.

Table 7
Simulated ACT-R, Vanilla-Each

<table>
<thead>
<tr>
<th>Lockout</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>15.9</td>
<td>13.8</td>
<td>12.9</td>
<td>12.4</td>
<td>12.1</td>
<td>11.2</td>
<td>10.3</td>
<td>9.2</td>
</tr>
<tr>
<td>400</td>
<td>15.5</td>
<td>13.3</td>
<td>12.5</td>
<td>11.5</td>
<td>11.3</td>
<td>10.4</td>
<td>9.4</td>
<td>8.0</td>
</tr>
<tr>
<td>3200</td>
<td>12.7</td>
<td>10.7</td>
<td>10.0</td>
<td>9.2</td>
<td>8.6</td>
<td>9.2</td>
<td>8.4</td>
<td>7.2</td>
</tr>
</tbody>
</table>

Expected values after 48 trials for Encode-strategies 1–8 by lockout condition using ACT-R's default expected value equation and updating after each Encode-Strategy. See text for explanation of the simulation.

Table 8
Simulated ACT-R, Vanilla-Once

<table>
<thead>
<tr>
<th>Lockout</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2.47</td>
<td>1.21</td>
<td>0.62</td>
<td>0.41</td>
<td>-0.38</td>
<td>-0.71</td>
<td>-1.10</td>
<td>-1.81</td>
</tr>
<tr>
<td>400</td>
<td>0.99</td>
<td>-0.52</td>
<td>-0.90</td>
<td>-1.42</td>
<td>-1.82</td>
<td>-1.95</td>
<td>-2.07</td>
<td>-2.72</td>
</tr>
</tbody>
</table>

Expected values after 48 trials for Encode-strategies 1–8 by lockout condition using ACT-R's default expected value equation and updating once per trial. See text for explanation of the simulation.
7.3. Conclusions for Blocks World

Although we believe the Mixed-Weighted model is a general solution to similar problems, we do not believe that we have adequately modeled the Blocks World data.

Under the Mixed-Weighted scheme, a key to a strategy’s success or failure is its ability to retrieve from memory the items it has encoded. Currently we have run our models with optimized learning on (col t). This is the default for ACT-R models. However, in related research (Sims & Gray, 2004) we have come to believe that optimized learning overestimates the amount that can be retrieved in situations such as Blocks World where there is a long interval between an item’s early encoding and rehearsal versus its later retrieval.

Indeed, for the larger Encode-strategies there is a sizable period of time between the encoding and rehearsal of the first encoded block and the encoding and rehearsal of the last block on that round; this period, of course, precedes the long placement period. We believe that more realistic forgetting might work to drive the 0-Lock condition in the Mixed-Weighted model to rely more on lower Encode-strategies so that the number placed for 0-Lock declines to somewhere closer to the human data (see Fig. 2).

7.4. General conclusions and summary

Credit assignment in ACT-R is a binary value and the exact conditions for success and failure must be specified in advance. In this paper we have explored replacing the binary value with a scalar value. For all three alternative schemes the amount of success is not pre-specified, but is based on the number of blocks placed. Similarly, for the All-Weighted and Mixed-Weighted schemes the amount of failure is not pre-specified, but is calculated as the difference between the number of blocks encoded and the number successfully placed.

A strength of ACT-R’s default scheme for expected value is that the model changes strategies at approximately the same rate as humans do. This rate of change contrasts with the much longer training periods typically required by both Reinforcement Learning and Connectionist approaches. By working within the current ACT-R expected value scheme, our alternatives also gain the advantage of working in human time.

Our proposal contrasts with a more radical proposal to replace ACT-R’s expected value equation with a Reinforcement Learning approach (Fu & Anderson, 2004). Our favored revision, the Mixed-Weighted scheme, works within the basic framework of ACT-R’s current expected value equation. For data that are well fit by the current, Vanilla, expected value equation it is unclear to us how these fits would be affected by adopting our Mixed-Weighted scheme. For models in which success or failure is truly a binary decision, on success trials the Mixed-Weighted scheme would produce the exact effect of the current expected value equation. However, although on failure trials Mixed-Weighted would update $P$ as per Eq. (3) (the default); its effect on $C$ needs to be explored. During a failure trial the numerator for $C$ would continue to grow whereas the denominator would not (see Eq. (5)). This effect on $C$ would lead to a greater decline in a strategy’s expected value than would occur under the default scheme. Whether this greater decline would produce a poorer or better fit to human data is an empirical question that would need to be explored across a wide variety of phenomena.

In conclusion, we find ACT-R’s default expected value equation too restrictive. It is surprising to us that it has worked as well as it has. These past successes highlight the fact that there are a large number of circumstances in which binary success/failure is all that is required to explain human adaptation to the task environment. However, we view these circumstances as a special case of the larger set of adaptations and we would prefer a scalar approach to success and failure that would encompass the binary case.

We are not convinced that our approach will ultimately be successful. We are convinced that an exploration within ACT-R’s current formulas for expected value should be conducted before more radical changes are considered. Hence, the contribution of our work is to highlight important limitations to ACT-R’s credit assignment mecha-
nism and to begin discussion of whether that mechanism can be adjusted or must be abandoned.

Acknowledgements

The work reported was supported by a grant from the Office of Naval Research ONR #N000140310046. Additional support was provided by a subcontract to Rensselaer Polytechnic Institute from contract #MDA-904-03-C-0408 to Booz Allen Hamilton from the Advanced Research and Development Activity (ARDA). Thanks to Wai-Tat Fu for running human subjects as well as many other contributions to this project. This paper represents a substantial rewriting and expansion of a paper published in the Proceedings of the International Conference on Cognitive Modeling as: Gray, W. D., Schoelles, M. J., & Sims, C. R. (2004). Learning to choose the most effective strategy: Explorations in expected value. In M. C. Lovett, C. D. Schunn, C. Lebiere, & P. Munro (Eds.), 6th International Conference on Cognitive Modeling – ICCM2004 (pp. 112–117). Pittsburgh, PA: Erlbaum.

References


